DIMENSIONAL ANALYSIS OF ROBOT SOFTWARE WITHOUT DEVELOPER ANNOTATIONS

by

John-Paul William Calvin Ore

A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professors Sebastian Elbaum and Carrick Detweiler

Lincoln, Nebraska

August, 2019

DIMENSIONAL ANALYSIS OF ROBOT SOFTWARE
WITHOUT DEVELOPER ANNOTATIONS

John-Paul William Calvin Ore, Ph. D.

University of Nebraska, 2019

Advisers: Sebastian Elbaum and Carrick Detweiler

Robot software risks the hazard of *dimensional inconsistencies.* These inconsistencies occur when a program incorrectly manipulates values representing real-world quantities. Incorrect manipulation has real-world consequences that range in severity from benign to catastrophic. Previous approaches detect dimensional inconsistencies in programs but require extra developer effort and technical complications. The extra effort involves developers creating *type annotations* for every variable representing a real-world quantity that has physical units, and the technical complications include toolchain burdens like specialized compilers or type libraries.

To overcome the limitations of previous approaches, this thesis presents novel methods to detect dimensional inconsistencies without developer annotations. We start by empirically assessing the difficulty developers have in making type annotations. In a human study of 83 subjects, we find that developers are only 51% accurate and require more than 2 minutes per annotation. We further find that type suggestions have a significant impact on annotation accuracy. We find that when showing developers annotation suggestions, three suggestions are better than a single suggestion because they are as helpful when correct and less harmful when incorrect. Since developers struggle to make type annotations accurately, we present a novel method to infer physical unit types without developer annotations.

This is novel because it is the first method to detect dimensional inconsistencies in ROS `C++` without developer annotations, and this is important because robot software and ROS are increasingly used in real-world applications. Our method leverages a property of robotic middleware architecture that reuses standardized data structures, and we implement our method in an open-source tool, PHRIKY. We evaluate our method empirically on a corpus of 5.9 M lines of code and find that it detects real inconsistencies with an 87% TP rate. However, our method only assigns physical unit types to 25% of variables, leaving much of the annotation space unaddressed. To overcome these limitations, we extend our method to utilize uncertain evidence in identifiers using probabilistic reasoning. We implement our new probabilistic method in a tool PHYS and find that it assigns units to 75% of variables while retaining a TP rate of 82%. We present the first open dataset of dimensional inconsistencies in open-source robotics code, to our knowledge. Lastly, we identify extensions to our work and next steps for software tool developers to build more powerful robot software development tools.

COPYRIGHT

ACKNOWLEDGMENTS

Profound thanks to my advisors, Carrick and Sebastian, for their enduring patience and encouragement. Thank you, Matthew B. Dwyer, for sharing your deep insights and love of wisdom.

Thank you to my family Charles, Constance†, Heidi, Janna, Jon, Todd, Zoie, Kira, Fiona, and Ursula, for pointing to the guiding stars, cheerleading, cajoling, gentle reminders like 'Eyes on the Prize,' and your relentless love.

Words fail to express the depth of my gratitude to my brilliant wife, Dr. Aimee Allard. I'm looking forward to our journey together, hand-in-hand, on days both dark and bright, venturing in good courage on paths as yet unknown. Deep thanks as well to Aimee's family, Gregory and Carol Allard and my new brother-in-law Erik Allard for their kind support.

Thank you to my dear colleagues and friends for their assistance and feedback, including: Adam Plaucha, Adam Taylor, Ajay Shankar, Anne Rutledge, Ashraful Islam, Brittany Duncan, Carl Hildebrandt, Chandima Fernando, David Anthony, David Current, Evan Beachly, Gwendolyn Krieser, Hengle Jiang, Jim Higgins, Jinfu Leng, Jonathan Saddler, Justin Bradley, Lola Masson, Mitchell Gerrard, Nic Warmenhoven, Nishant Sharma, Pedro Albuquerque, Rubi Quiñones, Sayali Kate, Siya Kunde, Urja Acharya, William Wimsatt, and Xiangyu Zhang.

"For wisdom is better than rubies;

and all the things that may be desired are not to be compared to it."

Proverbs 8:11

GRANT INFORMATION

This work was partially supported by USDA-NIAF #2013-67021-20947, USDA-NIFA #2017-67021-25924, AFOSR #FA9550-10-1-0406, NSF #CCF-1718040, NSF #CCF-1526652 NSF #IIS-1116221, and NSF #IIS-1638099.

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these agencies.

Table of Contents

List of Figures

List of Tables

1 Introduction

Advances in robotic technology may increase the safety, reliability, and productivity of myriad human endeavors. For robots, the inescapable link between sensing and actuation is software. Robot software can enable new capabilities, like self-adaptivity and advanced autonomy. However, the potential benefits of robotics are fettered by our inability to rapidly prototype and deploy reliable, resilient software systems.

Building reliable robot software is hard because of software complexity and interactions between software, hardware, the environment, and the real-world. Additionally, the arsenal tools for software assurance is only beginning to focus on robot-specific concerns, leaving a gap between assurance about the runtime behavior of the software and assurance of the runtime behavior of the physical system. One concern for robot software is violating the rules of *dimensional analysis*. Essentially, dimensional analysis specifies that you can only add or compare quantities that are of the same kind, or *dimension*. Each physical dimension can also be represented by a unit of measure, such as time being measured in *seconds* (s) in the SI System [2]. All sensor values and all actuator commands are quantified in terms of physical units, such as *meter* (m) or *radian-per-second* ($\mathrm{rad\,s^{-1}}$). For robot software to be correct, every mathematical manipulation, assignment, or comparison of physical units must be correct. Further, when different software

```
189 float computeDistance(geometry_msgs::Pose goal, geometry_msgs::Pose current)
190 {
191     float dist = (goal.position.x - current.position.x)*(goal.position.x - current.position.x)
        + (goal.position.y - current.position.y) + (goal.position.y - current.position.y)
        + (goal.position.z - current.position.z) + (goal.position.z - current.position.z);
```

meters squared

meters

Figure 1.1: Code snippet from SoftBank's Romeo robot [1] containing a dimensional inconsistency detected by our tool Phriky, subsequently acknowledged and patched by the developers. *package*: ros-aldebaran *source*: `https://git.io/v6Xll`, *fixed source*: `https://git.io/v6xkH`

components exchange data, both must agree on what each element of exchanged data means in the real world [3]. Getting the physical units correct can be hard for developers to always get right.

Consider the simple code snippet in Figure 1.1 belonging to the 'Romeo' robot [1]. The expression on line 191 calculates the distance between the current position and the goal by multiplying and adding several values. These values are represented by the datatype `double`. The code compiles without complaint as all variables have the same programming type. However, this distance function incorrectly adds m to m^2, which is physically meaningless, and called a *dimensional inconsistency* or simply *inconsistency* in this work. The inconsistency in how the units are combined in the code constitutes a fault that will go undetected by the type system, likely to manifest later as incorrect behavior. Furthermore, this code might pass tests because it can be coincidentally correct at two or almost correct (a weaker version of a test oracle) for very small values of x, y, and z, making it difficult to detect the fault until the robot does something very wrong.

When a robotic software system incorrectly manipulates physical units, it can have real-world consequences, as shown in these three examples: 1) an interplanetary robot incinerated in Mars' atmosphere [4] after being sent a rocket-thrust command in *pounds-force* when it was expecting *Newtons*; 2) Air Canada 143 ran out of fuel mid-air [5] after being loaded with insufficient fuel when new avionics software had been updated to metric while the ground refueling system used Imperial units; and 3) the Cygnus spaceship aborted a docking procedure with the International Space Station [6] (ISS) after their GPS data strucutes were found to be unsynchronized when using two different time attributes. These high-profile, real-world consequences might represent only a fraction of all the times system developers encountered these hazards since this work finds dimensional inconsistencies in 6% of open-source robotics software repositories (§ 6).

Over the years, many solutions have been proposed to ensure that programs never contain dimensional inconsistencies. Already in 1978, Loveman and Karr [7] proposed protecting programs from these kinds of defects by employing a *type system*, a kind of logical framework that specifies rules for correctly handling data and operations that ensures a desirable property, called *type safety.* From a theoretical perspective, the problem of avoiding dimensional inconsistency in software programs is solved. However, in practice, developers often choose not to employ type systems because type systems require extra time and tools—a burden many developers are unwilling to bear. The burden requires developers to add extra information to every identifier in the program, specifically the physical unit type information, called a *type annotation*. Over the years, developers have "voted with their keyboards" [8] (see § 3.2.3.3) and chosen to build robot software without physical unit type annotations. These annotations have an anecdotal reputation of imposing an *annotation burden*, but there has been little empirical evidence on how accurately and quickly developers make type annotations.

Overall, this work seeks to better understand the burden of making type annotations, propose new methods for dimensional analysis of robotic programs without type annotations, and measure how frequently these dimensional inconsistencies occur in real-world software.

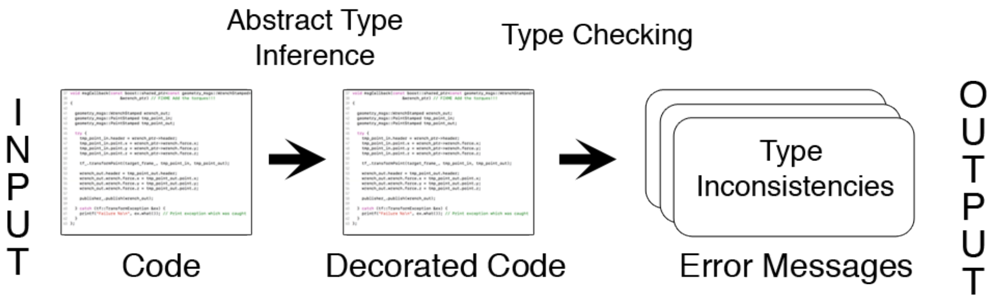


Figure 1.2: High-level overview of the proposed approach, Abstract Type Inference and Type Checking

To help quantify the burden of making type annotations, we design a human study and approximate the type annotation task using an online test, administered to 83 participants with programming experience recruited using Amazon's Mechanical Turk [9]. We find that developers are only 51% accurate when making type annotations and take nearly two minutes for each annotation. This result implies that making type annotations is a difficult, time-consuming process. But without complete physical unit type annotations, developers risk dimensional inconsistencies.

To help developers avoid dimensional inconsistencies without type annotations, we describe a method of inferring physical units type using evidence from shared message data structures common in the robotics software community (see § 2.4). By encoding physical unit conventions about these shared message data structures, we can automatically infer the physical unit type for some program variables. We find that our method is able to use the inferred types to detect dimensional inconsistencies with an 87% true positive rate, with no additional developer effort. As shown in Figure 1.2, we will do this with *Abstract Type Inference* (ATI). The figure shows untyped code as an input, and we use information from the robotics domain and information available in variable names to automatically infer the physical unit types for program variables. As shown in the figure, once we have inferred the physical unit types, we propagate these types through the dataflow of the code and detect type inconsistencies. Our method requires a one-time effort to link attributes of shared messages to their corresponding physical unit types. However, this one-time effort benefits all developers who use the shared messages, reducing duplicated work.

To shed light on how frequently these dimensional inconsistencies occur, we analyze a corpus of 5.9 M lines of code in open-source repositories that use these shared messages. We find that 6% (211/3,484) of repositories contain dimensional inconsistencies. We further find that 75% (267/357) of the dimensional inconsistencies we detect occur when developers use shared message contrary to their specified physical unit type, hindering interoperability. These findings are the first, to our knowledge, to measure how frequently dimensional inconsistencies occur in robot software.

1.1 Contributions

The contributions of this work include the following:

First, a study of developers showing that they correctly annotate variables with physical unit types only 51% of the time and require two minutes to make a single correct annotation. We find that correct suggestions significantly improve annotation accuracy. The study further determined that when showing developers annotation suggestions, three suggestions are better than a single suggestion because they are as helpful when correct and less harmful when incorrect.

Second, a method to automatically infer physical units for ROS variables and detect dimensional inconsistencies. This is novel because it is the first method to detect dimensional inconsistencies in ROS C++ without developer annotations, and this is important because robot software and ROS are increasingly used in real-world applications.

Third, an implementation of this method in an open-source tool PHRIKY, and an evaluation of PHRIKY showing an 87% True Positive (TP) rate in 231 open-source systems.

Fourth, a large-scale empirical study of PHRIKY on a corpus of 5.9 M lines of code. We find at least 6% (211/3,484) of repositories contain inconsistencies.

Fifth, an improvement to the detection power of PHRIKY using evidence in variable names combined with evidence from shared libraries in an open-source tool PHYS. PHYS was a collaborative effort previously published in [10]. The other authors contributions includes creating a substring similarity metric, using probabilistic graphical models for abstract type inference and formulating probabilistic constraints, choosing prior probabilities for various kinds of evidence according to norms in probabilistic reasoning, contributing to the core programming of PHYS, and contributing to the evaluation and debugging of PHYS's results. My contributions to PHYS in the work presented here includes guiding the extension of PHRIKY to reason probabilistically about type assignments in the tool PHYS, contributing to the evaluation and debugging of PHYS's results, creating the code corpus used during evaluation, examining and classifying the inconsistencies detected by PHYS, the comparison between PHYS and PHRIKY, and the proposed extensions to PHYS to make it an annotation tool.

Sixth, an empirical study of PHYS on 108 files to determine two things: 1) PHYS can infer units for 82% of variables; and, 2) PHYS detects dimensional inconsistencies with 82% accuracy in a corpus of 60 files.

Seventh, a detailed discussion of the design considerations required to extend Phys into a physical unit type annotation assistant. Specifically, we propose a new physical unit type annotation format, and ordering of the annotation worklist that balances the benefit of the information gained with cost of interrupting a developers current context.

Finally, an open dataset of physical inconsistencies identified by the tool Phys. To our knowledge, this is the first open dataset of dimensional inconsistencies.

1.2 Outline of Dissertation

The rest of this work is organized as follows. Chapter 2 presents background information about the SI System, dimensional analysis, and physical unit types, and gives motivating real-world code examples showing dimensional inconsistencies. Chapter 3 discusses related work and how dimensional analysis and physical unit types have previously been addressed in software engineering. Chapter 4 describes an empirical study of developers, investigating how accurately and quickly they make correct physical unit type annotations. Chapter 5 describes an improved method of ATI for physical unit types that capitalizes on assumptions about shared data structures commonly used in robotic systems. Chapter 6 details the result of an empirical study of a 5.9 MLOC software corpus, investigating how frequently dimensional inconsistencies occurs and what kinds exist. Chapter 7 describes a method for physical unit type inference using variable names that addresses some of shortcomings revealed during the empirical study. Finally, Chapter 8 discusses our contributions and identifies future work.

This work includes previously published material, specifically:

- Assessing the accuracy of type annotations (§ 4) appeared in [11] and an extension of the work is currently under submission.
- Inferring types from ROS messages (§ 5) appeared in [12].
- Phriky (§ 5.3) appeared in [13].
- An empirical evaluation on a large code corpus (§ 6) appears in [14].
- Phys (§ 7.2) appeared in [10] and is joint work with Dr. Xiangyu Zhang and Sayali Kate of Purdue University.

2 Background

This chapter presents background information in four areas: 1) physical units in the SI System and how they are used in programs; 2) dimensional analysis and inconsistencies; 3) how type checking is used to perform dimensional analysis using type annotations; and, 4) robotic middleware and the Robot Operating System (ROS).

2.1 Physical Units and the SI System

Physical phenomena are quantified in comparison to one another. The comparison is usually made with respect to a standardized quantity or unit, such as *meter-per-second* or *furlongs-per-fortnight*. The set of units we consider are the seven *base units* of the International System of Units (SI) [15], as shown in Equation 2.1. We also include *radian*, *degree*, and *quaternion* because they are widely used. The seven base units can be combined to represent other physical quantities and these combinations are called *derived units*. For example, the Newton is the SI unit of force and is a derived unit. One Newton can be expressed in terms of its equivalent base units, $(\textit{kilogram} * \textit{meter}) * (\textit{second} * \textit{second})^{-1}$, or equivalently $\mathrm{kg\,m\,s^{-2}}$.

To express units more formally, we extend the convention from Allen [16] that models units as types and defines a simplified *unit type language*:

$$ut ::= kilogram \mid meter \mid second \mid mole \mid ampere \mid kelvin \mid candela$$
$$\mid radian \mid degree \mid quaternion \mid unity \mid ut_1 * ut_2 \mid ut^{-1} \mid \delta \quad (2.1)$$

The binary operator '$*$' means multiplication, *unity* is identity, ut^{-1} is a unit's inverse, and δ represents the unknown unit. In the rest of this work, we omit '$*$' for brevity and adopt the convention that successive units are multiplied. The units *radian*, *degree*, and *quaternion* are equivalent to unity with dimensionless units meter-per-meter [17], but developers know and use them. The unknown unit δ is useful in expressing and tracking uncertainty in units. The grammar *ut* generates the set of all possible unit assignments.

2.2 Dimensional Analysis & Inconsistencies

All mathematical expressions must adhere to the rules of *dimensional analysis* for the results to be consistent with the physical world. The rules were first identified in 1822 by Fourier [18] but formalized in 1922 by Bridgeman [19]. These rules govern how physical quantities may be correctly combined, compared, and manipulated. Further, dimensional analysis abstracts quantities by *kind*, for example, all distances are lengths regardless of whether they're measured in SI Unit *meters* (m) or Imperial Unit *feet* (ft). The units *meter* and *feet* are lengths but different 'units-of-measure' or 'physical units'. The dimension is independent of the physical units. Essentially there are three rules in dimensional analysis:

1. Consider each quantity as a combination of one or more dimensions.

2. Only add/compare like with like.

3. Multiply quantities by adding exponents.

The simple logic of rule 2 is one we seek to enforce in programs. This work enforces the consistency rules of dimensional analysis but reports inconsistencies in terms of physical units, because developers are more familiar with physical units.

The rules of dimensional analysis still apply even if developers use quantities like *pound-feet*, from the Imperial unit system. Dimensional analysis applies because it is more general than any particular system of units. For example, adding $feet$ to $meters$ is dimensionally consistent though still physically incorrect. To be correct, quantities must be both dimensionally consistent and quantified in the same unit of measure. For quantities to be compatible, they must first be converted (scaled) to the same units. In this work, we focus the SI unit system because it is standard in the scientific and robot software communities [20]. We leave consideration of different units as a possible extension to our work (see § 7.8.4).

Every base unit in the SI system corresponds to a base dimension. For example, the base unit *meter* has a base dimension of length. Other measurements of length, like *smoots* or *furlongs*, have different units than *meter*s but the same dimension length. Based on dimensional analysis, we define rules for addition, comparison, and assignment as shown in Table 2.1, where $ut_{1,2} \in ut$.

Essentially, dimensional analysis specifies that you can only add or compare quantities with the same dimension. As shown in Table 2.1 the dimensional analysis rule for addition corresponds to Equation 2.2 and the rule for comparison corresponds to Equation 2.3. The rule in Equation 2.4 extends dimensional analysis to the software domain, because dimensional analysis has no notion of assignment.

INCONSISTENCY TYPE	DEFINITION
Addition of Inconsistent Units	$ut_1\{+,-\}ut_2 \rightarrow \{\text{consistent}\} \Leftrightarrow (ut_1 = ut_2) \quad (2.2)$
Comparison of Inconsistent Units	$ut_1\{<,>,\leq,\geq,=,\neq\}ut_2 \rightarrow \{\text{consistent}\} \Leftrightarrow (ut_1 = ut_2) \quad (2.3)$
Assignment of Multiple Units	$(ut_1 \leftarrow ut_2) \rightarrow \{\text{consistent}\} \Leftrightarrow (ut_1 = ut_2) \quad (2.4)$

Table 2.1: Dimensional inconsistencies types and their definitions.

```
736 void callback (const geometry_msgs::Twist &msg) {
737 // TODO: fix this it is ugly!!
738 // (divide ground truth from GPS!!)
739     if (! enableAbsoluteError) {
740         current_position.x = msg->linear.x;
741         current_position.y = msg->linear.y;
742         current_position.z = msg->linear.z;
743         }
744     desired_position.x = msg->angular.x;
745     desired_position.y = msg->angular.y;
746     desired_position.z = msg->angular.z;
747     }
```

Figure 2.1: Inconsistent assignment. ROS Message *Twist*, designed for linear and angular velocities, instead used for positions in lines 740-746. Comment from source.

Operations that violate Equations. 2.2-2.4 are called *dimensional inconsistencies* in this work.

We now show code examples for each of the three dimensional inconsistencies types shown in Table 2.1.

2.2.1 Assignment of Multiple Units

The code example shown in Figure 2.1 shows an assignment on lines 740 with the variable `current_position.x` being assigned a value from `msg->linear.x`. Both variables have the data type `float`, but they represent quantities with different physical units. Variable `msg->linear.x` is part of a class called `Twist`, declared in a shared library (see § 2.4 for more on shared libraries) `geometry_msgs` and specified to have physical units *meters-per-second*, while `current_position.x` is part of a class called `point` with attributes specified to have physical units `meters`. Because the specified units are different than the units being assigned, this code does not satisfy Equation 2.4 and is therefore inconsistent. Notice the comment "`TODO: fix this it is ugly!!`" on line 737, showing that some developer noticed this problem. We call this kind of inconsistency *assignment of multiple units.* As is, this code implicitly converts from one unit to another. At best, this inconsistency will make the code harder to maintain and understand. At worst, this implicit conversion might lead to unintended system behavior.

2.2.2 Comparison of Inconsistent Units

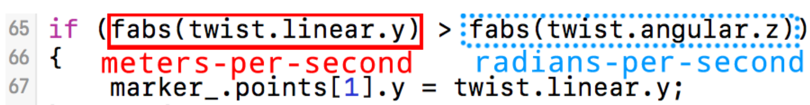


Figure 2.2: Inconsistent comparison. *package:*ros-teleop *source:*`https://git.io/v6Xld`

A second example is shown in Figure 2.2 line 65 where system developers compare two variables' magnitudes. The comparison is between `twist.linear.y` and `twist.angular.z`. The `Twist` data structure is defined in `geometry_msgs`, a shared library. The variable `linear.y` has units *meters-per-second* while the variable `angular.z` has units *radians-per-second*. This comparison does not satisfy Equation 2.3 and is therefore inconsistent, and we call this *comparison of inconsistent units*. The system developer might have a reason to make this comparison, but such choices in code are suspicious and should be conspicuously documented and justified, especially for shared code.

2.2.3 Addition of Inconsistent Units

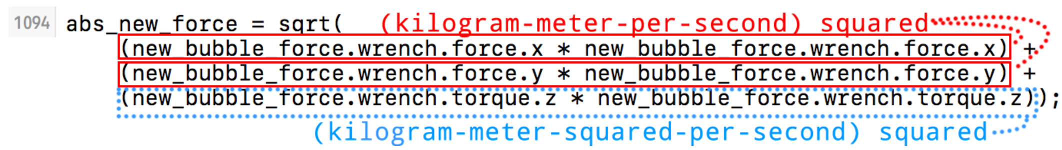


Figure 2.3: Inconsistent addition. Adds force to torque in distance metric.
package:eband_local_planner *source*: `https://git.io/v6X8T`

Figure 2.3 shows another example of an inconsistency on line 1094 in an addition expression. This sums the squares of three quantities: `force.x`, `force.y`, and `torque.z`. The problem with this expression is that the units for *force* ($\mathrm{kg\,m\,s^{-2}}$) are different than the units for *torque* ($\mathrm{kg\,m^2\,s^{-2}}$). Adding the square of *force* to the square of `torque` is not consistent by Equation 2.2. We call this *addition of inconsistent units*.

In the case shown in Figure 2.3, the developers *intentionally add units of different types to achieve a desired behavior*, specifically, to implement Quinlan and Khatib's 'elastic-band' controller [21]. This code, for example, creatively adds *force* to *torque* to limit the total *'force-torque'* exerted by a system. In the developer's defense, this calculation might behave as intended given input that implicitly normalizes these values. However, adding quantities with dissimilar units is generally devoid of physical meaning. Without explanation, this code might be considered a bewildering hack that works on one particular system, in one particular circumstance. Since this code is intentional, then the dimensional inconsistency reveals the existence of latent assumptions about the physical system. These assumptions hinder code re-use since system developers must duplicate the system and environment or risk unintended behavior.

These examples illustrate how dimensional inconsistencies—addition of inconsistent units, comparison of inconsistent units, and assignment of multiple units—can result in programs that are difficult to understand and maintain, incorrect, or hard to reuse.

2.3 Dimensional Analysis through Type Checking

Broadly, this work seeks to detect software faults by performing dimensional analysis through type checking programs. Type checking was first proposed by Milner, who said, "well-typed programs cannot go wrong" [22]. Milner observed that computer programs could be written in languages that assign *types* to data structures. These types add extra information about data structures and can be used in conjunction with a type theory to specify allowable operations and interactions. Applying type systems to programs requires developers to add some kind of type association, such as during variable declaration or by making type annotations. Making type associations takes time and makes the program more complicated, but yields benefits to developers such as fewer defects, easier maintenance, improved usability, as has been shown empirically (see § 3.2).

At a high level, there are three parts to type checking. First, type systems specify how types may interact while upholding desirable properties. In this work our type system is based on dimensional analysis and the SI System of units. Next, type associations connect identifiers in the program to a type in the type system. Lastly, type enforcement mechanisms check that the typed program conforms to the type systems' rules. There are multiple ways to connect identifiers to types. Primitives like `string`, `float`, and `int` are supported by many type languages and developers associate variables to a type when the variable is declared or assigned.

When a developer seeks to associate a program identifier with a type, she can do so in several ways, such as type support libraries, languages, or language extensions. For all these ways, developers must add extra information to associate an identifier to the correct type. Ideally, a developer might have *a priori* knowledge of every identifier's type, but this is not always the case. In many situations, such when reading or maintaining code, developers determine the correct type by reasoning about code operations and interactions in the type domain. Developers reason using the available evidence to infer a type for an identifier.

We define the **type annotation task** as follows. Let T be the set of types in some type domain and V be the set of program variables. Then the type annotation task is to find the function f that maps from program variables to types, such that:

$$f : \mathsf{V} \to \mathsf{T} \tag{2.5}$$

We assume the set of types T contains the empty element ϵ to account for the case when a program variable does not have a type. Finding the type annotation function f is usually a manual process and requires developers to find evidence to link program variables to types.

Definition 1. ***Type Annotation Task:*** *find a function from program variables to types.*

There are at least four kinds of code evidence developers could use to find f and reason about types: variable names, comments, code operations, and context. For example, in the code:

```
linVel = 0.42;
```

The variable name `linVel` provides a hint that this variable represents a linear velocity with physical unit type *meter-per-second* ($\mathrm{m\,s^{-1}}$) because it contains the substrings `lin` and `Vel`. The substring `lin` might be linear and the substring `Vel` might mean velocity.

Code comments can also contain useful clues. Consider, for example, the comment following this code:

```
goal_tolerance = 0.01; //one cm
```

The comment `one cm`, which might stand for *centimeter*, together with the value `0.01` provides evidence that `goal_tolerance`'s type is *meter* (m).

Code operations provide evidence for how variables interact with respect to the type domain. In the code:

```
x = x_vel * duration;
```

The physical unit type of `x` is inferred from evidence in the expression on the right-hand-side as the result of the multiplication expression. If `x_vel`'s type is linear velocity measured in *meter-per-second* ($\mathrm{m\,s^{-1}}$) and `duration`'s type is *second* (s), then `x`'s type must be *meter* (m).

The context surrounding a variable can provide useful clues for types. For example, domain specific libraries can define data structures with domain-defined physical unit types that, when used with other code, create a context in which other variables' types can be inferred. This kind of contextual clue is used by the type inference tool Phriky [13]. Variables that interact with shared libraries' data structures can then be inferred by flow. These contexts are limited in that not all program variables come from or interact with shared libraries.

Contexts, code operations, comments, and identifier names can help developers determine a unit type for a variable, but not all variables have a corresponding type in the type domain (their type is ϵ, the empty element). For example, Boolean values (`bool`) and program counters (`int`) rarely imply a physical unit type. Some values are *dimensionless*, a magnitude without physical units, such a scaling factor or ratio.

Determining whether a variable has a unit type is the first part of the annotation burden, followed by assigning the correct unit type. We denote the developer effort of time and energy to perform the type annotation task as the *type annotation burden*.

Definition 2. ***Type Annotation Burden:*** *The time and effort by developers to associate an identifier to a type, if any.*

2.4 Robotic Message-oriented Middleware: The Robot Operating System (ROS)

Robotic system developers recognized that a lack of a standard way to represent ubiquitous physical data, like range sensor readings and motor actuator commands, made software less reusable or modular [23, 3]. To improve robot software modularity, in 2001 NASA's CLARAty architecture [23] introduced a message-passing software architecture. In this message-passing architecture, reusable libraries specify data structures commonly used to exchange sensor and actuator values between software components [24]. Sensor and actuator data are attributes of classes defined in shared libraries. Shared libraries are code intended to be reused and shared across multiple contexts, often by many separate developers. The classes defined in shared libraries have attributes that are quantified in terms of physical units.

This message-passing architecture has now been widely adopted by the robotics community, especially in a popular framework called the Robot Operating System[1] (ROS) [25]. ROS programs are used increasingly in both academic and industrial

[1]As of July 2018, ROS has +5600 citations, monthly downloads of 16 M packages and 2 M web pageviews. Source: `http://wiki.ros.org/Metrics`

robots, including industrial automation at Boeing [26] and autonomous driving at BMW [27], and contains many variables representing quantities measured in physical units. ROS is specified to use the SI system [28].

These variables with physical units are attributes of classes specified in shared libraries. For example, a shared library for navigation is `nav_msgs`, for geometric relationships, the shared library is `geometry_msgs`, and for sensor values, the shared library is `sensor_msgs`. Within these libraries, there are a variety of attributes such as `geometry_msgs::twist.linear.x` ($\mathrm{m\,s^{-1}}$), `nav_msgs::odometry.angular.y` ($\mathrm{s^{-1}}$), and `sensor_msgs::imu.angular_velocity_covariance` ($\mathrm{s^{-2}}$). Note that the '::' symbol is particular to C++ and indicates a 'contained within' relationship. This link between data structures and their corresponding physical units is how we can apply dimensional analysis to programs without developer annotations..

Before ROS software is run, it is usually organized into ROS launch files. ROS launch files are a way to organize and interconnect separate computational concerns that together perform a unified purpose when executed. ROS launch files can start and control part of a system, a whole system, or multiple systems. A launch file is an XML file with named parameters that identifies separate, individually executable binary files that will all be executed simultaneously.

The tool HAROS [29], short for '**H**igh-**A**ssurance **ROS**' is a pluggable framework for running static analysis tools on ROS code. HAROS uses the launch file to identify sets of files, each set is a separate compilation unit.[2] As of 2019, the HAROS framework has been adopted by Open Robotics [30] (the maintainers of ROS) as the official code analysis framework for ROS. HAROS is both a static analysis tool and an umbrella tool that runs a collection of other static analysis tools (called

[2] A compilation unit is code, perhaps in multiple files, that the compiler treats as one logical unit.

'plugins' in Haros). One of the key features of Haros is its ability to statically identify how code is connected in a ROS computation graph given a ROS launch files [31]. In § 7.8.2, we propose an extension to our work that would enable compatibility with Haros.

Now that we have the necessary background, we turn to previous efforts that are related to our work.

3 Related work

Previous efforts relate to this work in several ways. We first discuss efforts related to robot software and shared library message data structures. Next, we discuss how dimensional analysis is applied to programs, both with and without manual type annotations. Finally, we look at work relating to helping developers make type annotations, the required link between program variables and type checking.

3.1 Unit Types in Robot Software

Support for standardizing the physical units used in robotic software was proposed in 2003 by Vaughn, Gerkey, and Howard [32] expanding on the ideas of NASA's `CLARAty` architecture (see § 2.4). However, static type checking specifically for robotics software was not implemented until 2007 by Biggs [33], to our knowledge. Biggs used custom `C++` type libraries to support dimensional analysis and the technique required manual type annotations. Like Biggs's work, we target robotic software, but unlike Biggs' work, we seek to be IDE-independent and type annotation free.

The importance of standard message formats between software components (like in the robotic middleware described in § 2.4) for checking unit interoperability was first identified by Damevski [3], and later emphasized in robot software by Walck *et al.* [34], Jung *et al.* [35], and Magyar *et al.* [24]. We exploit that ROS defines message structures in shared libraries that have physical unit types by specification. This might seem to limit the applicability of our proposed approach to just ROS programs, but we observe that other robotic message-passing-middle frameworks beyond ROS similarly adopt this standard message-passing design pattern, including: Orocos [36], OpenRTM [37], MOOS [38], and Yarp [39]. Because ROS is the most commonly used of these robotic middleware frameworks, we target ROS for impact.

3.2 Dimensional Analysis in Programs with Type Checking

In this section, we first discusses work related to type checking and its effectiveness, then describe how type checking is used to perform dimensional analysis. Next, we discuss methods to perform type checking both with and without developer annotations. Lastly, we discuss methods to help developers make type annotations.

3.2.1 Type Checking and Empirical Studies of their Effectiveness

One of the best and most time-tested methods of determining if a program has desirable properties is type checking [40, 41, 42, 43]. Many empirical studies confirm the benefits of type systems. Prechelt and Tichy [44] compared the impact of static type checking on student programmers using Ansi C and K&R, where Ansi C's compiler type checked procedure arguments and found significantly fewer defects in programs written with static type checking. Like this work, we are interested in empirical measurement of types, but unlike this work we use existing code artifacts (in § 4) rather than newly created ones and we assume a robotics domain with physical unit types. Spiza *et al.* [45] demonstrated that using type names alone helps an API's usability, even with no type enforcement mechanism. Hannenberg *et al.* [46] showed that programs using static type checking are easier to maintain. Rojas and Fraser's [47] work emphasized the importance of semantically useful names. We likewise find that variable names contain useful clues (see § 4.3.5), but unlike their work, we also find that a misunderstood name can lead to incorrect type annotations and false dimensional inconsistencies. The empirically-measured benefits of type systems can come at the cost to developers in time and effort to make type annotations.

3.2.2 Dimensional Analysis in Programs

For physical units and dimensional analysis, in 1978 Karr and Loveman [7][1] advocated for the design of programming languages with support for unit types, but required a separate type for every physical unit.

We instead specify physical types using a vector to represent the exponents of the seven base SI Units, an idea first proposed by Gonzalez *et al.* [49], yielding a compact representation of all possible units. Many subsequent efforts, including this work, use this vector representation because it allows multiplication by adding exponents. Novak *et al.* [50] presented a generalized algorithm for converting between different units-of-measure. We consider units-of-measure (i.e., *kilometer* vs. *millimeter*) to be natural extension of our work (se § 7.8.4). The next theoretical advance came when Allen showed that physical units form an Abelian group [16][2] that can be represented as a formal language. We adopt Allen's convention and show physical units as a formal language (see § 2.1).

[1]Karr and Loveman identify Cheatham's work [48] from 1960 as the earliest idea of incorporating dimensional analysis into programming languages.

[2]Abelian groups are finite or infinite sets with a binary operation (for units, multiplication) that satisfy associativity, commutativity, closure, and have identity and inverse elements.

Recent work by Xiang, Knight, and Sullivan [51] proposed type checking of 'Real-World Types' [52], a superset of dimensional analysis [19]. This includes 35 different real-world types and 97 type rules. Their analysis requires that an analyst examine all program tokens to decide what type rules apply and what needs to be annotated. Like their work, our work goes beyond dimensional analysis because we also check rotational representations like quaternions, common in the robotic domain. We also look for inconsistent use of data structures contrary to their specification (Equation 2.4). Unlike their work, the various techniques proposed in this work (§ 5 and § 7) do not require developer annotations.

3.2.3 Checking With Type Annotations

Most previous efforts impose both annotation burdens and toolchain burdens. The toolchain burdens include specialized compilers or dependencies on unit type libraries. Table 3.1 shows a summary of various efforts to enable dimensional analysis in programming languages. As shown in the table, some languages like `F#` have full language support for physical unit types and dimensional analysis built in by design, although they require type annotations. This section addresses methods requiring type annotations.

3.2.3.1 Full Programming Language Support.

Specialized language support for units based on the ideas of Allen [16] is built into the `Java` variant `Fortress` [80]. More recently, unit consistency as envisioned by Kennedy [89] has been built into `F#`. We observe that the open-source robotics community appears to have limited adoption of these languages, with our search [14] of $3,484$ open-source robotics repositories yielding no instances of either `Fortress` or

Table 3.1: Programming Languages and Support for Dimensional Analysis.

LANGUAGE	WITH TYPE ANNOTATIONS				WITHOUT ANNOTATIONS
	Full Support	Extension	Type Library	Rewriting	
Ada		[53] [54] [55][49] [56] [57]			
C		[58][50]		[59][60][61] [62] [63]	[64, 65]
C++			[66][67][68][33]		PHRIKY [12][13], PHYS [10]
Eiffel		[69]			
F#	[70]				
Fortran		[71] [72] [73]			
Haskell		[74]	[75]		
Java		[76] [77]	[16]	[78]	[79]
Java Fortress	[80]				
Lisp		[81]			
Pascal		[82][83] [84] [85] [86]			
Python			[87] [88]		

F# files. F# is not supported by Open Robotics (the maintainers of ROS) nor is there an indication that support for F# is planned, and Fortress might be supported by ROS in the future, but currently Java is supported only experimentally [90].

3.2.3.2 Programming Language Extensions.

Early efforts in the 1970-80s proposed programming language extensions to support dimensional analysis and physical unit checking of programs. All these works extend the target language and therefore require special compiler extensions to run. These efforts also require a type annotation for each variable representing a physical quantity. Gehani [82, 83] proposed extending Pascal, House [85] identified that Gehani's ideas of type consistency could be checked entirely at compile time, Agrawal *et al.* [84] built a dimensional analysis package for Pascal, Manner *et al.* [53] built an extension for Ada that required a separate type definition for each physical unit meaning that additional unit types had to be added to support particular applications, Baldwin *et al.* [86] implemented physical units for `Pascal`, Both Hilfinger *et al.* [55] and Rogers *et al.* [57] built `Ada` packages to support static physical unit type checking. Umrigar *et al.* [58] created a compiler that supported dimensional analysis on a non-standard version of `C++`, and Delft *et al.* [77] made an extension for `Java`. Unlike our work, all of these efforts impose toolchain *and* annotation burdens on developers. Notably, Orchard *et al.* [72] built a `Fortran` dimensional analysis tool that identifies 'critical variables' that would provide the most information when annotated and reduces the annotation burden by 80%. Orchard *et al.*identify this subset by framing the annotation problem as performing Gaussian Elimination on the set of linear equations formed by multiplication interactions between program variables *in log-space* [72].

We likewise would like to explore the impact of prioritizing variables for annotation. There have been so many 'units-of-measure' academic papers that Bennich *et al.* [91]' work 'The next 700 Unit-of-Measure Checkers' identified the vast quantity and variety of efforts, highlighting the missed opportunities for reusing existing analysis frameworks.

3.2.3.3 Type Libraries.

Several efforts seek to provide dimensional analysis using type libraries. Unlike program extensions, these efforts do not require specialized compilers and instead rely on an extensible type mechanism built into the language. These efforts create a dependency on the type libraries themselves and still require type annotations. Macpherson [56] built a library for dimensional analysis alone, relying on a newer version of `Ada` that supported type libraries. Cmelik *et al.* [66] use `C++` class templating to implement dimensional types that can be statically checked.

Brown *et al.* [67] created a static type library for `C++` tailored to the needs of the Fermilab research institute, including modes for high-energy and quantum physics. Jiang and Su's `Osprey` [63] targets `Java` and uses constraint solving to infer physical unit types for unknown variables. Schnable *et al.* [68] built `boost::units` for `C++`, but our analysis of 213 open source systems in § 5 finds only 3% of systems reference `boost::units`. Unlike these efforts, our work imposes no extra toolchain or annotation burden.

3.2.3.4 Mechanism to Transform Annotations

To avoid the dependency problems with type libraries, Chen, Feng, Hills, and Roşu released a series of papers [92, 93, 59, 60, 94, 61] describing a 'Program Rewriting' or 'pluggable policies' approach to enable dimensional analysis with static type checking. This technique still requires that developers undertake the effort of making type annotations, but puts the annotation (or type association) in program comments. Putting the type association in comments makes the program compatible with the compiler without adding dependencies, thereby avoiding the toolchain burden. Program rewriting uses type associations in these comments to transform the program to a version that can be statically checked for dimensional inconsistencies. Unlike their work, we encode the type association in a kind of lookup table (see the 'Mapping' in § 5.2). Also unlike their approach, we infer physical unit types from shared robot software libraries and variable names rather than requiring developers to make type associations manually.

3.2.3.5 Specialized Tools for Domain Specific Dimensional Analysis Support.

In addition to these extensions for general programming languages, many efforts built targeted support for dimensional analysis into domain-specific tools.

Khanin *et al.* [95] implemented dimensional analysis in `Mathematica`, Antoniu *et al.* [96] built a spreadsheet checker `XeLda`, Hinsley [97] implemented dimensional analysis in `Plancktonica`—a system for biological oceanographic computing, Broman *et al.* [98] built an extension for `Modelica`, Cooper *et al.* [99] devised a physical units checking enforcement mechanism for the modeling tool `CellML`, Maehne *et al.* [100] created an extension to `SystemC-AMC` that uses the type annotation library `boost::units` to declare domain-specific types for Systems-on-Chips, such as the micro (μ) distances, forces, and voltages. Roy *et al.* [101] built a tool SimCheck to check for dimensional inconsistencies in `Simulink` programs unit type annotations, and Owre *et al.* [102] built a tool DimSim also to check `Simulink` programs but with support for compositional analysis, meaning it could reason about physical units not previously declared as types but created through mathematical operations. Ou *et al.* [103] built a type system extension for physical units in `C++` related to computer graphic rendering that implemented dimensional analysis through type checking, but requiring type annotations. Eliasson [104] also built support for dimensional analysis in biological models in the language `CellML`. Griffioen built the `Pacioli` language as a proof-of-concept for statically typed matrices [105]. Nanjundappa *et al.* [106] built a *correct-by-construction* type extension to the modeling language MRICDF (Multi-Rate Instantaneous Channel-connected Data Flow) used for synthesis of embedded system software. Krings *et al.* [107] built a physical unit type library for the `B` modeling language using annotations inside *pragmas*, a kind of compiler directive.

Like these approaches, we target a particular domain (robotics), but unlike these approaches, our method works on a general programming language `C++` and imposes no toolchain burdens. Also, unlike our work, all these extensions require type annotations.

3.2.4 Checking Without Type Annotations

We now discuss efforts that do not require type annotations, as seen on the right-hand side of Table 3.1. In 2005, Gao, McCammant, and Ernst [64, 65] proposed a method to perform dimensional analysis without annotations by clustering program variables semantically, using dataflow analysis to determine variable interactions, and inferring inconsistencies by detecting flow between semantically distant clusters. The underlying assumption of their approach is that semantic similarity metrics like WordNet [108], which is based on large-scale analysis of natural language used on the internet, accurately identifies the similarity or dissimilarity of program identifiers. Since this technique relies on semantic similarities of any words, it is somewhat more general than dimensional analysis or even real-world types. For example, their technique could detect that adding a variable named `turtle` to a variable named `girdle` is likely inconsistent because the words are too semantically distant. This approach was expanded in 2015 the tool Ayudante [79]. Unfortunately, Ayudante did not report an extensive evaluation and the tool is not available to our knowledge. Like this effort, we seek to use information in variable names and combine this with dataflow analysis. Our investigations into this approach indicate that the greater generality of this approach comes at a cost within the physical units domain. Although words like 'speed' and 'velocity' are semantically similar by WordNet, within the robotics domain 'speed 'and 'velocity' can mean either linear or angular movement, a non-trivial and critical difference. Unlike their effort, we specialize for the robotics domain.

Another similar effort is the tool `UniFi` [78] that infers dimensions automatically by mining a program for contradictory variable type usages, much in the same manner as Lackwit [109], but for dimensional analysis. For example, if a program contains two variables n and m, and two statements $n = n + m$ and $n = n * m$, then if n and m have any physical unit type there exists a dimensional inconsistency, no matter what units n and m might have. Like those tools, we infer and propagate abstract types through assignment and detect inconsistent usage. Unlike their work, we can detect inconsistencies without requiring at least two contradictory usages of the same variable. For example, our approach can detect the addition of inconsistent units in Figure 1.1 even if these variables were used only once in this program, whereas UniFi would not detect this inconsistency.

Several efforts infer types using uncertain information in variable names and detect type violations such as Raychev *et al.* [110] and Xu *et al.* [111] both of which target Python programs. Overall, we propose a technique similar to these in § 7, but one key difference is that we model the constructive nature of physical unit types that can create new types (derived units, discussed in § 2.1) through multiplication and division. Our work also incorporates robotics domain knowledge that lets us infer some physical units with very high certainty, because they are specified in ROS message types. Dash *et al.* [112]'s tool RefiNym uses semantic information in variable names to suggest type refinements. Like their work we seek to use evidence in variable names but unlike their work we want to make an initial type inference rather than a refinement. Hellerdoorn *et al.* [8] learn type patterns from annotated TypeScript code to predict types in JavaScript. They report $\approx 70\%$ accuracy. Unlike their work, which uses a relatively simple type system (int, string, etc.), the type system of physical units is more complicated. Further, they have a large corpus ($+100,000$ files) of typed code to mine whereas we do not. Malik *et al.* [113] attempt to learn TypeScript types directly from identifiers, but their 84% type prediction accuracy would cause an unacceptable number of false positive inconsistencies to be useful. Again, unlike their work, we do not have a corpus from which to learn an association between identifiers and types.

3.3 Helping Developers Annotate Code with Tools

This section discusses techniques and tools to help developers make code annotations, and relates to our discussion of a type annotation tool extension presented in § 7.8.1. This related work also indicates that making manual annotations is a burden for developers, and therefore motivates both our inquiry into quantifying the annotation burden discussed in § 4 and our methods to apply type checking without developer annotations in § 5 and § 7. Unlike the previous sections, the annotations and types discussed here are broader than physical unit types, and the analysis is not based on dimensional analysis but instead based on type checking more generally.

3.3.1 Type Qualifiers

Type qualifiers extend an existing type system and require annotations to link identifiers to the qualified type. To help developer reason about the consequences of applying type qualifiers, Vakilian *et al.* created the interactive tool Cascade [114]. The authors found that Cascade works best when the developer and automated tool work together when compared with an automated tool working alone. Cascade is 'universal' in that it can apply any type qualifier that works with the Checker framework [115]. Further, Cascade is 'speculative,' meaning that it shows developers the potential consequences of assigning a type qualifier while developers add qualifier annotations. Shankar *et al.* [116] built a type qualifier system and inference-based type checker for legacy `C` programs to detect format string vulnerabilities. They use a special version of C called CQUAL [117] that includes an extensible type qualifier framework. Their approach requires manual annotation but uses flow to find tainted, unsecure strings. Like their work we use flow and seek to detect problems with existing systems, but unlike their work we do not require language variants or specialized compilers. Greenfieldboyce and Foster [118] proposed a similar type qualifier inference tool for Java, called JQUAL. JQUAL parameterizes the precision of the analysis, specifically for optional *context-sensitive* and *field-sensitive* analysis. Our analysis aims at a more lightweight analysis, leaving exploration of performance/precision trade-offs for future work.

3.3.2 Type Annotation Burden in Java and Javascript

Chalin *et al.* [119] report anecdotal evidence for the difficulty of annotating 'non-nullity' in large Java codebases to motivate their work on automatic annotation. Also in Java, Dietl, Ernst, and Müller [120] identify the type annotation burden as a primary motivation for their work on static type inference for Generic Universe Types. In JavaScript, Gao, Bird, and Barr [121] examined how type annotations can detect bugs, and quantified their annotation burden in terms of a *time tax* and *token tax*. The authors measured their annotation effort and reported the time and number of tokens to annotate to detect one bug. Using their *token tax* (token-annotation-per-bug) and *time tax* (time-per-bug), we infer their time per single annotation to be 127.8 s for TypeScript and 135.8 s for Flow. TypeScript and Flow are versions of JavaScript with a type system, type annotations, and a type enforcement mechanism (described in § 2.3). In our empirical study of developers making phyical unit type annotations in § 4.3.2, we measure a very similar 136.0 s for a single type annotation in § 4.3.2. Generally, we assume that we can help developers by automatically inferring types as opposed to making developers do all type annotations by hand. Like their work, we are interested in the cost of type annotation, but unlike their work, we measure the time for a population of 71 individuals and not just the three authors themselves and our work is in a very different domain, namely, physical unit types.

This concludes our discussion of related work. In the next chapter, we discuss an empirical study showing that developers struggle to make type annotations correctly.

4 Study of Developers: Better Understanding the Type Annotation Burden

This chapter presents an empirical study of developers that measures how quickly and accurately developers make type annotations. Type annotations are the link between program identifiers and their corresponding type in a type system. We discuss our research questions, methodology for the empirical study, and details of how the study is conducted including the sample population and code artifacts. We then present results for timing, accuracy, and an examination of how and why developers choose particular type annotations. We finish with an examination of code attributes that might influence the difficulty of particular annotations.

The material presented in this section extends material previously published in [11] by presenting new results for the impact of three suggestions versus a single suggestion.

4.1 Introduction

Type checking, as discussed in § 2.3, is one of the best and time-tested methods of ensuring that software has desirable properties. To enable the power of type checking, developers must have a way to associate identifiers to their corresponding type. Traditionally, developers make this association by adding type annotations.

Like code annotations generally, making type annotations is an onerous burden for developers. It is burdensome for several reasons, including that developers must first determine what needs to be annotated and then assign a correct type, all part of the *annotation burden* [119] (see Definition 2). But...*how* burdensome? In spite of the 'common knowledge' that making annotations is burdensome, we lack empirical evidence of how and why it is burdensome. By richly characterizing the factors that influence the difficulty of making annotations, we aim to help researchers and tool builders better target future solutions.

We concretize this work in the physical unit type domain (see § 2.1– 2.1), which is just one type domain amoung many that vary greatly in complexity [41]. Picking any single type domain might threaten how our results generalize (see § 4.4.1.2). However, no matter the type domain, we observe that developers must still reason about how code operations impact types in the type domain.

This chapter reports on an empirical study of 83 subjects to answer these questions about type annotations:

RQ$_1$ How accurately do subjects assign types?

RQ$_2$ How quickly can subjects make correct type annotations?

To address these foundational questions, we design an empirical study wherein we show subjects a code snippet with a variable that might require a type annotation. In our study, subjects choose a type annotation for the indicated variable from a drop-down list of frequently occurring domain types, and then subjects are required to provide an explanation for why they chose a type.

We believe that, in the near future, an increasing number of automated tools will be able to suggest type annotations. In cases when automated methods cannot determine the exact type with certainty, this leaves the developer to finalize the type given one or more suggestions. We find no previous work measuring the impact of suggestions on annotation accuracy. Therefore we ask:

RQ3 What is the impact of suggestions, both beneficial and detrimental?

RQ4 How does the impact of a single suggestion compare to three suggestions?

To better understand *why* and *how* developers assign a type, we pose a qualitative research question:

RQ5 Why do developers choose a particular type?

We address this question by requiring subjects to provide a detailed explanation of each type annotation, and organizing their responses using Grounded Theory [122].

Our findings are:

- Subjects' type annotation accuracy is only 51%.
- It takes more than two minutes to make a correct type annotation (136 s), so even smaller programs might take hours to manually annotate.
- Suggestions have a strong impact on annotation accuracy, with a single suggestion increasing accuracy to 73% when correct and reducing accuracy to 28% when incorrect.
- Three suggestions outperform one suggestion with respect to overall accuracy, because three incorrect suggestions are less harmful than a single incorrect suggestion, and three suggestions (correct first) benefits accuracy nearly as much as one correct suggestion.
- Providing multiple suggestions does not significantly increase the time to make a correct annotation.

- The main reason subjects provide for assigning a type is variable names, both for correct and incorrect annotations, while reasoning about the abstract domain together with variable names is more likely to credited for correct annotations.
- By analyzing code attributes, we find that making an incorrect annotation is significantly less likely in the presence of good identifier names.
- Annotation accuracy goes down as the number of variables involved in reasoning about the domain type assignment goes up.
- Identifying what variables need to be typed is valuable to developers.

4.2 Methodology: Accuracy, Duration, and Suggestions

Determining whether a variable has a unit type is the first part of the annotation burden, followed by assigning the correct type from type domain. In this work, we measure the time and accuracy of type annotations as proxies for the effort of the type annotation burden (see § 2.3). In this section, we describe both our research questions and how we address these questions with an experiment using a test instrument made from code artifacts. We discuss the experimental design by first showing how we find code artifacts from open-source repositories, then how code artifacts become test questions, and how subjects are recruited. Finally, we describe the phases of the study.

4.2.1 Research Questions

To better understand how developers make type annotations, we pose several research questions. By answering these questions, we seek empirical evidence for the accuracy and timing of the type annotation burden, the impact of suggestions, and the reasons developers make type annotations.

We now discuss each research question in detail.

4.2.1.1 RQ_1 How Accurately Do Subjects Assign Types?

This question seeks to measure the accuracy of developers assigning types to program identifiers. We do this for two reasons. Firstly, to determine if there is empirical evidence supporting the claim that the type annotation task is difficult for developers. Secondly, to establish a baseline accuracy for developers to make type annotations without automated support, which helps quantify the expected utility of a tool that automatically suggests types annotations.

4.2.1.2 RQ_2 How Quickly Can Subjects Make Correct Type Annotations?

This question helps us better assess how much time is required to correctly associate a type to an identifier. From this measurement, we might extrapolate the time required to annotate whole programs, and thereby better understand the temporal dimension of the type annotation burden.

4.2.1.3 RQ_3 What is the Impact of Suggestions, Both Beneficial and Detrimental?

Suggestions are important because we believe that developers will increasingly work together with type annotation tools to infer and suggest types. We imagine these kinds of tools might use sources of evidence with various degrees of certainty, such as domain knowledge, identifiers, comments, and context [123]. These sources could provide useful clues but are uncertain.

4.2.1.4 RQ_4 How Does the Impact of a Single Suggestion Compare to Three Suggestions?

We ask about the impact of three suggestions to determine if there is a difference between the effects of a single suggestion and multiple suggestions. We choose three because previous work suggests developers consider only the top few recommendations [124].

4.2.1.5 RQ_5 Why Do Developers Choose a Particular Type?

Unlike the previous research questions, RQ_5 is qualitative and asks *why* and *how* developers choose a type? In § 2.3, we identify possible sources of information developers might use to determine a type, but this question seeks to elicit the developer's reasoning and thought process for making particular type assignments. Once subjects have selected a type annotation, they are then required to provide an open-ended explanation. We collect explanations because we want to better understand how subjects reason about choosing a type, both when the type annotation is correct and when it is incorrect.

These are the research questions we address in this work. The next section describes the experiment we conduct to address these questions.

4.2.2 Experimental Setup

To answer our research questions, we design an experiment to replicate the type annotation. We considered a range of experimental options, including an in-person test with developers in a controlled environment. However, we reasoned that a web-based test would allow us to 'cast a wider net' and recruit more subjects drawn from a larger pool, since in a web-based test might cost less to administer per subject and could reach our to subjects accross the world.

We therefore design a web-based instrument that addresses all our research questions at the same time. In our experiment, we administer to developers a web-based test with questions based on code artifacts. Each test is a collection of 10 questions, drawn from a pool of 20 code artifacts. We apply treatments to questions to explore our research questions.

4.2.2.1 Type Domain and Code Artifacts

There are myriad type domains each with specific challenges and characteristics. For all type domains, developers seeking to assign a type annotation must reason about the abstract type and choose a type. We choose to instantiate the type annotation task (§ 2.3) within the domain of physical unit types, described in § 2.1. We choose physical units for several reasons: this domain is generally accessible to anyone with some physics background and includes all software systems that interact with real-world quantities, like robot and cyber-physical software. Further, subjects might have some previous exposure to physical unit quantities, and we have a special interest in robot software.

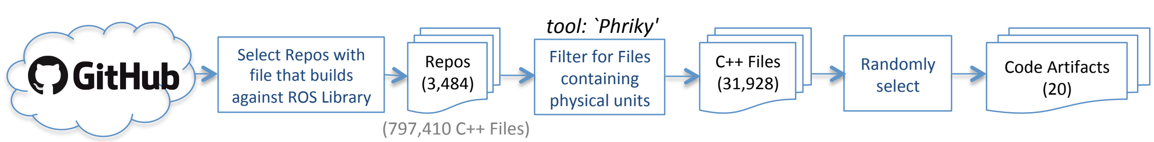


Figure 4.1: Process by which code artifacts are selected from the code corpus.

We collected code artifacts for our test instrument from a universe of open-source robot and cyber-physical software (see discussion of ROS [25, 14] in § 2.4) during February and March of 2018. As shown in Figure 4.1, this code is available on GitHub and repositories are selected for inclusion because some file contains a string matching the name of a ROS library, such as 'geometry_msgs::Pose.' The strings that match the names of a ROS library are typically part of a C++ #include statement. When a string matches the name of a ROS library, it is likely that the file containing that string uses the shared data structures defined in those ROS libraries, and therefore that file contains other variables implicitly representing physical quantities that can be annotated with a physical unit type. In the code corpus there are 797,410 C++ files based on matching by case-insensitive filename suffixes (.cpp, .c++, .cxx, .cc) from 3,484 repositories. We narrowed these 797,410 C++ files to 31,928 first using the tool Phriky [13] to find C++ files containing physical unit types using the command:

```
python ./phriky-units.py --only-find-files-with-units
```

This command uses the mechanism of recognizing ROS libraries mentioned above. We then exclude files that did not compile because of parser errors. From these 31,608 files, we randomly selected functions. We only allowed functions that met the following criterion: 1) no 'getter' or 'setter' functions; 2) more than 10 lines of code; 3) explore distinct types; and 4) code that had interactions between different types. We established these criterion to ensure variety, capture interactions in the type domain, and to avoid trivially easy artifacts. We repeatedly selected and screened functions until we had 20 artifacts from 20 projects. Within each artifact, we randomly select a variable using a random number generator to pick line numbers within the function until we land on an assignment statement for type annotation, reviewing the selection to ensure a variety of different types. All code artifacts used in this study are in Appendix D.

Table 4.1 shows the most common physical unit types found in a large corpus of open-source robot software [14], in decreasing order of frequency. In the table,

Table 4.1: Physical unit types used in our study. COVERED indicates that a physical unit type is a correct answer in our study.

PHYSICAL UNIT TYPE	DESCRIPTION	SYMBOL	COVERED
meters	distance	m	✓
second	time	s	
quaternion	3-D rotation	q	✓
radians-per-second	angular velocity	$\mathrm{rad\,s^{-1}}$	✓
meters-per-second	linear velocity	$\mathrm{m\,s^{-1}}$	
radians	2-D rotation	rad	✓
meters-per-second-squared	acceleration	$\mathrm{m\,s^{-2}}$	✓
kilogram-meters-squared-per-second-squared	torque	$\mathrm{kg\,m^2\,s^{-2}}$	✓
meters-squared	area	$\mathrm{m^2}$	✓
degrees (360°)	rotation	deg°	
radians-per-second-squared	angular acceleration	$\mathrm{rad\,s^{-2}}$	✓
meters-squared-per-second-squared	velocity covariance	$\mathrm{m^2\,s^{-2}}$	✓
kilogram-meter-per-second-squared	force	$\mathrm{kg\,m\,s^{-2}}$	
kilogram-per-second-squared-per-ampere	magnetic flux density	$\mathrm{kg\,s^{-2}\,A^{-1}}$	✓
Celsius	temperature	°C	
kilogram-per-second-squared	spring constant	$\mathrm{kg\,s^{-2}}$	✓
kilogram-per-meter-per-second-squared	air pressure	$\mathrm{kg\,m^{-1}\,s^{-2}}$	
lux	luminous emittance	lx	
kilogram-squared-per-meter-squared-per-second-to-the-fourth	force covariance	$\mathrm{kg^2\,m^{-2}\,s^{-4}}$	

the 'COVERED' column denotes whether our study used that physical unit type as a correct answer.

4.2.2.2 Treatments

To answer our research questions, we make a small change, called a treatment, to questions included in our online test instrument, where the difference between these small changes helps us measure their impact on question accuracy and timing. A sample question and artifact from our test is shown in Figure 4.2. The code in the figure shows a callback function in a reactive software system. As shown in the figure, we provide a visual indicator for the variable to be annotated, and this visual indicator corresponds to the line number referenced in the question

```
28 // Function for conversion of quaternion to roll pitch and yaw. The angles
29 // are published here too.
30 void MsgCallback(const geometry_msgs::PoseStamped msg) {
31   geometry_msgs::Quaternion GMquat;
32   GMquat = msg.pose.orientation;
33
34   // the incoming geometry_msgs::Quaternion is transformed to a tf::Quaterion
35   tf::Quaternion quat, quattemp;
36   tf::quaternionMsgToTF(GMquat, quattemp);
37   //    ROS_INFO("quat.x =%f, quat.y=%f, quat.z=%f, quat.w=%f", quattemp.x(),
38   //    quattemp.y(), quattemp.z(),quattemp.w());
39   quat =
40       tf::Quaternion(quattemp.x(), -quattemp.z(), quattemp.y(), quattemp.w());
41
42   // the tf::Quaternion has a method to acess roll pitch and yaw
43   double roll, pitch, yaw;
44   tf::Matrix3x3(quat).getRPY(roll, pitch, yaw);
45
46   // the found angles are written in a geometry_msgs::Vector3
47   geometry_msgs::Vector3 anglesmsg;
48   anglesmsg.z = yaw;
49   anglesmsg.y = roll;
50   anglesmsg.x = -pitch;
51
52   // this Vector is then published:
53   rpy_publisher.publish(anglesmsg);
54   ROS_INFO("published pitch=%.1f, roll=%.1f,  yaw=%.1f",
55             anglesmsg.x * 180 / 3.1415926, anglesmsg.y * 180 / 3.1415926,
56             anglesmsg.z * 180 / 3.1415926);
57 }
```

VISUAL INDICATOR OF VARIABLE TO BE ANNOTATED

What are the units for **anglesmsg.z** on line 48? QUESTION

SUGGESTION (Might not be correct): SUGGESTION

1. `kilogram-meter-per-second-squared (kg m s^-2)`

DROP-DOWN

Figure 4.2: A code artifact used in the study. This test question shows treatment T_3, an incorrect suggestion. All code artifacts used in this work are available at `https://doi.org/10.5281/zenodo.3247869` and in Appendix D.

text below the code artifact. Below the question text, the figure shows an incorrect suggestion (treatment type T_3). Other questions might have no suggestion (T_1), a correct suggestion (T_2), or multiple suggestions (T_4-T_6). Finally, at the bottom of the question is a drop-down box with several annotation options of physical unit types, discussed shortly below in § 4.2.4. We seek to approximate the annotation task described in §2.3 by asking developers to choose a type for a variable in the code artifact. We measure both the accuracy and time it takes for the developer to select an annotation.

Treatments. To address our research questions, we apply to each question one of six treatments, abbreviated as T_1 through T_6:

T_1**: No suggestion (control).** A question with the suggestion section not included. T_1 is intended to approximate the base task, examining a variable and its immediate context, and then determine which type from a type domain applies, if any.

T_2**: One correct suggestion.** A question with a correct suggestion immediately above the drop-down box, where the text of the suggestion exactly matches one option in the drop-down. The suggestion is accompanied by the caveat: *"SUGGESTION (Might not be correct)."* We include this caveat to encourage subjects to approach suggestions with skepticism.

T_3**: One incorrect suggestion.** This treatment is identical to T_2 except the suggestion is incorrect. The incorrect suggestion has the same caveat as in T_2 and matches one option in the drop-down box. This incorrect suggestion is chosen randomly from Table 4.1 (excluding the correct answer). Treatment T_3 is shown in Figure 4.2.

T_4**: Three suggestions, correct first.** A question with three suggestions immediately above the drop-down box. The suggestions are each on their own line and are enumerated 1, 2, 3. All suggestions exactly match one option in the drop-down.

T_5**: Three suggestions, correct not first.** This treatment is identical to T_4 except the second or third option exactly matches an option in the drop down. We randomly placed the correct option either second or third.

T_6**: Three suggestions, none correct.** This treatment is identical to T_4 except that *none* of the three suggestions is correct.

Treatment T_1 answers both RQ_1 and RQ_2. T_1 answers RQ_1 by measuring a baseline accuracy and answers RQ_2 by measuring how long annotations take without a suggestion. Treatment T_1 is the control for RQ_3 and establishes a baseline accuracy and timing for our research question about the impact of suggestions. To address RQ_3, we compare the accuracy and timing of questions with treatment T_1 to those treatments with suggestions, T_2–T_6. Also addressing the portion of RQ_3 pertaining to multiple suggestions, we compare the accuracy and timing of questions with treatment T_2 (one correct suggestion) to T_4 (three suggestions, first correct), and we compare the accuracy and timing of questions with treatment T_3 (one incorrect suggestion) to the union of T_5 (three suggestions, correct not first) and T_6 (three suggestions, none correct). For the qualitative question, RQ_5, after every question we require subjects to provide an open-ended textual explanation of their reasons for choosing a type. We examine all the explanations utilizing Grounded Theory [122] to answer RQ_5. Our **independent variable** is the kind of suggestion, if any, and the **dependent variables** are response accuracy and duration.

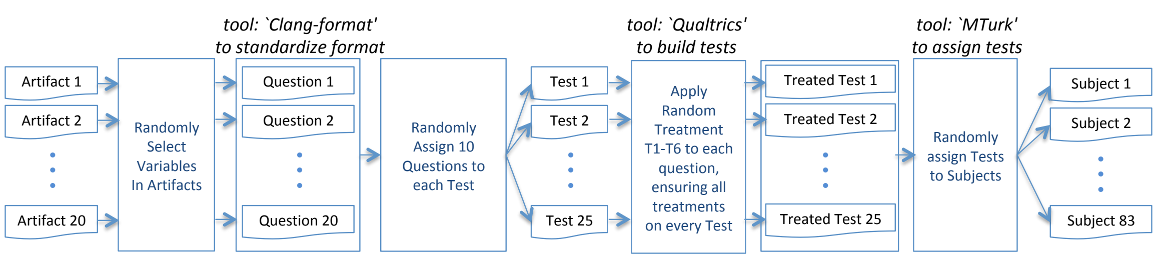


Figure 4.3: Experimental design showing how code artifacts become test instruments applied to subjects.

4.2.2.3 Experimental Design

As shown in Figure 4.3, our experimental design is *completely randomized* [125], helping to mitigate confounding effects. In this design, we randomly select a variable to annotate in each of 20 code artifacts, creating 20 questions. We randomly assign subsets of 10 questions from the 20 questions to create 25 tests, balancing the tests so that each question appears the same number of times. We then apply treatments T_1–T_6 randomly to questions in tests, ensuring that each test has at least one, and no more than three, of each treatment. We then apply tests randomly to subjects.

4.2.3 Subject Sample Population

We recruited subjects using Mechanical Turk (MTurk), an online marketplace for labor that is popular for many kinds of empirical research [126, 127] including software engineering [128, 129, 130]. MTurk subjects are appropriate for studies requiring neurological diversity [131, 132], meaning that we want to capture various ways of thinking about the task so that our results might generalize.

One caveat in collecting demographics on MTurk is that respondents have been shown to fabricate demographic answers to qualify [133] and receive compensation. Therefore, we clearly state during demographic questions that demographic answers will not be used to determine eligibility. We warn users to watch for random 'attention checks' [134], which are simple questions designed to have an obvious answer that could only be answered incorrectly by subjects who are not paying attention. We ask them to watch for attention checks, because the idea of attention checks, even without implementing them, has been shown to improve performance (we do not assess or enforce attention).

We pre-screen our subjects using recommended best practices that have been shown not to bias behavioral experiments [135]. The pre-screening has three requirements: 1) successfully complete at least 500 MTurk tasks, which means these subjects are amoung the most the most serious turkers; 2) have at least 90% accuracy on those tasks; and, 3) correctly complete our pretest with two annotation questions. We pay subjects $2.00 USD to complete the pretest and $10.00 USD to complete the main ten-question test. Paying subjects just on correct answers seems like an incentive to provide better answers, but we do not do this because it has been shown to be ineffective [136], meaning participants do not perform better when rewarded financially only for correct answers. We encourage subjects not to rush and to provide thoughtful explanations.

Table 4.2: Reported demographics for 83 Subjects.

YEARS EXPERIENCE	PROGRAMMING `C`, `C++`, `C#`, Java	EMBEDDED SYSTEMS, CYBER-PHYSICAL, ROBOTICS
< 1	22 (27%)	63 (76%)
$1 - 5$	44 (53%)	17 (20%)
$5+$	17 (20%)	3 (4%)

We ask three demographic questions during the pretest to try to better understand our subjects' previous experience and to see if these demographics correlate with performance. Table 4.2 shows a summary of the demographics for our 83 subjects. We ask about experience with (mostly) statically typed languages: *"How many years of programming experience in languages like `C`, `C++`, `C#`, `Java`?"* More than half our subjects (53%, 44/83) report 1–5 years experience with these languages. We then ask about embedded system programming: *"Years of experience programming embedded systems or robotic systems or cyber-physical systems (Things that move or sense)?"* Only 24% (20/83) of subjects report one or more years of experience with embedded systems. And thirdly, we inquire about previous experience with code annotation: *"Have you used any code annotation frameworks?"* If subjects report having previous annotation experience, we further ask them to indicate which frameworks they have used. Only 16% (13/83) of subjects indicate experience with annotation frameworks such as '`Resharper/Jetbrains`', '`JSR 308`', and '`SAL/MSDN`'. In § 4.3.1, we examine the impact of demographics on annotation accuracy.

4.2.4 Test Instrument Details

4.2.4.1 Type Annotations Options in the Drop-down Menu.

The contents of the drop-down menu include the 19 physical unit types listed in Table 4.1, plus `OTHER` and `NO UNITS`. We include `OTHER` to allow subjects to think beyond the options we have provided, and `NO UNITS` captures cases when the variable to be annotated does not belong in the type domain. The `OTHER` option is useful for less common types (i.e. *kilogram-meter-squared-per-second-cubed-per-ampere*, more commonly known as *voltage*, an answer to one of our questions that was correct only 33% (2/6) of the time). The `NO UNITS` option is important because the type annotation task first requires developers to identify whether a variable belongs in the domain before selecting a type. The order of the elements in the drop-down menu is randomized every time a subject sees a question. Randomizing the order helps mitigate the threat of response order bias [137].

4.2.4.2 Question Timing.

We instrument our web test to collect timing information for each question. Our test consists of alternating multiple choice and open-ended questions. In the multiple-choice question, subjects assign a type (if any) to a variable. Then in the open-ended questions, subjects explain why they selected that type. By tracking how long it takes for subjects to finalize their multiple-choice type assignment by clicking 'next', we can answer RQ_2. The time to provide an explanation is not included in our answers to RQ_2, and although we track it, it is not part of our results. We do not limit the time to answer individual questions and instead limit the total test duration to four hours.

4.2.4.3 Suggestions.

We provide suggestions (as shown in Figure 4.2) to answer RQ_3 and to assess the impact of future tools that might help developers make type annotations. All suggestions are drawn from the union of the units types in Table 4.1 along with `NO UNITS` and `OTHER`. The exact suggestion depends on the treatment. Please see § 4.2.2.3 for how suggestions are used for treatments. We randomize incorrect suggestions *per test*, so that each question and treatment receives an assortment of suggestions.

4.2.4.4 Explanations.

To answer RQ_5, we require subjects to provide textual explanations for why they chose a particular type. We record explanations because we want to better understand the sources of evidence and how that evidence is used. After subjects have finalized their type selection, we again show them the code artifact but with their answer and an open-ended text box. We notify subjects in the instructions that good explanations are required to successfully complete the test.

4.2.5 Utilized Tools

We use off-the-shelf tool in our experiments and analysis:

4.2.5.1 Phriky

Phriky [13] is a static analysis tool to detect physical unit type inconsistencies in ROS `C++` code. As shown in Figure 4.1, we only use Phriky to select the pool of `C++` files that use use as artifacts. We identify files with physical units by invoking Phriky with the `--only-find-files-with-units` command line parameter.

4.2.5.2 Clang-format

`Clang-format` [138] is a tool to format `C++` code in a standard way. As shown in Figure 4.3, we use Clang-format to ensure that code artifacts shown to subjects are formatted clearly and uniformly.

4.2.5.3 Qualtrics

`Qualtrics` [139] is a web-based survey tool. As shown in Figure 4.3, we build our test instrument using Qualtrics and use several of its features, such as: 1) tracking the time required by subjects to assign annotations; 2) ensuring that all questions are answered; 3) randomizing the question order by subject; 4) randomizing the order of options in the drop-down box for every question; 5) preventing the same IP address from taking the test; 6) recording subject's responses; and, 7) creating unique IDs used to pay subjects. As shown in Figure 4.5, we use Qualtric's API to immediately notify us when a subject passes the pretest so we can evaluate their explanations and grant them access to the main test. We grant access using Mechanical Turk.

4.2.5.4 Mechanical Turk

Mechanical Turk [9] (MTurk) is a marketplace for online labor. We use MTurk to recruit and pay subjects for both the pretest and main test, and retain only anonymized identifiers for remuneration as required by our IRB (# 20170817412EX, shown in Appendix B). We use MTurk to control access to our tests using MTurk's 'Qualification' mechanism where we can designate subjects as having passed the pretest as a necessary prerequisite to see the 'main test' task.

4.2.5.5 MySQL

We use a relational database MySQL to organize and track tests, questions, suggestions, demographics, and explanations. Our database schema is shown in Appendix G. We use `MySQL` to store data, but to analyze it we use the `R` language.

4.2.5.6 RStudio

RStudio [140] is a statistical analysis tool that we use `RStudio` [141] to analyze data. We utilized standard packages such as `nnet` for our binomial log-linear response model [142](`multinom`), the `binom` package [143] for confidence internals, and the `aov` function to perform ANOVA on questions about timing.

4.2.6 Study Phases

We conducted our study in April of 2018, and conducted a follow-on study in September of 2018.

Our study has three phases: a test evaluation phase, a main test deployment phase, and a main test follow-on phase.

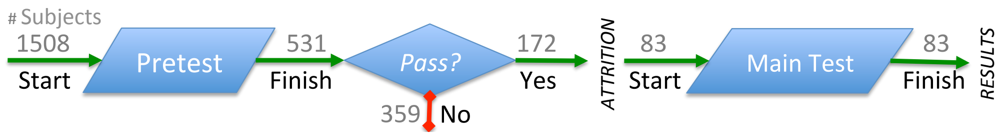


Figure 4.4: Number of Subjects at each point during Phases Two and Three combined.

4.2.6.1 Phase One: Evaluation and Refinement of the Test Instrument.

During the evaluation and refinement phase, we deploy an initial version of the test to 27 subjects. This initial version has no suggestions (Treatment T_1). The purpose of this evaluation is to make sure the questions can be answered correctly by some subjects, are not trivial, and to identify areas where our instructions were unclear. We made several iterative improvements to our test instrument based on this initial evaluation: 1) identified two trivial questions and replaced them with more difficult ones; 2) added text to qualify that suggestions *"Might not be correct"*; 3) added to demographic questions that answers would not be used to screen participants by adding the text *"NOT GRADED OR SCORED,"* in accordance with MTurk best practices [133]; 4) visually identified the variable to be annotated using colorblind-safe yellow markers as shown in Figure 4.2; 5) ensured that the question order was randomized per subject; 6) modified the test so that every annotation question was followed by a required, open-ended question about why developers made the annotations they did. The data collected during the evaluation phase is used only for evaluation and refinement, and all 27 evaluation test subjects were excluded from the deployment phase of the experiment.

4.2.6.2 Phase Two: Deployment of Pretest & Main Test.

Subjects must pass a pretest and provide good explanations to qualify for our experiment. The pretest serves several purposes. Firstly, it ensures that every subject has some chance to complete the annotation task in the main test, and that the explanations will be coherent. Two pretest subjects were excluded from the main test for providing useless explanations such as 'asdf' or 'nope' even though they correctly identified the physical unit type. Secondly, the pretest is a kind of tutorial and includes two practice questions to familiarize subjects with the mechanics of the web test instrument.

4.2.6.3 Phase Three: Follow-On Survey.

As we analyzed the results from the previous phases, we realized that RQ_4 would be a natural evolution of our work, and that we had not collected the required data. Therefore we collected these additional responses to measure the impact of three suggestions. This phase is identical to Phase Two except that more questions had treatments T_4–T_6.

Figure 4.4 shows the number of subjects in Phases Two and Three combined. As shown in Figure 4.4, 1508 subjects started the pretest, but only 531 finished it, indicating that many subjects opted out of the task. Of those that finished the pretest, 32.4% of subjects (172/531) passed the pretest. For the pretest, we gave subjects 30 minutes. As shown in Figure 4.5, after subjects complete the pretest, we review the answers and explanations within 15 minutes and enabled subjects to then immediately take the main test. We found that immediately qualifying passing subjects for the main test noticeably reduced attrition. After passing the pretest, subjects could begin the main test anytime within the next 36 hours, and had to complete the main test within four hours once started. During Phase Two and Three, we received 833 responses to the main test.

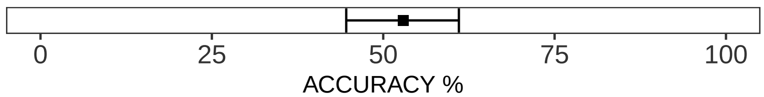


Figure 4.6: Manual annotation accuracy for control treatment T_1.

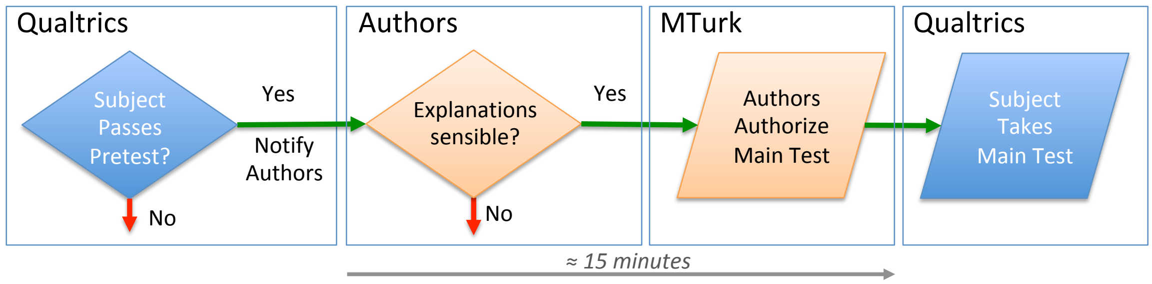


Figure 4.5: Pretest review and approval process showing Qualtrics, Authors, and MTurk.

4.3 Results

This section presents the results of our research questions presented in § 4.2.1. We describe results for accuracy and time, then discuss the impact of suggestions, and finish with qualitative results for why developers make the annotations they do.

4.3.1 Accuracy

Treatment T_1 is the control for our experiments. In T_1, subjects performed the annotation task without suggestions. As shown in Figure 4.6, the average accuracy for assigning unit types to identifiers is 51% (71/138), ±8.5% (Agresti-Coull) [144]. Our results strongly support the commonly-held opinion [145, 119] that the annotation task is difficult without assistance.

4.3.1.1 Subjects' Demographics Have Small Impact on Accuracy

We asked subjects about their previous experience with programming languages, embedded systems, and annotation frameworks, as discussed in § 4.2.3. Subjects with 5+ years of experience with programming languages (`C`, `C++`, `C#`, `Java`, 17 subjects) had a slightly higher accuracy of 56% vs 50% for both of the other groups, but without significance ($p = 0.554$). Subjects who reported *the least* experience with embedded systems ($N = 53$) had a slightly higher accuracy of 53% compared to 45% for subjects with 1–5 years experience, but again without significance ($p = 0.829$).

RQ$_1$ Results: Manually assigning type annotations is error-prone (51% accurate, ±8.5%).
Implication: If we rely on manual annotation alone, then type checking of physical units will be worthless for many developers.

4.3.2 Timing

Using the accuracy of responses with the control treatment T_1, we group questions into three groups by difficulty. The groups are Easy 100% − 75% correct, Medium 75% − 25%, and Hard 25% − 0%. We grouped questions this way to explore how difficulty correlates to other aspects, like timing and the impact of suggestions. Detailed accuracy and timing results for questions arranged by difficulty and treatment are shown in Appendix A, Table 9.1, including the response accuracy (percentage and fraction) and timing (mean and median). The table shows results by question, and the code artifacts corresponding to each question are available at `https://doi.org/10.5281/zenodo.3247869` or in Appendix D.

Figure 4.7 shows the time to make a single correct type annotation, with some outliers capped. Our timing data contains outliers, perhaps because we allowed subjects four hours to complete the test and administered the test via the web and therefore could not observe how subjects spent their time. To address the furthest outliers, we use Tukey's interquartile 'gate' range method [146]. This method specifies that values beyond $k = 3$ times the interquartile range plus the third quartile ($Q3$) are outliers, but we use an even more conservative $k = 6$. Overall, we capped two long duration responses (2/138) greater than 961 s ($961 = Q3 + k(Q3 - Q1)$) to 529 s, the sample mean's 95% value.

As shown in Figure 4.7, making a single correct annotation takes 136.0 s (median=108.6 s). Incorrect annotations take longer than correct annotations but without significance ($p = 0.184$). Overall, subjects took approximately two minutes for a single annotation for both correct and incorrect answers. The figure shows that question difficulty appears to have the largest impact on timing for Easy and Hard questions. Correct answers to Easy questions took an average of 112.3 s whereas Hard questions took 219.7 s, but it should be noted that there were few correct answers to Hard questions, so we measure no significance between the timing for Easy and Hard correct answers ($p = 0.282$).

If we extrapolate from a single correct annotation taking approximately two minutes, then the 20 randomly selected files that contained our artifacts would take 62 hours to annotate (1645 variables counted using Cppcheck version 1.80 [147]). The smallest program in these 20 would take less than a half hour (11 variables), and the largest would take almost eight hours (1,645 variables).

The time required to make a correct annotation measured by our experiment does not include the additional time required to determine what variables do not belong to the type domain, troubleshoot incorrect annotations, or maintain type annotations during evolution.

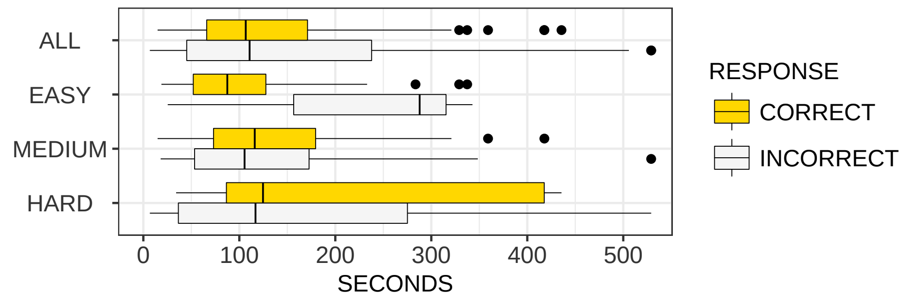


Figure 4.7: The quantity of time required for a single annotation question under treatment T_1 (No Suggestion, the control), grouped by question difficulty and correctness.

TREATMENT (SUGGESTION)	% CORRECT	RESPONSES	SUBJECTS	RISK RATIO CONFIDENCE INTERVAL
T_1 No Suggestion (control)	51%	140	72	
T_2 One Correct	73%	139	69	
T_3 One Incorrect	28%	142	58	
T_4 Three, 1st Correct	66%	136	66	
T_5 Three, Correct 2nd or 3rd	58%	146	69	
T_6 Three, None Correct	47%	141	69	

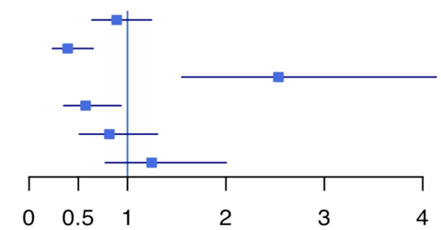


Table 4.3: Annotation accuracy and 'Risk Ratio' by question treatment. The Risk Ratio shows a 95% log-linear confidence interval for how likely subjects are to make an incorrect type annotation. A value of 1 means the subject has a 50% chance of making an incorrect annotation.

RQ_2 Timing Results: The type annotation task is time-intensive (mean=136.0 s, median=108.6 s for a single variable).

Implication: Applying type annotations is time-intensive.

4.3.3 Impact of Suggestions on Accuracy

Our results for accuracy are based on responses to a multiple choice question where the answer can either be correct or incorrect, a binomial outcome. Because we want to measure how treatments (suggestions) impact accuracy, we need a mathematical model to quantify the impact. We use a binomial log-linear response model [142]. The impact of suggestions results in a type annotation that is either *correct* $= 1$ or *incorrect* $= 0$. The model outputs a 'Risk Ratio' interval that quantifies the likelihood of choosing an incorrect type annotation because of the treatment applied to a question.

4.3.3.1 RQ_3 Results: Impact of a Single Suggestion on Accuracy

Table 4.3 shows the risk ratios for suggestions. A risk ratio >1 in our study means an increased risk of assigning an incorrect type. The impact of suggestions varies significantly by treatment. A single correct suggestion (T_2) increases accuracy with significance compared to no suggestions (T_1) ($p < 0.05$). For a single correct suggestion, the risk of annotating incorrectly is reduced on average by a factor of 0.4, meaning an increased accuracy of 73% compared to 51% with no suggestion. The impact of a single correct suggestion is significant when compared to the control of no suggestions (see Table 4.4).

A single incorrect suggestion (T_3) increases the risk of making an incorrect suggestion with significance compared to no suggestion (T_1). Treatments T_2 and T_3 are also different from each other with significance as shown in Table 4.4. When subjects were asked to annotate a variable in the presence of a single incorrect suggestion (treatment T_3), 30% of incorrect responses (30/98) 'took the bait' and answered with the provided incorrect suggestion.

	T_1	T_2	T_3	T_4	T_5	T_6
T_1		0.001	0.001	0.05	-	-
T_2			0.0001	-	0.01	0.0001
T_3				0.0001	0.0001	0.01
T_4					-	0.01
T_5						-
T_6						

Table 4.4: Pairwise comparison of p-values of binomial Z tests between treatments. We only show p-values where $p <= 0.05$, our threshold for significance.

RQ_3 Results: Suggestions have a significant impact on developer type annotation accuracy—a positive impact when the suggestion is correct, and a negative impact when incorrect.
Implication: Automated tools that can suggest type annotations could be significantly helpful to developers.

4.3.3.2 RQ_4 Results: Impact of Three Suggestions on Accuracy

To answer RQ_4 and to better understand the difference between making a single suggestion with making multiple suggestions, we examined the impact of three suggestions with treatments T_4, T_5, and T_6 (See § 4.2.2.3 for details on treatments).

Three suggestions, first correct (T_4) is significantly more accurate than no suggestions (T_1), and when compared to a single correct suggestion we found no statistically significant difference (see Table 4.4 for p-value comparisons). Three suggestion with the correct suggestion 2nd or 3rd (T_5) is not as helpful as a single correct suggestion (T_2), but T_5 does not appear to harm accuracy since T_1 and T_5 are not significantly different.

Treatment T_6, three incorrect suggestions, is not as helpful as T_2 or T_4 (correct suggestions first), but is also not particularly harmful when compared to T_1. Further, we measure T_6 as causing significantly less detriment to accuracy than T_3, a single incorrect suggestion. Where there were three incorrect suggestions (treatment T_6), they only took the bait 12% of the time (17/147), and for three suggestions, correct 2nd or 3rd (treatment T_5), subjects' responses matched a suggestion 7% of the time (10/141). T_5 and T_6 are not significantly different than the control, meaning that showing three suggestions with the first correct can help and showing three with none correct is not measurably different from showing no suggestions.

RQ$_3$ Accuracy Results: Suggestions have a significant impact on type annotation accuracy. Three suggestions (correct first) improves accuracy nearly as much as a single correct suggestion.
Implication: Automated type annotation tools should show multiple suggestions, since multiple suggestions help nearly as much as a single suggestion when correct and three suggestions hurt significantly less when incorrect.

4.3.3.3 Accuracy by Question Difficulty

Figure 4.8 shows the range of accuracy for all treatments by question difficulty. Correct suggestions (T_2) benefit all questions compared to T_1, with similar improvements for HARD (+33%) and MEDIUM (+26%) questions, while only helping EASY questions by +7%. As shown in the figure, an incorrect suggestion T_3 reduces accuracy for EASY (−53%) and MEDIUM (−49%) questions with little impact on HARD questions. This makes sense because subjects who could correctly deter-

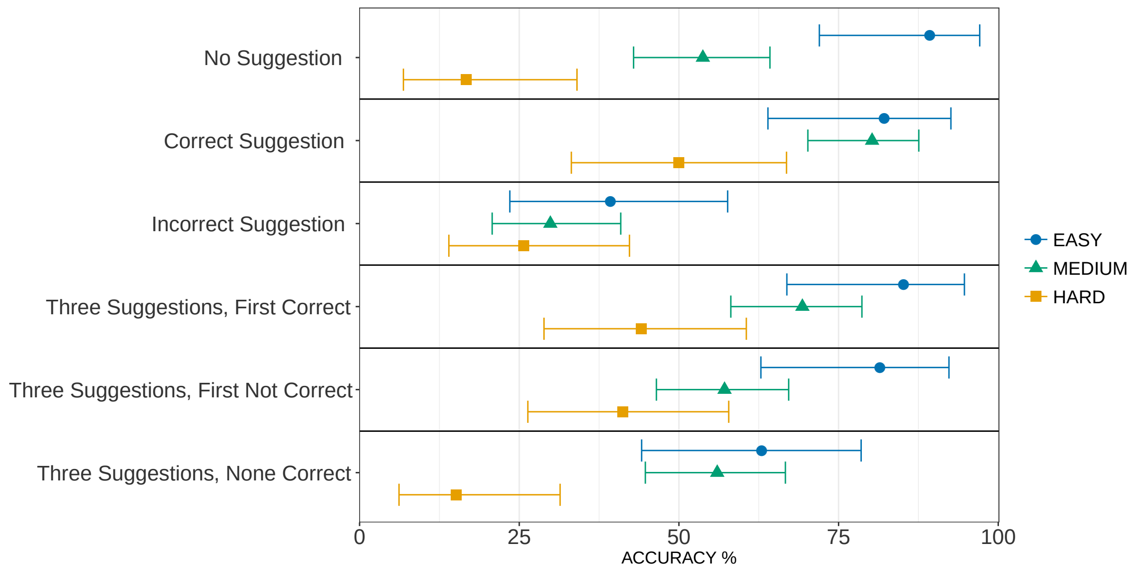


Figure 4.8: Annotation accuracy per treatment and question difficulty. The intervals indicate 95% confidence levels.

mine the correct type annotation for a HARD question already had evidence that eliminated the incorrect suggestion. A single correct suggestion (T_2) has a similar distribution of accuracy to three suggestions, first correct (T_4). Therefore, showing three suggestions is nearly as helpful as showing a single suggestion, as can also be seen in the risk ratios in Table 4.3. Notice that three suggestions, none correct (T_6), causes less harm than a single incorrect suggestion (T_3) for EASY and MEDIUM difficulty questions.

4.3.3.4 Analysis of Incorrect Answers

For incorrect answers, the most common mistake overall was `NO UNITS`, accounting for 31% (115/376) of incorrect answers. This means that the subject believes that the variable in question does not belong to the type domain of physical units. The next most common incorrect answer was *meters* at 10% (37/376), followed by *other* at 9% (35/376).

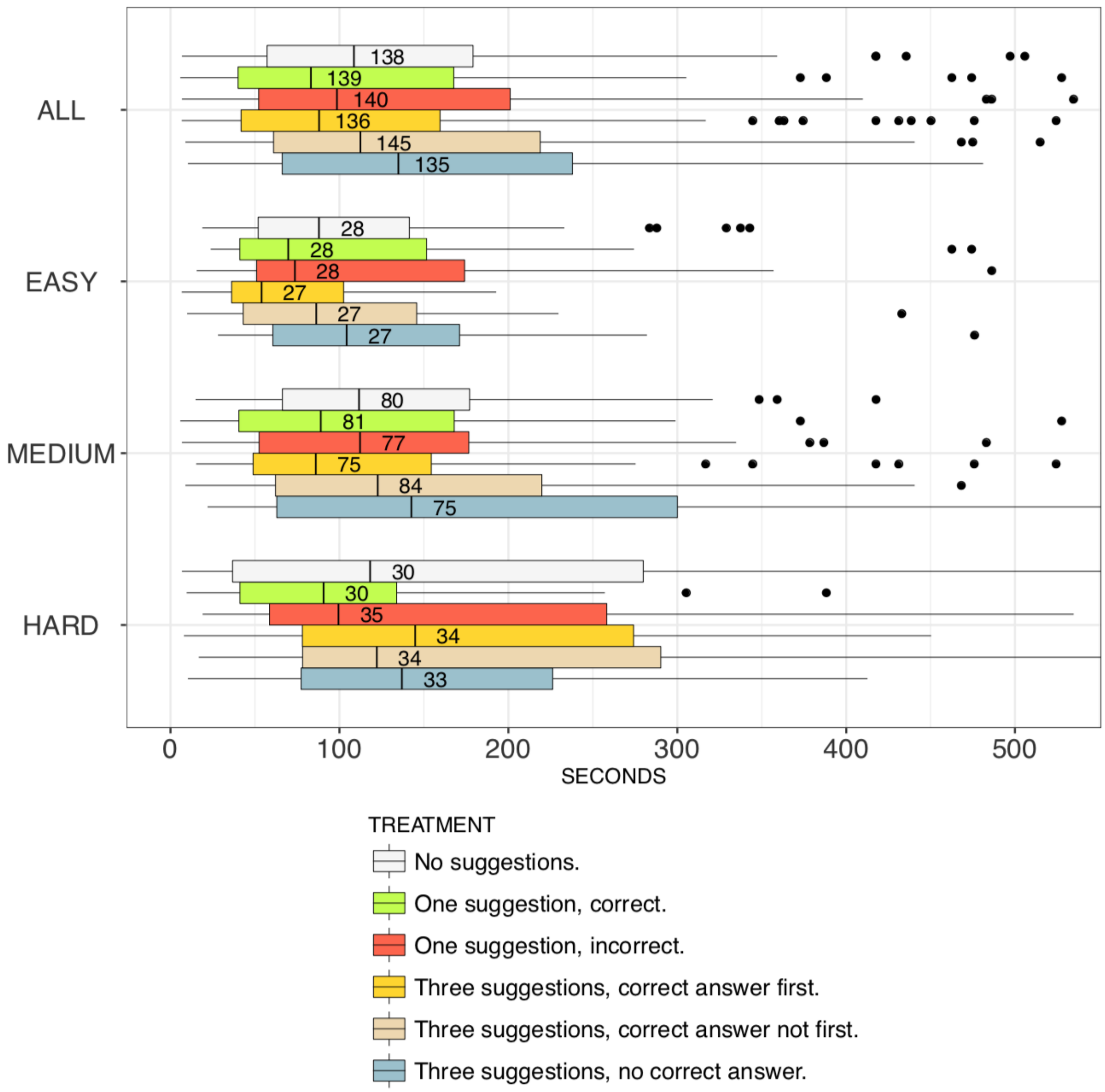


Figure 4.9: Time required to provide a single correct annotation, broken down by difficulty and treatment. The number inside each box indicates the observation count.

4.3.4 Impact of Suggestions on Timing

Figure 4.9 shows how suggestions impact the duration required to provide correct annotations. We examine correct annotations because we want to determine if providing suggestions significantly delays developer's ability to annotate correctly. As shown in the Figure, the annotation accuracy is grouped by question difficulty along with the category All. For All, correct annotations are speediest in treatment T_2 (mean=126.1 s), compared to 33% longer with T_3 (incorrect suggestions, mean=168.5 s)) and 8% longer with T_1 (no suggestion, mean=136.0 s). The difference between the time between T_2 and T_3 is not significant ($p = 0.220$). The slowest distribution of responses comes for treatment T_6 (Three suggestions, no correct answer). This might be because evaluating the incorrect suggestions requires extra time. Overall, the time differences for all treatments lack significance, meaning suggestions do not incur a time penalty.

Correct suggestions (T_2, T_4) have little impact on the timing of Easy questions. This small impact intuitively makes sense since Easy questions are aided less by a correct suggestion. A single correct suggestion tends to reduce the time required for Hard questions, as shown in Figure 4.9, although this difference lacks statistical significance because our dataset contains few (5) correct answers to Hard questions (T_1) (see Appendix A for details). Incorrect suggestions (T_3) as well as three suggestions (T_4–T_6) increase the tendency toward longer annotation times for both Medium and Hard questions, but without significance.

RQ$_3$ Impact of Suggestions on Timing: Suggestions do not impact developers' time to make correct annotations.
Implication: An annotation tool that provides suggestions would not significantly increase the time required to make correct annotations.

4.3.5 Qualitative Results: Clues for Choosing a Type

GROUNDED THEORY	CORRECT RESPONSES						INCORRECT RESPONSES						TOTAL	
EXPLANATION CATEGORY	T_1	T_2	T_3	T_4	T_5	T_6	T_1	T_2	T_3	T_4	T_5	T_6	#	%
Names only	36	54	17	40	36	27	35	20	44	23	21	20	373	48%
Math reasoning and names	20	24	18	26	32	23	5	4	12	9	10	19	202	26%
Not in type domain	4	10	1	7	8	6	19	13	25	4	18	18	133	17%
Code comments	11	9	2	5	6	5	3	-	-	-	2	-	43	5%
Used suggestion	-	5	-	2	1	-	-	-	12	-	1	-	21	3%
Type depends on input	-	-	-	-	-	-	5	2	2	-	1	1	11	1%

Table 4.5: Summary of type annotation explanations for 783 answers.

Annotation explanations are *qualitative*, unlike the results from § 4.3.1–4.3.4 that are *quantitative*. To better understand how and why developers choose a type annotation, we explore the explanations subjects provided after each annotation using a Grounded Theory [122] approach. In Grounded Theory, the goal is to categorize all of the elements into distinct groups where each group has a 'label'. In this approach, rather than start with pre-defined labels, we instead examine each explanation successively in random order and assign and create labels simultaneously so that categories emerge from the data organically. The process is iterative, during the first iteration we identified 12 categories. With each successive iteration we merged labels until, after three iterations, we converged to six labels as shown in Table 4.5. Explanations might receive multiple labels, such as a variable name reinforced by a comment. The table shows the labels we identified broken down by treatment and whether the annotation was correct.

The most common explanation subjects gave was 'Names only', providing almost half (48%) of all responses. The importance of high-quality identifiers is well-supported [47], and our results confirm that identifiers are a significant factor in how subjects make the semantic connection between variables and their types, for example:

> Q_{13}: *The name of the left part of the expression is msg.linear_acceleration.z. I trust the [person] who coded this and thus I think that this would be in units of linear acceleration (meters per second squared).*

> Q_{17}: *At least I hope 'torque' is referring to torque.*

Note that 'Names only' is also the most common explanation for incorrect type annotations, indicating that the clues in variable names can be misleading, confusing, or insufficient. Our results regarding the importance of variable names should be taken with a grain of salt, however, because every code artifact in our study had some identifier, but not all artifacts had comments or mathematical reasoning.

The second most common explanation is 'Math reasoning and names,' accounting for 26% of explanations, such as:

> *Q_4: vx * cos(th) - vy * sin(th) will give a quantity in m / s. Since dt is a quantity in seconds, multiplying by that will yield meters.*

As shown in Table 4.5, subjects providing incorrect answers are less likely to identify 'Math reasoning and names' as their reason for choosing a type. However, some subjects cite math reasoning and then bungle the maths:

> *Q_4: Meters per second times dt would cause the seconds to cancel out and the meters to square*

Where *"cause...the meters to square"* is not correct.

'Not in type domain' is the third most common explanation provided for choosing a type (17% overall) but 73% (97/133) of these responses are incorrect. It appears that subjects were unable to see how the variable in question belonged to the type domain. This raises an important question in the overall process of assigning types: which variables should be typed? It appears that future tool developers could aide the type annotation task simply by helping find these variables.

The fourth most commonly cited reason is 'Code comments' (5%), with comments much more likely to be cited with correct answers ($N = 36$) than incorrect answers ($N = 5$). Note that only 2/20 of our code artifacts contained comments

(Q_6 and Q_8 in Appendix D). This might indicate that although code comments are effective in providing a clue to the correct type annotation, the overall lack of comments limits this as a factor.

Only 3% of explanations (21/833) explicitly say they took the provided suggestion. However, this is likely only a lower bound because 63/463 (14%) of incorrect responses matched an incorrect suggestion, and subjects might not have admitted to using the suggestion.

Qualitative Results: The main clues for type selection are variable names and reasoning over code operations, and names together with math operations are more likely used in correct type annotations.
Implication: An automated method to suggested type annotations should leverage multiple sources of evidence.

4.4 Threats

In this section we discuss both the external threats and internal threats. External threats related to how this study might generalize, and internal threats are factors about how we conducted the study that might bias our results.

4.4.1 External Threats

4.4.1.1 Subjects Not Representative of Developers.

Our subjects are recruited from Mechanical Turk and might not represent developers more generally, even if MTurk is appropriate for research seeking neurological diversity [131, 132]. We mitigate this threat by requiring subjects to correctly annotate two code artifacts during the pretest and provide good explanations for choosing a type. To answer the pretest questions correctly, subjects must comprehend code, understand the annotation task, and correctly identify the physical unit type. However, we asked subjects to annotate code they did not write, and second, subjects were not trained on this task. Applying type annotations to someone else's previously existing code can happen when developers seek to improve overall code quality by gradually evolving untyped code into typed code [148]. Since subjects were not specifically trained to annotate physical unit types, our results likely underestimate the true accuracy of trained developers. Training subjects to perform this task could improve accuracy, but we wanted to establish that the basic annotation task is not trivially easy.

4.4.1.2 Fidelity of the Annotation Task.

We concretize this work in the domain of physical unit types as described in § 2.1– 2.1, and this type domain might not generalize to the type annotation task more generally vary greatly in complexity. We note that no matter the type domain, developers must still reason about interactions in the type system and how code operations impact types in the type domain. Additionally, we deliver the annotation task through a web-based test, which might not represent how developers apply annotations in an IDE. Also, our work measures how non-authors make type annotations, likely underestimating accuracy for authors. We observe that our measurements apply to the case where the pre-existing, non-typed code is gradually evolved towards greater type safety to increase reliability. To better account for this difference in future work, we could observe code authors.

4.4.1.3 Generality of Code Artifacts.

The code artifacts we selected might not generalize to all code that needs type annotation. We mitigate this threat by randomly selecting code artifacts from a corpus of $31,928$ files. Moreover, all our code artifacts are strongly-typed (`C++`), although applying type annotations to non-strongly-typed languages also involves reasoning about the type domain and how code operations impact types. We limit the scope of analysis to functions, and the time and accuracy might differ for larger scopes.

4.4.2 Internal Threats

4.4.2.1 MTurk Used to Recruit Subjects.

Research about using MTurk for scientific studies [133] indicates that subjects falsify demographic data to participate and get paid. We sought to mitigate this effect by repeatedly indicating that answers to demographic questions, including questions about experience, would not impact eligibility for participation, and that questions are "NOT GRADED OR SCORED." Additionally, we screened subjects on their ability to complete previous tasks provided a financial incentive ($2.00 USD pretest, $10.00 USD main test) hopefully sufficient for them to undertake the task with seriousness.

4.4.2.2 Bias from Code Context.

The code artifacts we show to subjects are limited to a function, whereas additional context might be helpful in determining the correct type. We mitigate this threat by testing our questions during an evaluation phase (see § 4.2.6) to ensure that it is possible to infer the correct units with the available information.

4.4.2.3 Classification of Questions into Easy/Medium/Hard.

We organize our results by three levels of question difficulty Easy $100 - 75\%$ correct, Medium $75 - 25\%$, Hard $25 - 0\%$. Using these three levels as defined might distort how we present results, because our question difficulty levels are extrinsic in practice but we use an internal measure to define them. We mitigate this threat by exploring several groupings (two through five groups, using $66 - 33\%$ for Medium) and finding they all exhibit similar facets of the same underlying contours. We settled on these three groupings because they were the simplest grouping that retains how question difficulty impacts accuracy and timing across treatments.

4.4.2.4 Format of the Test Instrument.

We frame the type annotation task in a question and answer format, and this format loses some of the context in which developers make type annotations, especially as a developer gradually annotates code in a single file or project and becomes more familiar with it. Further, we truncate 10 of 20 artifacts because they were much longer than what would fit on a standard desktop or laptop display. We mitigated the impact of this threat by verifying that the question could be correctly answered using the available information during the test phase one (described in § 4.2.6). This format might have unforeseen impacts on subjects. We help mitigate this threat by refining the test with visual markers.

4.4.2.5 Ordering of Suggestions by 'Goodness'.

We did not study the impact of sorting the suggestions by 'goodness' or 'closeness', and instead randomized the order of suggestions when possible. A real suggestion mechanism will likely order suggestions by some metric rather than showing suggestions in a random order. We mitigate this by observing that suggestion mechanisms likely will not achieve an ideal suggestion ordering, especially not at first. Also, our experiments seek to establish baseline measurements (under approximations) as a starting point for future refinement.

4.4.2.6 Common Names for Physical Unit Types.

Some of the physical unit types in our study have common names, such as the type *kilogram-meters-per-second-squared* being more commonly known as *force*. In our study, we use the fully-explicit, long form of the name. To help mitigate the case when subjects do not connect the full name with the common name, we examine every explanation and when subjects indicate the common name in the explanation and select `OTHER` as an answer, we consider the answer to be correct. Overall, we deemed 7/414 incorrect answers as correct because of an explanation that correctly identifies the common name.

4.4.2.7 Duration of Test.

We allow four hours for subjects to complete the main test. During this time window, subjects might take breaks or perform other tasks. Since our test instrument is web-based and remotely administered, we cannot distinguish between long interludes spent thinking about questions from time spent in other activities. This long duration might give rise to 'ceiling effects,' with longer overall time estimates to complete tasks. We mitigate this threat by identifying and capping some timing outliers (described in § 4.3.2). Its also important to note that our timing only captures the time to choose an annotation, whereas we are aware that a developer might spend additional time troubleshooting incorrect annotations.

4.4.3 Conclusion Threats

4.4.3.1 Statistical Significance.

Some of our hypotheses that exhibit clear trends lack statistical significance ($p > 0.05$) because we do not have enough responses in some categories. For example, the timing for Hard questions indicates that they take longer to answer correctly, but we cannot conclude that this has significance because there are few correct answers to Hard questions. Additionally, when we segment the data by demographics (§ 4.2.3), the samples in some segments are too small to be representative. We could address these kinds of limitations in the future by deploying more tests and actively monitoring results during testing, to help balance the response distributions by reassigning questions to subjects.

4.5 Discussion: Code Attributes' Impact on Annotation Accuracy

In this section, we identify and discuss several code attributes that might impact developers ability to make a correct type annotation. We then perform a data analysis of how each attribute impacts accuracy and present the results.

4.5.1 Code Attributes

We examine several code attributes to determine if these features could make type annotations more difficult. We identified five attributes. The first three are binomial (either True or False) and the last two are real-valued, discrete quantities. We now discuss each in more detail.

4.5.1.1 'Has Good Identifiers'

For this code attribute, we examined each identifier transitively connected to the variable to be annotated. For each of these identifiers, we determined whether the identifiers speak to the type domain or contain substrings that contribute a useful clue to the type. Based on the presence of these semantic clues, we designate the question as having 'good identifiers.' For example, the code artifact in Figure 4.2 is designated 'good' because `anglemsg` contains the substring 'angle' and the right-hand-side `yaw` is semantically connected to angles. These clues together narrow the search even without considering code operations. Conversely, some identifiers contain scant semantic meaning, such as `x2`, `pt`, `k` and `av`. Others like `tmp_point_out.point.x` have semantic meaning but are misleading in the artifact context (Q_{10}) which is about *force*. We found 'Good identifiers' in 13/20 questions (3, 4, 5, 6, 8, 9, 11, 12, 14, 16, 17, 18, 19, in Appendix D).

4.5.1.2 'Is Truncated'

For this attribute, we noted whether the artifact shown to subjects in the main test (§ 4.2.6) is truncated. We truncated some artifacts because they were significantly longer than what could fit on a typical desktop screen ($\approx$768 pixels) of our web-based survey instrument. At most we showed 36 LOC. Artifacts are truncated in 10 of 20 questions (4, 7, 9, 11, 13, 14, 15, 17, 18, 20, in Appendix D).

4.5.1.3 'Requires Multiline Reasoning'

For this attribute, we noted whether the operations impacting the type of the variable to be annotated are contained within a single line. The artifact shown in Figure 4.2 requires multi-line reasoning because only by considering the code operations on lines 55 and 56 can the type on line 48 be determined to be *radians* and not *degrees_360*. Multiline reasoning is required for 9/20 questions (1, 3, 4, 6, 7, 10, 15, 19, 20, in Appendix D).

4.5.1.4 LOC in Artifact

For this attribute, we count the number of non-blank, non-comment lines of code in each artifact using CLOC [149], a code lines counting tool. Unlike the previous code attributes, this attribute is not binomial but real-valued. The 20 artifacts have a 19.6 LOC on average ($\sigma = 10.0$).

4.5.1.5 Number of Variables Involved with Type Annotation

For this attribute, we compute the number of variables that participate in the reasoning by starting with the variable to be annotated and counting all variables in the backwards data dependence slice within the code artifact. We do this because the number of variables that are transitively involved might correlate with complexity of the reasoning, and the more ways that reasoning can go wrong. The 20 artifacts have an average of 4.8 variables involved ($\sigma = 4.3$).

4.5.2 Results of Code Attributes' Impact

Figure 4.10 shows how some binomial (True or False) code attributes impact accuracy for responses to questions without suggestions (T_1). We conducted this analysis *a posteriori* to explore how these code attributes impact accuracy. As shown in Figure 4.10, of the code attributes we examined, only 'Has Good Identifiers' appears to have an effect on accuracy but with only 90% confidence ($p = 0.08$)[1], yet again emphasizing the value of high-quality identifiers. Likewise, truncated

[1] In all other parts of this work, our significance threshold is $p < 0.05$, denoting 95% confidence.

artifacts do not have a significant impact ($p = 0.18$) but this might be worth further consideration in future studies that measure whether withholding the surrounding context negatively impacts accuracy. We did not measure a significant impact for 'Requires Multiline Reasoning' ($p = 0.26$), which we found surprising because of anecdotal experiences with particularly challenging type annotations that required multi-line reasoning. We save further refinement of the role of context in type annotation accuracy for future work.

Figure 4.11 shows the negligible impact of increasing lines of code in the code artifact. However, as shown in Figure 4.12, the number of variable involved shows negative correlation with accuracy, indicating the difficulty developers face when annotating a variable that depends on the interplay of several elements of the type domain. Note that we only have 20 code samples and that having a small number of instances with larger variables might bias the correlation. Further note that if we remove the observations with the largest numbers of variables then the correlation is close to zero. As shown in the figure, the more variables involved in an annotation, the more difficult it is to assign a type annotation correctly. This might be a way to rank variables needing annotation by difficulty.

All of these code attributes very likely require further, larger-scale studies to definitively characterize their impact on accuracy.

Summary

In this chapter we presented work that, to our knowledge, is the first to quantify the type annotation burden. This work contributes to the limited empirical evidence in the literature about code annotation more generally. We analyzed code attributes of the artifacts and provided new empirical evidence of the benefits of high-quality

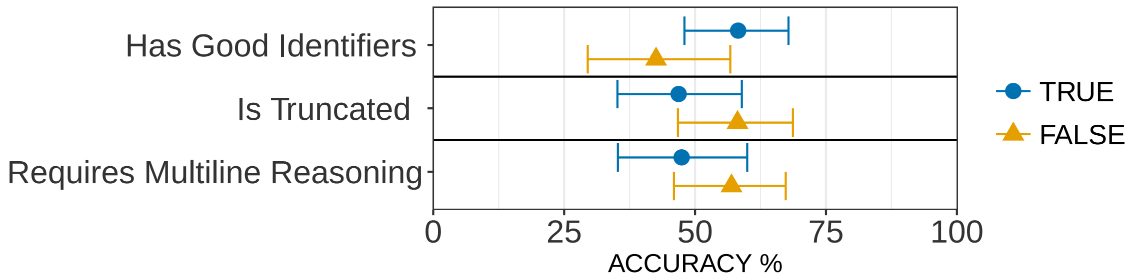


Figure 4.10: Code attributes impact on annotation accuracy for questions without suggestions (T_1).

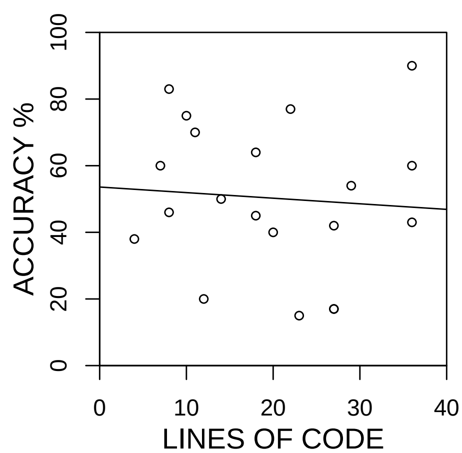


Figure 4.11: Accuracy by lines of code in the artifacts.

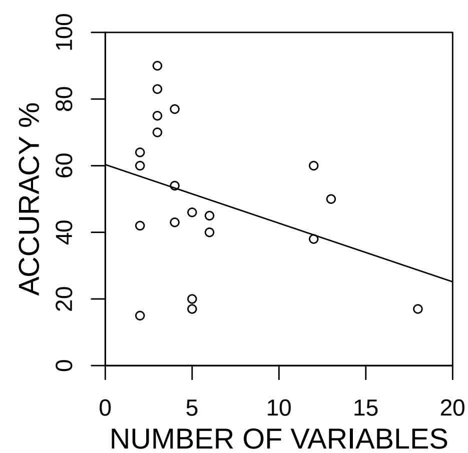


Figure 4.12: Accuracy by the number of variables involved.

identifiers to annotation type accuracy. We also examined code attributes such as 'requires multi-line reasoning', the size of the code artifact in LOC, and the number of variables involved.

Our work strongly supports that type annotations are difficult for developers to assign correctly and that correct suggestions significantly improve accuracy. herefore, the next chapter examines a method to help developers detect dimensional inconsistencies without developer annotations.

5 Method to Infer Types without Developer Annotations

Since dimensional inconsistencies are a real hazard to robot software, but type annotations are difficult for developers, we develop an approach to automatically infer types for some program variables and detect dimensional inconsistencies without developer annotations.

In this chapter, we identify the challenges in automatically inferring physical unit types, propose a method that exploits an architectural feature of robot message-passing middleware, describe our implementation of our method in a tool Phriky, evaluate Phriky on a corpus of open-source robot software, and identify threats and limitations of this approach.

The work presented in this section was previously published in [12].

5.1 Challenges

The challenge of this approach is to find a source of physical unit type information while imposing neither the hassle of manual annotations nor the burden or a specialized toolchain. This approach capitalizes on an architectural feature of ROS [25] as discussed in § 2.4. Our approach requires a one-time effort of building a *mapping* from attributes in shared libraries in ROS to units (instead of annotating every program that uses the shared library). As our approach analyzes a program, the mapping enables the automatic annotation of program variables with physical units, and applies rules from dimensional analysis to detect dimensional inconsistencies. The challenge is to yield a low-enough false-positive rate to justify the value of its findings[1].

5.2 Approach Overview

5.2.1 One-time Mapping from Class Attributes in Shared Program Libraries to Units.

The goal of mapping is to assign physical units to physical attributes in shared libraries. By *physical attributes* we mean class attributes or fields, structures, and class function return values found in shared libraries that represent quantities measured in physical units. For example, Figure 5.1 shows the contents of the `Inertia.msg` data structure from the shared library `geometry_msgs`. As shown in the figure, the variable `m` is a physical attribute of the message class with physical unit type *kilogram* (kg). The attribute `com` references another data structure `Vector3`, which can take on different physical units depending on the context and itself has attributes `x`, `y`, and `z`. As an attribute of the `Inertia` message, `com`'s attributes all have the physical unit type *meters* (m). Likewise, the attributes `ixx`–`izz` all have units kilogramm². Unfortunately, most ROS message structures do not include units in the comments for each attribute, but instead the physical meaning of the attributes is described in the documentation for the shared library.

Rather than annotating physical attributes at the point they are defined in shared libraries, this approach instead decouples this 'mapping' between physical

[1]Both Bessey and Hovemeyer *et al.* used $<$ 20% as a baseline [150, 151]

```
# Mass [kg]
float64 m

# Center of mass [m]
geometry_msgs/Vector3 com

# Inertia Tensor [kg-m^2]
#     | ixx ixy ixz |
# I = | ixy iyy iyz |
#     | ixz iyz izz |
float64 ixx
float64 ixy
float64 ixz
float64 iyy
float64 iyz
float64 izz
```

Figure 5.1: Inertial message class from shared library `geometry_msgs`.

attributes and units from the shared libraries. By decoupling the relationship between shared class attributes and physical units from the shared libraries, system developers do not need annotated copies of those libraries, reducing the toolchain burden. Further, this avoids the reliance on unit-aware type libraries, compilers, or languages—all of which hinder re-use. When compared to individual system developers annotating program variables with physical units at declaration, this approach requires a single effort that can be broadly reused to enable dimensional inconsistency detection in every system that uses those shared libraries. This approach has larger benefits at larger scales. Overall, the purpose of mapping is to achieve the same effect as if the entire user base of the shared libraries were to agree to apply physical unit types in the shared libraries.

More formally, the mapping is a binary relation between two sets: the set of physical attributes $\mathsf{PHYS\_ATTRIB}$ (where physical attributes are identified by fully qualified names (FQNs) in the shared libraries) and the set of unit types ut (Equation 2.1):

$$\mathsf{R}_{\mathsf{mapping}} \subseteq (\mathsf{PHYS\_ATTRIB} \times ut) \tag{5.1}$$

We implement this binary relation R_{mapping} as a lookup table for ROS message types. The complete mapping is shown in Appendix C.

5.2.2 External Mapping Cost.

The upfront effort to create the external mapping is slightly more than applying in-line, manual type annotations to physical attributes in shared libraries, because of the effort to encode the mapping in an external data structure that can be used programmatically. This additional effort is justified by the benefits mentioned above. Compared to annotating attributes in shared libraries, an external mapping introduces no reliance on unit-aware type libraries, compilers, or languages. When compared to annotating programs that use shared libraries, the single effort to create the mapping is much less than the repeated effort by every system developer to separately annotate program variable declarations for those shared libraries.

Creating the core mapping took 3-4 days and was aided because of our extensive familiarity with ROS. An initial investigation of similar cyber-physical middleware like OROCOS [36], OpenRTM [37], MOOS [38], and YARP [39] indicates that a mapping for these domains would require a similar effort. In total, we mapped 246 total physical attributes (class attributes or function return values) from 82 classes across 7 shared libraries, as shown in Appendix C. These physical attributes mapped to 17 distinct derived units. Finally, we encoded the fully qualified name of the physical attributes and its corresponding physical unit to create the mapping.

5.2.3 Algorithm for Lightweight Detection of Unit Inconsistencies

Using this mapping, we present an algorithm LightweightDetectDimensionalInconsistency (Algorithm 1) for dimensional inconsistency detection utilizing this mapping. Some functions of Algorithm 1 that require further explanation are described in the text below.

The analysis examines a program one procedure at a time and is flow-sensitive, path-insensitive, context-insensitive, and intra-procedural. Flow-sensitive means the analysis takes into account the sequential order of statements. Path-insensitive means the analysis does not consider how branch outcomes can result in different program states. Context-insensitive means the analysis does not consider the calling context for procedures, and therefore is intra-procedural. Although the analysis is intra-procedural, it analyzes procedures in reverse call-graph order so we can know what units a procedure returns, if any. In these cases, the approach applies the units returned by the function at its call point. Note that for math procedures like `atan2` we encode the return units, *radians*, into the mapping (see Appendix C Table 9.3 for details). Further, the analysis accumulates information about globally scoped variables during analysis.

A dataflow analysis is often defined using states, a transfer function, a lattice, and a join operation [152]. The states represent knowledge at entry/exit points of blocks, a transfer function calculates changes to the state during that block, the lattice represents all possible abstract states arranged in a power-set hierarchy, and the join function calculates the state at the entry to a block by 'joining' the states that flow into that block in the control flow graph. In contrast, the analysis has only a single state (as opposed to multiple states that must be joined), *State*, that enters and exits every statement. We have a single state because our analysis is path-insensitive, meaning we never split the state at branches. *State* is a set of tuples representing variable unit assignments, $\{(var, \{units\}), ...\}$ where $var \in \mathsf{VAR}$, the set of program variables and $\{units\} \subset ut$, the unit type language of Equation 2.1. A power-set lattice representation of the abstract state is a poor fit because physical units form an Abelian group, and therefore have no 'top' or 'bottom,' and therefore we instead use a unit type language (Equation 2.1).

Statements are analyzed sequentially (flow sensitive) without regard to control flow (path insensitive). At a program point, the units of a variable in *State* are the union of: 1) any units specified by the mapping because the variable is of a type that belongs to a shared library and represents a physical class attribute; 2) previous unit assignments. The transfer function from before a statement (the '*in*' state) to after the statement (the '*out*' state) is the union of: 1) the previous state; 2) the evaluation of the units resulting from the RHS expression of assignment and return statements. Since there is only one state, the join operation is unnecessary. If a program path branches, and a variable were assigned different units based on the branch taken, then the analysis reports an 'Assignment of Multiple Units' inconsistency (see § 2.2.1). This would be a false positive because the analysis over-approximates the space of possible executions, but it still is likely bad programming hygiene to use the one variable to mean two different physical concepts.

5.2.3.1 Algorithm Overview.

Algorithm 1 takes as input a program P and relation $\mathsf{R}_{\mathsf{mapping}}$ from Equation 5.1. During the loop in lines 5-10, the algorithm processes each program statement once. It detects the three kinds of dimensional inconsistencies (see § 2.2) in two ways: 1) within a statement for addition/comparison inconsistencies; and 2) by

Algorithm 1 Lightweight physical dimensional inconsistency detection over program P

Require: Program P and unit mapping $R_{mapping}$.
Ensure: Set of inconsistencies.

```
1:  function LightweightDetectDimensionalInconsistency(P, R_mapping)
2:      DI ← ∅                                         ▷ Dimensional Inconsistencies
3:      State ← ∅
4:      sortedFunctions ← Preprocess(P)
5:      for function ∈ sortedFunctions do
6:          for statement ∈ function do
7:              statement ← AnnotateWithUnits(statement, State, R_mapping)
8:              statement ← EvaluateExpressions(statement)
9:              DI ← DI ∪ DetectExpInconsistency(statement)
10:             State ← State ∪ TransferFunction(statement)
11:     DI ← DI ∪ DetectMultipleUnitInconsistencies(State)
12:     return DI

13: function TransferFunction(statement)
14:     newUnits ← getRHSUnits(statement)
15:     if newUnits = ∅ then
16:         return ∅
17:     if isAssignment(statement) then
18:         return {(getLHSVar(statement), newUnits)}
19:     else if isReturn(statement) then
20:         return {(functionName, newUnits)}
21:     return ∅
```

analyzing variables in the final version of *State* for multiple unit assignments to one variable.

Preprocess. In line 4, the algorithm preprocesses program P by constructing a context-insensitive call graph (without alias analysis) and performing a reverse topological sort, to analyze functions bottom-up. If the call graph contains a cycle, an edge of the cycle is removed from the call graph until no cycles are found. If the topological sort yields a partial order, the approach breaks ties arbitrarily and examines only the first ordering for simplicity and because we do not seek for our analysis to be sound. We examined multiple orderings on several sample programs and did not find differences signifiant enough to justify making the analysis 5-10 times slower for negligible gains. The output is an ordered list of functions.

AnnotateWithUnits. In line 7, this function traverses a statement's Abstract Syntax Tree (AST) and applies unit annotations to variables, when possible. We assume the existence of a relation between the set of program variables VAR and the set of physical attributes PHYS_ATTRIB:

$$\mathsf{R}_{\mathsf{typeOf}} \subseteq (\mathsf{VAR} \times \mathsf{PHYS\_ATTRIB}) \tag{5.2}$$

The relation in Equation 5.2 is commonly provided by a compiler front end, and in Phriky this is provided by Cppcheck [147]. Using the composition of this relation with the mapping from Equation 5.1 we have:

$$\mathsf{R}_{\mathsf{unitsOf}} \equiv (\mathsf{R}_{\mathsf{mapping}} \circ \mathsf{R}_{\mathsf{typeOf}}) \subseteq (\mathsf{VAR} \times ut) \tag{5.3}$$

Where $\mathsf{R}_{\mathsf{unitsOf}}$ is the composition of the relations in Equation 5.1 and Equation 5.2, thereby linking program variables to units.

Program variables can be annotated with units from either a prior assignment statement listed in *State* or when the variable's type is found in $\mathsf{R}_{\mathsf{unitsOf}}$. The function AnnotateWithUnits first checks for units in *State* and if no units are found, checks $\mathsf{R}_{\mathsf{unitsOf}}$. If neither structure yields units, then the variable is annotated with δ, the unknown unit. An example of unit annotation using $\mathsf{R}_{\mathsf{unitsOf}}$ is shown in the dotted boxes of Figure 5.2. These variables can be annotated because their variable type belongs to the shared library `geometry_msgs` that declares a class `WrenchStamped` with physical class attributes included in $\mathsf{R}_{\mathsf{mapping}}$. The composed relation $\mathsf{R}_{\mathsf{unitsOf}}$ connects variable `force.x` to the units $\mathrm{kg\,m\,s^{-2}}$.

EvaluateExpressions. This function visits a statement's AST and attempts to resolve the units of expressions using the unit resolution rules shown in Table 5.1. It works from the leaves up, matching expressions to unit resolution rules and annotating the interior nodes of the AST with units. It continues to apply unit resolution rules in a loop until no changes are made. These rules apply when variables or expressions with units are combined and manipulated.

Note an important difference between the rule for multiplication and the one for addition: during multiplication, if one operand has known units but the other is δ, the unknown unit, we pessimistically assume the result is unknown; during addition, if one operand is known and the other is δ, we optimistically assume the result is the known unit. The reason multiplication is pessimistic is that there is only one way for multiplication to yield the same units, and many ways for the result to be different. Multiplication only yields the same units when multiplied by a scalar with *unity* as the unit, and assuming that every unknown variable involved in multiplication is a scalar leads to many false positives. The reason addition (and equivalently subtraction) is optimistic is that the resulting sum must have the same units as the known operand or be inconsistent, and we cannot conclude the sum is inconsistent because δ is unknown. Further, any subsequent dimensional inconsistency based on an optimistic assumption about the physical unit type resulting from addition must still be valid because of Euclid's first axiom: "things that are equal to the same thing are equal to each other."

An example of how the function EvaluateExpressions works is shown in Figure 5.2. The units in the dotted boxes were applied in AnnotateWithUnits, and the three multiplications near the bottom of the AST match the multiplication rule in Table 5.1. By the multiplication rule, we add the exponents of the units of the operands, yielding the unit annotations on the three '$*$' symbols. Next, the rules match the '+' symbol up the tree, and apply the addition resolution rule, yielding the union of the operands' units. This function continues to apply unit resolution rules until no more changes can be made. This function only adds additional unit annotations and does not detect dimensional inconsistency in the expressions, which happens in the next function.

DetectExpInconsistency. This function applies the unit consistency tests from Equation 2.2 (addition) and Equation 2.3 (comparison) to expressions within a single statement. This function scans a statement's AST looking for inconsistencies like those in Figures 1.1,2.2,2.3, and 5.2. The example in Figure 5.2 shows an dimensional inconsistency detected while evaluating an addition expression.

As shown in Table 5.1, the dimensional inconsistency detection has a 'confidence' that can be either High or Low, High if the units of all variables in the expression are known and Low if the expression contains δ, the unknown unit. Figure 5.2 shows the detection of inconsistent addition of $\mathrm{kg}^2\,\mathrm{m}^2\,\mathrm{s}^{-4}$ to $\mathrm{kg}^2\,\mathrm{m}^4\,\mathrm{s}^{-4}$ with High confidence.

TransferFunction. The transfer function in this analysis can only add new information to the state, and only for assignment or return statements. For assignment statements, the function getRHSUnits at line 14 simply returns the units annotating the '=', and otherwise returns the empty set. In line 10 of Algorithm 1, *State* is updated as the union of *State* and the output of TransferFunction.

DetectMultipleUnitInconsistencies. Scanning *State* at line 11 of Algorithm 1 can reveal *assignment of multiple units* inconsistencies. This kind of inconsistency comes from two sources 1) variables assigned units contrary to their specification in the $\mathsf{R}_{\mathsf{mapping}}$; and 2) variables assigned different units at different points in the program.

When *State* contains multiple units for a variable this function reports inconsistencies with either Low or High confidence, based on the presence of δ in a variable's units. This function reports High confidence if at least two units without δ are assigned to a program variable.

EXPRESSION	CONDITION	RESULT	INCONSISTENT	CONFIDENCE
$ut_1\{*, \div\}ut_2$		ut_3 =add/subtract exponents of ut_1 and ut_2		
$ut_1\{*, \div\}\delta$		$ut_1 * \delta$ (pessimistic)		
$ut_1\{+, -\}ut_2$	$ut_1 = ut_2$	ut_1		
$ut_1\{+, -\}ut_2$	$ut_1 \neq ut_2$	$ut_1 \cup ut_2$	Yes, by Equation 2.2	High
$ut_1\{+, -\}\delta * ut_2$	$ut_1 \neq ut_2$	$ut_1 \cup \delta * ut_2$	Yes, by Equation 2.2	Low
$ut_1\{+, -\}\delta$		ut_1 (optimistic)		
$pow(ut_1, n)$	$n \in \mathbb{R}$	multiple each ut_1 exponent by n		
$sqrt(ut_1)$		divide each ut_1 exponent by 2		
$sqrt(\delta)$		δ		
$ut_1\{<, >, \geq, \leq, \neq\}ut_2$	$ut_1 = ut_2$	none		
$ut_1\{<, >, \geq, \leq, \neq\}ut_2$	$ut_1 \neq ut_2$	none	Yes, by Equation 2.3	High
$ut_1\{<, >, \geq, \leq, \neq\}\delta * ut_2$	$ut_1 \neq ut_2$	none	Yes, by Equation 2.3	Low
$ut_1\{<, >, \geq, \leq, \neq\}\delta$		none		
$\{floor, ceil, (f)abs\}(ut_1)$		ut_1		
$\{min, max\}(ut_1, ut_2)$		$ut_1 \cup ut_2$		
$\{min, max\}(ut_1, \delta)$		ut_1		
$(Boolean)\ ?\ ut_1\ :\ ut_2$		$ut_1 \cup ut_2$ (ternary operator)		

Table 5.1: Unit resolution rules used in Algorithm 1 function EvaluateExpressions on line 8, and the inconsistency rules are used to detect addition/comparison inconsistencies in function DetectExpInconsistency on line 9.

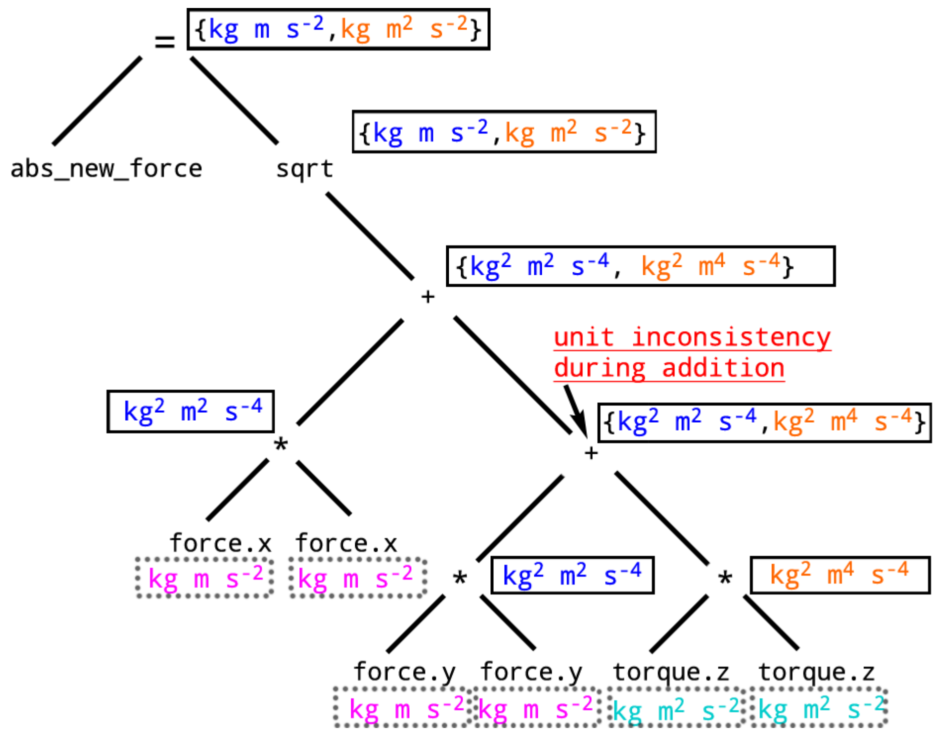


Figure 5.2: Example of a statement's AST from the code in Figure 2.3 with the shared class fully qualified name `WrenchStamped::wrench` omitted for simplicity. Figure shows unit annotation of variables by the relation R_{unitsOf} (dotted boxes), and evaluation of expressions' units toward the root by unit resolution rules in Table 5.1 (solid boxes).

5.3 Implementation: Phriky

We implement the approach from § 5.2 in a tool call PHRIKY. Our implementation detects dimensional inconsistencies in C++ code written for ROS. We explored the trade-offs between precision, speed, and scalability and aimed for a lightweight analysis. The architecture follows the approach, is implemented in 3,300 lines of python, and can be run from the command-line.

PHRIKY utilizes CPPCHECK as a C++ preprocessor and parser [147], invoked with default parameters and includes directories:

```
cppcheck --dump -I ../include myfile.cpp
```

The *dump* option generates an XML file containing:

1. Every program statement as a separate abstract syntax tree.

2. Token list.

3. Symbol database including functions, variables, classes, and all scopes.

Cppcheck can explore multiple compilation configurations (different `#define` values), but in the reported results we only consider the default system configuration. We considered using a more powerful preprocessor and parser framework, Clang, and then implementing our analysis as a Clang plugin, but instead chose Cppcheck because Cppcheck works even without a complete compilation unit. Having results without a complete compilation unit allows us to analyze a wide variety of open-source code without having to resolve all its dependencies. Phriky also uses `NetworkX` [153] to topologically sort the call graph.

We use a visitor pattern in each statement's AST to apply units and evaluate expressions with unit resolution rules. During implementation, we realized that *radian* and *quaternion* require special handling: during multiplication, *radian* and *quaternion* act as *unity* since their units are *meters-per-meter* as discussed in § 2.1; during addition, they are 'coherent units of measure' [17], meaning that they cannot be added to a dissimilar unit, even though they are dimensionless.

An example inconsistency message for the code in Figure 2.3 reads:

> `Addition of inconsistent units on line 1094 with High confidence.`
> `Attempting to add` $kg^2m^2s^{-4}$ `to` $kg^2m^4s^{-4}$.

We consolidate error messages to report only the first dimensional inconsistency for a particular variable.

5.3.1 Termination and Complexity of Phriky

Preprocessing requires a linear pass over the program to construct the context-insensitive call graph, and topologically sorting the call graph is $O(|V| + |E|)$ with a worst case $O(|E^2|)$ when detecting and removing cycles. The loop in lines 5-10 analyzes each statement once and is linear in the size of the input program AST. Annotating a statement's variables with units, evaluating expressions, detecting expression dimensional inconsistencies, and the transfer function are linear in the size of a statement's AST. After the loop, detecting multiple dimensional inconsistencies requires a linear scan of *State*, and *State* is as large as the number of program variables. Putting it all together, the worst case for the algorithm is quadratic in time and space. Termination is guaranteed because EVALUATEEXPRESSIONS (see § 5.2.3.1) applies unit resolution rules at most h times where h is the height of a statement's AST.

5.4 Research Questions

We ask:

- RQ_6: How effective is PHRIKY at detecting dimensional inconsistencies?
- RQ_7: Are the dimensional inconsistencies detected by PHRIKY problematic to real robotic system developers?

We ask RQ_6 to better understand how effective PHRIKY is at detecting these kinds of inconsistencies. We ask RQ_7 to better understand the relevance of these kinds of inconsistencies to real robot software developers.

CORPUS SOURCE	# of REPOSITORIES
Total ROS.org "Indigo" Projects Links	2416
Live Git Repos	649 of 2416
Git REPOS with C++ FILES	436 of 649
REPOS with C++ FILES AND ROS UNITS	213 of 436

Table 5.2: ROS Open-Source Repositories

5.5 Results

To answer RQ_6 we run Phriky on a corpus of publicly available robotic systems and then hand-label the results as True and False Positives.

5.5.1 Analysis of Robotic Software Corpus

The maintainers of ROS published a list of public software repositories using ROS in academic and industrial robots. The list, published at `http://www.ros.org/browse/list.php`, includes projects at various stages of development, and for a wide variety of purposes: mobile robot navigation, collision detection libraries for robotic arms, drivers for depth cameras, control software for flying robots—a diverse set.

Table 5.2 shows statistics about this corpus. At the time we gathered this corpus in early 2017, there were $2,416$ projects linked from the 'Indigo' version of ROS. Of these $2,416$ links, 649 were linked to live Git repositories. ROS supports `C++`, `Python`, and a few projects with `LISP` and `Java`, but the majority are in `C++`, so we focused on those. Of the 649 live Git repositories, 436 contained `C++` files. Of these 436, we found 213 repositories with systems containing shared libraries with physical attributes. For this initial work, we proceed with all ROS `geometry`, `navigation`, `transform`, `sensor`, and `time` libraries.

	High Confidence			Low Confidence		
INCONSISTENCY TYPE	TP	FP	TP%	TP	FP	TP%
Assignment of Multiple Units	33	6	85%	55	83	40%
Addition of Inconsistent Units	5	0	100%	9	20	31%
Comparison of Inconsistent Units	2	0	100%	0	4	0%
TOTAL	40	6	87%	64	107	37%

Table 5.3: Classification of dimensional inconsistencies found by PHRIKY. Note: this table presents precision and not recall, because recall requires false negatives (FN) that are unknown in our corpus.

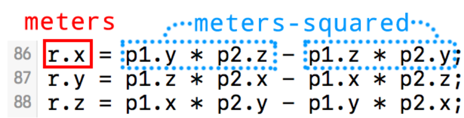


Figure 5.3: Figure showing a cross-product operation, a False Positive corner case.

5.5.2 RQ$_6$ Results: Phriky Effectiveness

We individually examined each inconsistency reported by PHRIKY, reviewing the source code surrounding each reported line, and labeled each one as either 'True Positive' (TP) or 'False Positive' (FP). Note that labeling inconsistencies as TP or FP lets us calculate precision, but the number of 'False Negatives' (FN) is unknown and therefore we cannot calculate recall (see § 8.2.2). This labeling process required several rounds of iterations as the analysis of some inconsistencies led us to question and re-analyze previous labels.

Our results are summarized in Table 5.3. The overall TP rate, computed as $TP\% = 100 * TP/(TP + FP)$, for HIGH confidence dimensional inconsistencies is 87%. This includes the three types of inconsistencies (see § 2.2), assignment of multiple units, comparison of inconsistent units, and addition of inconsistent units. As we noted earlier, for one of the cases where we contacted the authors of the code for clarification, shown in Figure 1.1, the inconsistency was acknowledged as a fault by the developers within 90 minutes and patched within 36 hours. Within the HIGH confidence TP, we found a TP rate of 84.6% for variables assigned multiple units.

The False Positives with High confidence all detect redundant implementations of vector cross-products and outer-products that are already provided by the ROS API, where *meters-squared* intentionally equals *meters*. Figure 5.3 contains one such case in line 90, which is frequently used and deemed correct by system developers. In general, Phriky handles vectors like any other quantity and detects inconsistent addition, comparison, and assignment. We believe we could modify Phriky to detect and ignore this special case, but we would have to be careful not to blind Phriky to unintentional assignments of *meters-squared* to *meters*, therefore for now we accept these kinds of FP.

The overall TP rate for Low confidence dimensional inconsistencies is 37.45%, with about 50% more low confidence TP (64) than High confidence TP (40). The low TP rate is caused mainly by the large number of variables and constants with implicit units not found in the mapping.

5.5.2.1 Causes of "Assignment of Multiple Units" Inconsistencies

We observe that "Assignment of Multiple Units" inconsistencies can have at least two distinct causes:

1. Variable re-use (like `temp` variables).
2. Disagreement between the units defined in the mapping (from the documentation) and the actual units used.

Variable re-use was identified as one category of causes of dimensional inconsistencies by Jiang and Su [63], where they broadly identified the kinds of dimensional inconsistencies in programs. However, disagreements between a data structure's specification and its use was not identified by Jiang and Su likely because they were not present in the software artifacts they examined. The first to identify software component interfaces as a source of dimensional inconsistencies was Damevski [3], to our knowledge.

We currently do not distinguish between these causes but believe they could be separated automatically by observing whether the units come directly from the mapping and whether they are assigned only one kind of unit in the program.

5.5.3 RQ_7 Results: Developer Survey

We conducted a survey to obtain an initial assessment of whether these kinds of dimensional inconsistencies are problematic to robotic software developers. Specifically, some dimensional inconsistencies, like variable reuse and using a physical attribute to store a quantity against its specification, might be poor programming style, but also might not warrant a high-priority bug report. Therefore we wanted to assess the severity of these kinds of inconsistencies. Our survey instrument consists of eight questions (see Appendix E for the complete survey), each showing a code artifact similar to those in Figures 1.1 and 2.3, drawn from dimensional inconsistencies detected by Phriky.

For each code artifact, we asked *"Is the dimensional inconsistency on line [X] problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?"*, with a choice of responses: 'yes', 'maybe', and 'no'. After each question, the respondents could add an open-ended explanation. The order of the questions was randomized for each respondent.

The target population for our survey included either heads of academic robotics research labs or their senior research associates. These labs publish regularly in top robotic conferences and use ROS extensively. We sent our survey to ten labs and

Table 5.4: Summary of survey responses to whether dimensional inconsistencies found by Phriky are 'problematic.'

Question #	YES	MAYBE	NO
1	6	2	2
2	8	1	1
3	3	5	2
4	9	0	1
5	6	3	1
6 (Figure 5.4)	2	8	0
7 (Figure 2.3)	7	3	0
8	4	5	1
TOTALS	45	27	8
%	56%	34%	10%

received ten responses from six of the labs. We recognize the sample population size is small and may not generalize, and a larger, more nuanced study might be justified in the future.

Our results are shown in detail in Table 5.4. Overall, 56% of responses indicate that 'yes', these dimensional inconsistencies are problematic. The 'yes' responses included explanations from *'The addition of different units means nothing in real world'* to *'just bad programming.'* This fits with our assessment that many dimensional inconsistencies require attention or at least special explicit justification.

The 'maybe' responses (34%) included explanations, such as *'If the angular radius is unity, then OK, otherwise could lead to error'*, identifying a special case when the code *could be correct*, or *'I don't know when you'd like to compute this.'* Several 'maybe' responses indicated the possibility of a special circumstance when the dimensional inconsistency might not be problematic. In these cases Phriky indicates a possible constraint on the circumstances under which the code behaves correctly, and for the dimensional inconsistencies detected by Phriky, these special circumstances were never mentioned in the code comments, to our knowledge.

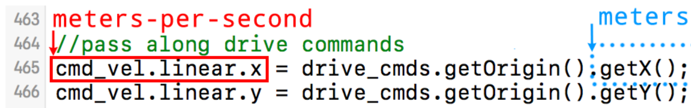


Figure 5.4: Inconsistent assignment.

Of 'no' responses (not problematic), half came from one respondent (4 of 8), who explained: *'The problem I see is that the proposed method will get hung up in hacks that actually are workable solutions and it might be impossible for the average coder to fix these issues.'* We contend that detecting 'workable' 'hacks' is still valuable, especially for junior developers lacking the hard-earned experience necessary to recognize them in the first place.

Questions 6 and 7 from Table 5.4 of the survey are also presented in this work as Figure 5.4 and 2.3. Notice for question 6 that most respondents said 'maybe', and this code artifact shows dimensional inconsistency by assignment to a data type with a different physical unit specification, which is perhaps more an issue of code maintenance and reuse since it only uses a technically incorrect data container. However in question 7 (Figure 2.3) most respondents said 'yes problematic', and this code artifact contains addition of inconsistent units, which is perhaps more concerning because it might be incorrect. We believe that identifying both of these kinds of dimensional inconsistencies has value to system developers.

At the end of the survey we let respondents write an open-ended 'overall' feedback to these kinds of dimensional inconsistencies. The most critical respondent stated *'Overall a lot of dimensional inconsistencies will happen for control or optimization reasons and sometimes ... cannot be avoided,'* while the most laudatory stated *'This tool is amazing! At the very worst, it find out questionable programming practice that needs additional documentation. Most of the time, it finds bugs or hacky heuristics.'*

Overall, is spite of the limited size our population, and that this population does not represent industrial system developers, most responses affirm our assertion that the kinds of dimensional inconsistencies detected by this approach are problematic.

5.5.4 Scale and Speed.

We ran PHRIKY on 213 systems containing ROS physical units, analyzing 934, 124 non-blank non-commented lines of C/C++ as reported by CLOC [149]. Analyzing all systems took 108 minutes (61 minutes to parse the files with CPPCHECK and 47 minutes to perform the analysis), with an average analysis time of 31 seconds per system, when running on a MacBook Pro ('early 2015') with a 2.9 GHz Intel Quad Core i5 processor, and 16 GiB of memory. We only utilized a single core during evaluation, although this could be easily parallelized since the files and analysis are independent.

5.6 Threats and Limitations

5.6.1 Self-labeling.

We rely on self-labeled TP and FP. We used multiple authors to review each inconsistency independently. Low confidence TPs were directly or transitively involved with partial information and were harder to identify, so we assumed the LOW confidence inconsistencies were FP until proven to be a TP.

5.6.2 False Negatives.

We cannot measure recall because the total number of faults due to improper units in the software corpus is unknown. We could address this threat by seeding faults.

5.6.3 Limitations.

While designed to be as fast and lightweight as possible while detecting useful inconsistencies, Phriky has limitations in applicability, soundness, and completeness. This approach is unsound because it includes infeasible sets of variable-unit assignments in *State* as it ignores control flow. This approach is incomplete because *State* misses some variable-unit assignments in loops and because it is not path-sensitive. Further, the approach does not attempt symbolic analysis that could reason about statements like $(UNIT^n)^{(-n)}$. The key limitation is that the only evidence for physical units is the mapping and this only applies units to physical attributes identified and correctly assigned beforehand.

5.6.4 Summary

In this chapter, we examined a method of detecting dimensional inconsistencies that capitalizes on some program variables being attributes of shared ROS message libraries. This enables us to get some evidence about the unit types for free. However, not all program variables are ROS message class attributes, so the number of program variables that can be labeled with unit information limits Phriky's power. To quantify this limitation, we instrumented Phriky to count the number of variables that do not have unit types but likely represent real-world quantities (`float` and `double` variable types), and found Phriky only addresses 24% of variables. We estimated 24% by manually annotating 924 variables from 30 programs and counting how many variables Phriky assigns units to (see § 7.4.1).

That Phriky labels only 24% of variables is a weakness that motivates our efforts in § 7 to find additional sources of evidence for type inference.

The method described in this chapter, implemented in PHRIKY, can detect dimensional inconsistencies that developers deem problematic. However, we do not know how frequently dimensional inconsistencies occur, making it hard for developers to understand the scope of the problem. Therefore, in the next chapter we apply PHRIKY to a corpus of 5.9 *M* lines of code.

6 Study of Inconsistencies in 5.9 Million Lines of Code

PHRIKY can detect dimensional inconsistencies, and the consequences of such inconsistencies exhibit a range of severities, from mild to occasionally catastrophic [4]. There does not seem to exist, however, an estimate of how frequently dimensional inconsistencies occur. Consider the 3,484 repositories of the Robot Operating System (ROS) [25] code we study in this chapter. These repositories have hundreds of thousands of program points where variables represent physical quantities including time, distance, angles, torques, Teslas, and others.

The work presented in this section was published in [14].

6.1 Study Overview and Research Questions

We investigate the following research questions:

- **RQ**$_8$: How frequently do dimensional inconsistencies occur in programs that use ROS?
- **RQ**$_9$: What units are used in ROS, and what does this tell us about how ROS is used?

- **RQ$_{10}$**: What ROS Message classes are most commonly used with incorrect units?

To address these research questions, we designed a study to apply our dimensional inconsistency and physical unit detection tool, Phriky, to a large-scale software corpus.

6.1.1 Software Corpus

We sought to build a corpus of ROS code with physical units specified by standard ROS message types, because ROS messages have attributes defined to have units, and because detecting dimensional inconsistencies requires units. We constructed the software corpus for inconsistencies in the same manner described in § 4.2.2.1. GitHub is one of the largest collections of open-source code available and has been used as the basis of other large-scale software studies [43]. To find ROS code with units, we used the GitHub code search API to submit keyword queries for each ROS message type defined at `http://wiki.ros.org/common_msgs`, and extracted the repository names from the results. We conducted the search and built the corpus in mid-2017. In total we found 4,736 repositories that contained search hits on ROS-related terms. Of this, 73% or 3,484 repositories contain compilable `C++` code that uses the ROS messages defined to have physical units. Within these 3,484 repositories, we found a total of $20,843$ files with units containing $5,950,839$ lines of `C++` code as measured using the tool Cloc (`http://cloc.sourceforge.net`). We provide a complete list of repositories used in this study in Appendix H.

The corpus contains $\approx$ 30% of duplicate code. We consider two files to be duplicates if they have the same `md5` hash. Since we evalute code duplication at the file level, we likely underestimate the amount of code duplication that occurs at the function or statement level, since files might only be different by one character and have a different hash. We decided to leave duplicates in the corpus because we wanted to assess the frequency of units in code that is re-used across ROS developers.

6.2 Results

6.2.1 RQ_8 Results: Dimensional Inconsistency Frequency.

We detected dimensional inconsistencies in 211 of the 3,484 repositories, or 6%. Granted, some of these inconsistencies might be FP, since Phriky has an 87% TP rate 5.5.2, which might cause a slight overestimate. However, 6% might be an underestimate because we do not know how many dimensional inconsistencies exist in these repositories, only how many Phriky detects.

This 6% answers RQ_8, and this result shows that dimensional inconsistencies lurk in a non-trivial number of repositories.

6.2.2 RQ_9 Results: Kinds of Inconsistencies

Dimensional inconsistencies in software appear in several forms, and the most common in ROS is the 'Assignment of Multiple Units' type (defined in § 2.2), as shown in Table 6.1. This inconsistency represents 75% (267/357) of all inconsistencies found by Phriky, and is most likely to occur with *meters* and *meters-per-second*, as shown in the table. The *meters-squared* associated with 'Addition of Inconsistent Units' are usually caused by improperly formed distance metrics (Euclidean distances), like that shown in Figure 1.1. These distance metrics are either typos or combinations of dissimilar units, which can behave correctly because of implicit

INCONSISTENCY TYPE	COUNT	UNITS	MOST FREQUENT UNITS COUNTS
Assignment of Multiple Units (Equation 2.4)	267	m $\mathrm{m\,s^{-1}}$ $\mathrm{s^{-1}}$ quaternion $\mathrm{m^2}$ rad $\mathrm{kg\,m\,s^{-2}}$	204 171 71 30 27 15 4
Addition of Inconsistent Units (Equation 2.2)	61	$\mathrm{m\,s^{-1}}$ m $\mathrm{s^{-1}}$ quaternion m rad $\mathrm{m^2\,s^{-2}}$	34 32 14 10 6 5 1
Comparison of Inconsistent Units (Equation 2.3)	29	$\mathrm{m\,s^{-1}}$ $\mathrm{s^{-1}}$ m $\mathrm{m^2}$ $\mathrm{m^2\,s^{-1}}$ s	21 6 6 4 2 1

Table 6.1: Dimensional Inconsistencies by Type with the most frequently involved units. Note that multiple units can be involved with one inconsistency.

constraints on the values that effectively normalize the values. However, these implicit assumptions hinder portability and might introduce faults when these assumptions change. The comparison of inconsistent units happens for a variety of reasons, but most often involve velocities and inconsistent interactions with time.

All inconsistency types were more likely to be caused by interactions between simple units, such as *seconds*, *meters*, *meters-per-second*, and *quaternions*. The more sophisticated units (combination of three or more base units) like *torque* are used less frequently in the corpus and account for an even smaller percentage of inconsistencies, suggesting that either the developers who work with sophisticated units are more careful not to cause dimensional inconsistencies, or the space for inconsistencies across those units is smaller. Further, many inconsistencies are caused when developers use ROS message types contrary to their specification.

This might not manifest as incorrect behavior if these misused data structures are used consistently. However, these data structures can cause confusion when sharing or maintaining code.

UNIT NAME	SI UNIT	REPO COUNT	FILE COUNT	UNIT USAGE by ROS MSG DEFINITION	UNIT INFERRED USAGE by ASSIGNMENT
meter	m	2,669	9,930	112,538	19,525
second	s	2,433	9,939	85,299	9,573
quaternion (rotation)	*(dimensionless)*	2,078	6,169	49,449	2,749
angular velocity	s^{-1}	1,790	4,313	17,645	1,363
velocity	$\mathrm{m\,s}^{-1}$	1,598	3,961	21,885	2,078
radian (angle)	*(dimensionless)*	1,106	3,133	159	21,557
acceleration	$\mathrm{m\,s}^{-2}$	355	456	1,580	171
torque	$\mathrm{kg\,m}^{2}\,\mathrm{s}^{-2}$	257	403	2,373	18
area or *pose covariance*	m^{2}	187	314	333	770
degree 360 *(angle)*	*(dimensionless)*	172	232	844	68
angular acceleration	s^{-2}	168	199	544	3
acceleration covariance	$\mathrm{m}^{2}\,\mathrm{s}^{-4}$	156	183	495	0
Newton (force)	$\mathrm{kg\,m\,s}^{-2}$	154	606	2,366	29
Tesla (magnetic induction)	$\mathrm{kg\,s}^{-2}\,\mathrm{A}^{-1}$	46	52	151	10
Celsius (temperature)	°C	37	40	42	2
Pascal (pressure)	$\mathrm{kg\,m}^{-1}\,\mathrm{s}^{-2}$	17	21	23	2
lux	lx	12	12	12	0
Pascal covariance	$\mathrm{kg}^{2}\,\mathrm{m}^{-2}\,\mathrm{s}^{-4}$	3	3	3	0

Table 6.2: Most common physical units used in 20,843 files across 3,484 open-source repositories in 5.9M lines of code, based on units from both ROS Messages and units inferred in the code by PHRIKY.

6.2.3 Units Used and Frequencies

Table 6.2 shows the frequency of physical units used in ROS code. By 'Unit Usage by ROS Msg Definition' we mean the number of program points where a variable has units because it is a ROS Message attribute or the result of a known math operator, like `atan2`. By 'Unit inferred usage by assignment' we mean the number of program points where a variable has units not based on a ROS Message definition but instead inferred by the context of the program as the result of assignment statements and mathematical operations. This distinction is important because it tends to separate the units used externally in ROS Messages to communicate between nodes from those used internally in a ROS node during computation.

At a high level, Table 6.2 shows that simpler units are used more frequently, in more repos and files, and used more frequently during computations. There are some exceptions to this overall trend, including for *meters-squared*, *force*, *torque*, and *radians*, as we now discuss.

The radian unit, as shown in Table 6.2, is the most common way to represent an angle, but notice that it is used more times as an inferred unit (21,557) than as a ROS Message definition (159). This suggests that robot software developers make extensive use of this representation of an angle, but that ROS does not have a standard way to represent it within ROS nodes. The radian's inferred usage comes mostly from the result of math operators such as `atan2`, `acos`, or `asin`.

Force ($\mathrm{kg\,m\,s^{-2}}$) is only found in 4% (154/3,484) of repositories, but is used 2,395 times. Likewise *torque* ($\mathrm{kg\,m^2\,s^{-2}}$) is found in 7% (257/3,484) of repositories and used 2,391 times. This means most ROS projects do not measure, compute, or communicate about *forces* and *torques*, or that many users are not using standard message types for force and torque. However, repositories that use *force* and *torque* perform several calculations and manipulations on these quantities. This might suggest that $< 10\%$ of ROS projects involve systems like robot arms, where *force* and *torque* measurements are more common.

Meters-squared (area or pose covariance) is used by definition 333 times and inferred 770 times. The inferred uses are usually Euclidean distance metrics, while the use by definition is position covariance. Although these quantities have the same units, they represent different kinds of quantities and should not be combined or compared, but in this case dimensional analysis would not detect this, because they have the same units.

These results address RQ9, and indicate that the more sophisticated units (like *force* and *torque*) are used in less than 10% of repositories, and that most ROS code achieves its goals using a combination of less complex units.

6.2.4 ROS Message Classes Most Likely to be Used with the Wrong Units.

Phriky detects when ROS Messages are used with units contrary to their specification, often the result of interactions between two conflicting sources of unit information. In our case, this interaction occurs because of a mismatch between the units specified by the ROS Message type, and the units actually assigned to the variables of the ROS Message.

To help identify the ROS classes most likely to be used together inconsistently, we plotted the pairs of ROS Message classes involved in inconsistencies in Figure 6.1. Note that this figure would not show dimensional inconsistencies such as those from Figure 2.3 because that inconsistency only involved units that origi-

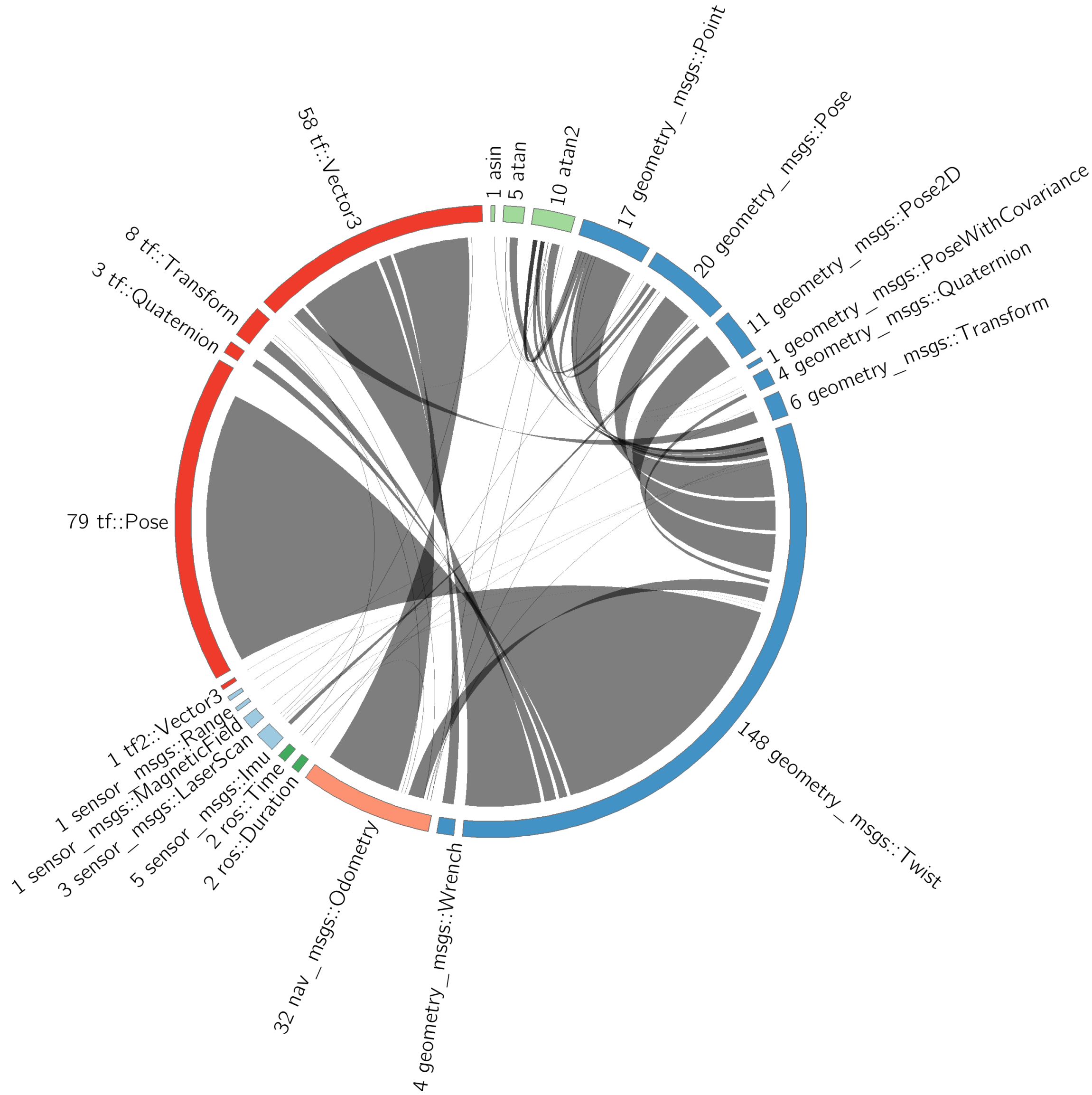


Figure 6.1: Pairs of ROS Message classes involved with dimensional inconsistencies. Edges between ROS Message classes indicate an instance of inconsistent usage involving these two classes. Numbers preceding the ROS Message class indicate the number of inconsistencies. Stamped and unstamped messages were combined.

nated from one ROS Message class, `geometry_msgs::Wrench`. This figure shows an edge drawn between classes to indicate a pairwise inconsistent interaction. For example, the inconsistent usage shown in Figure 2.1 results in a edge between geometry_msgs::Twist and geometry_msgs::Pose. Some ROS Messages types have two subtypes, stamped and unstamped, which are identical other than a timestamp attribute. Figure 6.1 combines stamped and unstamped messages for simplicity.

As shown in Figure 6.1, usage of geometry_msgs::Twist accounts for 41% (148/357) of all inconsistent ROS Message usage, and is used most frequently in combination with tf:Pose and tf::Vector3. Also note the inconsistencies between tf::Vector3 and nav_msgs::Odometry, that often happen with the velocity portion of Odometry, much in the same way as happens with Twist. This answers RQ_{10}.

6.3 Practical Implications

6.3.1 Use Standardized ROS Units

Our study found that standardized ROS units are used in 70% (3,484/4,736) of the accessed repositories, with units related to position, time, and velocity making the bulk of the units we identified (they are 2.4 times more common than the rest of the units combined). As mentioned, the usage estimate is an under-approximation, as many declared variables containing physical units do not employ the standardized ROS units. For example, we found that variables named 'time' and 'duration' are defined with type ros::Time or ros::Duration in 39% (4,123/10,530) of the instances those variable names are used, otherwise they do not have standardized ROS units that could be leveraged by our dimensional analysis. Not using standardized units negatively impacts reuse, making code comprehension more difficult, and undermining the application of tools like Phriky that can help to detect dimensional inconsistencies.

6.3.2 Run an Automated Checker To Detect Dimensional Inconsistencies in Code

Even a lightweight inconsistency detection tool like PHRIKY, which requires no additional effort for code annotation or migration, can detect certain dimensional inconsistencies with high confidence. On a MacBook Pro ('Early 2015') 2.9 GHz Intel i5 with 16 GiB of memory, it can analyze approximately 150 lines of code per second, its operation is trivially parallelizable, and it can be easily integrated as part of standard building processes. So, even for practitioners that have been hesitant to invest in code annotations or specialized libraries usage, there is little reason not to run a tool like PHRIKY.

6.3.3 Avoid Common Anti-Patterns

Since `geometry_msgs::Twist` is the most misused ROS Message type, we performed an additional analysis of how `Twist` is used by ROS developers.

We modified PHRIKY to track assignments made to variables of type `Twist`. `Twist` has 6 attributes: 3 linear velocity components `x,y,z` and three angular velocity components `x,y,z`. For every `Twist` message in the corpus, we tracked which of these 6 attributes were written during programs, and the results are shown in Table 6.3.

As shown in Table 6.3, `Twist` is mostly used for 2-D planar robots (2-D in this case means that the program never writes to attribute `linear.z`). This usage is not inconsistent in itself, since `Twist` is intentionally overloaded to mean either 2-D or 3-D velocities (Euclidean dimensions). However, many of these instances also use `angular.z` to store the heading, not angular velocity as intended. As shown in the figure, developers add the content of `Twist` directly to `Pose`, as a kind of

			`twist.linear.`			`twist.angular.`		
USAGE	TOTAL	COUNT	`x`	`y`	`z`	`x`	`y`	`z`
2-D planar	2,591	1,172	✓					✓
		1,101	✓	✓				✓
		201						✓
		117	✓					
3-D	1,534	1213	✓	✓	✓	✓	✓	✓
		169	✓	✓	✓			✓
		152	✓	✓	✓			

Table 6.3: Usage of `geometry_msgs::Twist` showing majority of 2D planar usage of a 3D structure. A '✓' indicates an attribute was written, and a blank means the attribute was never written. Table does not show read-only instances.

'delta.' Phriky detects this dimensional inconsistency because the physical unit types do not match. Overall, Table 6.3 shows that `Twist` is used in many different and sometimes inconsistent ways, making it difficult for others consuming such messages to correctly interpret what `Twist` means. This might indicate the need to revisit the overload of the structure of this message.

Summary

In this chapter, we found that Phriky detects dimensional inconsistencies in 6% (211/3, 484) of open-source robot software repositories we examined. This means that dimensional inconsistencies happen frequently enough to justify the effort to build and improve automated tools to help developers detect and avoid them, even though 6% is an underestimate. Further, in § 5.6.4 we estimated that Phriky only assigns physical unit types to 24% of variables that likely could represent real-world quantities. This means Phriky has no knowledge of what inconsistencies could be hiding in interactions between untyped varialbes. Therefore, in the next chapter we present an improved method of inferring and predicting physical unit types for more program variables, thereby increasing the power to detect inconsistencies.

7 Improved Physical Unit Inference with a Probabilistic Method

Since PHRIKY can infer units for only 24% of variables, its power to detect dimensional inconsistencies is limited. To address this limitation, we extend our approach from § 5 to utilize evidence of physical unit types available in variable names. However, this evidence is uncertain because developers might give an incorrect or uninformative name (i.e., `result`). The extended approach presented in this chapter uses probabilistic graphical models [154] to combine uncertain evidence in variable names with type evidence through program dataflow analysis [152] and evidence from ROS message types.

The work presented in this section was published in [10], a group effort. The other authors contributions includes: 1) creating a substring similarity metric; 2) using probabilistic graphical models for abstract type inference and formulating probabilistic constraints; 3) choosing prior probabilities for various kinds of evidence according to norms in probabilistic reasoning; 4) contributing to the core programming of PHYS; and, 5) contributing to the evaluation and debugging of PHYS's results.

My contributions to the work presented here includes: 1) guiding the extension of PHRIKY to reason probabilistically about type assignments in the tool PHYS; 2) contributing to the evaluation and debugging of PHYS's results; 3) creating the code

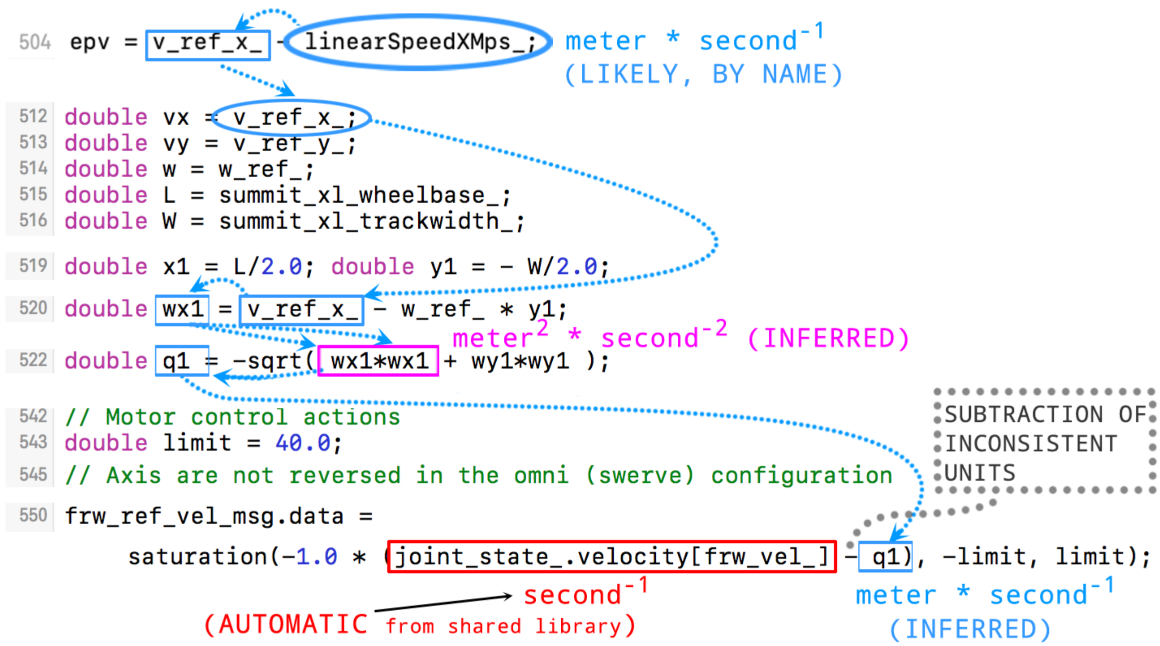


Figure 7.1: Code example where PUT type inconsistency can be detected by adding evidence in variable names.

corpus used during evaluation; 4) examining and classifying the inconsistencies detected by Phys; 5) the comparison between Phys and Phriky; and, 6) proposed extensions to Phys to make it an annotation tool in § 7.8.

7.1 Challenges

The dimensional inconsistency in Figure 7.1 line 550 cannot be detected by Phriky, because Phriky does not have sufficient information to determine the physical unit type for variable q1. On line 550, the variable `joint_state_.velocity` has units *per-second* because it is defined in a shared ROS message library. However, $q1$ has no units from the shared ROS message libraries and no units from flow, making it impossible for Phriky to detect the dimensional inconsistency, 'subtraction of inconsistent units.' This inconsistency is interesting because the code commands a wheel to turn, and it would turn in the right direction. However, the developers on line 550 forgot to scale the angular velocity of the wheel by the radius of the wheel by multiplying by the radius. This is coincidentally correct when then wheel diameter is near 1 m. Otherwise, it turns too fast or too slow based on the scale of the real robot. In this kind of scenario, the system developers might blame the lower-level controller or the tuning parameters. Changing the tuning parameters, especially increasing the gains, might make the system more susceptible to instabilities.

The challenge is to overcome the limitations of Phriky, which can only assign physical units to 28% of variables. This approach uses information in variable names so it can assign physical unit types to more variables and detect more dimensional inconsistencies. The first insight is that variables representing physical quantities are often given an informative name. For example, in Figure 7.1 line 504, the variable `linearSpeedXMps` contains the substring `speed`, implying a type of $\mathrm{m\,s^{-1}}$. The second insight is that although variable names can have useful evidence, this evidence is only partially reliable. Therefore, we must treat this evidence probabilistically.

7.2 Approach Overview and Implementation in Phys

To collect and combine uncertain information, our approach: 1) collects observations (also called *beliefs*) using substring matching of variable names against a pre-existing list of likely string fragments (shown in Appendix F) with prior probabilities; 2) collects evidence from the ROS message libraries; 3) collects evidence from dataflow, such as assignment and mathematical operations; 4) combines and propagates these beliefs using the sum-product belief propagation algorithm [155], finding for each variable the most likely physical unit type, if any.

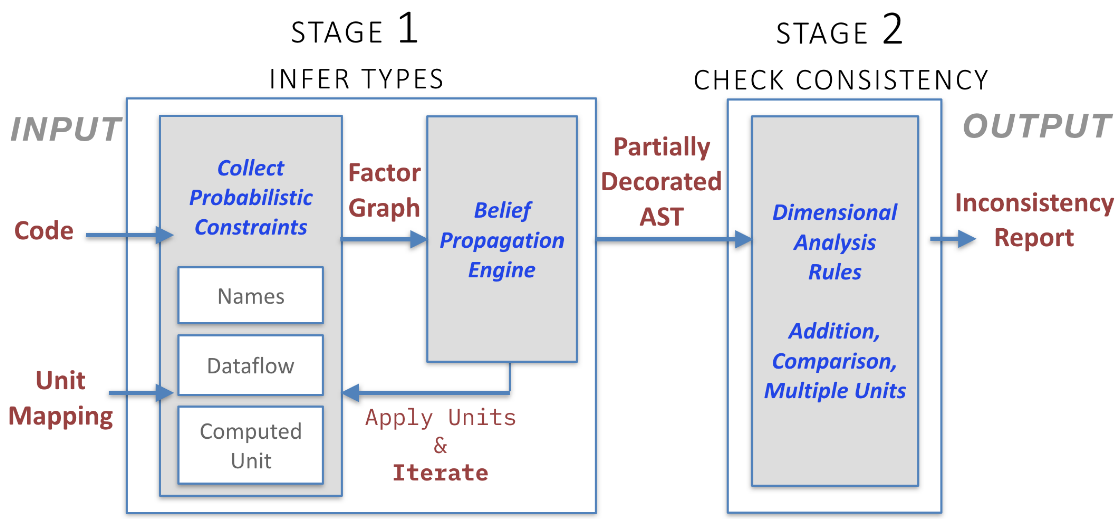


Figure 7.2: High-level overview of PHYS.

Figure 7.2 shows a high-level overview of PHYS.

7.2.1 Stage 1: Infer Physical Unit Types.

As shown in Figure 7.2, PHYS, like PHRIKY (see § 5), takes as input `C++` code written with ROS and a 'Predefined Unit Map' (the *mapping* from § 5.2), simply a dictionary between attributes of ROS message data structure classes and physical unit types.

PHYS preprocesses and traverses the code much in the same way PHRIKY does (see § 5.3). However, the key difference is that during code traversal PHYS collects evidence called 'probabilistic constraints,' as shown on the left-hand side of Figure 7.2. These probabilistic constraints are a way to represent evidence, or beliefs, about physical unit types within a probabilistic mathematical framework. As shown in Figure 7.2, the probabilistic constraints are used to construct a factor graph, a graphical representation that connects all the physical unit type evidence in the program. The factor graph can then be input to a 'belief propagation engine,' an off-the-shelf solver that determines the most likely physical unit type for each variable.

The factor graph is made of probabilistic constraints, and Phys collects three main kinds: Names, Dataflow, and Computed Unit. We now discuss each type in detail.

7.2.1.1 Name Constraints

We infer Name constraints when identifiers (also called 'names') contain substrings that are similar to substrings in a predefined, heuristically determined table. This table, called the 'name assumptions table,' links common physical unit names to physical unit types. This table is shown in full in Appendix F. Name constraints encode assumptions about identifiers names into a probability distribution, also called a belief.

For example, when Phys encounters a variable `linearSpeedXMps`, it finds the closest match in the name assumptions table, specifically `speed`, and adds a constraint that expresses the belief that `linearSpeedXMps`'s physical unit type is *meter-per-second* ($\mathrm{m\,s^{-1}}$). More formally, Phys finds the probability P_{name} equal to the highest scoring match for variable, var, in the assumptions list, A, according to the similarity metric:

$$P_{name} = \max_{s \in A} \frac{len(LCS(var, s, k))}{MAX\_LEN\_SUFFIX} \tag{7.1}$$

Where $len(LCS(var, s, k))$ is the length of a longest common substring (LCS) between var and substring s, and k is the minimal length of match we allow ($k = 3$, determined empirically). If the variable length is less than k, then we assume the name contains no evidence for any physical unit type. Then the quantity $len(LCS(var, s, k))$ is divided by the length of the longest entry in the assumptions table, $MAX\_LEN\_SUFFIX = 12$. For variable `linearSpeedXMps` and name assumption `speed`, the score would be $5/12$, or 0.42.

This value, $P_{name} = 0.42$ becomes a probabilistic name constraint that says `linearSpeedXMps`'s physical unit type is *meter-per-second* with likelihood:

$$P(var, unit) = (0.5 + 0.5 * P_{name}) \tag{7.2}$$

Where Equation 7.2 simply normalizes the probability to a scale where 0 means 'absolutely false', 1 means 'absolutely true,' and 0.5 means 'neutral.' This results in name constraint of $P($`linearSpeedXMps`, *meter-per-second*$= 0.71$ Even if the name is a perfect match to a name in the assumptions table, we still assign a maximum confidence value of $P(var, unit) = 0.7$, a heuristic adopted by our co-authors in previous work [111]. This is because naming constraints are the least reliable evidence for physical units types when compared with constraints derived from flow and code operations. Intuitively, if we assumed perfect confidence in a name, with $P_{name} = 1.0$, then there is no room for doubt, and *we want some doubt* because sometimes variable names are wrong.

7.2.1.2 Dataflow

Phys generates Dataflow constraints based on assignment statements, such as `x=y`. In this case, Phys adds a dataflow constraint that says *'the physical unit type of x should be the same as the physical unit type of y.'* More formally:

$$P(y, unit) \overset{0.95}{\longleftrightarrow} P(x, unit) \tag{7.3}$$

Under the hood, a dataflow assignment would be initialized with a confidence of $P_{name} = 0.95$, because probabilistic reasoning algorithms (like the Sum-Product algorithm [155] that we use for Phys) work better when they have some small margin of uncertainty [154, 111]. Note that the implication goes in both directions, and we use the heuristic of 0.95 to indicate a strong belief.

Phys also adds dataflow constraints as a result of addition, subtraction, comparison, `min()`, and `max()`, because these code operations are evidence that the operands have the same physical unit type, or else they must be dimensional inconsistencies.

7.2.1.3 Computed Unit

Phys adds computed unit constraints based on mathematical (or 'computed') code operations. Phys encodes how multiplying and dividing quantities with known physical unit types results in a new unit based on the outcome of the computation.

For example, if `x=y*z`, then the physical unit type of `x` must be the product of the units of `y` and `z`. Phys adds a constraint expressing the belief that a variable, *var* has the computed unit, *cu*:

$$\Big(P(y = unit_1, z = unit_2) \xrightarrow{0.95} P(var, cu)\Big) \Leftrightarrow (unit_1\{*, \div\}unit_2 = cu) \tag{7.4}$$

Computed units are also used by Phys to express the result of known mathematical operations, such as `sqrt(`var`)` and `pow(`var`, `exp`)`. If we know the physical unit type of the argument *var* to `sqrt`, then we know the resulting physical unit type, and likewise, if we know the physical unit type for *var* and the exponent *exp*.

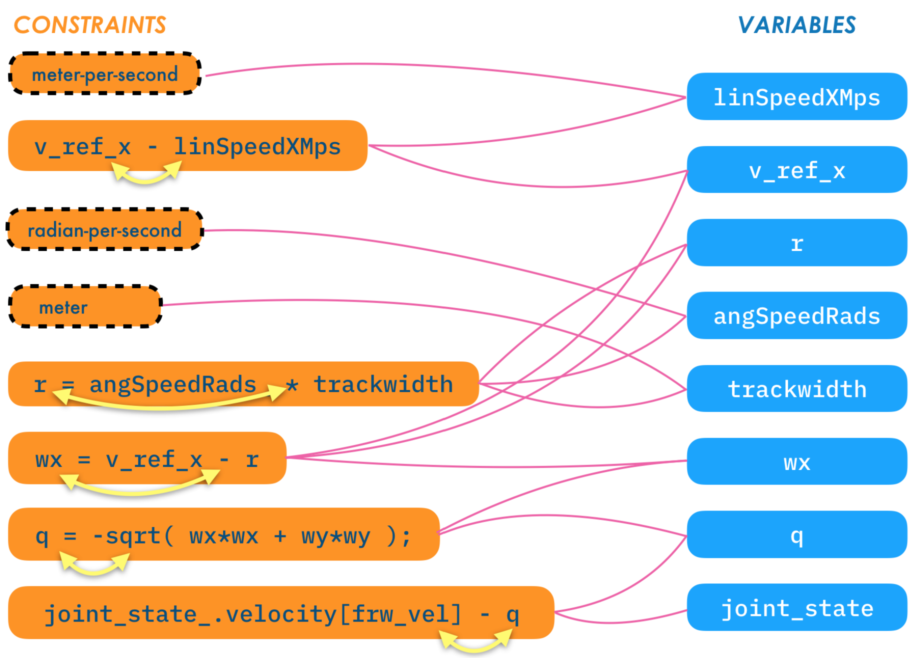


Figure 7.3: Factor graph constructed by Phys for the probabilistic constraints and variables detected in the code shown in Figure 7.1. Dotted boundaries on nodes indicate a Name Constraint.

7.2.2 Building the Factor Graph

Phys collects evidence, or beliefs, from all over the program in the form of probabilistic constraints. Phys combines these beliefs into a graph, called a factor graph. A factor graph is also a bipartite graph, meaning all the node in the graph belong to one of two sets, and edges are only allowed between nodes in different sets [156]. Let our two sets be called *CONSTRAINTS* and *VARS*.

We construct the graph as follows: For each probabilistic constraint *con*, add a node to set *CONSTRAINTS*. For each variable *var* in the program, add a node to set *VARS*. Then if a probabilistic constraint *con* has evidence about a variable *var*, add an edge between the node for *con* and the node for *var*.

For example, consider the code in Figure 7.1. The corresponding factor graph that Phys constructs is shown in Figure 7.3. As shown in the figure, the left-hand side shows the set of nodes *CONSTRAINTS* formed by the probabilistic constraints from the code in Figure 7.1. In Figure 7.1, line 405, the code has a variable `linearSpeedXMps`. Phys adds a name constraint (see § 7.2.1.1) for this variable because `linearSpeedXMps` contains the substring 'speed' that likely means *meter-per-second* from the assumptions table (see Appendix F). The code on line 405 also has the expression `v_ref_x_ - linearSpeedXMps`. For this expression, Phys adds a dataflow constraint because `v_ref_x_` and `linearSpeedXMps` are operands of a subtraction operation, as shown near the top left of Figure 7.3.

Overall, Figure 7.3 shows how nodes in the *CONSTRAINTS* set connect to their corresponding node in the *VARIABLES* set.

Continuing with the approach overview in Figure 7.2, the factor graph is input to the *Belief Propagation Engine*. The belief propagation engine runs the Sum-Product [155] algorithm on the factor graph that calculates, for each variable, a distribution over the set of possible unit types. Note that the distribution over the set of units is uncertain and represents a consideration of all available evidence in the program. Further, Sum-Product finds the most likely units for all variables in the program when considering all physical unit type assignments collectively.

If the most likely variable is above a certain likelihood threshold (we use 0.6), then Phys applies the most likely unit as shown by the 'Apply Units' of Stage 1 shown in Figure 7.2. Phys then uses these newly inferred units and runs the whole visitor pattern (see § 5.3) traversal of the program again, trying to leverage the previously inferred physical unit types to infer new physical unit types.

Once a fixed-point or a bounded number of iterations (four) is complete, Phys moves to Stage 2, as shown in Figure 7.2. We chose four iterations based on observations of 30 sample programs, in which no program took more than four iterations to converge. Other datasets might require more iterations or we could change Phys to warn developers when Phys bounds the number of iterations. For additional examples and details about how to convert probabilistic constraints into factors graphs, please see [10].

7.2.3 Stage 2: Unit-inconsistency Detection.

The dimensional inconsistency detector scans the annotated abstract syntax tree (AST) for dimensional inconsistencies in the same way Phriky does (see § 2.2), seeking inconsistent addition/subtraction, comparison, or assignment. Phys then outputs a list of inconsistencies and optionally, a list of physical unit type assignments to variables.

7.2.4 Complexity and Termination of Phys

Preprocessing builds a context-insensitive call graph, and topologically sorting this graph is $O(|V| + |E|)$, worst case $O(|E^2|)$ when removing cycles. Collecting probabilistic constraints involves at most h loops over each statement where h is the height of the statement's AST. The probabilistic inference engine implements an approximate solution to the sum-product message passing algorithm [157] that is quadratic. Collecting probabilistic constraints and the sum-product run within a loop bounded by a constant (four times). After the loop, detecting inconsistencies involves a linear scan of program variables and the program's AST. Overall, complexity is quadratic in time and space. This approach terminates because we bound the loops to collect probabilistic constraints and run the sum-product algorithm.

7.3 Implementation

Phys uses the same third-party software as Phriky (see § 5.3) plus these additional tools: `NLTK` [158] is used to parse identifier strings into smaller units and to identify parts-of-speech, and `libDAI` [159] is used as the probabilistic inference engine.

Our code is available at `https://unl-nimbus-lab.github.io/phys/`.

Our implementation uses many 'magic parameters', as mentioned in § 7.2.1. To find these values, we explored a range of values and determined empirically that we had the best results for detecting dimensional inconsistencies when the evidence from variable names is the weakest evidence. We tested our parameters for variable names and a balance between 'dialing up' the confidence in names to infer more physical unit types and 'dialing down' the confidence to avoid false positive dimensional inconsistencies. We address the threats caused by 'magic parameters' in § 7.5.

7.4 Evaluation of Phys

Our evaluation of Phys asks two questions:

- **RQ_{11}.** How effectively can Phys infer physical unit types for variables compared to Phriky?
- **RQ_{12}.** How well can Phys detect dimensional inconsistencies?

We evaluate Phys on a set of ROS `C++` files selected randomly from a ROS-based project available on GitHub. These files were not used during the evaluation of Phriky in § 5.5 but were included in our large-scale analysis of 5.9 *MLOC* in § 6. We use 30 files to answer RQ_{11} because of the manual annotation effort, described shortly below, and 60 files for RQ_{12}, dimensional inconsistency detection.

```
<physical_unit_annotation file="labbot_teleoperation_twist.cpp" linenr="35" id="0x11" name="nh"
    units="0,0,0,0,0,0,0,0,0,0,0" isVar="true" isConstant="false" varId="0x7fe2b148c5d0" hasUnits="false" />
<physical_unit_annotation file="labbot_teleoperation_twist.cpp" linenr="42" id="0x2" name="x"
    units="1,-1,0,0,0,0,0,0,0,0,0" isVar="true" isConstant="false" varId="0x7fe2b148c870" hasUnits="true" />
```

Figure 7.4: XML encoding of manual physical unit type annotations using during evaluation.

7.4.1 Results of Comparison of Physical Unit Type Inference in Phriky vs Phys

The robotics programs in our experiments are devoid of physical unit information. We begin by manually annotating every program variable in these 30 programs, and use this as ground truth. Overall we annotated 924 variables in these 30 programs, taking approximately two days. To help ensure that the manual annotations are correct, at least two authors from [10] reviewed each type annotation independently. We only consider variables that might have physical units, because some variables do not represent any physical quantity, e.g., a `for` loop index variable.

Additionally, we assume that `integer` variables are dimensionless. The process of annotation involved using Cppcheck as a preprocessor to identifier all the variables. Then for every `double` or `float` variable in the list, we examined evidence such as the variable name, code operations involving that variable, the surrounding context, and interactions between variables. For each variable, we determined whether it had any physical unit type (whether it belongs to the physical units type domain) and if so, what units it had.

As shown in Figure 7.4, we implemented the compact vector representation of physical units proposed in [49] and encoded it in an XML file. The figure shows two lines from a larger file, representing the manual physical unit type annotations for two variables. The first line is for a variable `nh` of type `ros::NodeHandler`

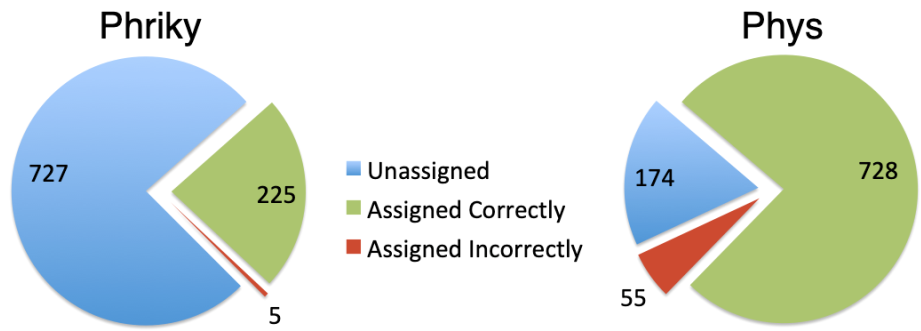


Figure 7.5: Comparison of Phys and Phriky's ability to infer and assign physical units for variables in 30 sample programs.

that does not have units. The second line is for a variable `x` of type `float` that has units *meter-per-second*. Notice that *meter-per-second* is encoded as `units="1,-1,0,0,0,0,0,0,0,0"`, where each number represents an exponent for each of the seven base units plus *radians*, *degrees_360*, and *quaternions*. After making this structured encoding, we modified Phys and Phriky to read these physical unit type annotations. This allowed us to automate the process of counting how many variables are assigned physical unit types by Phriky and Phys.

Phys reports a ranked list of units for each variable, but for this experiment, we consider only the top unit. Variables that are ROS message types from the mapping 7.2 are not included in the evaluation, because we just want to focus on quantifying the improvement of Phys over Phriky.

To address RQ_{11}, figure 7.5 shows that Phys assigns physical units to 82% (783/957) of variables in this dataset as compared to 24% (230/957) for Phriky. Of the 783 variables Phys assigns types to, it assigns type correctly to 93% of variables. Phriky assigns the correct type to 98% of variables.

To answer RQ_{12}, we evaluate the ability of Phys to detect high-confidence dimensional inconsistencies. We consider an inconsistency to be high-confidence

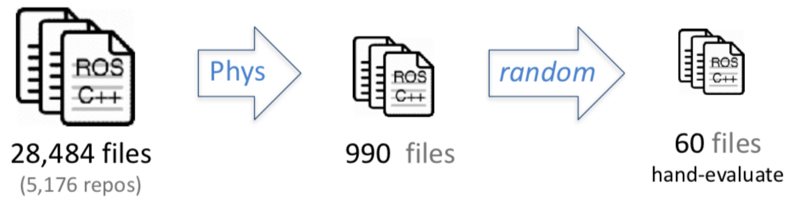


Figure 7.6: Source of files used to evaluate Phys.

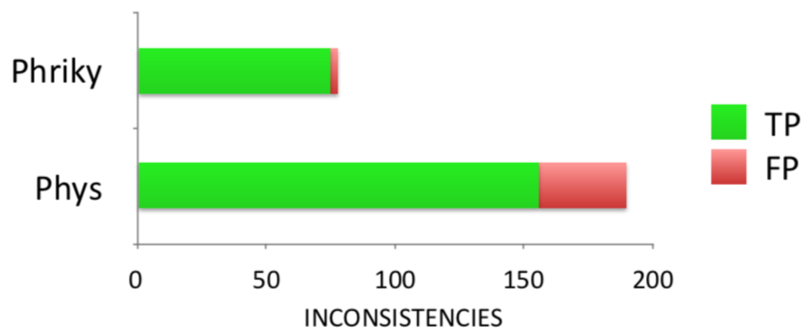


Figure 7.7: Comparison of Phriky and Phys ability to detect dimensional inconsistencies.

only if all physical units in the inconsistent expression are known, meaning no unknown units for variables or constants, the same as for Phriky (see § 5.2).

7.4.1.1 Experiment Setup.

We compute the TP rate of the reported inconsistencies for both Phys and Phriky, using a set of 60 randomly selected files. We use a different set of 60 files from the 30 files used in § 7.4.1 because those 30 files were used for testing during development. An overview of the file selection process is shown in Figure 7.6. As shown in the figure, we run Phys on 28,484 ROS-based projects available on GitHub. Phys reports inconsistencies in 990 files (≈ 3.5% of files with units as reported by Phriky). We then randomly selected 60 files for which inconsistencies were reported by Phys to form the evaluation set.

7.4.2 RQ_{12} Results: Phys Detected Inconsistencies.

Figure 7.7 shows the summarized results, with Phys having a TP rate of 82% on this dataset of 60 files. Phys detects 103.3% more inconsistencies compared to Phriky, including every inconsistency that Phriky detects. This makes sense because Phys is only adding information to what Phriky can already infer. Phys finds 156 true positive inconsistencies in 45 files, whereas Phriky finds only 75 in 24 files. Phys has 28 FP on this dataset, significantly more than Phriky's 7. This might be because Phys is parameterized to detect inconsistencies by making a trade-off between type inference power and inconsistency detection. By allowing Phys to infer more types, we can detect more inconsistencies but at the cost of more false positives. Overall, Phys overcomes limitations of Phriky and opens the door to future analyses that utilize evidence from even more uncertain sources, such as code comments or deep learning of type patterns.

7.5 Threats and Limitations

7.5.1 Self-labeling.

We self-label both variable physical unit types and TP or FP for inconsistencies. We mitigate this threat for type annotations by having multiple authors review the type assignments and also used Phys to show inconsistencies when a physical unit type needed correction. As in § 5.6, to mitigate this threat with inconsistencies, the authors evaluated inconsistencies independently and compared results.

7.5.2 Overfitting.

We assume English for variable names. Our substring matching assumes 'speed' could mean either linear or angular velocity (different abstract types), hence there is a threat of overfitting. We mitigate this threat by using a small list (41 entries).

7.5.3 Predefined Confidence Values and 'Magic Parameters.'

We use three predefined confidence values for the constraints collected in Stage 1 of Figure 7.2. We tested a range of values experimentally and found that the results are not particularly sensitive to the values, except for the name constraints. The confidence value for the name constraint seems to be a 'dial' that increases the number of variables that are assigned a physical unit type, but when 'dialed' too high, can cause an excessive number of false positive dimensional inconsistencies. We determined a confidence value for names empirically by examining Phys' results on a randomly selected set of files not used during the rest of the evaluation.

7.5.4 False Negatives Limitation.

As in § 5.6, the number of false negatives in the dataset is unknown, so we cannot calculate recall. To address this limitation, we will examine evaluating the approach after seeding faults (see § 8.2.2).

7.5.5 Generality Limitation.

Like with Phriky, this approach relies on having some initial abstract type information for physical units, in our case the ROS shared message libraries. However, this approach could also leverage type information from developer type annotations, even if the developer only provides type annotations for some variables. While our evaluation focuses on ROS `C++` software for impact, the technique generalizes for other robotic systems.

7.6 Open Dataset of Physical Inconsistencies

To help software researchers better study and understand dimensional inconsistencies, we created the first publicly available dataset of inconsistencies. This dataset contains 108 files and was published as part of [10] and is available at `https://doi.org/10.5281/zenodo.1310129`. For each file, the dataset includes the source, a link to the original GitHub repository including the line number containing an inconsistency, and our classification of inconsistencies as TP or FP.

The code artifacts in the dataset represent a wide variety of robotic applications, including but not limited to autonomous car navigation, quadrotor simulators, path planning, odometry, motor controllers, hardware interfaces, and teleoperation.

7.7 Discussion

7.7.1 Comparison of Phys and Phriky

The goal of this section is to highlight the improvement Phys makes over Phriky and to show where there is room for improvement, especially if Phys could work with developers by making physical unit type suggestions. As shown in § 4.3.3, suggestions have a significant impact on a developer's ability to make type annotations correctly. Moreover, Phys makes multiple suggestions, ordered by confidence, of which we consider only the top three to be consistent with our study (see § 4).

Table 7.1 shows the type predictions made by PHRIKY and PHYS for the 20 variables in the questions of our developer study (§ 4.2.2). The tool PHRIKY makes predictions for 10/20 (50%) of the variables, but only 6/10 (60%) suggestions are correct, whereas PHYS makes predictions for 15/20 (75%) variables and gets 11/15 (73%) correct as the first guess (highest confidence), 13/15 (87%) correct in the first or second guess, and 2/15 (13%) completely wrong. The results on this dataset contain several variables with *radians*, which can be hard for PHRIKY and might explain its weaker performance.

In general, these tools make errors for the following reasons: failing to account for the surrounding context, failing to consider the possibility that a variable might be dimensionless (like `ratio_to_consume` in Table 7.1), and missing domain-specific nuances in the identifiers. As shown in Table 7.1, PHRIKY guesses incorrectly about `robotSpeed.angular.z` which from the surrounding context in Q_5 is about *angular* rotation and not *linear* rotation as PHRIKY supposed. Some variables, like `ratio_to_consume` are not in the type domain (Table 7.1, Q_3) but PHYS believes it is (it has no units). Further, some variable names like `w` (Q_6) seem to have very little semantic information, but within the robotics software domain, `w` is used to represent the similar looking ω (omega), which often means *angular velocity* [160]. PHYS gets right all the variables that PHRIKY does, plus several like `delta_x` and `xi`. However PHYS, like PHRIKY, struggles when variable names have little semantic information, like `x2`, `w`, `av`, and `x`. Both of the suggestions PHYS gets right on the 2nd guess, `motor_.voltage[1]` and `dyaw`, indicate that PHYS is on the right track and shows promise as a type annotation assistant.

Q#	DIFFICULTY	VARIABLE NAME	CORRECT TYPE	SUGGESTIONS PHRIKY	PHYS 1^{st}	PHYS 2^{nd}	PHYS 3^{rd}
12	EASY	`pose.orientation`	q	✓	✓		
9		`delta_d`	m	✓	✓		
5		`robotSpeed.angular.z`	$\mathrm{rad\,s^{-1}}$	✗	✓		
15		`x2`	$\mathrm{m^2}$				
4	MEDIUM	`delta_x`	m		✓		
6		`w`	$\mathrm{rad\,s^{-1}}$				
16		`av`	$\mathrm{rad\,s^{-1}}$				
8		`path_move_tol_`	m		✓		
2		`springConstant`	$\mathrm{kg\,s^{-2}}$		✗	✗	✗
3		`ratio_to_consume`	NO UNITS		✗	✗	✗
7		`x`	NO UNITS				
10		`wrench_out.wrench.force.y`	$\mathrm{kg\,m\,s^{-2}}$	✓	✓		
11		`data->gyro_z;`	$\mathrm{m\,s^{-2}}$	✓	✓		
14		`xi`	m		✓		
18		`motor_.voltage[1]`	$\mathrm{kg\,m\,s^{-3}\,A^{-1}}$	✗	✗	✓	
1	HARD	`return`	m				
13		`angular_velocity_covariance`	$\mathrm{rad\,s^{-2}}$	✗	✓		
17		`torque`	$\mathrm{kg\,m^2\,s^{-2}}$	✓	✓		
19		`anglesmsg.z`	rad	✓	✓		
20		`dyaw`	rad	✗	✗	✓	

Table 7.1: Correct types for each question compared to PHRIKY and PHYS unit annotations. Ordered by question difficulty. The original questions are in Appendix D.

7.7.2 Implications for Future Tool Developers

Future tool developers should consider the following implications:

- Determining if a variable needs a type annotation is valuable because developers struggle to identify which variables belong in the type domain.
- Finding an order for annotating variables can be valuable because assigning a type for some variables transitively implies the type of others.
- Figure 4.6 shows that correct suggestions improve the accuracy of HARD type annotations the most, and therefore developers could maximize the impact of their tools by focusing on potentially HARD variables. As mentioned in § 4.5,

a possible way to determine the difficulty could be the number of variables involved.

- A tool that suggests types might simultaneously suggest an improved variable name.

7.8 Possible Extensions to Phys

In this section, we discuss several possible extensions to Phys. The content presented in this section was not previously published in [10]. The first extension enables Phys to become a physical unit type annotation assistant. The second extension makes Phys compatible with other static analysis frameworks, especially for the ROS development community. The third extension would enable Phys to suggest improved variable names. The fourth extension would expand Phys's analysis to include units-of-measure [70] and real-world types [52].

7.8.1 Extending Phys: Towards a Type Annotation Tool

The overall vision for a physical unit annotation tool is within an IDE. Phys is currently a command-line-interface tool, but research suggests [114] that IDE-based annotation assistants help developers more than batch processing tools. We imagine that the IDE tool would provide visual cues, like highlights, that could alert developers to untyped identifiers and potential inconsistencies. The IDE plugin could build on the strengths of the Checker Framework [115] that shows a visual representation of speculative consequences [161] for the current type assignments.

The vision for these extensions is to enable Phys to work together with developers to apply supplemental physical type annotations, only for those variables that Phys cannot infer with high confidence since research shows that tools and developers together outperform tools alone [114]. Our proposed extensions have two parts: 1) a physical unit annotation format that specifies the type annotations developers can add to program comments; and 2) a method to order annotation worklists that takes into consideration the cost of interrupting a developer's current context [162, 163, 164].

7.8.1.1 Annotation Tool Consideration 1: Comments & Format

There are many ways to associate an identifier with its physical unit type [54, 55, 68, 165], including type declarations from special class libraries or language extensions. We adopt the method of Hills, Feng, and Roşu [61] who proposed including physical unit type annotations in comments. As discussed in § 3.2.3, their work used a rewriting engine to modify the code into typed code, whereas in our proposed method the comments are read directly by a preprocessor as an input to Phys.

Putting supplemental type annotations in comments is appealing for several reasons: 1) it imposes no toolchain burden; 2) the annotated code remains completely portable to contexts that do not use the type annotations. Putting type annotations in comments would be familiar to users of Microsoft's `TypeScript` [166]. But using comments for type annotations is not without potential downsides, because type annotations could displace useful developer insights at that comment location. However, avoiding the toolchain burden is appealingly aligned with ROS's philosophy of least constraints (*"ROS is designed to be as thin as possible"* [167]). We only intend to add type annotations in code for those variables whose physical unit type cannot otherwise be inferred.

To read comments, either PHYS or CPPCHECK would have to be significantly modified. PHYS uses CPPCHECK as it preprocessor, and CPPCHECK does not include program comments in its 'dump' output that PHYS takes as input. Modifying the open-source CPPCHECK, although technically possible, creates an undesirable design dependency on the modified version. Otherwise, PHYS would have to be modified to use another preprocessor.

Retooling PHYS with a new preprocessor like CLANG would allow PHYS to read comments during analysis. Further, CLANG is part of a powerful analysis framework (LLVM). However, CPPCHECK fails gracefully in the presence of an incomplete compilation unit whereas CLANG does not. In either case, the physical unit type annotations contained in comments would have to follow a standard, machine-readable format. Therefore, we present our annotation format as a context-free grammar in Backus-Naur form [168]:

$$\begin{aligned} \langle annotation \rangle &::= \langle u \rangle \mid dimensionless \mid blended \\ \langle u \rangle &::= \langle u \rangle \, \langle u \rangle \mid \langle base\_unit \rangle \, \langle power \rangle \mid per \; \langle base\_unit \rangle \, \langle power \rangle \\ \langle base\_unit \rangle &::= kilogram|meter|second|mole|ampere|kelvin|candela| \\ &\quad\; radian|degree|quaternion \\ \langle power \rangle &::= squared \mid cubed \mid tothe \; (i \in \mathbb{N}) \mid \text{``''} \end{aligned} \tag{7.5}$$

As shown in Equation 7.5, this grammar creates strings that are physical unit annotations. This grammar builds on ideas from the LaTeX package `siunitx` [169], specifically the '*squared*', '*cubed*', and '*tothe*' ('to the') literals for exponents. The package `siunitx` is used to format quantities in SI units in text documents.

Notice that the grammar in Equation 7.5 is similar to the 'unit types' grammar in Equation 2.1, that likewise expresses the space of possible unit types. Like the grammar in Equation 2.1, the grammar in Equation 7.5 generates equivalent strings,[1] for example:

$$\textit{meter per second} \;\equiv\; \textit{per second meter} \tag{7.6}$$

Both of these strings represent the same physical phenomena because the strings in Equation 7.6 represent the mathematical expression ms^{-1} and $s^{-1}m$, respectively. Since the terms are multiplied, the commutativity property ensures that any ordering of the base terms is equivalent. We adopt the convention that all '*per*' terms shall be placed to the right so that *meter per second* is preferred to the equivalent *per second meter*.

By convention, we adopt the shortest form of the equation. For example, these strings both represent a distance:

$$\textit{meter} \equiv \textit{meter meter per meter} \tag{7.7}$$

On the right-hand side, '*meter per meter*' would divide out, leaving only '*meter*'. Although equivalent, we prefer the simplest, shortest form.

These annotations can be used to specify the physical unit type at variable declaration:

```
double accel_threshold; //phys: meter per second squared
```

[1]The language in Equation 7.5 has the property that any string has an infinite number of distinct equivalent strings, but in practice this number is finite. Bruce Hamilton (Hewlett-Packard Laboratories) [170] observed that the exponent for any useful physical unit is between -7 and $+7$, inclusive.

By annotating types at declaration, instead of at every use, we leverage the ability of automated software tools to propagate units with flow. Notice the `phys:` prepended to the type annotation. We suggest using a sentinel to notify PHYS that the comment contains a physical unit type annotation. The eagle-eyed reader will note that some languages (including `C++`) allow developers to declare multiple variables in a single line. We realize that it might cause an inconvenience, but for now, we would require one variable declaration per line so that each variable has into own type annotation (only if the type cannot otherwise be inferred). Declaring one variable per line might help with program readability as well.

Even if an identifier does not have units, our format enables developers to specify that no units are present:

```
float scaling_factor = 0.42; //phys: dimensionless
```

Note that this example also shows a constant, 0.42, and that adding physical type annotation could be extended to constants as well.

Further, developers occasionally intentionally blend units (see the code in Figure 2.3 in § 2.2.3) and the '*blended*' annotation empowers developers to declare that a dimensional inconsistency on the current line is intensional.

Optionally, the grammar in Equation 7.5 could include common names like *force* or abbreviations such as F for *force* ($\mathrm{kg\,m\,s^{-2}}$) and Hz for *frequency* ($\mathrm{s^{-1}}$).

7.8.1.2 Annotation Tool Consideration 2: Worklist Ordering

The annotation task in Definition 1 (§ 2.2) specifies *what* developers must do but not *how* they should do it. This design consideration is motivated by Orchard *et al.* [72] who observed that nearly 80% of type annotations in `Fortran` programs could be inferred given a 'critical subset' of manually annotated variables (see

§ 3.2.3.2). Manually annotating this critical set maximally amplifies the information provided by each developer annotation.

Assuming there are multiple program variables that lack type information, then the question arises: how best to order the type annotation worklist to maximize annotation accuracy? Every identifier in the critical set 'unlocks' the understanding of some other variable. We could rank the unannotated variables based on the number of other variables they unlock, plus the number of variables *those* variables unlock. This metric alone would allow us to rank potential annotations greedily. However, this greedy ordering ignores the human part of the 'automated tool/human developer' team.

We propose an annotation ordering that takes both information gain and human factors into account:

- Maximum information gained from each annotation, denoted as **I**. The information gain **I** is defined as an annotation that maximally reduces the number of untyped variables.
- The cost of switching the developer's current context, denoted as **C**. The context-change penalty **C** is defined as the distance between the current annotation scope and the new annotation scope.

Combining **I** and **C** gives us:

$$\arg\max x \left(\alpha \mathbf{I_x} - \beta \mathbf{C_x}\right) \tag{7.8}$$

Where $\mathbf{I_x}$ is the total transitive information gained for an identifier x, $\mathbf{C_x}$ is the penalty for switching developer context, and α and β are tuning parameters that might be left to the preference of the developer. The next annotation in this ideal worklist is the one with the highest score in the formula: The cost of calculating **I** and **C** depends on the precision of the static analysis used (i.e. alias analysis is generally expensive, but increases precision).

Additionally, Equation 7.8 could be modified to account for the predicted difficulty based on code attributes like those identified in § 4.5 by adding an 'expected difficulty' term. Ordering the annotations by expected difficulty would allow developers additional worklist flexibility.

Extending Phys to be an annotation assistant would likely require overcoming additional technical challenges not addressed here, like determining *what parts* of Phys' analysis should be run based on the current changeset, and *how often* Phys should run the full analysis. Additionally, these proposed extensions would have to be evaluated empirically to determine if the tool is useful in helping developers assign type annotations quickly and accurately. We leave these and other non-trivial implementation details for future work, and here propose two initial parts of the overall design, the annotation format, and the worklist ordering. Together, the two proposed Phys extensions lay a foundation for a future physical unit annotation tool.

7.8.2 Extending Phys: Compatibility with Existing Analysis Frameworks

Currently, Phys requires the developer to identify target ROS files individually and the scope of the analysis is only within a complication unit. The ROS analysis tool Haros (see § 2.4), by contrast, identifies how information flows between separate compilation units, determining statically the ROS graph through which information flows in messages (also see § 2.4).

By extending Phys to consider a set of interrelated files, Phys could potentially leverage information in one compilation unit to inform the analysis in another. This is useful in several ways. Firstly, for custom messages, where the attributes of the shared data structure are not defined in ROS shared libraries, and therefore, their physical units must be inferred or predicted. Evidence in one compilation unit might be leveraged in another. Secondly, if Phys was informed by how information flows between these separate files, it could use units associated with an attribute of a message data structure in one file and use those units is code in another file that reads that same kind of message.

Modifying Phys to make it compatible with Haros entails creating an interface on Phys. This interface would return Phys inconsistency messages in a manner than Haros expects. Currently, Haros runs plugins on each file of a launch file individually, but the authors of Haros have expressed intent to make all files available simultaneously to plugins. The larger change is modifying Phys to reason across compilation units. This would likely entail a two-step process. The first step finds evidence from each complication unit and determines what evidence generalizes to other programs, for example, custom message data structures. The second step would be running Phys with this added evidence and, when new evidence about a shared data-structure is inferred, re-running Phys on other files that use the shared data structure.

7.8.3 Extending Phys: Suggesting Improved Variable Names

Another future direction is to help developers improve the quality of identifiers with respect to a type system. The semantic-based technique we explored with Phys (§ 7) measures how closely an identifier matches pre-defined substrings (see

Appendix F). This method could be extended to find identifiers whose physical unit types can be strongly inferred by flow or context, but whose variable name has a low score for the inferred type. This indicates an identifier name that could be improved. Such a tool might both find identifiers and suggest improvements to the current identifier that specify the abstract type.

7.8.4 Extending Phys: Beyond Dimensional Analysis with Units-of-Measure and Real-World Types

Handing scaled versions of SI units, such as *kilo*-meters and *centi*-meters would make the analysis more power and could catch inconsistencies that have the same dimension (*length*) but are different units-of-measure. This problem of scaling quantities within the SI system is nearly equivalent to using units-of-measure from other systems, such as Imperial units. In both cases, the units are dimensionally equivalent but incompatible because of a scaling factor. The non-trivial complexity of addressing units-of-measure is discussed in [63], and before devoting significant effort, it might be worth conducting an empirical study of a large-scale corpus to estimate how often other units are used and therefore how impactful an automated solution might be.

We are also inspired by the work of Xiang, Sullivan, and Knight [52] who widen the ideas of dimensional analysis to include closer semantic ties to the real world, with rules that are a superset of dimensional analysis, such as *'a magnetic heading cannot be used in an expression with a true heading.'* One key difference with their work is that we seek to automatically infer additional, real-world types, whereas Xiang *et al.* depend on manual developer annotations.

Summary

In this chapter, we presented a method of inferring physical units that leverages evidence in both shared component interfaces and identifier names. We further demonstrated that this method, implemented in a tool Phys, assigns physical units to $\approx$ 75% of program variables as opposed to only $\approx$ 24% for Phriky. We also identified possible extensions to Phys, such as an annotation ordering and format, that could be the foundation for a physical unit annotation tool.

Next, we discuss the overall contributions of this work and identify future work.

8 Conclusions and Future Directions

In this chapter, we identify, summarize, and itemize the contributions of the overall work. Next, we describe several promising opportunities for future research. Finally, we conclude by commenting on the work as a whole.

8.1 Contributions

This work presents several major contributions:

First, in § 4, we presented a study of developers showing that they correctly annotate variables with physical unit types only 51% of the time and require two minutes to make a single correct annotation. We also empirically determined that three physical unit type suggestions are better than one. This is because three helps nearly as much as a single correct suggestion, but that three suggestions with none correct hurts developers' accuracy less than a single incorrect suggestion.

Second, in § 5, we showed a method to automatically infer physical units for ROS variables and detect dimensional inconsistencies. It further showed an implementation of this method in an open-source tool Phriky, and an evaluation of Phriky showing an 87% True Positive (TP) rate in 231 open-source systems.

Third, in § 6, we described a large-scale empirical study of Phriky on a corpus of 5.9 *M* lines of open-source robot software to determine how frequently dimensional inconsistencies occur in this corpus and found at least 6% ($211/3,484$)

of repositories contain inconsistencies. We further identified the main kinds of dimensional inconsistencies, finding that the most common inconsistency is the misuse of data structures from the ROS message libraries.

Fourth, in § 7, we showed a new tool, Phys, that improved the detection power of Phriky by using additional evidence in variable names with probabilistic reasoning. We conducted an empirical study of Phys on 108 files and found: 1) Phys can infer units for 82% of variables; and, 2) Phys can detect dimensional inconsistencies with 82% accuracy.

Lastly, in § 7.6, we presented an open dataset of physical inconsistencies identified by the tool Phys. To our knowledge, this is the first open dataset of these kinds of inconsistencies.

8.2 Future Directions

8.2.1 Role of Context in Type Annotation Accuracy

Developers making physical unit type annotations consider multiple sources of evidence before assigning a type. The results of our research question RQ5 (see § 4.3.5) on why developers choose a particular type shows that variable names and reasoning about code operations are key sources of evidence. In our study, we showed subjects code artifacts with whole or truncated functions (see § 4.2.2.1) without giving subjects access to the surrounding context, such as the rest of the program or repository, or the target system for the software. Our study did not seek to address important questions about the role of context in assigning a type correctly.

Context matters during software development [171] because it more fully describes the information available to developers when they are reasoning about program types. A future developer study might take a broader view of the type annotation task, and have developers make several type annotations in the same program while showing, for example, a backwards data-dependency slice for an untyped variable or other usages of that variable in different parts of the program. Such a study might help future tool developers make better tools by revealing the contextual clues that maximally increase annotation accuracy or speed.

8.2.2 False Negatives and Seeding Faults

One way to evaluate the effectiveness of static analysis tools is using metrics of 'False Positives' (FP) and 'False Negatives' (FN). FPs and FNs correspond to Type I and Type II errors, respectively. The FP rate (FP_{rate}) is defined as:

$$FP_{\text{rate}} = \frac{\text{detected real faults}}{\text{\# of detected faults}} \tag{8.1}$$

In this work we evaluate the FP rate for both Phriky (§ 5.5) and Phys (§ 7.4) by evaluating detected inconsistencies by hand. However, we cannot determine the FN rate because this would require knowing all the inconsistencies in a given dataset. The FN rate (FN_{rate}) is defined as:

$$FN_{\text{rate}} = \frac{\text{detected faults}}{\text{\# of faults}} \tag{8.2}$$

To address the problem of unknown faults, software researchers proposed *fault seeding* [172, 173] to approximate the FN rate. In this technique, faults (in our case, dimensional inconsistencies) are intentionally introduced into programs. By counting the number of seeded inconsistencies that a method detects, we can estimate the FN rate. The FN rate is estimated by:

$$FN_{\text{rate}} \approx \frac{\text{detected seeded faults}}{\text{\# seeded faults}} \tag{8.3}$$

The challenge of introducing seeded inconsistencies is that the accuracy of this technique depends on seeding faults are representative of real faults. This requires tester skill, understanding, and experience with these kinds of faults.

Introducing realistic faults is challenging because examples of dimensional inconsistencies are poorly documented in the literature, and our open dataset of dimensional inconsistencies is the first of its kind, to our knowledge. However, our dataset of dimensional inconsistencies (See § 7.6) as well as all the inconsistencies identified in this work were discovered by running Phriky and Phys on open-source robot software. Therefore it is possible that there are types of dimensional inconsistencies lurking in programs that have escaped detection using our tools. These kinds of inconsistencies are therefore unlikely to be introduced by us during seeding, thereby inflating our estimate of FN_{rate}.

A more complete understanding of FNs in dimensional inconsistencies would be helpful in understanding the detection power of Phriky and Phys.

8.2.3 Exploring the Performance / Precision Trade-off

Phriky implements a lightweight analysis with an eye toward continuous integration (see § 5.2.3). Likewise Phys, which is built on Phriky, inherits many trade-offs that favor speed over precision. Even though our analysis is lightweight, both Phriky and Phys detect real dimensional inconsistencies with $> 80\%$ accuracy.

However, there might be deeper inconsistencies (dimensional or units-of-measure or real-world inconsistency, see § 7.8.4) that our analysis overlooks because we favor speed. A lightweight analysis is useful to detect and avoid shallower inconsistencies. A deeper, slower, and more precise analysis might occasionally be justified for increased assurance of physical systems that can have dangerous or expensive, real-world consequences. To make our analysis more precise, we could make it interprocedural, context-sensitive, path-sensitive, or by performing alias analysis (also called 'points-to' analysis).

Both interprocedural and context-sensitive analysis can be computationally expensive because they require constructing an interprocedural control flow graph and trigger a new analysis at every call point and during recursion or loops. They consider the dataflow into and out from a procedure at its call points, including the impact on the global program state. Anecdotally, we have yet to see any dimensional inconsistencies that would have been detected by either interprocedural or context-sensitive analysis. Further, we instrumented Phriky to detect procedures whose arguments have different physical unit types at different call points and found a few procedures such as a `sign(x)` function that returns whether `x` is positive or negative, or a `bound(x, max, min)` function that limits `x` to values within a specified minimum and maximum. Procedures like `sign()` and `bound()` take arguments with different physical unit types yet they are not dimensionally inconsistent. Finding one or more examples in the wild with an empirical study might motivate the time and effort to implement and perform interprocedural or context-sensitive analysis.

As yet, adding path-sensitivity does not appear promising because our current lightweight analysis over-approximates feasible program paths yet causes few false positives. If there were deeper inconsistencies along particular program paths, we believe they would also be contained within an over-approximation of those paths. Further, over-approximating paths might be an acceptable solution because it reveals cases where the same variable has different physical unit types along different paths, a kind of 'code smell' that might hinder code comprehension.

Alias analysis seems to be a promising way to detect deeper inconsistencies because alias analysis helps determine what complex variables, like pointers and complex data structures, may or must mean. In general, our analysis performs best when we know what all the variables mean. Since many ROS programs use `C++`, leveraging the alias analysis already built into the `LLVM` compiler infrastructure for `C++` might enable new versions of PHYS to efficiently explore performance/precision trade-offs.

Overall, PHRIKY and PHYS detect real inconsistencies even with a lightweight analysis, but increasing the precision of the analysis might reveal deeper inconsistencies.

8.2.4 Code-Aware Robot Simulation and Scenario Generation

High-resolution physical simulations provide essential and cost-effective validation of robotic systems. However, robotic simulation is intentionally and architecturally separate from the inner workings of the software that reads sensors and commands actuators. Recent standardizations in robot simulation tools (SDFormat) provide a data structure through which concerns in the simulation can be linked to concerns in code. For example, the code implies that system behavior depends on temperature, but the simulation does not model temperature. PHYS reveals the physical concerns present in code when it infers physical unit types. This connection between simulations and programs can enrich robot simulation by making it aware of what is happening in the code.

8.2.5 Connecting Programs to the Real World.

Program analysis relies on several powerful transformations of source code into abstractions: control flow graphs, dependency graphs, abstract syntax trees. These representations lift program analysis into abstract representations that model critical program properties, like domination and reachability. Recent work in robotics and artificial intelligence (AI) leverage *hypergraphs* that connect multiple levels of abstractions. For example, map coordinates (x_1, y_1) have an edge to a semantic graph node (*BlueChair*), and a trivial robot path can be represented by $[(x_1, y_1), (x_2, y_2), (x_3, y_3)]$ or $(BlueChair, YellowHallway, RedDoor)$. These kinds of abstractions have been fruitful for advances in AI, and this idea is to extend existing program analysis graphs with connections to semantic understandings of programs in the current context.

The idea is to leverage Phys to bridge the gap between entities in program analysis and entities in the real world. Knowing a variable's physical unit type helps ground its meaning in the real world. This might enable program analysis to reason about the interplay of variable values and the future state of the physical system, such as: *"the integrator term of the PID will not wind up beyond threshold X in region Y of the map given the current plan and state estimation, with confidence Z."* Our work here begins to bridge this gap by inferring physical unit types, and future work seeks to make far-reaching connections. Inferring the semantic meaning of code within the context of an environment at runtime might be a gateway to assuring critical safety properties of autonomous systems.

8.3 Conclusions

This work seeks to activate the power of type checking in untyped contexts. We focus our efforts in the robotics domain because reliable robot software is a key barrier to unlocking the tremendous potential of robotic systems.

We motivated the problem with real-world examples. We demonstrated in a user study of 83 subjects that developers struggle to assign types correctly. We then proposed a method to infer physical unit types without developer annotations and showed that this method detects real inconsistencies with a high TP rate (87%). Using an implementation of this method called Phriky, we conducted the first large-scale analysis of dimensional inconsistencies in open source robot software. We found inconsistencies in 6% of repositories.

Building on Phriky, we implemented a probabilistic method in a tool called Phys that uses uncertain evidence in identifiers together with evidence from middleware interfaces to dramatically increase (triple) the number variables for which we could infer physical unit types. We used Phys to create the first open dataset of dimensional inconsistencies.

These are vibrant and novel contributions in an increasingly important area. However, recent tragedies with the Boeing 737 MAX-8 [174] point to a failure in our ability to ensure system reliability in the complex interplay of humans, software, hardware, and the environment—even within the safety-critical domain of aviation with a rigorous safety assurance process. As more complex cyber-physical and robotic systems enter the mainstream, working ever more closely with human partners, the process of creating software will likely continue to exhibit a tension between prototyping with 'fast-and-loose' software and delivering high-assurance software. Efforts that start as prototypes but then transition eventually towards higher-assurance are acutely challenged and could benefit from automated software tools. As automated software development tools mature, future systems will become increasingly reliant on developers working with automated tools during all stages of a system's life cycle. Our work here is a small step toward realizing the vision of creating reliable robot software more easily, rapidly, and economically.

Bibliography

[1] SoftBank, "Romeo, the research robot from Aldebaran," 2016. (document), 1.1, 1

[2] BIPM, *Le Systme international d'units / The International System of Units ('The SI Brochure')*. Bureau international des poids et mesures, eighth ed., 2006. 1

[3] K. Damevski, "Expressing measurement units in interfaces for scientific component software," in *Proceedings of the 2009 Workshop on Component-Based High Performance Computing*, CBHPC '09, (New York, NY, USA), pp. 13:1–13:8, ACM, 2009. 1, 2.4, 3.1, 5.5.2.1

[4] A. G. Stephenson, D. R. Mulville, F. H. Bauer, G. A. Dukeman, P. Norvig, L. LaPiana, P. Rutledge, D. Folta, and R. Sackheim, "Mars climate orbiter mishap investigation board phase I report, 44 pp," *NASA, Washington, DC*, 1999. 1, 6

[5] G. H. Lockwood, *Final Report of the Board of Injury: Investigating the Circumstances of an Accident Involving the Air Canada Boeing 767 Aircraft C-GAUN that Effected an Emergency Landing at Gimli, Manitoba on the 23rd Day of July, 1983*. Government of Canada, 1985. 1

[6] C. Bergin and P. Harding, "Cygnus delays ISS berthing following GPS discrepancy NASASpaceFlight.com," Sept. 2013. 1

[7] M. Karr and D. B. Loveman, III, "Incorporation of Units into Programming Languages," *Commun. ACM*, vol. 21, pp. 385–391, May 1978. 1, 3.2.2

[8] V. J. Hellendoorn, C. Bird, E. T. Barr, and M. Allamanis, "Deep learning type inference," in Leavens *et al.* [175], pp. 152–162. 1, 3.2.4

[9] "Amazon Mechanical Turk (MTurk)," 2018. `https://www.mturk.com`, (accessed 24 April 2019). 1, 4.2.5.4

[10] S. Kate, J.-P. Ore, X. Zhang, S. Elbaum, and Z. Xu, "Phys: Probabilistic Physical Unit Assignment and Inconsistency Detection," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, (New York, NY, USA), pp. 563–573, ACM, 2018. 1.1, 1.2, 3.1, 7, 7.2.2, 7.4.1, 7.6, 7.8

[11] J.-P. Ore, S. Elbaum, C. Detweiler, and L. Karkazis, "Assessing the type annotation burden," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 190–201, ACM, 2018. 1.2, 4

[12] J.-P. Ore, C. Detweiler, and S. Elbaum, "Lightweight Detection of Physical Unit Inconsistencies Without Program Annotations," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2017, (New York, NY, USA), pp. 341–351, ACM, 2017. 1.2, 3.1, 5

[13] J.-P. Ore, C. Detweiler, and S. Elbaum, "Phriky-units: A Lightweight, Annotation-free Physical Unit Inconsistency Detection Tool," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2017, (New York, NY, USA), pp. 352–355, ACM, 2017. 1.2, 2.3, 3.1, 4.2.2.1, 4.2.5.1

[14] J. P. Ore, S. Elbaum, and C. Detweiler, "Dimensional inconsistencies in code and ROS messages: A study of 5.9m lines of code," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 712–718, Sept. 2017. 1.2, 3.2.3.1, 4.2.2.1, 6

[15] I. B. o. Weights, Measures, B. N. Taylor, and A. Thompson, "The International System Of Units (SI)," 2001. 2.1

[16] E. Allen, D. Chase, V. Luchangco, J.-W. Maessen, and G. L. Steele, Jr., "Object-oriented Units of Measurement," in *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '04, (New York, NY, USA), pp. 384–403, ACM, 2004. 2.1, 3.2.2, 3.2.3.1, 3.1

[17] I. Mills, B. N. Taylor, and A. Thor, "Definitions of the units radian, neper, bel and decibel," *Metrologia*, vol. 38, no. 4, p. 353, 2001. 2.1, 5.3

[18] J. Fourier, *Theorie analytique de la chaleur, par M. Fourier*. Chez Firmin Didot, pre et fils, 1822. 2.2

[19] P. W. Bridgman, *Dimensional Analysis*. Yale University Press, 1922. 2.2, 3.2.2

[20] O. S. R. Foundation, *ROS Enhancement Proposal 103*, July 2010 (accessed 27 July 2016). 2.2

[21] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation, Atlanta, Georgia, USA, May 1993*, pp. 802–807, IEEE Computer Society Press, 1993. 2.2.3

[22] R. Milner, “A Theory of Type Polymorphism in Programming,” *J. Comput. Syst. Sci.*, vol. 17, no. 3, pp. 348–375, 1978. 2.3

[23] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, and H. Das, “The CLARAty architecture for robotic autonomy,” in *Aerospace Conference, 2001, IEEE Proceedings.*, vol. 1, pp. 1–121, IEEE, 2001. 2.4

[24] G. Magyar, P. Sink, and Z. Krizsn, “Comparison study of robotic middleware for robotic applications,” in *Emergent Trends in Robotics and Intelligent Systems*, pp. 121–128, Springer, 2015. 2.4, 3.1

[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, vol. 3.2, p. 5, Kobe, Japan, 2009. 2.4, 4.2.2.1, 5.1, 6

[26] ROS Industrial Consortium, “Current Members - ROS Industrial,” 2016. 2.4

[27] Open Source Robotic Foundation, “Automated Driving with ROS at BMW,” 2016. 2.4

[28] O. S. R. Foundation, *ROS Enhancement Proposal 103*. July 2010. 2.4

[29] A. Santos, A. Cunha, N. Macedo, and C. Lourenço, “A framework for quality assessment of ROS repositories,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, pp. 4491–4496, IEEE, 2016. 2.4

[30] Open Robotics, “Open Robotics,” 2019. 2.4

[31] A. Santos, A. Cunha, and N. Macedo, “Static-time extraction and analysis of the ROS computation graph,” in *3rd IEEE International Conference on Robotic*

Computing, IRC 2019, Naples, Italy, February 25-27, 2019, pp. 62–69, IEEE, 2019. 2.4

[32] R. T. Vaughan, B. P. Gerkey, and A. Howard, "On device abstractions for portable, reusable robot code," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 3, pp. 2421–2427, IEEE, 2003. 3.1

[33] G. Biggs, *Designing an application-specific programming language for mobile robots*. PhD Thesis, ResearchSpace@ Auckland, 2007. 3.1, 3.1

[34] G. Walck, U. Cupcic, T. O. Duran, and V. Perdereau, "A case study of ROS software re-usability for dexterous in-hand manipulation," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, 2014. 3.1

[35] M. Y. Jung, M. Balicki, A. Deguet, R. H. Taylor, and P. Kazanzides, "Lessons learned from the development of component-based medical robot systems," *Journal of Software Engineering for Robotics*, vol. 5, no. 2, pp. 25–41, 2014. 3.1

[36] H. Bruyninckx, "Open robot control software: the OROCOS project," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, pp. 2523–2528, IEEE, 2001. 3.1, 5.2.2

[37] N. Ando, T. Suehiro, and T. Kotoku, "A software platform for component based RT-system development: OpenRTM-Aist," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pp. 87–98, Springer, 2008. 3.1, 5.2.2

[38] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010. 3.1, 5.2.2

[39] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet another robot platform," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, p. 8, 2006. 3.1, 5.2.2

[40] J. C. Reynolds, "Towards a theory of type structure," in *Programming Symposium, Proceedings Colloque sur la Programmation, Paris, France, April 9-11, 1974*, pp. 408–423, 1974. 3.2.1

[41] L. Cardelli, "Type Systems," *ACM Computuing Surveys*, vol. 28, no. 1, pp. 263–264, 1996. 3.2.1, 4.1

[42] B. C. Pierce, *Types and programming languages*. MIT Press, 2002. 3.2.1

[43] B. Ray, D. Posnett, P. T. Devanbu, and V. Filkov, "A large-scale study of programming languages and code quality in GitHub," *Commun. ACM*, vol. 60, no. 10, pp. 91–100, 2017. 3.2.1, 6.1.1

[44] L. Prechelt and W. F. Tichy, "A Controlled Experiment to Assess the Benefits of Procedure Argument Type Checking," *IEEE Trans. Software Eng.*, vol. 24, no. 4, pp. 302–312, 1998. 3.2.1

[45] S. Spiza and S. Hanenberg, "Type names without static type checking already improve the usability of APIs (as long as the type names are correct): an empirical study," in *13th International Conference on Modularity, MODULARITY '14, Lugano, Switzerland, April 22-26, 2014*, pp. 99–108, 2014. 3.2.1

[46] S. Hanenberg, S. Kleinschmager, R. Robbes, . Tanter, and A. Stefik, "An empirical study on the impact of static typing on software maintainability," *Empirical Software Engineering*, vol. 19, no. 5, pp. 1335–1382, 2014. 3.2.1

[47] E. Daka, J. M. Rojas, and G. Fraser, "Generating unit tests with descriptive names or: would you name your children thing1 and thing2?," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis, Santa Barbara, CA, USA, July 10 - 14, 2017*, pp. 57–67, 2017. 3.2.1, 4.3.5

[48] T. Cheatham, "Handling fractions and n-tuples in algebraic languages," in *Communications of the ACM*, vol. 3, pp. 391–391, ASSOC COMPUTING MACHINERY 1515 BROADWAY, NEW YORK, NY 10036, 1960. 1

[49] D. W. Gonzalez and T. Peart, "Applying Dimensional Analysis," *Ada Lett.*, vol. XIII, pp. 77–86, July 1993. 3.2.2, 3.1, 7.4.1

[50] G. S. Novak, "Conversion of units of measurement," *IEEE Transactions on Software Engineering*, vol. 21, no. 8, pp. 651–661, 1995. 3.2.2, 3.1

[51] J. Xiang, J. Knight, and K. Sullivan, "Is My Software Consistent with the Real World?," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 1–4, Jan. 2017. 3.2.2

[52] J. Xiang, J. Knight, and K. Sullivan, "Real-World Types and Their Application," in *Proceedings of the 34th International Conference on Computer Safety, Reliability, and Security - Volume 9337*, SAFECOMP 2015, (New York, NY, USA), pp. 471–484, Springer-Verlag New York, Inc., 2015. 3.2.2, 7.8, 7.8.4

[53] R. Mnner, “Strong Typing and Physical Units,” *SIGPLAN Not.*, vol. 21, pp. 11–20, Mar. 1986. 3.1, 3.2.3.2

[54] A. Dreiheller, B. Mohr, and M. Moerschbacher, “Programming Pascal with Physical Units,” *SIGPLAN Not.*, vol. 21, pp. 114–123, Dec. 1986. 3.1, 7.8.1.1

[55] P. N. Hilfinger, “An Ada Package for Dimensional Analysis,” *ACM Trans. Program. Lang. Syst.*, vol. 10, pp. 189–203, Apr. 1988. 3.1, 3.2.3.2, 7.8.1.1

[56] G. W. Macpherson, “A Reusable Ada Package for Scientific Dimensional Integrity,” *Ada Lett.*, vol. XVI, pp. 56–63, May 1996. 3.1, 3.2.3.3

[57] P. Rogers, “Dimensional Analysis in Ada,” *Ada Lett.*, vol. VIII, pp. 92–100, Sept. 1988. 3.1, 3.2.3.2

[58] Z. D. Umrigar, “Fully static dimensional analysis with C++,” *SIGPLAN Notices*, vol. 29, no. 9, pp. 135–139, 1994. 3.1, 3.2.3.2

[59] G. Rosu and F. Chen, “Certifying measurement unit safety policy,” in *18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings.*, pp. 304–309, Oct. 2003. 3.1, 3.2.3.4

[60] F. Chen, G. Rou, and R. P. Venkatesan, “Rule-based Analysis of Dimensional Safety,” in *Proceedings of the 14th International Conference on Rewriting Techniques and Applications*, RTA’03, (Berlin, Heidelberg), pp. 197–207, Springer-Verlag, 2003. 3.1, 3.2.3.4

[61] M. Hills, F. Chen, and G. Roşu, “A Rewriting Logic Approach to Static Checking of Units of Measurement in C,” *Electronic Notes in Theoretical Computer Science*, vol. 290, pp. 51–67, Dec. 2012. 3.1, 3.2.3.4, 7.8.1.1

[62] D. Donaghy and T. Crick, "Physical Type Tracking through Minimal Source-Code Annotation," *CTIT*, vol. 2, p. 251, 2014. 3.1

[63] L. Jiang and Z. Su, "Osprey: A Practical Type System for Validating Dimensional Unit Correctness of C Programs," in *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, (New York, NY, USA), pp. 262–271, ACM, 2006. 3.1, 3.2.3.3, 5.5.2.1, 7.8.4

[64] P. Guo and S. McCamant, "Annotation-less unit type inference for C," *Final Project, 6.883: Program Analysis*, 2005. 3.1, 3.2.4

[65] P. J. Guo, J. H. Perkins, S. McCamant, and M. D. Ernst, "Dynamic Inference of Abstract Types," in *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, ISSTA '06, (New York, NY, USA), pp. 255–265, ACM, 2006. 3.1, 3.2.4

[66] R. F. Cmelik and N. H. Gehani, "Dimensional analysis with C++," *IEEE Software*, vol. 5, no. 3, pp. 21–27, 1988. 3.1, 3.2.3.3

[67] W. E. Brown, "Introduction to the SI library of unit based computation," in *Presented at*, 1998. 3.1, 3.2.3.3

[68] M. Schabel and S. Watanabe, "Boost Units," 2010. 3.1, 3.2.3.3, 7.8.1.1

[69] "EiffelUnits - Library of Units of Measurement." 3.1

[70] A. Kennedy, "Types for units-of-measure in F#: invited talk," in *Proceedings of the 2008 ACM SIGPLAN workshop on ML*, pp. 1–2, ACM, 2008. 3.1, 7.8

[71] G. W. Petty, "Automated computation and consistency checking of physical dimensions and units in scientific programs," *Software: Practice and Experience*, vol. 31, no. 11, pp. 1067–1076, 2001. 3.1

[72] D. Orchard, A. Rice, and O. Oshmyan, "Evolving Fortran types with inferred units-of-measure," *Journal of Computational Science*, vol. 9, pp. 156–162, July 2015. 3.1, 3.2.3.2, 7.8.1.2

[73] M. Contrastin, A. Rice, M. Danish, and D. Orchard, "Units-of-Measure Correctness in Fortran Programs," *Computing in Science & Engineering*, vol. 18, pp. 102–107, Dec. 2015. 3.1

[74] T. Muranushi and R. A. Eisenberg, "Experience report: Type-checking polymorphic units for astrophysics research in Haskell," in *ACM SIGPLAN Notices*, vol. 49, pp. 31–38, ACM, 2014. 3.1

[75] A. Gundry, "A Typechecker Plugin for Units of Measure: Domain-specific Constraint Solving in GHC Haskell," in *Proceedings of the 2015 ACM SIGPLAN Symposium on Haskell*, Haskell '15, (New York, NY, USA), pp. 11–22, ACM, 2015. 3.1

[76] E. Wyk and Y. Mali, "Generative and Transformational Techniques in Software Engineering II," pp. 442–456, Berlin, Heidelberg: Springer-Verlag, 2008. 3.1

[77] A. Van Delft, "A Java extension with support for dimensions," *Software Practice and Experience*, vol. 29, no. 7, pp. 605–616, 1999. 3.1, 3.2.3.2

[78] S. Hangal and M. S. Lam, "Automatic dimension inference and checking for object-oriented programs," in *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, pp. 155–165, 2009. 3.1, 3.2.4

[79] I. U. Haq, J. Caballero, and M. D. Ernst, "Ayudante: identifying undesired variable interactions," in *Proceedings of the 13th International Workshop on Dynamic Analysis, WODA@SPLASH 2015, Pittsburgh, PA, USA, October 26, 2015*, pp. 8–13, 2015. 3.1, 3.2.4

[80] E. Allen, D. Chase, J. Hallett, V. Luchangco, J.-W. Maessen, S. Ryu, G. L. Steele Jr, S. Tobin-Hochstadt, J. Dias, C. Eastlund, and others, "The Fortress language specification," *Sun Microsystems*, vol. 139, p. 140, 2005. 3.2.3.1, 3.1

[81] R. Cunis, "A Package for Handling Units of Measure in Lisp," *SIGPLAN Lisp Pointers*, vol. V, pp. 21–25, Apr. 1992. 3.1

[82] N. Gehani, "Units of measure as a data attribute," *Computer Languages*, vol. 2, no. 3, pp. 93–111, 1977. 3.1, 3.2.3.2

[83] N. H. Gehani, "Ada's derived types and units of measure," *Software: Practice and Experience*, vol. 15, pp. 555–569, June 1985. 3.1, 3.2.3.2

[84] M. B. Agrawal and V. K. Garg, "Dimensional Analysis in Pascal," *SIGPLAN Not.*, vol. 19, pp. 7–11, Mar. 1984. 3.1, 3.2.3.2

[85] R. T. House, "A proposal for an extended form of type checking of expressions," *The Computer Journal*, vol. 26, no. 4, pp. 366–374, 1983. 3.1, 3.2.3.2

[86] G. Baldwin, "Implementation of Physical Units," *SIGPLAN Not.*, vol. 22, pp. 45–50, Aug. 1987. 3.1, 3.2.3.2

[87] N. J. Goldbaum, J. A. ZuHone, M. J. Turk, K. Kowalik, and A. L. Rosen, "unyt: Handle, manipulate, and convert data with units in python," *arXiv preprint arXiv:1806.02417*, 2018. 3.1

[88] Grecco, Hernan E., "Pint," 2018. 3.1

[89] A. Kennedy, *Programming languages and dimensions*. No. 391 in UCAM-CL-TR-391, PhD Thesis, University of Cambridge, 1996. 3.2.3.1

[90] Open Robotics, "ROS Introduction," 2019. 3.2.3.1

[91] O. Bennich-Björkman and S. McKeever, "The next 700 unit of measurement checkers," in *Proceedings of the 11th ACM SIGPLAN International Conference on Software Language Engineering*, SLE 2018, (New York, NY, USA), pp. 121–132, ACM, 2018. 3.2.3.2

[92] M. Hills, F. Chen, and G. Rosu, "An Abstract Semantics Approach to Unit Safety," 3.2.3.4

[93] F. Chen, G. Rosu, and R. P. Venkatesan, "Checking Dimensional Safety Policies Dynamically and Statically," 3.2.3.4

[94] M. d'Amorim, M. Hills, F. Chen, and G. Rosu, "Automatic and Precise Dimensional Analysis," tech. rep., 2005. 3.2.3.4

[95] R. Khanin, "Dimensional Analysis in Computer Algebra," in *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation*, ISSAC '01, (New York, NY, USA), pp. 201–208, ACM, 2001. 3.2.3.5

[96] T. Antoniu, P. A. Steckler, S. Krishnamurthi, E. Neuwirth, and M. Felleisen, "Validating the Unit Correctness of Spreadsheet Programs," in *26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom*, pp. 439–448, 2004. 3.2.3.5

[97] W. R. Hinsley, *Planktonica: A system for doing biological oceanography by computer*. PhD Thesis, University of London, 2005. 3.2.3.5

[98] P. Aronsson and D. Broman, "Extendable physical unit checking with understandable error reporting," in *Proceedings of the 7th International Modelica Conference; Como; Italy; 20-22 September 2009*, pp. 890–897, Linkping University Electronic Press, 2009. 3.2.3.5

[99] J. Cooper and S. McKeever, "A model-driven approach to automatic conversion of physical units," *Software: Practice and Experience*, vol. 38, pp. 337–359, Apr. 2008. 3.2.3.5

[100] T. Maehne and A. Vachoux, "Supporting dimensional analysis in SystemC-AMS," in *2009 IEEE Behavioral Modeling and Simulation Workshop*, pp. 108–113, Sept. 2009. 3.2.3.5

[101] P. Roy, "SimCheck: An Expressive Type System for Simulink," Apr. 2010. 3.2.3.5

[102] S. Owre, I. Saha, and N. Shankar, "Automatic Dimensional Analysis of Cyber-Physical Systems," in *FM 2012: Formal Methods* (D. Giannakopoulou and D. Mry, eds.), Lecture Notes in Computer Science, pp. 356–371, Springer Berlin Heidelberg, 2012. 3.2.3.5

[103] J. Ou and F. Pellacini, "SafeGI: Type Checking to Improve Correctness in Rendering System Implementation," *Computer Graphics Forum*, vol. 29, pp. 1269–1277, June 2010. 3.2.3.5

[104] D. Eliasson, *Units of Measurement in a Modelica Compiler*. LU-CS-EX 2016-32, 2016. 3.2.3.5

[105] P. R. Griffioen, "Type Inference for Array Programming with Dimensioned Vector Spaces," in *Proceedings of the 27th Symposium on the Implementation*

and Application of Functional Programming Languages, IFL '15, (New York, NY, USA), pp. 4:1–4:12, ACM, 2015. 3.2.3.5

[106] M. Nanjundappa and S. K. Shukla, "Verification of unit and dimensional consistencies in polychronous specifications," in *Proceedings of the 2014 Forum on Specification and Design Languages (FDL)*, vol. 978-2-9530504-9-3, pp. 1–8, Oct. 2014. 3.2.3.5

[107] S. Krings and M. Leuschel, "Inferring Physical Units in B Models," in *Software Engineering and Formal Methods* (R. M. Hierons, M. G. Merayo, and M. Bravetti, eds.), Lecture Notes in Computer Science, pp. 137–151, Springer Berlin Heidelberg, 2013. 3.2.3.5

[108] G. A. Miller, "WordNet: A Lexical Database for English," *Commun. ACM*, vol. 38, pp. 39–41, Nov. 1995. 3.2.4

[109] R. O'Callahan and D. Jackson, "Lackwit: A program understanding tool based on type inference," in *In Proceedings of the 19th International Conference on Software Engineering*, Citeseer, 1997. 3.2.4

[110] V. Raychev, M. Vechev, and A. Krause, "Predicting Program Properties from "Big Code"," in *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, (New York, NY, USA), pp. 111–124, ACM, 2015. 3.2.4

[111] Z. Xu, X. Zhang, L. Chen, K. Pei, and B. Xu, "Python Probabilistic Type Inference with Natural Language Support," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, (New York, NY, USA), pp. 607–618, ACM, 2016. 3.2.4, 7.2.1.1, 7.2.1.2

[112] S. K. Dash, M. Allamanis, and E. T. Barr, "Refinym: using names to refine types," in Leavens *et al.* [175], pp. 107–117. 3.2.4

[113] R. S. Malik, J. Patra, and M. Pradel, "Nl2type: inferring javascript function types from natural language information," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019* (G. Mussbacher, J. M. Atlee, and T. Bultan, eds.), pp. 304–315, IEEE / ACM, 2019. 3.2.4

[114] M. Vakilian, A. Phaosawasdi, M. D. Ernst, and R. E. Johnson, "Cascade: A Universal Programmer-Assisted Type Qualifier Inference Tool," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp. 234–245, 2015. 3.3.1, 7.8.1

[115] M. M. Papi, M. Ali, T. L. C. Jr., J. H. Perkins, and M. D. Ernst, "Practical pluggable types for java," in *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2008, Seattle, WA, USA, July 20-24, 2008* (B. G. Ryder and A. Zeller, eds.), pp. 201–212, ACM, 2008. 3.3.1, 7.8.1

[116] U. Shankar, K. Talwar, J. S. Foster, and D. Wagner, "Detecting format string vulnerabilities with type qualifiers," in *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, SSYM'01, (Berkeley, CA, USA), USENIX Association, 2001. 3.3.1

[117] J. S. Foster, M. Fähndrich, and A. Aiken, "A theory of type qualifiers," in *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Atlanta, Georgia, USA, May 1-4, 1999* (B. G. Ryder and B. G. Zorn, eds.), pp. 192–203, ACM, 1999. 3.3.1

[118] D. Greenfieldboyce and J. S. Foster, "Type qualifier inference for java," in *Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2007, October 21-25, 2007, Montreal, Quebec, Canada* (R. P. Gabriel, D. F. Bacon, C. V. Lopes, and G. L. S. Jr., eds.), pp. 321–336, ACM, 2007. 3.3.1

[119] P. Chalin and P. R. James, "Non-null References by Default in Java: Alleviating the Nullity Annotation Burden," in *ECOOP 2007 - Object-Oriented Programming, 21st European Conference, Berlin, Germany, July 30 - August 3, 2007, Proceedings*, pp. 227–247, 2007. 3.3.2, 4.1, 4.3.1

[120] W. Dietl, M. D. Ernst, and P. Müller, "Tunable static inference for generic universe types," in *Proceedings of the 25th European Conference on Object-oriented Programming*, ECOOP'11, (Berlin, Heidelberg), pp. 333–357, Springer-Verlag, 2011. 3.3.2

[121] Z. Gao, C. Bird, and E. T. Barr, "To type or not to type: quantifying detectable bugs in JavaScript," in *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*, pp. 758–769, 2017. 3.3.2

[122] A. Strauss and J. M. Corbin, *Basics of qualitative research: Grounded theory procedures and techniques.* Sage Publications, Inc, 1990. 4.1, 4.2.2.2, 4.3.5

[123] M. Gasparic, G. C. Murphy, and F. Ricci, "A context model for ide-based recommendation systems," *Journal of Systems and Software*, vol. 128, pp. 200–219, 2017. 4.2.1.3

[124] C. Parnin and A. Orso, "Are automated debugging techniques actually helping programmers?," in *Proceedings of the 20th International Symposium*

on Software Testing and Analysis, ISSTA 2011, Toronto, ON, Canada, July 17-21, 2011, pp. 199–209, 2011. 4.2.1.4

[125] R. E. Kirk, *Experimental design*. Wiley Online Library, 1982. 4.2.2.3

[126] J. Bohannon, “Mechanical turk upends social sciences,” 2016. 4.2.3

[127] N. A. Smith, I. E. Sabat, L. R. Martinez, K. Weaver, and S. Xu, “A convenient solution: Using MTurk to sample from hard-to-reach populations,” *Industrial and Organizational Psychology*, vol. 8, no. 2, pp. 220–228, 2015. 4.2.3

[128] K. T. Stolee and S. Elbaum, “Exploring the Use of Crowdsourcing to Support Empirical Studies in Software Engineering,” in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM ’10, (New York, NY, USA), pp. 35:1–35:4, ACM, 2010. 4.2.3

[129] R. M. de Mello, P. C. da Silva, P. Runeson, and G. H. Travassos, “Towards a Framework to Support Large Scale Sampling in Software Engineering Surveys,” in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM ’14, (New York, NY, USA), pp. 48:1–48:4, ACM, 2014. 4.2.3

[130] K. Mao, L. Capra, M. Harman, and Y. Jia, “A survey of the use of crowdsourcing in software engineering,” *Journal of Systems and Software*, vol. 126, pp. 57 – 84, 2017. 4.2.3

[131] W. Mason and S. Suri, “Conducting behavioral research on Amazons Mechanical Turk,” *Behavior research methods*, vol. 44, no. 1, pp. 1–23, 2012. 4.2.3, 4.4.1.1

[132] R. Jia, Z. R. Steelman, and B. H. Reich, "Using Mechanical Turk Data in IS Research: Risks, Rewards, and Recommendations," *CAIS*, vol. 41, p. 14, 2017. 4.2.3, 4.4.1.1

[133] I. P. Kan and A. Drummey, "Do imposters threaten data quality? An examination of worker misrepresentation and downstream consequences in Amazon's Mechanical Turk workforce," *Computers in Human Behavior*, vol. 83, pp. 243–253, 2018. 4.2.3, 4.2.6.1, 4.4.2.1

[134] D. J. Hauser and N. Schwarz, "Attentive Turkers: MTurk participants perform better on online attention checks than do subject pool participants," *Behavior research methods*, vol. 48, no. 1, pp. 400–407, 2016. 4.2.3

[135] K. A. Thomas and S. Clifford, "Validity and Mechanical Turk: An assessment of exclusion methods and interactive experiments," *Computers in Human Behavior*, vol. 77, pp. 184–197, 2017. 4.2.3

[136] W. A. Mason and D. J. Watts, "Financial incentives and the performance of crowds," *SIGKDD Explorations*, vol. 11, no. 2, pp. 100–108, 2009. 4.2.3

[137] M. J. McClendon, "Acquiescence and recency response-order effects in interview surveys," *Sociological Methods & Research*, vol. 20, no. 1, pp. 60–103, 1991. 4.2.4.1

[138] "Clang: a C language family frontend for LLVM," 2018. `https://clang.llvm.org`, (accessed 1 May 2019). 4.2.5.2

[139] "Qualtrics," 2018. `https://www.qualtrics.com`, (accessed 3 May 2019). 4.2.5.3

[140] R Core Team, *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2013. 4.2.5.6

[141] C. Gandrud, *Reproducible research with R and R studio*. Chapman and Hall/CRC, 2016. 4.2.5.6

[142] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. New York: Springer, fourth ed., 2002. 4.2.5.6, 4.3.3

[143] S. Dorai-Raj, *binom: Binomial Confidence Intervals For Several Parameterizations*. 2014. 4.2.5.6

[144] A. Agresti and B. A. Coull, "Approximate is better than exact for interval estimation of binomial proportions," *The American Statistician*, vol. 52, no. 2, pp. 119–126, 1998. 4.3.1

[145] C. Flanagan and K. R. M. Leino, "Houdini, an Annotation Assistant for ESC/Java," in *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings*, pp. 500–517, 2001. 4.3.1

[146] J. W. Tukey, *Exploratory data analysis*, vol. 2. Reading, Mass., 1977. 4.3.2

[147] D. Marjamäki, *Cppcheck: a tool for static C/C++ code analysis*, 2018 (accessed 3 May 2019). 4.3.2, 5.2.3.1, 5.3

[148] A. Takikawa, D. Feltey, B. Greenman, M. S. New, J. Vitek, and M. Felleisen, "Is Sound Gradual Typing Dead?," in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, (New York, NY, USA), pp. 456–468, ACM, 2016. 4.4.1.1

[149] A. Danial, "Count Lines Of Code," 2018. 4.5.1.4, 5.5.4

[150] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler, "A few billion lines of code later: using static analysis to find bugs in the real world," *Communications of the ACM*, vol. 53, no. 2, pp. 66–75, 2010. 1

[151] D. Hovemeyer, J. Spacco, and W. Pugh, "Evaluating and tuning a static analysis to find null pointer bugs," in *ACM SIGSOFT Software Engineering Notes*, vol. 31, pp. 13–19, ACM, 2005. 1

[152] F. Nielson, H. R. Nielson, and C. Hankin, *Principles of program analysis*. Springer, 1999. 5.2.3, 7

[153] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 5.3

[154] D. Koller and N. Friedman, *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. 7, 7.2.1.2

[155] J. Pearl, "Fusion, Propagation, and Structuring in Belief Networks," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, 1986. 7.2, 7.2.1.2, 7.2.2

[156] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009. 7.2.2

[157] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 498–519, 2001. 7.2.4

[158] E. Loper and S. Bird, "NLTK: the natural language toolkit," *CoRR*, vol. cs.CL/0205028, 2002. 7.3

[159] J. Mooij, "libDAI - A free and open source C++ library for Discrete Approximate Inference in graphical models," 2010. 7.3

[160] P. A. Tipler and G. Mosca, *Physics for scientists and engineers*. Macmillan, 2007. 7.7.1

[161] K. Muslu, Y. Brun, R. Holmes, M. D. Ernst, and D. Notkin, "Speculative analysis of integrated development environment recommendations," in *Proceedings of the 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2012, part of SPLASH 2012, Tucson, AZ, USA, October 21-25, 2012* (G. T. Leavens and M. B. Dwyer, eds.), pp. 669–682, ACM, 2012. 7.8.1

[162] M. Czerwinski, E. Horvitz, and S. Wilhite, "A diary study of task switching and interruptions," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 175–182, ACM, 2004. 7.8.1

[163] G. Mark, V. M. Gonzalez, and J. Harris, "No task left behind?: examining the nature of fragmented work," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 321–330, ACM, 2005. 7.8.1

[164] G. Mark, D. Gudith, and U. Klocke, "The cost of interrupted work: more speed and stress," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 107–110, ACM, 2008. 7.8.1

[165] M. Krisper, J. Iber, T. Rauter, and C. Kreiner, "Physical Quantity: Towards a Pattern Language for Quantities and Units in Physical Calculations," in

Proceedings of the 22Nd European Conference on Pattern Languages of Programs, EuroPLoP '17, (New York, NY, USA), pp. 9:1–9:20, ACM, 2017. 7.8.1.1

[166] G. M. Bierman, M. Abadi, and M. Torgersen, "Understanding typescript," in *ECOOP 2014 - Object-Oriented Programming - 28th European Conference, Uppsala, Sweden, July 28 - August 1, 2014. Proceedings* (R. E. Jones, ed.), vol. 8586 of *Lecture Notes in Computer Science*, pp. 257–281, Springer, 2014. 7.8.1.1

[167] Open Robotics, "ROS/Introduciton - ROS Wiki," 2019. 7.8.1.1

[168] D. E. Knuth, "Backus normal form vs. backus naur form," *Commun. ACM*, vol. 7, pp. 735–736, Dec. 1964. 7.8.1.1

[169] J. Wright, "siunitx, A comprehensive (SI) units package," 2019. 7.8.1.1

[170] B. Hamilton, *A compact representation of units.* Hewlett-Packard Laboratories, Technical Publications Department, 1996. 1

[171] G. C. Murphy, "The need for context in software engineering (ieee cs harlan mills award keynote)," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 5–5, ACM, 2018. 8.2.1

[172] H. Mills, "On the statistical validation of computer programs," *IBM FSD Report*, 1972. 8.2.2

[173] J. C. Knight and P. Ammann, "An experimental evaluation of simple methods for seeding program errors," in *Proceedings, 8th International Conference on Software Engineering, London, UK, August 28-30, 1985.* (M. M. Lehman, H. Hünke, and B. W. Boehm, eds.), pp. 337–342, IEEE Computer Society, 1985. 8.2.2

[174] Wikipedia, *Boeing 737 MAX groundings*, July (accessed 10 July 2019). 8.3

[175] G. T. Leavens, A. Garcia, and C. S. Pasareanu, eds., *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, ACM, 2018. 8, 112

9 Appendicies

A Detailed Accuracy and Timing Statistics

Table 9.1: Accuracy and time for questions by treatment.

Q#	DIFFICULTY	CONTROL				TREATMENTS																			
						ONE SUGGESTION								THREE SUGGESTIONS											
		T_1				T_2				T_3				T_4				T_5				T_6			
		Correct		Time (s)		Correct		Time (s)		Correct		Time (s)		Correct		Time (s)		Correct		Time (s)		Correct		Time (s)	
		%	Fraction	Mean	Median	%	Fraction	Mean	Median	%	Fraction	Mean	Median	%	Fraction	Mean	Median	%	Fraction	Mean	Median	%	Fraction	Mean	Median
12	Easy	100	6/6	76	70	83	5/6	111	36	33	2/6	162	121	100	6/6	54	42	83	5/6	162	62	67	4/6	95	57
9		90	9/10	113	90	80	8/10	112	70	67	4/6	93	68	100	6/6	52	34	100	6/6	124	68	83	5/6	287	134
15		83	5/6	169	141	83	5/6	122	103	40	4/10	125	102	78	7/9	90	87	89	8/9	109	114	33	2/6	150	133
5		83	5/6	144	82	83	5/6	237	155	17	1/6	116	49	67	4/6	66	56	50	3/6	109	91	67	6/9	99	104
All Easy		89	25/28	124	88	82	23/28	141	70	36	10/28	124	74	74	23/27	68	54	81	22/27	124	86	63	17/27	152	104
6	Medium	67	4/6	134	130	75	6/8	156	103	50	3/6	146	76	89	8/9	187	93	33	2/6	133	87	44	4/9	131	65
4		67	4/6	153	102	80	8/10	151	105	20	2/10	223	146	50	3/6	244	148	67	6/9	258	354	33	2/6	187	178
16		67	4/6	64	65	90	9/10	200	72	33	2/6	104	77	67	4/6	81	61	89	8/9	170	138	70	7/10	242	125
8		64	7/11	130	141	90	9/10	98	79	33	2/6	163	103	67	4/6	84	71	67	4/6	170	102	67	4/6	120	71
3		60	6/10	302	233	83	5/6	202	139	17	1/6	150	123	67	4/6	68	76	100	6/6	184	139	83	5/6	138	142
2		60	6/10	120	105	33	2/6	75	54	20	2/10	72	58	50	3/6	92	94	33	2/6	102	124	50	3/6	191	77
7		50	3/6	226	103	80	8/10	155	153	17	1/6	86	69	67	4/6	193	150	33	3/9	191	112	20	1/5	399	297
10		43	3/7	87	105	83	5/6	97	100	33	2/6	184	184	89	8/9	125	146	44	4/9	156	110	83	5/6	278	222
11		33	2/6	151	128	100	3/3	52	65	67	6/9	171	112	50	3/6	88	65	44	4/9	145	128	83	5/6	203	135
18		33	2/6	167	50	100	6/6	126	125	33	2/6	264	218	67	6/9	285	145	67	4/6	162	120	83	5/6	271	182
14		33	2/6	106	101	67	4/6	75	42	0	0/6	75	53	83	5/6	54	37	56	5/9	87	57	11	1/9	246	220
All Medium		51	41/80	153	112	77	65/84	140	90	28	21/74	143	108	69	52/75	144	86	57	48/84	161	123	56	42/75	215	143
19	Hard	17	1/6	213	201	50	3/6	90	85	17	1/6	174	83	67	6/9	143	80	33	2/6	118	73	50	3/6	239	188
1		17	1/6	245	188	67	4/6	56	52	40	4/10	258	175	67	4/6	115	83	33	2/6	187	98	17	1/6	96	65
17		17	1/6	54	32	33	2/6	198	126	57	4/7	233	111	67	4/6	270	182	43	3/7	198	195	0	0/6	136	134
13		17	1/6	130	90	50	3/6	99	67	0	0/6	156	146	0	0/6	231	193	56	5/9	222	198	11	1/9	231	143
20		17	1/6	234	196	50	3/6	231	168	0	0/6	111	84	14	1/7	244	273	33	2/6	302	283	0	0/6	330	311
All Hard		17	5/30	175	118	50	15/30	135	91	23	8/35	196	99	44	15/34	196	145	41	14/34	206	122	15	5/33	208	137
All Questions		53	73/138	152	109	74	103/139	136	83	31	43/140	156	99	66	90/136	142	88	58	84/145	165	112	47	64/135	201	135

B IRB for Human Study in § 4.2.3

IRB# 20170817412EX
Date Approved: 08/07/2017
Valid Until: 08/06/2022

UNIVERSITY OF Nebraska Lincoln

Computer Science and Engineering Department

IRB#17412

Assessing Programmers' Abilities to Infer Physical Units

Purpose:
This research project will aim to assess how well programmers can determine the physical units associated with program variables. Participants in the states of Nebraska and Alabama must be at least 19 years old or older to participate, participants in the state of Mississippi must be at least 21 years old to participate, and participants in all other states must be 18 years old to participate. You are invited to participate in this study because you are familiar with computer programs.

Procedures:
You will be asked to view software code samples and determine the physical units (like 'meters' or 'seconds') associated with program variables. You will be asked to take a pre-test to determine that you meet the study criteria, and you will be excluded from the rest of the study if you do not pass the pre-test. The procedures will last for ~20 minutes, and will be conducted on your computer.

Benefits:
There are no direct benefits to you as a research participant; however, the benefits to science and/or society may include better understanding of how programmers reason about physical units.

Risks and/or Discomforts:
There are no known risks or discomforts associated with this research.

Confidentiality:
Any information obtained during this study that could identify you will be kept strictly confidential.

Identifiable files will be kept until compensation has been distributed and then deleted. Non-identifiable survey results will be kept at least 1-year and possibly indefinitely. The data will be stored electronically through a secure server and will only be seen by the research team. The MTurk worker ID will **not** be shared with anyone. The MTurk work ID will only be collected for the purposes for distributing compensation and will not be associated with survey responses. Note that any work done on MTurk can be linked to a workers public profile, as described in https://www.mturk.com/mturk/privacynotice.

Compensation:
You will receive $0.25 for passing the pre-test and $0.25 for each correctly answered question of the 20 questions up to a total of $5.50.

Opportunity to Ask Questions:
You may ask any questions concerning this research and have those questions answered before agreeing to participate in or during the study. Or you may contact the investigator(s) at the phone numbers below. Please contact the University of Nebraska-Lincoln Institutional Review Board at (402) 472-6965 to voice concerns about the research or if you have any questions about your rights as a research participant.

Freedom to Withdraw:
Participation in this study is voluntary. You can refuse to participate or withdraw at any time without harming your relationship with the researchers or the University of Nebraska-Lincoln, or in any other way receive a penalty or loss of benefits to which you are otherwise entitled.

256 Avery Hall / P.O. 880115 / Lincoln, NE 68588-0115
(402) 472-2401 / FAX (402) 472-7767

IRB Approval
IRB# 20170817412EX
Date Approved: 08/07/2017
Valid Until: 08/06/2022

Consent, Right to Receive a Copy:
You are voluntarily making a decision whether or not to participate in this research study. Your signature certifies that you have decided to participate having read and understood the information presented. You will be given a copy of this consent form to keep.

Participant Name:

____
(Name of Participant)

Consent:

☐ By marking this checkbox, I hereby consent to participate in this study.

Date: ________________

Name and Phone number of investigator(s)

John-Paul Ore, Principal Investigator Office: (402) 882-2118
Sebastian Elbaum, Secondary Investigator Office: (402) 472-6748

C Phriky's Mapping

LIBRARY	CLASS	ATTRIBUTE	PHYSICAL UNITS
geometry_msgs	Accel	angular	s^{-2}
geometry_msgs	Accel	linear	$m\,s^{-2}$
geometry_msgs	AccelStamped	angular	s^{-2}
geometry_msgs	AccelStamped	linear	$m\,s^{-2}$
geometry_msgs	AccelStamped	stamp	s
geometry_msgs	AccelWithCovariance	angular	s^{-2}
geometry_msgs	AccelWithCovariance	linear	$m\,s^{-2}$
geometry_msgs	AccelWithCovarianceStamped	angular	s^{-2}
geometry_msgs	AccelWithCovarianceStamped	linear	$m\,s^{-2}$
geometry_msgs	AccelWithCovarianceStamped	stamp	s
geometry_msgs	Inertia	com	m
geometry_msgs	Inertia	ixx	$kg\,m^{-2}$
geometry_msgs	Inertia	ixy	$kg\,m^{-2}$
geometry_msgs	Inertia	ixz	$kg\,m^{-2}$
geometry_msgs	Inertia	iyy	$kg\,m^{-2}$
geometry_msgs	Inertia	iyz	$kg\,m^{-2}$
geometry_msgs	Inertia	izz	$kg\,m^{-2}$
geometry_msgs	Inertia	m	kg
geometry_msgs	InertiaStamped	com	m
geometry_msgs	InertiaStamped	ixx	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	ixy	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	ixz	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	iyy	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	iyz	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	izz	$kg\,m^{-2}$
geometry_msgs	InertiaStamped	m	kg
geometry_msgs	InertiaStamped	stamp	s
geometry_msgs	Point	x	m
geometry_msgs	Point	y	m
geometry_msgs	Point	z	m
geometry_msgs	Point32	x	m
geometry_msgs	Point32	y	m
geometry_msgs	Point32	z	m

geometry_msgs	PointStamped	stamp	s
geometry_msgs	PointStamped	x	m
geometry_msgs	PointStamped	y	m
geometry_msgs	PointStamped	z	m
geometry_msgs	PointStampedPtr	stamp	s
geometry_msgs	PointStampedPtr	x	m
geometry_msgs	PointStampedPtr	y	m
geometry_msgs	PointStampedPtr	z	m
geometry_msgs	Polygon	points	m
geometry_msgs	PolygonStamped	points	m
geometry_msgs	PolygonStamped	stamp	s
geometry_msgs	Pose	orientation	*quaternion*
geometry_msgs	Pose	position	m
geometry_msgs	Pose2D	theta	rad
geometry_msgs	Pose2D	x	m
geometry_msgs	Pose2D	y	m
geometry_msgs	PoseArray	orientation	*quaternion*
geometry_msgs	PoseArray	position	m
geometry_msgs	PoseArray	stamp	s
geometry_msgs	PoseStamped	orientation	*quaternion*
geometry_msgs	PoseStamped	position	m
geometry_msgs	PoseStamped	stamp	s
geometry_msgs	PoseWithCovariance	orientation	*quaternion*
geometry_msgs	PoseWithCovariance	position	m
geometry_msgs	PoseWithCovarianceStamped	orientation	*quaternion*
geometry_msgs	PoseWithCovarianceStamped	position	m
geometry_msgs	PoseWithCovarianceStamped	stamp	s
geometry_msgs	Quaternion	w	*quaternion*
geometry_msgs	Quaternion	x	*quaternion*
geometry_msgs	Quaternion	y	*quaternion*
geometry_msgs	Quaternion	z	*quaternion*
geometry_msgs	QuaternionStamped	stamp	s
geometry_msgs	QuaternionStamped	w	*quaternion*
geometry_msgs	QuaternionStamped	x	*quaternion*
geometry_msgs	QuaternionStamped	y	*quaternion*
geometry_msgs	QuaternionStamped	z	*quaternion*
geometry_msgs	Transform	rotation	*quaternion*

geometry_msgs	Transform	translation	m
geometry_msgs	TransformStamped	rotation	*quaternion*
geometry_msgs	TransformStamped	stamp	s
geometry_msgs	TransformStamped	translation	m
geometry_msgs	Twist	angular	s^{-1}
geometry_msgs	Twist	linear	$m\,s^{-1}$
geometry_msgs	TwistStamped	angular	s^{-1}
geometry_msgs	TwistStamped	linear	$m\,s^{-1}$
geometry_msgs	TwistStamped	stamp	s
geometry_msgs	TwistWithCovariance	angular	s^{-1}
geometry_msgs	TwistWithCovariance	linear	$m\,s^{-1}$
geometry_msgs	TwistWithCovarianceStamped	angular	s^{-1}
geometry_msgs	TwistWithCovarianceStamped	linear	$m\,s^{-1}$
geometry_msgs	TwistWithCovarianceStamped	stamp	s
geometry_msgs	Wrench	force	$kg\,m\,s^{-2}$
geometry_msgs	Wrench	torque	$kg\,m^2\,s^{-2}$
geometry_msgs	WrenchStamped	force	$kg\,m\,s^{-2}$
geometry_msgs	WrenchStamped	stamp	s
geometry_msgs	WrenchStamped	torque	$kg\,m^2\,s^{-2}$
nav_2d_msgs	Twist2D	theta	rad
nav_2d_msgs	Twist2D	x	$m\,s^{-1}$
nav_2d_msgs	Twist2D	y	$m\,s^{-1}$
nav_2d_msgs	Twist2D32	theta	rad
nav_2d_msgs	Twist2D32	x	$m\,s^{-1}$
nav_2d_msgs	Twist2D32	y	$m\,s^{-1}$
nav_2d_msgs	Twist2D32Stamped	theta	rad
nav_2d_msgs	Twist2D32Stamped	x	$m\,s^{-1}$
nav_2d_msgs	Twist2D32Stamped	y	$m\,s^{-1}$
nav_msgs	GridCells	cell_height	m
nav_msgs	GridCells	cell_width	m
nav_msgs	GridCells	stamp	s
nav_msgs	MapMetaData	map_load_time	s
nav_msgs	MapMetaData	resolution	m
nav_msgs	MapMetaData	x	m
nav_msgs	MapMetaData	y	m
nav_msgs	MapMetaData	z	rad
nav_msgs	OccupancyGrid	map_load_time	s

nav_msgs	OccupancyGrid	resolution	m
nav_msgs	OccupancyGrid	stamp	s
nav_msgs	OccupancyGrid	x	m
nav_msgs	OccupancyGrid	y	m
nav_msgs	OccupancyGrid	z	rad
nav_msgs	Odometry	angular	s^{-1}
nav_msgs	Odometry	linear	$m\,s^{-1}$
nav_msgs	Odometry	orientation	*quaternion*
nav_msgs	Odometry	position	m
nav_msgs	Odometry	stamp	s
nav_msgs	Path	orientation	*quaternion*
nav_msgs	Path	position	m
nav_msgs	Path	stamp	s
ros	Duration	nsec	s
ros	Duration	sec	s
ros	Rate	rate	s^{-1}
ros	Time	nsec	s
ros	Time	sec	s
sensor_msgs	BatteryState	capacity	A s
sensor_msgs	BatteryState	cell_voltage	$kg\,m^2\,A^{-1}\,s^{-3}$
sensor_msgs	BatteryState	charge	A s
sensor_msgs	BatteryState	current	A
sensor_msgs	BatteryState	design_capacity	A s
sensor_msgs	BatteryState	stamp	s
sensor_msgs	BatteryState	voltage	$kg\,m^2\,A^{-1}\,s^{-3}$
sensor_msgs	FluidPressure	fluid_pressure	$kg\,m^{-1}\,s^{-2}$
sensor_msgs	FluidPressure	stamp	s
sensor_msgs	FluidPressure	variance	$kg\,m^2\,s^{-2}$
sensor_msgs	Illuminance	illuminance	$cd\,m^{-2}$
sensor_msgs	Illuminance	stamp	s
sensor_msgs	Illuminance	variance	$cd^2\,m^{-4}$
sensor_msgs	Imu	angular_velocity	s^{-1}
sensor_msgs	Imu	angular_velocity_covariance	s^{-2}
sensor_msgs	Imu	linear_acceleration	$m\,s^{-2}$
sensor_msgs	Imu	linear_acceleration_covariance	$m^2\,s^{-4}$
sensor_msgs	Imu	orientation	*quaternion*
sensor_msgs	Imu	stamp	s

sensor_msgs	JointState	effort	$kg\,m^2\,s^{-2}$
sensor_msgs	JointState	position	rad
sensor_msgs	JointState	stamp	s
sensor_msgs	JointState	velocity	s^{-1}
sensor_msgs	LaserEcho	echoes	m
sensor_msgs	LaserScan	angle_increment	rad
sensor_msgs	LaserScan	angle_max	rad
sensor_msgs	LaserScan	angle_min	rad
sensor_msgs	LaserScan	range_max	m
sensor_msgs	LaserScan	range_min	m
sensor_msgs	LaserScan	ranges	m
sensor_msgs	LaserScan	scan_time	s
sensor_msgs	LaserScan	stamp	s
sensor_msgs	LaserScan	time_increment	s
sensor_msgs	MagneticField	magnetic_field	$kg\,A^{-1}\,s^{-2}$
sensor_msgs	MagneticField	magnetic_field_covariance	$kg^2\,A^{-2}\,s^{-4}$
sensor_msgs	MagneticField	stamp	s
sensor_msgs	MultiDOFJointState	stamp	s
sensor_msgs	MultiEchoLaserScan	angle_increment	rad
sensor_msgs	MultiEchoLaserScan	angle_max	rad
sensor_msgs	MultiEchoLaserScan	angle_min	rad
sensor_msgs	MultiEchoLaserScan	range_max	m
sensor_msgs	MultiEchoLaserScan	range_min	m
sensor_msgs	MultiEchoLaserScan	ranges	m
sensor_msgs	MultiEchoLaserScan	scan_time	s
sensor_msgs	MultiEchoLaserScan	stamp	s
sensor_msgs	MultiEchoLaserScan	time_increment	s
sensor_msgs	NavSatFix	altitude	m
sensor_msgs	NavSatFix	latitude	*degrees_360*
sensor_msgs	NavSatFix	longitude	*degrees_360*
sensor_msgs	NavSatFix	position_covariance	m^2
sensor_msgs	NavSatFix	stamp	s
sensor_msgs	PointCloud	points	m
sensor_msgs	PointCloud	stamp	s
sensor_msgs	PointCloud2	points	m
sensor_msgs	PointCloud2	stamp	s
sensor_msgs	PointCloud2Iterator	points	m

sensor_msgs	PointCloud2Iterator	stamp	s
sensor_msgs	Range	field_of_view	rad
sensor_msgs	Range	max_range	m
sensor_msgs	Range	min_range	m
sensor_msgs	Range	range	m
sensor_msgs	Range	stamp	s
sensor_msgs	Temperature	stamp	s
sensor_msgs	Temperature	temperature	°C
sensor_msgs	Temperature	variance	°C^2
sensor_msgs	TimeReference	stamp	s
sensor_msgs	TimeReference	time_ref	s
shape_msgs	Mesh	vertices	m
shape_msgs	SolidPrimitive	dimensions	m
stereo_msgs	DisparityImage	T	m
stereo_msgs	DisparityImage	stamp	s
tf	Pose	getOrigin	m
tf	Pose	getRotation	*quaternion*
tf	Quaternion	getW	*quaternion*
tf	Quaternion	getX	*quaternion*
tf	Quaternion	getY	*quaternion*
tf	Quaternion	getZ	*quaternion*
tf	StampedTransform	getOrigin	m
tf	StampedTransform	getRotation	*quaternion*
tf	StampedTransform	stamp_	s
tf	Transform	getOrigin	m
tf	Transform	getRotation	*quaternion*
tf2	Pose	getOrigin	m
tf2	Pose	getRotation	*quaternion*
tf2	Quaternion	getW	*quaternion*
tf2	Quaternion	getX	*quaternion*
tf2	Quaternion	getY	*quaternion*
tf2	Quaternion	getZ	*quaternion*
tf2	StampedTransform	getOrigin	m
tf2	StampedTransform	getRotation	*quaternion*
tf2	StampedTransform	stamp_	s
tf2	Transform	getOrigin	m
tf2	Transform	getRotation	*quaternion*

trajectory_msgs	JointTrajectory	accelerations	s^{-2}
trajectory_msgs	JointTrajectory	effort	$kg\,m^2\,s^{-2}$
trajectory_msgs	JointTrajectory	positions	rad
trajectory_msgs	JointTrajectory	stamp	s
trajectory_msgs	JointTrajectory	time_from_start	s
trajectory_msgs	JointTrajectory	velocities	s^{-1}
trajectory_msgs	JointTrajectoryPoint	accelerations	s^{-2}
trajectory_msgs	JointTrajectoryPoint	effort	$kg\,m^2\,s^{-2}$
trajectory_msgs	JointTrajectoryPoint	positions	rad
trajectory_msgs	JointTrajectoryPoint	time_from_start	s
trajectory_msgs	JointTrajectoryPoint	velocities	s^{-1}
visualization_msgs	InteractiveMarker	lifetime	s
visualization_msgs	InteractiveMarker	orientation	*quaternion*
visualization_msgs	InteractiveMarker	position	m
visualization_msgs	InteractiveMarker	stamp	s
visualization_msgs	InteractiveMarkerControl	lifetime	s
visualization_msgs	InteractiveMarkerControl	orientation	*quaternion*
visualization_msgs	InteractiveMarkerControl	position	m
visualization_msgs	InteractiveMarkerControl	stamp	s
visualization_msgs	InteractiveMarkerFeedback	orientation	*quaternion*
visualization_msgs	InteractiveMarkerFeedback	position	m
visualization_msgs	InteractiveMarkerFeedback	stamp	s
visualization_msgs	InteractiveMarkerInit	lifetime	s
visualization_msgs	InteractiveMarkerInit	orientation	*quaternion*
visualization_msgs	InteractiveMarkerInit	position	m
visualization_msgs	InteractiveMarkerInit	stamp	s
visualization_msgs	InteractiveMarkerPose	orientation	*quaternion*
visualization_msgs	InteractiveMarkerPose	position	m
visualization_msgs	InteractiveMarkerPose	stamp	s
visualization_msgs	InteractiveMarkerUpdate	lifetime	s
visualization_msgs	InteractiveMarkerUpdate	orientation	*quaternion*
visualization_msgs	InteractiveMarkerUpdate	position	m
visualization_msgs	InteractiveMarkerUpdate	stamp	s
visualization_msgs	Marker	lifetime	s
visualization_msgs	Marker	orientation	*quaternion*
visualization_msgs	Marker	position	m
visualization_msgs	Marker	stamp	s

visualization_msgs	MarkerArray	lifetime	s
visualization_msgs	MarkerArray	orientation	*quaternion*
visualization_msgs	MarkerArray	position	m
visualization_msgs	MarkerArray	stamp	s

Table 9.2: PHRIKY's Mapping of that relates attributes of classes defined in shared libraries to physical unit types.

PROCEDURE	PHYSICAL UNITS
`atan2`	rad
`acos`	rad
`asin`	rad
`atan`	rad
`toSec`	s
`toNSec`	s
`quatToRPY`	rad
`getRoll`	rad
`getPitch`	rad
`getYaw`	rad

Table 9.3: PHRIKY's Mapping that relates known procedures to physical unit types.

D Code Artifacts and Questions for Developer Study of the Type Annotation Burden

INTRO

TASK ACCEPTANCE CRITERION:

1. Don't rush.
2. Do much better than random.
3. Provide good explainations.

*Watch for random **attention checks (ACs)**.*

10 QUESTIONS. **START WHEN READY.**

Block 1

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
77 float gdist(pcl::PointXYZ pt, const Eigen::Vector4f &v) {
78   return sqrt((pt.x - v(0)) * (pt.x - v(0)) + (pt.y - v(1)) * (pt.y - v(1)) +
79               (pt.z - v(2)) * (pt.z - v(2))); //
80 }
```

What are the units for `return` on line #78?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
77 float gdist(pcl::PointXYZ pt, const Eigen::Vector4f &v) {
78   return sqrt((pt.x - v(0)) * (pt.x - v(0)) + (pt.y - v(1)) * (pt.y - v(1)) +
79               (pt.z - v(2)) * (pt.z - v(2))); //
80 }
```

What are the units for **`return`** on line #78?

Your Answer: ***${q://QID5/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 2

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
40 void ContactDirection::adjustSpringConstant(double k) {
41   if (fabs(k) <= 1e-3) {
42     ROS_ERROR("Spring constant must be non-zero");
43   } else {
44     springConstant = k;
45   }
46 }
```

What are the units for **springConstant** on line #44?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
40 void ContactDirection::adjustSpringConstant(double k) {
41   if (fabs(k) <= 1e-3) {
42     ROS_ERROR("Spring constant must be non-zero");
43   } else {
44     springConstant = k;
45   }
46 }
```

What are the units for **springConstant** on line #44?

Your Answer: ***${q://QID7/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 3

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
122 /// Updates the vehicles position along the path.
123 ///
124 void DummyDrone::update(const ros::TimerEvent &event) {
125   boost::lock_guard<boost::mutex> guard(lock);
126   if (is_moving) // If task was not preempted go towards goal.
127   {
128     // Compute new position
129     double x_diff = goal_pose.position.x - current_pose.position.x;
130     double y_diff = goal_pose.position.y - current_pose.position.y;
131     double z_diff = goal_pose.position.z - current_pose.position.z;
132     double segment_length =
133         sqrt(x_diff * x_diff + y_diff * y_diff + z_diff * z_diff);
134     double ratio_to_consume = speed * _update_rate / segment_length;
135
136     if (1 <= ratio_to_consume) // We made the goal
137     {
138       current_pose.position.x = goal_pose.position.x;
139       current_pose.position.y = goal_pose.position.y;
140       current_pose.position.z = goal_pose.position.z;
141       is_moving = false;
142     } else {
143       current_pose.position.x =
144           current_pose.position.x + ratio_to_consume * x_diff;
145       current_pose.position.y =
146           current_pose.position.y + ratio_to_consume * y_diff;
147       current_pose.position.z =
148           current_pose.position.z + ratio_to_consume * z_diff;
149       // t = segment_length / speed;
150       // ROS_INFO_STREAM("Time to target: " << t << " seconds.");
151       // aseta_task_management::PhotoWaypointFeedback _feedback;
152       // _feedback.time_to_target = t;
153       // as.publishFeedback(_feedback);
154     }
155     flewn_path.push_back(current_pose);
156   }
157
158   // Publish the new path
159   publishPath();
160 }
```

What are the units for `ratio_to_consume` on line #134?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
122 /// Updates the vehicles position along the path.
123 ///
124 void DummyDrone::update(const ros::TimerEvent &event) {
125   boost::lock_guard<boost::mutex> guard(lock);
126   if (is_moving) // If task was not preempted go towards goal.
127   {
128     // Compute new position
129     double x_diff = goal_pose.position.x - current_pose.position.x;
130     double y_diff = goal_pose.position.y - current_pose.position.y;
131     double z_diff = goal_pose.position.z - current_pose.position.z;
132     double segment_length =
133         sqrt(x_diff * x_diff + y_diff * y_diff + z_diff * z_diff);
134     double ratio_to_consume = speed * _update_rate / segment_length;
135
136     if (1 <= ratio_to_consume) // We made the goal
137     {
138       current_pose.position.x = goal_pose.position.x;
139       current_pose.position.y = goal_pose.position.y;
140       current_pose.position.z = goal_pose.position.z;
141       is_moving = false;
142     } else {
143       current_pose.position.x =
144           current_pose.position.x + ratio_to_consume * x_diff;
145       current_pose.position.y =
146           current_pose.position.y + ratio_to_consume * y_diff;
147       current_pose.position.z =
148           current_pose.position.z + ratio_to_consume * z_diff;
149       // t = segment_length / speed;
150       // ROS_INFO_STREAM("Time to target: " << t << " seconds.");
151       // aseta_task_management::PhotoWaypointFeedback _feedback;
152       // _feedback.time_to_target = t;
153       // as.publishFeedback(_feedback);
154     }
155     flewn_path.push_back(current_pose);
156   }
157
158   // Publish the new path
159   publishPath();
160 }
```

What are the units for `ratio_to_consume` on line #134?

Your Answer: ***${q://QID12/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 4

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
30 void opticalflowCallback(const optical_flow::OpticalFlow::ConstPtr &flow) {
31   // Filling out the velocities
32   current_time = flow->header.stamp;
33   double dt = (current_time - last_time).toSec();
34   if (fabs(dt) > MAX_DELTA_TIME) {
35     last_time = current_time;
36     return; // This is for the first iteration when 'last_time' is undefined
37   }
38
39   vx = flow->velocity_x;
40   vy = flow->velocity_y;
41   z = flow->ground_distance;
42
43   double flow_accuracy =
44       1 / (flow->quality +
45            0.001); // 0.001 is for not dividing by zero (0.004 - 1000)
46   double delta_x = (vx * cos(th) - vy * sin(th)) * dt;
47   double delta_y = (vx * sin(th) + vy * cos(th)) * dt;
48   double delta_th = vth * dt;
```

Assume **`vx`** and **`vy`** are **`meters-per-second.`** What are the units for **`delta_x`** on line #46?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
30 void opticalflowCallback(const optical_flow::OpticalFlow::ConstPtr &flow) {
31   // Filling out the velocities
32   current_time = flow->header.stamp;
33   double dt = (current_time - last_time).toSec();
34   if (fabs(dt) > MAX_DELTA_TIME) {
35     last_time = current_time;
36     return; // This is for the first iteration when 'last_time' is undefined
37   }
38
39   vx = flow->velocity_x;
40   vy = flow->velocity_y;
41   z = flow->ground_distance;
42
43   double flow_accuracy =
44       1 / (flow->quality +
45            0.001); // 0.001 is for not dividing by zero (0.004 - 1000)
46   double delta_x = (vx * cos(th) - vy * sin(th)) * dt;
47   double delta_y = (vx * sin(th) + vy * cos(th)) * dt;
48   double delta_th = vth * dt;
```

Assume **`vx`** and **`vy`** are **`meters-per-second.`** What are the units for **`delta_x`** on line #46?

Your Answer: ***${q://QID15/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 5

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
98  void JoyController::joyCallback(const sensor_msgs::JoyConstPtr &joystick) {
99    geometry_msgs::Twist robotSpeed;
100   robotSpeed.linear.x = joystick->axes[1] * this->maxLinearVel;
101   robotSpeed.linear.y = 0;
102   robotSpeed.linear.z = 0;
103   robotSpeed.angular.x = 0;
104   robotSpeed.angular.y = 0;
105   robotSpeed.angular.z = joystick->axes[0] * this->maxAngularVel;
106
107   this->robotSpeedPub.publish(robotSpeed);
108 }
```

What are the units for `robotSpeed.angular` on line #105?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*


```
98  void JoyController::joyCallback(const sensor_msgs::JoyConstPtr &joystick) {
99    geometry_msgs::Twist robotSpeed;
100   robotSpeed.linear.x = joystick->axes[1] * this->maxLinearVel;
101   robotSpeed.linear.y = 0;
102   robotSpeed.linear.z = 0;
103   robotSpeed.angular.x = 0;
104   robotSpeed.angular.y = 0;
105   robotSpeed.angular.z = joystick->axes[0] * this->maxAngularVel;
106
107   this->robotSpeedPub.publish(robotSpeed);
108 }
```

What are the units for `robotSpeed.angular` on line #105?

Your Answer: ***${q://QID16/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 7

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
49 void EncoderCallback(const ras_arduino_msgs::Encoders::ConstPtr &msg) {
50   // obtain encoder data(sensor data)
51   delta_encoder1 = msg->delta_encoder1;
52   delta_encoder2 = msg->delta_encoder2;
53
54   // calculate estimated velocity of two wheels
55   estimated_w_left =
56       (delta_encoder1 * 2 * M_PI * control_frequency) / ticks_per_rev;
57   estimated_w_right =
58       (delta_encoder2 * 2 * M_PI * control_frequency) / ticks_per_rev;
59
60   // Update angular and linear velocity
61   w = -(estimated_w_right - estimated_w_left) * wheel_radius / base;
62   v = (estimated_w_right + estimated_w_left) * wheel_radius / 2;
63 }
```

What are the units for `w` on line #61?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
49 void EncoderCallback(const ras_arduino_msgs::Encoders::ConstPtr &msg) {
50   // obtain encoder data(sensor data)
51   delta_encoder1 = msg->delta_encoder1;
52   delta_encoder2 = msg->delta_encoder2;
53
54   // calculate estimated velocity of two wheels
55   estimated_w_left =
56       (delta_encoder1 * 2 * M_PI * control_frequency) / ticks_per_rev;
57   estimated_w_right =
58       (delta_encoder2 * 2 * M_PI * control_frequency) / ticks_per_rev;
59
60   // Update angular and linear velocity
61   w = -(estimated_w_right - estimated_w_left) * wheel_radius / base;
62   v = (estimated_w_right + estimated_w_left) * wheel_radius / 2;
63 }
```

What are the units for **w** on line #61?

Your Answer: ***${q://QID20/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 8

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
199 void ExpandMap::expandObstacle(const nav_msgs::OccupancyGrid &map_in) {
200   local_map = map_in;
201
202   vector<int8_t>::iterator itr;
203   for (itr = local_map.data.begin(); itr != local_map.data.end(); itr++) {
204     *itr = FREE;
205   }
206
207   for (int xi = 0; xi < (int)map_in.info.height; xi++) {
208     for (int yi = 0; yi < (int)map_in.info.width; yi++) {
209       // if the cell is LETHAL
210       if (map_in.data[xi + map_in.info.width * yi] != FREE) {
211         // expand the LETHAL cells with respect to the circle radius
212         list<MapIndex>::iterator litr;
213         for (litr = expanded_circle.begin(); litr != expanded_circle.end();
214              litr++) {
215           int x = xi + litr->i, y = yi + litr->j;
216           if (x >= 0 && x < (int)local_map.info.height && y >= 0 &&
217               y < (int)local_map.info.width &&
218               map_in.data[xi + map_in.info.width * yi] >
219                   local_map.data[x + map_in.info.width * y]) {
220             local_map.data[x + map_in.info.width * y] =
221                 map_in.data[xi + map_in.info.width * yi];
222           }
223         }
224       }
225     }
226   }
227 }
```

What are the units for **x** on line #215?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
199 void ExpandMap::expandObstacle(const nav_msgs::OccupancyGrid &map_in) {
200   local_map = map_in;
201
202   vector<int8_t>::iterator itr;
203   for (itr = local_map.data.begin(); itr != local_map.data.end(); itr++) {
204     *itr = FREE;
205   }
206
207   for (int xi = 0; xi < (int)map_in.info.height; xi++) {
208     for (int yi = 0; yi < (int)map_in.info.width; yi++) {
209       // if the cell is LETHAL
210       if (map_in.data[xi + map_in.info.width * yi] != FREE) {
211         // expand the LETHAL cells with respect to the circle radius
212         list<MapIndex>::iterator litr;
213         for (litr = expanded_circle.begin(); litr != expanded_circle.end();
214              litr++) {
215           int x = xi + litr->i, y = yi + litr->j;
216           if (x >= 0 && x < (int)local_map.info.height && y >= 0 &&
217               y < (int)local_map.info.width &&
218               map_in.data[xi + map_in.info.width * yi] >
219                   local_map.data[x + map_in.info.width * y]) {
220             local_map.data[x + map_in.info.width * y] =
221                 map_in.data[xi + map_in.info.width * yi];
222           }
223         }
224       }
225     }
226   }
227 }
```

What are the units for **x** on line #215?

Your Answer: *${q://QID22/ChoiceGroup/SelectedChoices}*

Explain why you made that selection:

Block 10

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
37 // constructor: fill in default param values (changeable via "set" fncs)
38 TrajBuilder::TrajBuilder() {
39   dt_ = default_dt; // 0.02; //send desired-state messages at fixed rate, e.g.
40                     // 0.02 sec = 50Hz
41   // dynamic parameters: should be tuned for target system
42   accel_max_ = default_accel_max;         // 0.5; //1m/sec^2
43   alpha_max_ = default_alpha_max;         // 0.2; //1 rad/sec^2
44   speed_max_ = default_speed_max;         // 1.0; //1 m/sec
45   omega_max_ = default_omega_max;         // 1.0; //1 rad/sec
46   path_move_tol_ = default_path_move_tol; // 0.01; // if path points are within
47                                           // 1cm, fuggidaboutit
48
49   // define a halt state; zero speed and spin, and fill with viable coords
50   halt_twist_.linear.x = 0.0;
51   halt_twist_.linear.y = 0.0;
52   halt_twist_.linear.z = 0.0;
53   halt_twist_.angular.x = 0.0;
54   halt_twist_.angular.y = 0.0;
55   halt_twist_.angular.z = 0.0;
56 }
```

What are the units for `path_move_tol_` on line #46?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
37 // constructor: fill in default param values (changeable via "set" fncs)
38 TrajBuilder::TrajBuilder() {
39   dt_ = default_dt; // 0.02; //send desired-state messages at fixed rate, e.g.
40                   // 0.02 sec = 50Hz
41   // dynamic parameters: should be tuned for target system
42   accel_max_ = default_accel_max;         // 0.5; //1m/sec^2
43   alpha_max_ = default_alpha_max;         // 0.2; //1 rad/sec^2
44   speed_max_ = default_speed_max;         // 1.0; //1 m/sec
45   omega_max_ = default_omega_max;         // 1.0; //1 rad/sec
46   path_move_tol_ = default_path_move_tol; // 0.01; // if path points are within
47                                           // 1cm, fuggidaboutit
48 
49   // define a halt state; zero speed and spin, and fill with viable coords
50   halt_twist_.linear.x = 0.0;
51   halt_twist_.linear.y = 0.0;
52   halt_twist_.linear.z = 0.0;
53   halt_twist_.angular.x = 0.0;
54   halt_twist_.angular.y = 0.0;
55   halt_twist_.angular.z = 0.0;
56 }
```

What are the units for `path_move_tol_` on line #46?

Your Answer: ***${q://QID28/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 11

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
134 int fazTrajetoria(Robo *barco, geometry_msgs::Pose2D *posicao_objetivo) {
135
136   int i, delta_x, delta_y;
137   float delta_d;
138   float theta, traj_x, traj_y;
139   float vel_linear, vel_angular;
140
141   // definindo angulo trajetoria
142   delta_y = posicao_objetivo->y - barco->getPosicao().y;
143   delta_x = posicao_objetivo->x - barco->getPosicao().x;
144   theta = atan(delta_y / delta_x) * 180 / PI; // angulo em GRAUS;
145
146   barco->limpaTrajetoria();
147
148   // calculando distância
149   delta_d = distancia2pts(barco->getPosicao(), *posicao_objetivo);
150
151   for (i = 1; i <= (delta_d / RESOLUCAO); i++) {
152
153     traj_x = cos(RESOLUCAO) * i + barco->getPosicao().x;
154     traj_y = sin(RESOLUCAO) * i + barco->getPosicao().y;
155
156     barco->addPontoTrajetoria(traj_x, traj_y, theta);
157   }
158
159   // velocidade linear igual a zero, o barco so vai rodar
160   vel_angular = 0.0;
```

What are the units for `delta_d` on line #149?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
134 int fazTrajetoria(Robo *barco, geometry_msgs::Pose2D *posicao_objetivo) {
135
136   int i, delta_x, delta_y;
137   float delta_d;
138   float theta, traj_x, traj_y;
139   float vel_linear, vel_angular;
140
141   // definindo angulo trajetoria
142   delta_y = posicao_objetivo->y - barco->getPosicao().y;
143   delta_x = posicao_objetivo->x - barco->getPosicao().x;
144   theta = atan(delta_y / delta_x) * 180 / PI; // angulo em GRAUS;
145
146   barco->limpaTrajetoria();
147
148   // calculando distância
149   delta_d = distancia2pts(barco->getPosicao(), *posicao_objetivo);
150
151   for (i = 1; i <= (delta_d / RESOLUCAO); i++) {
152
153     traj_x = cos(RESOLUCAO) * i + barco->getPosicao().x;
154     traj_y = sin(RESOLUCAO) * i + barco->getPosicao().y;
155
156     barco->addPontoTrajetoria(traj_x, traj_y, theta);
157   }
158
159   // velocidade linear igual a zero, o barco so vai rodar
160   vel_angular = 0.0;
```

What are the units for `delta_d` on line #149?

Your Answer: ***${q://QID30/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 12

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
37 void msgCallback(const boost::shared_ptr<const geometry_msgs::WrenchStamped>
38                      &wrench_ptr) // FIXME Add the torques!!!
39 {
40
41   geometry_msgs::WrenchStamped wrench_out;
42   geometry_msgs::PointStamped tmp_point_in;
43   geometry_msgs::PointStamped tmp_point_out;
44
45   try {
46     tmp_point_in.header = wrench_ptr->header;
47     tmp_point_in.point.x = wrench_ptr->wrench.force.x;
48     tmp_point_in.point.y = wrench_ptr->wrench.force.y;
49     tmp_point_in.point.z = wrench_ptr->wrench.force.z;
50
51     tf_.transformPoint(target_frame_, tmp_point_in, tmp_point_out);
52
53     wrench_out.header = tmp_point_out.header;
54     wrench_out.wrench.force.x = tmp_point_out.point.x;
55     wrench_out.wrench.force.y = tmp_point_out.point.y;
56     wrench_out.wrench.force.z = tmp_point_out.point.z;
57
58     publisher_.publish(wrench_out);
59
60   } catch (tf::TransformException &ex) {
61     printf("Failure %s\n", ex.what()); // Print exception which was caught
62   }
63 };
```

What are the units for **`wrench_out.wrench.force.y`** on line #55?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
37 void msgCallback(const boost::shared_ptr<const geometry_msgs::WrenchStamped>
38                      &wrench_ptr) // FIXME Add the torques!!!
39 {
40
41   geometry_msgs::WrenchStamped wrench_out;
42   geometry_msgs::PointStamped tmp_point_in;
43   geometry_msgs::PointStamped tmp_point_out;
44
45   try {
46     tmp_point_in.header = wrench_ptr->header;
47     tmp_point_in.point.x = wrench_ptr->wrench.force.x;
48     tmp_point_in.point.y = wrench_ptr->wrench.force.y;
49     tmp_point_in.point.z = wrench_ptr->wrench.force.z;
50
51     tf_.transformPoint(target_frame_, tmp_point_in, tmp_point_out);
52
53     wrench_out.header = tmp_point_out.header;
54     wrench_out.wrench.force.x = tmp_point_out.point.x;
55     wrench_out.wrench.force.y = tmp_point_out.point.y;
56     wrench_out.wrench.force.z = tmp_point_out.point.z;
57
58     publisher_.publish(wrench_out);
59
60   } catch (tf::TransformException &ex) {
61     printf("Failure %s\n", ex.what()); // Print exception which was caught
62   }
63 };
```

What are the units for `wrench_out.wrench.force.y` on line #55?

Your Answer: ***${q://QID32/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 13

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
1  void onImuData(logImu *data) {
2    sensor_msgs::Imu msg;
3    msg.header.stamp = ros::Time::now();
4    msg.header.frame_id = m_tf_prefix + "/base_link";
5    msg.orientation_covariance[0] = -1;
6
7    msg.orientation.x =
8        cos(data->yaw / 2) * cos(data->pitch / 2) * cos(data->roll / 2) +
9        sin(data->yaw / 2) * sin(data->pitch / 2) * sin(data->roll / 2);
10   msg.orientation.y =
11       sin(data->yaw / 2) * cos(data->pitch / 2) * cos(data->roll / 2) -
12       cos(data->yaw / 2) * sin(data->pitch / 2) * sin(data->roll / 2);
13   msg.orientation.z =
14       cos(data->yaw / 2) * sin(data->pitch / 2) * cos(data->roll / 2) +
15       sin(data->yaw / 2) * cos(data->pitch / 2) * sin(data->roll / 2);
16   msg.orientation.w =
17       cos(data->yaw / 2) * cos(data->pitch / 2) * sin(data->roll / 2) -
18       sin(data->yaw / 2) * sin(data->pitch / 2) * cos(data->roll / 2);
19
20   // measured in deg/s; need to convert to rad/s
21   msg.angular_velocity.x = data->roll;  // degToRad(data->roll);
22   msg.angular_velocity.y = data->pitch; // degToRad(data->pitch);
23   msg.angular_velocity.z = data->yaw;   // degToRad(data->yaw);
24
25   msg.linear_acceleration.x = data->gyro_x;
26   msg.linear_acceleration.y = data->gyro_y;
27   msg.linear_acceleration.z = data->gyro_z;
```

What are the units for **`data->gyro_z`** on line #26?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
1  void onImuData(logImu *data) {
2    sensor_msgs::Imu msg;
3    msg.header.stamp = ros::Time::now();
4    msg.header.frame_id = m_tf_prefix + "/base_link";
5    msg.orientation_covariance[0] = -1;
6
7    msg.orientation.x =
8        cos(data->yaw / 2) * cos(data->pitch / 2) * cos(data->roll / 2) +
9        sin(data->yaw / 2) * sin(data->pitch / 2) * sin(data->roll / 2);
10   msg.orientation.y =
11       sin(data->yaw / 2) * cos(data->pitch / 2) * cos(data->roll / 2) -
12       cos(data->yaw / 2) * sin(data->pitch / 2) * sin(data->roll / 2);
13   msg.orientation.z =
14       cos(data->yaw / 2) * sin(data->pitch / 2) * cos(data->roll / 2) +
15       sin(data->yaw / 2) * cos(data->pitch / 2) * sin(data->roll / 2);
16   msg.orientation.w =
17       cos(data->yaw / 2) * cos(data->pitch / 2) * sin(data->roll / 2) -
18       sin(data->yaw / 2) * sin(data->pitch / 2) * cos(data->roll / 2);
19
20   // measured in deg/s; need to convert to rad/s
21   msg.angular_velocity.x = data->roll;  // degToRad(data->roll);
22   msg.angular_velocity.y = data->pitch; // degToRad(data->pitch);
23   msg.angular_velocity.z = data->yaw;   // degToRad(data->yaw);
24
25   msg.linear_acceleration.x = data->gyro_x;
26   msg.linear_acceleration.y = data->gyro_y;
27   msg.linear_acceleration.z = data->gyro_z;
```

What are the units for `data->gyro_z` on line #26?

Your Answer: ***${q://QID37/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 14

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
42 geometry_msgs::Pose xyPhi2Pose(double x, double y, double phi) {
43     geometry_msgs::Pose pose; // a pose object to populate
44     // convert from heading to corresponding quaternion
45     pose.orientation = convertPlanarPhi2Quaternion(phi);
46     pose.position.x = x; // keep the robot on the ground!
47     pose.position.y = y; // keep the robot on the ground!
48     pose.position.z = 0.0; // keep the robot on the ground!
49     return pose;
50 }
```

What are the units for **`pose.orientation`** on line #45?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*


```
42 geometry_msgs::Pose xyPhi2Pose(double x, double y, double phi) {
43     geometry_msgs::Pose pose; // a pose object to populate
44     // convert from heading to corresponding quaternion
45     pose.orientation = convertPlanarPhi2Quaternion(phi);
46     pose.position.x = x; // keep the robot on the ground!
47     pose.position.y = y; // keep the robot on the ground!
48     pose.position.z = 0.0; // keep the robot on the ground!
49     return pose;
50 }
```

What are the units for **`pose.orientation`** on line #45?

Your Answer: ***${q://QID39/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 15

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
119 imu_message_.header.frame_id = frame_id_;
120
121 // Angular velocity measurement covariance.
122 imu_message_.angular_velocity_covariance[0] =
123     imu_parameters_.gyroscope_noise_density *
124     imu_parameters_.gyroscope_noise_density;
125 imu_message_.angular_velocity_covariance[4] =
126     imu_parameters_.gyroscope_noise_density *
127     imu_parameters_.gyroscope_noise_density;
128 imu_message_.angular_velocity_covariance[8] =
129     imu_parameters_.gyroscope_noise_density *
130     imu_parameters_.gyroscope_noise_density;
131 // Linear acceleration measurement covariance.
132 imu_message_.linear_acceleration_covariance[0] =
133     imu_parameters_.accelerometer_noise_density *
134     imu_parameters_.accelerometer_noise_density;
135 imu_message_.linear_acceleration_covariance[4] =
136     imu_parameters_.accelerometer_noise_density *
137     imu_parameters_.accelerometer_noise_density;
138 imu_message_.linear_acceleration_covariance[8] =
139     imu_parameters_.accelerometer_noise_density *
140     imu_parameters_.accelerometer_noise_density;
141 // Orientation estimate covariance (no estimate provided).
142 imu_message_.orientation_covariance[0] = -1.0;
```

What are the units for `imu_message_.angular_velocity_covariance[8]` on line #128?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
119 imu_message_.header.frame_id = frame_id_;
120
121 // Angular velocity measurement covariance.
122 imu_message_.angular_velocity_covariance[0] =
123     imu_parameters_.gyroscope_noise_density *
124     imu_parameters_.gyroscope_noise_density;
125 imu_message_.angular_velocity_covariance[4] =
126     imu_parameters_.gyroscope_noise_density *
127     imu_parameters_.gyroscope_noise_density;
128 imu_message_.angular_velocity_covariance[8] =
129     imu_parameters_.gyroscope_noise_density *
130     imu_parameters_.gyroscope_noise_density;
131 // Linear acceleration measurement covariance.
132 imu_message_.linear_acceleration_covariance[0] =
133     imu_parameters_.accelerometer_noise_density *
134     imu_parameters_.accelerometer_noise_density;
135 imu_message_.linear_acceleration_covariance[4] =
136     imu_parameters_.accelerometer_noise_density *
137     imu_parameters_.accelerometer_noise_density;
138 imu_message_.linear_acceleration_covariance[8] =
139     imu_parameters_.accelerometer_noise_density *
140     imu_parameters_.accelerometer_noise_density;
141 // Orientation estimate covariance (no estimate provided).
142 imu_message_.orientation_covariance[0] = -1.0;
```

What are the units for `imu_message_.angular_velocity_covariance[8]` on line #128?

Your Answer: ***${q://QID46/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 16

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
37 map.data.resize(map.info.width *map.info.height);
38 for (int i = 0; i < map.info.width * map.info.height; i++) {
39   xi = map.info.origin.position.x + (i % map.info.width) * map.info.resolution
         +
40       map.info.resolution / 2;
41   yi = map.info.origin.position.y + (i / map.info.width) * map.info.resolution
         +
42       map.info.resolution / 2;
43
44   if (xi < range && xi > 0 && fabs(yi / xi) < tan(field_of_view / 2)) {
45     map.data[i] = 0;
46   } else {
47     map.data[i] = -1;
48   }
49 }
```

What are the units for `xi` on line #39?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*


```
37 map.data.resize(map.info.width *map.info.height);
38 for (int i = 0; i < map.info.width * map.info.height; i++) {
39   xi = map.info.origin.position.x + (i % map.info.width) * map.info.resolution
         +
40       map.info.resolution / 2;
41   yi = map.info.origin.position.y + (i / map.info.width) * map.info.resolution
         +
42       map.info.resolution / 2;
43
44   if (xi < range && xi > 0 && fabs(yi / xi) < tan(field_of_view / 2)) {
45     map.data[i] = 0;
46   } else {
47     map.data[i] = -1;
48   }
49 }
```

What are the units for `xi` on line #39?

Your Answer: ***${q://QID47/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 17

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
76 double delta_LookAhead = 0.0;
77 double m_len_c2r = 1.10;
78 double turning_rad = 0;
79 double heading = 0.0;
80 // int lookAheadIndex = 0;
81 double a = 0.19;
82
83 if (m_LocalSplinePath.size() > 1) {
84
85   double minDist = 99999;
86   int cIndex = -1;
87   int lookIndex = -1;
88   for (int i = 0; i < m_LocalSplinePath.size(); i++) {
89     double x2 = m_LocalSplinePath[i][0] - m_pos[0];
90     x2 *= x2;
91     double y2 = m_LocalSplinePath[i][1] - m_pos[1];
92     y2 *= y2;
93     double dist = sqrt(x2 + y2); // distŽÂ
94     if (dist < minDist) {
95       minDist = dist;
96       cIndex = i;
97     }
98   }
```

If the units for `m_pos[0]` on line #89 are **meters**, what are the units for `x2` on line #93?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*

Click Count: *0 clicks*

```
76 double delta_LookAhead = 0.0;
77 double m_len_c2r = 1.10;
78 double turning_rad = 0;
79 double heading = 0.0;
80 // int lookAheadIndex = 0;
81 double a = 0.19;
82
83 if (m_LocalSplinePath.size() > 1) {
84
85   double minDist = 99999;
86   int cIndex = -1;
87   int lookIndex = -1;
88   for (int i = 0; i < m_LocalSplinePath.size(); i++) {
89     double x2 = m_LocalSplinePath[i][0] - m_pos[0];
90     x2 *= x2;
91     double y2 = m_LocalSplinePath[i][1] - m_pos[1];
92     y2 *= y2;
93     double dist = sqrt(x2 + y2); // distŽÂ
94     if (dist < minDist) {
95       minDist = dist;
96       cIndex = i;
97     }
98   }
```

If the units for `m_pos[0]` on line #89 are **meters**, what are the units for `x2` on line #93?

Your Answer: ***${q://QID48/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 18

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
136 void send_attitude_ang_velocity(const ros::Time &stamp,
137                                 const Eigen::Vector3d &ang_vel) {
138   /* Q + Thrust, also bits noumbering started from 1 in docs
139    */
140   const uint8_t ignore_all_except_rpy = (1 << 7) | (1 << 6);
141   float q[4] = {1.0, 0.0, 0.0, 0.0};
142
143   auto av = UAS::transform_frame_baselink_aircraft(ang_vel);
144
145   set_attitude_target(stamp.toNSec() / 1000000, ignore_all_except_rpy, q,
146                       av.x(), av.y(), av.z(), 0.0);
147 }
```

What are the units for `av` on line #143?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*


```
136 void send_attitude_ang_velocity(const ros::Time &stamp,
137                                 const Eigen::Vector3d &ang_vel) {
138   /* Q + Thrust, also bits noumbering started from 1 in docs
139    */
140   const uint8_t ignore_all_except_rpy = (1 << 7) | (1 << 6);
141   float q[4] = {1.0, 0.0, 0.0, 0.0};
142
143   auto av = UAS::transform_frame_baselink_aircraft(ang_vel);
144
145   set_attitude_target(stamp.toNSec() / 1000000, ignore_all_except_rpy, q,
146                       av.x(), av.y(), av.z(), 0.0);
147 }
```

What are the units for `av` on line #143?

Your Answer: ***${q://QID53/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 19

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
245 for (unsigned int k = 0; k < pts; ++k) {
246   // rotate into user specified frame.
247   // frame_rot is identity if world is used.
248   math::Vector3 force = frame_rot.RotateVectorReverse(math::Vector3(
249       contact.forces[k].body1Force.x, contact.forces[k].body1Force.y,
250       contact.forces[k].body1Force.z));
251   math::Vector3 torque = frame_rot.RotateVectorReverse(math::Vector3(
252       contact.forces[k].body1Torque.x, contact.forces[k].body1Torque.y,
253       contact.forces[k].body1Torque.z));
```

What are the units for `torque` on line #251?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
245 for (unsigned int k = 0; k < pts; ++k) {
246   // rotate into user specified frame.
247   // frame_rot is identity if world is used.
248   math::Vector3 force = frame_rot.RotateVectorReverse(math::Vector3(
249       contact.forces[k].body1Force.x, contact.forces[k].body1Force.y,
250       contact.forces[k].body1Force.z));
251   math::Vector3 torque = frame_rot.RotateVectorReverse(math::Vector3(
252       contact.forces[k].body1Torque.x, contact.forces[k].body1Torque.y,
253       contact.forces[k].body1Torque.z));
```

What are the units for torque on line #251?

Your Answer: ***${q://QID55/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 20

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
129 if (wrench_.wrench.force.z > 0.0) {
130
131   double nominal_thrust_per_motor = wrench_.wrench.force.z / 4.0;
132   motor_.force[0] = nominal_thrust_per_motor -
133                     wrench_.wrench.torque.y / 2.0 / parameters_.lever;
134   motor_.force[1] = nominal_thrust_per_motor -
135                     wrench_.wrench.torque.x / 2.0 / parameters_.lever;
136   motor_.force[2] = nominal_thrust_per_motor +
137                     wrench_.wrench.torque.y / 2.0 / parameters_.lever;
138   motor_.force[3] = nominal_thrust_per_motor +
139                     wrench_.wrench.torque.x / 2.0 / parameters_.lever;
140
141   double nominal_torque_per_motor = wrench_.wrench.torque.z / 4.0;
142   motor_.voltage[0] = motor_.force[0] / parameters_.force_per_voltage +
143                       nominal_torque_per_motor / parameters_.torque_per_voltage;
144   motor_.voltage[1] = motor_.force[1] / parameters_.force_per_voltage -
145                       nominal_torque_per_motor / parameters_.torque_per_voltage;
146   motor_.voltage[2] = motor_.force[2] / parameters_.force_per_voltage +
147                       nominal_torque_per_motor / parameters_.torque_per_voltage;
148   motor_.voltage[3] = motor_.force[3] / parameters_.force_per_voltage -
149                       nominal_torque_per_motor / parameters_.torque_per_voltage;
```

What are the units for the expression `nominal_torque_per_motor / paramters_.torque_per_voltage` on line #144?

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
129 if (wrench_.wrench.force.z > 0.0) {
130
131   double nominal_thrust_per_motor = wrench_.wrench.force.z / 4.0;
132   motor_.force[0] = nominal_thrust_per_motor -
133                     wrench_.wrench.torque.y / 2.0 / parameters_.lever;
134   motor_.force[1] = nominal_thrust_per_motor -
135                     wrench_.wrench.torque.x / 2.0 / parameters_.lever;
136   motor_.force[2] = nominal_thrust_per_motor +
137                     wrench_.wrench.torque.y / 2.0 / parameters_.lever;
138   motor_.force[3] = nominal_thrust_per_motor +
139                     wrench_.wrench.torque.x / 2.0 / parameters_.lever;
140
141   double nominal_torque_per_motor = wrench_.wrench.torque.z / 4.0;
142   motor_.voltage[0] = motor_.force[0] / parameters_.force_per_voltage +
143                       nominal_torque_per_motor / parameters_.torque_per_voltage;
144   motor_.voltage[1] = motor_.force[1] / parameters_.force_per_voltage -
145                       nominal_torque_per_motor / parameters_.torque_per_voltage;
146   motor_.voltage[2] = motor_.force[2] / parameters_.force_per_voltage +
147                       nominal_torque_per_motor / parameters_.torque_per_voltage;
148   motor_.voltage[3] = motor_.force[3] / parameters_.force_per_voltage -
149                       nominal_torque_per_motor / parameters_.torque_per_voltage;
```

What are the units for the expression `nominal_torque_per_motor / paramters_.torque_per_voltage` on line #144?

Your Answer: ***${q://QID56/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 21

These page timer metrics will not be displayed to the recipient.
First Click: *0 seconds*
Last Click: *0 seconds*
Page Submit: *0 seconds*
Click Count: *0 clicks*

```
28 // Function for conversion of quaternion to roll pitch and yaw. The angles
29 // are published here too.
30 void MsgCallback(const geometry_msgs::PoseStamped msg) {
31   geometry_msgs::Quaternion GMquat;
32   GMquat = msg.pose.orientation;
33
34   // the incoming geometry_msgs::Quaternion is transformed to a tf::Quaterion
35   tf::Quaternion quat, quattemp;
36   tf::quaternionMsgToTF(GMquat, quattemp);
37   //     ROS_INFO("quat.x =%f, quat.y=%f, quat.z=%f, quat.w=%f", quattemp.x(),
38   //     quattemp.y(), quattemp.z(),quattemp.w());
39   quat =
40       tf::Quaternion(quattemp.x(), -quattemp.z(), quattemp.y(), quattemp.w());
41
42   // the tf::Quaternion has a method to acess roll pitch and yaw
43   double roll, pitch, yaw;
44   tf::Matrix3x3(quat).getRPY(roll, pitch, yaw);
45
46   // the found angles are written in a geometry_msgs::Vector3
47   geometry_msgs::Vector3 anglesmsg;
48   anglesmsg.z = yaw;
49   anglesmsg.y = roll;
50   anglesmsg.x = -pitch;
51
52   // this Vector is then published:
53   rpy_publisher.publish(anglesmsg);
54   ROS_INFO("published pitch=%.1f, roll=%.1f,  yaw=%.1f",
55            anglesmsg.x * 180 / 3.1415926, anglesmsg.y * 180 / 3.1415926,
56            anglesmsg.z * 180 / 3.1415926);
57 }
```

What are the units for **anglesmsg.z** on line #48?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
28 // Function for conversion of quaternion to roll pitch and yaw. The angles
29 // are published here too.
30 void MsgCallback(const geometry_msgs::PoseStamped msg) {
31   geometry_msgs::Quaternion GMquat;
32   GMquat = msg.pose.orientation;
33
34   // the incoming geometry_msgs::Quaternion is transformed to a tf::Quaterion
35   tf::Quaternion quat, quattemp;
36   tf::quaternionMsgToTF(GMquat, quattemp);
37   //    ROS_INFO("quat.x =%f, quat.y=%f, quat.z=%f, quat.w=%f", quattemp.x(),
38   //    quattemp.y(), quattemp.z(),quattemp.w());
39   quat =
40       tf::Quaternion(quattemp.x(), -quattemp.z(), quattemp.y(), quattemp.w());
41
42   // the tf::Quaternion has a method to acess roll pitch and yaw
43   double roll, pitch, yaw;
44   tf::Matrix3x3(quat).getRPY(roll, pitch, yaw);
45
46   // the found angles are written in a geometry_msgs::Vector3
47   geometry_msgs::Vector3 anglesmsg;
48   anglesmsg.z = yaw;
49   anglesmsg.y = roll;
50   anglesmsg.x = -pitch;
51
52   // this Vector is then published:
53   rpy_publisher.publish(anglesmsg);
54   ROS_INFO("published pitch=%.1f, roll=%.1f,  yaw=%.1f",
55            anglesmsg.x * 180 / 3.1415926, anglesmsg.y * 180 / 3.1415926,
56            anglesmsg.z * 180 / 3.1415926);
57 }
```

What are the units for **`anglesmsg.z`** on line #48?

Your Answer: ***${q://QID57/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:

Block 22

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

Click Count: *0 clicks*

```
48 int lticksl_ = lenc_.GetTicks(), rticksl_ = renc_.GetTicks();
49 ros::Time tl_ = ros::Time::now();
50 while (ros::ok()) {
51   // wheel/time delta
52   int lticksc_ = lenc_.GetTicks(), rticksc_ = renc_.GetTicks();
53   odom_.header.stamp = ros::Time::now();
54   double dt_ = (odom_.header.stamp - tl_).toSec();
55   int dl_ = lticksc_ - lticksl_, dr_ = rticksc_ - rticksl_;
56   if (dl_ > 1e6) {
57     if (lticksc_ < 1e6)
58       dl_ = lticksc_ + (MAX_ENCODER_VALUE - lticksl_);
59     else
60       dl_ = lticksl_ + (MAX_ENCODER_VALUE - lticksc_);
61   }
62   dl_ *= -1; // left wheel encoder spins CCW
63   if (dr_ > 1e6) {
64     if (rticksc_ < 1e6)
65       dr_ = rticksc_ + (MAX_ENCODER_VALUE - rticksl_);
66     else
67       dr_ = rticksl_ + (MAX_ENCODER_VALUE - rticksc_);
68   }
69   lticksl_ = lticksc_;
70   rticksl_ = rticksc_;
71   tl_ = odom_.header.stamp;
72
73   // wheel/platform movement
74   double ml_ = dl_ * M_PER_TICK, mr_ = dr_ * M_PER_TICK;
75   double dpos_ = (ml_ + mr_) / 2, dyaw_ = (mr_ - ml_) / AXLE_LENGTH;
```

What are the units for `dyaw_` on line #75?

These page timer metrics will not be displayed to the recipient.

First Click: *0 seconds*

Last Click: *0 seconds*

Page Submit: *0 seconds*

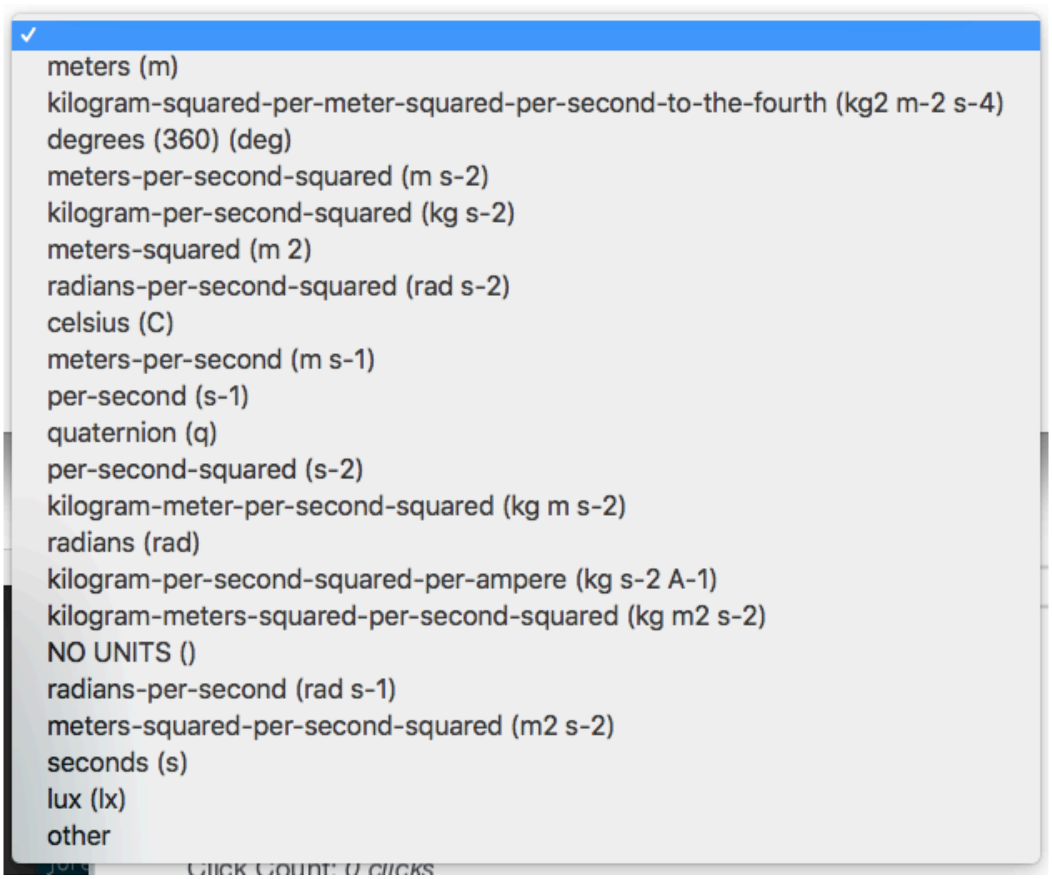
Click Count: *0 clicks*

```
48 int lticksl_ = lenc_.GetTicks(), rticksl_ = renc_.GetTicks();
49 ros::Time tl_ = ros::Time::now();
50 while (ros::ok()) {
51   // wheel/time delta
52   int lticksc_ = lenc_.GetTicks(), rticksc_ = renc_.GetTicks();
53   odom_.header.stamp = ros::Time::now();
54   double dt_ = (odom_.header.stamp - tl_).toSec();
55   int dl_ = lticksc_ - lticksl_, dr_ = rticksc_ - rticksl_;
56   if (dl_ > 1e6) {
57     if (lticksc_ < 1e6)
58       dl_ = lticksc_ + (MAX_ENCODER_VALUE - lticksl_);
59     else
60       dl_ = lticksl_ + (MAX_ENCODER_VALUE - lticksc_);
61   }
62   dl_ *= -1; // left wheel encoder spins CCW
63   if (dr_ > 1e6) {
64     if (rticksc_ < 1e6)
65       dr_ = rticksc_ + (MAX_ENCODER_VALUE - rticksl_);
66     else
67       dr_ = rticksl_ + (MAX_ENCODER_VALUE - rticksc_);
68   }
69   lticksl_ = lticksc_;
70   rticksl_ = rticksc_;
71   tl_ = odom_.header.stamp;
72
73   // wheel/platform movement
74   double ml_ = dl_ * M_PER_TICK, mr_ = dr_ * M_PER_TICK;
75   double dpos_ = (ml_ + mr_) / 2, dyaw_ = (mr_ - ml_) / AXLE_LENGTH;
```

What are the units for `dyaw_` on line #75?

Your Answer: ***${q://QID59/ChoiceGroup/SelectedChoices}***

Explain why you made that selection:


meters (m)
kilogram-squared-per-meter-squared-per-second-to-the-fourth (kg2 m-2 s-4)
degrees (360) (deg)
meters-per-second-squared (m s-2)
kilogram-per-second-squared (kg s-2)
meters-squared (m 2)
radians-per-second-squared (rad s-2)
celsius (C)
meters-per-second (m s-1)
per-second (s-1)
quaternion (q)
per-second-squared (s-2)
kilogram-meter-per-second-squared (kg m s-2)
radians (rad)
kilogram-per-second-squared-per-ampere (kg s-2 A-1)
kilogram-meters-squared-per-second-squared (kg m2 s-2)
NO UNITS ()
radians-per-second (rad s-1)
meters-squared-per-second-squared (m2 s-2)
seconds (s)
lux (lx)
other

E Code Artifacts and Questions for Developer Study of Inconsistency Severity

Finding Unit Inconsistencies between the Physical World and ROS Programs

We have found that many system failures arise when units present in the physical world are instantiated and manipulated in programs. We believe that part of the problem can be attributed to the fact that programs know little about the meaning of those units in the physical world.

Take the following example for instance. The following code will be deemed as correct by the compiler even though the quantities being added in Line 191 are incompatible. The resulting system will then be adding meters and meters squared which have no meaning in the physical world, and would likely constitute a fault in this program.

Example: adding inconsistent units

```
188
189 float computeDistance(geometry_msgs::Pose goal, geometry_msgs::Pose current)
190 {                                                                        meters²
191   float dist = (goal.position.x - current.position.x)*(goal.position.x - current.position.x)
192       + (goal.position.y - current.position.y) + (goal.position.y - current.position.y)
193       + (goal.position.z - current.position.z) + (goal.position.z - current.position.z);
194                                              meters
195   return dist;
196 }
197
```

We have built a tool that can find such unit inconsistencies in ROS code. The tool consumes source code and produces error messages when an inconsistency is found.

We now need your help in assessing whether this type unit mismatches between physical and program types are problematic in that they may cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems.

INSTRUCTIONS:
Please examine the following snippets of code (from real robots), indicate whether the type inconsistency is problematic, and justify your response.

There are a total of 8 snippets, representing different inconsistencies found by our tool. The assessment should take around 20 minutes.

If you have any questions, just email me at jore@cse.unl.edu

Thanks.

John-Paul

1.

m s^{-1} m

```
463
464  //pass along drive commands
465  cmd_vel.linear.x = drive_cmds.getOrigin().getX();
466  cmd_vel.linear.y = drive_cmds.getOrigin().getY();
467  yaw = tf::getYaw(drive_cmds.getRotation());
468
```

Inconsistent unit assignment to Twist.linear.x (meters per second) from Vector.x (meters)

1. **1a. Is the unit inconsistency on line 465 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**
   Full code available at: https://github.com/ros-planning/navigation/blob/ef077ba7c1b2870d030fece96103c1a41f21da8b/base_local_planner/src/trajectory_planner_ros.cpp#L465
   Mark only one oval.
   - ◯ Yes
   - ◯ No
   - ◯ Maybe

2. **1b. Briefly explain your answer**

2.

```
cmd_vel_inc = target_vel.linear.y - last_cmd_vel.linear.y;
if (odometry_vel.linear.y*target_vel.linear.y < 0.0)
{
  max_vel_inc = decel_lim_y*period;   // countermarch
}
else
{
  max_vel_inc = ((cmd_vel_inc*target_vel.linear.y > 0.0)?accel_lim_y:decel_lim_y *period;
}
if (std::abs(cmd_vel_inc) > max_vel_inc)
{
  cmd_vel.linear.y = last_cmd_vel.linear.y + sign(cmd_vel_inc)*max_vel_inc;
}

cmd_vel_inc = target_vel.angular.z - last_cmd_vel.angular.z;
```

m s^{-1}

rad s^{-1}

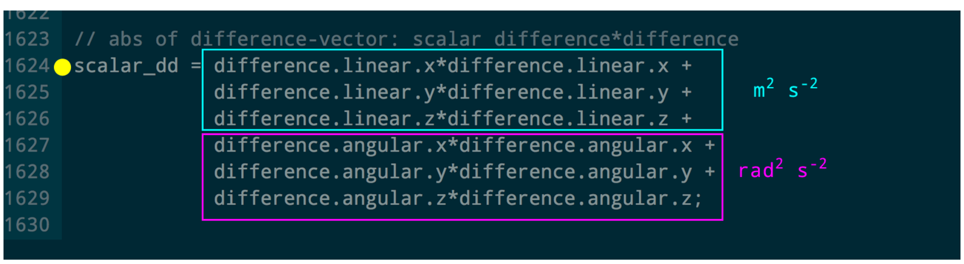
Reuse of cmd_vel_inc with different units might make the code more difficult to maintain.

3. **2a. Is the unit inconsistency on line 139 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**
Full code available at:
https://github.com/yujinrobot/yujin_ocs/blob/5e008dadc43272904fc26f07c34ddb9ced624094/yocs_velocity_smoother/src/velocity_smoother_nodelet.cpp#L139
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

4. **2b. Briefly explain your answer**

3.



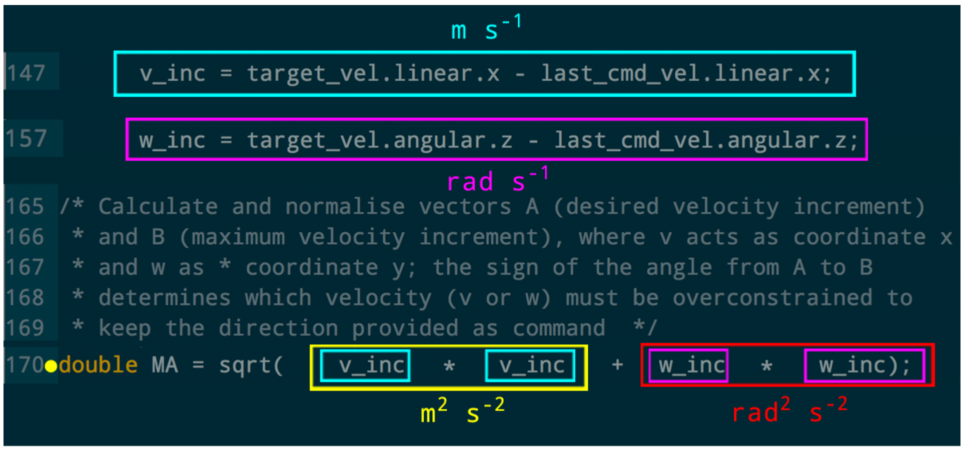
The terms in this addition do not have the same units.

5. **3a. Is the unit inconsistency on line 1624 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**
Full code available at: https://github.com/ros-planning/navigation_experimental/blob/39bedf6001dab97f39e6352ed369c8cf512fa1c7/eband_local_planner/src/eband_local_planner.cpp#L1621
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Maybe

6. **3b. Briefly explain your answer**

4.



The terms in this addition do not have the same units.

7. **4a. Is the unit inconsistency on line 170 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**

   Full code available at:
   https://github.com/yujinrobot/yujin_ocs/blob/92ef8086e61ff727bdfaaeaf0d8a068377db8272/yocs_velocity_smoother/src/velocity_smoother_nodelet.cpp#L170

   Mark only one oval.

   - Yes
   - No
   - Maybe

8. **4b. Briefly explain your answer**

5.

```
181  #define VX  vs.twist.linear.x   m s-1
182  #define VY  vs.twist.linear.y
183  #define VZ  vs.twist.linear.z
184  #define sp_X  pos_setpoint().x
       m s-1
390   VX = setpoint.pose.position.x - current_x;   m
391   VY = setpoint.pose.position.y - current_y;
392   VZ = setpoint.pose.position.z - current_z;
393   vel_sp_pub.publish(vs);
394  }
```

The assignment of meters in line 390 is inconsistent with meters per second expected for Twist.linear.x

9. **5a. Is the unit inconsistency on line 390 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**

Full code available at:
https://github.com/mavlink/mavros/blob/c2d4d13b7b6c436fbfdc6a9397817164d463a089/test_mavros/include/tests/offboard_control.h#L390

Mark only one oval.

- Yes
- No
- Maybe

10. **5b. Briefly explain your answer**

6.

```
double alpha_min = alpha.linear.x;
if(alpha.linear.y < alpha_min)
  alpha_min = alpha.linear.y; ...... m s-1
if(alpha.angular.z < alpha_min)        or
  alpha_min = alpha.angular.z; ...... rad s-1

result.linear.x = start.linear.x + alpha_min * (end.linear.x - start.linear.x);
  m s-1             m s-1                       m s-1          m s-1
```

alpha_min could be two different units, and in either case the units are inconsistent during addition.

11. **6a. Is the unit inconsistency on line 247 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**
    Full code available at:
    https://github.com/PR2/pr2_controllers/blob/32cbddad1d9edc40d03ff315772f55b00da46941/pr2_mechanism_controllers/src/pr2_base_controller.cpp#L247
    Mark only one oval.

    - Yes
    - No
    - Maybe

12. **6b. Briefly explain your answer**

7.

```
224          m s-1            m              s
225    f_lin_vel_right = f_delta_sr / dur_time.toSec();
226    f_lin_vel_left = f_delta_sl / dur_time.toSec();
227   }
228   else
229   {
230    ROS_ERROR("Division by Zero");
231   }
232
233
234            m                  m s-1
235   wheel_vel.point.x = f_lin_vel_left;
236   wheel_vel.point.y = f_lin_vel_right;
237
```

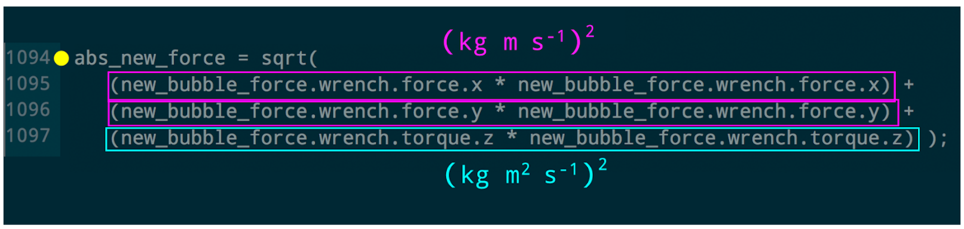
`wheel_vel.point.x (type: PointStamped) is meters but is assigned units meters per second.`

13. **7a. Is the unit inconsistency on line 235 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**
    Full code available at:
    https://github.com/inomuh/evapi_ros/blob/e0bca090dadada10d92766a6749ee29fd36040cc/evarobot_odometry/src/evarobot_odometry.cpp#L235
    Mark only one oval.

    - Yes
    - No
    - Maybe

14. **7b. Briefly explain your answer**

8.



The terms in this addition do not have the same units.

15. **8a. Is the unit inconsistency on line 1094 problematic (e.g., cause failures, increase cost of maintenance, make code more difficult to understand, or introduce interoperability problems)?**

Full code available at: https://github.com/utexas-bwi/eband_local_planner/blob/98a9d898c932705955f4a9be4f31d66fc6d273f8/src/eband_local_planner.cpp#L1094

Mark only one oval.

- Yes
- No
- Maybe

16. **8b. Briefly explain your answer**

17. **Overall Feedback:**

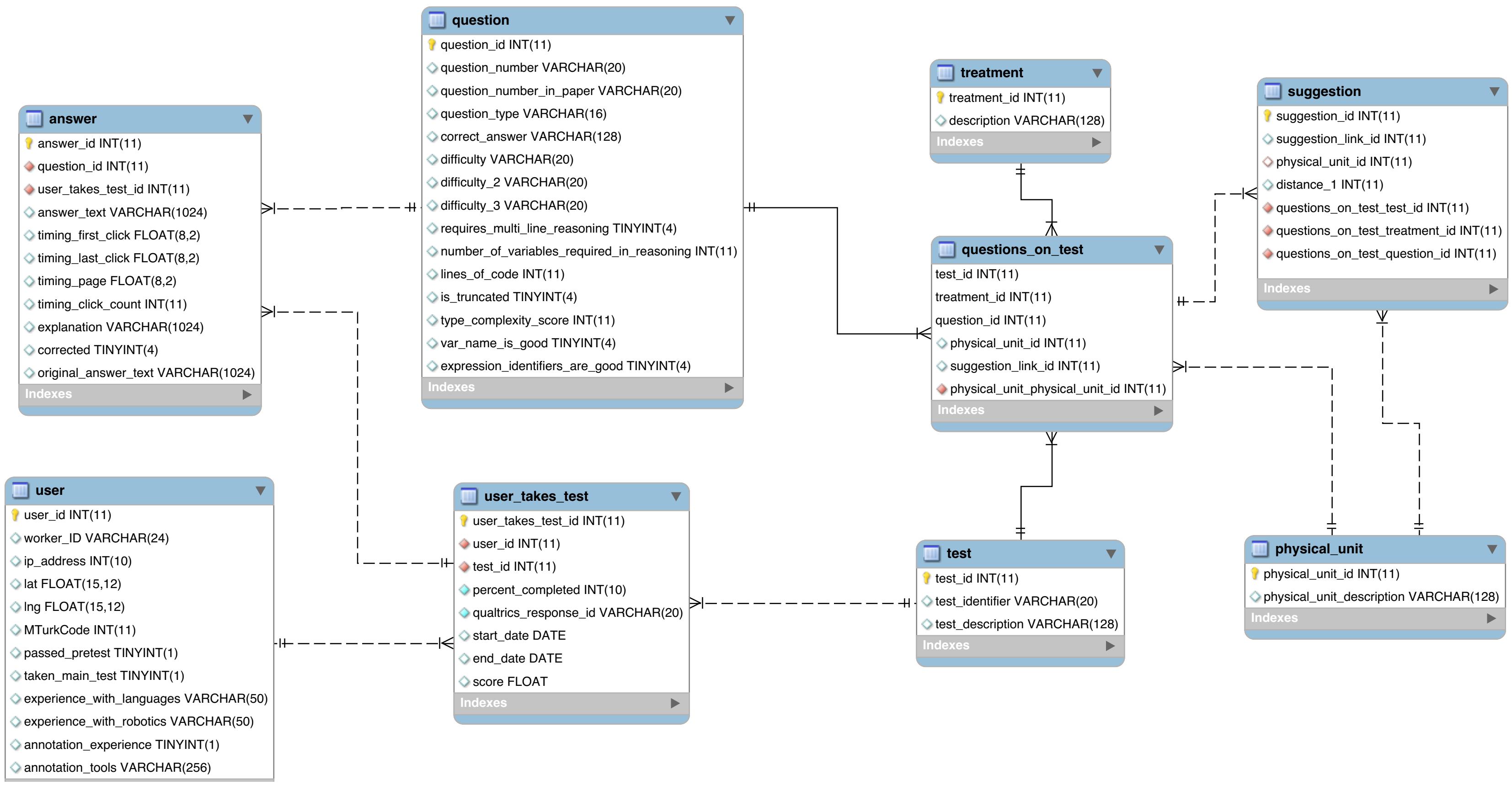
F Phys's Name Assumptions Table

SUBSTRING	PHYSICAL UNITS
position	m
distance	m
diameter	m
radius	m
length	m
width	m
height	m
depth	m
wheelbase	m
altitude	m
time	s
duration	s
period	s
age	s
angle	rad
theta	rad
roll	rad
pitch	rad
yaw	rad
latitude	*degree_360*
longitude	*degree_360*
speed	$\mathrm{m\,s^{-1}}$
velocity	$\mathrm{m\,s^{-1}}$
acceleration	$\mathrm{m\,s^{-2}}$
deceleration	$\mathrm{m\,s^{-2}}$
gravity	$\mathrm{m\,s^{-2}}$
force	$\mathrm{kg\,m\,s^{-2}}$
thrust	$\mathrm{kg\,m\,s^{-2}}$
energy	$\mathrm{kg\,m^2\,s^{-2}}$
effort	$\mathrm{kg\,m^2\,s^{-2}}$
torque	$\mathrm{kg\,m^2\,s^{-2}}$
area	$\mathrm{m^2}$
mass	kg
weight	kg

temperature	°C
frequency	s^{-1}
orientation	*quaternion*
tilt	rad
pan	rad
pressure	$kg\,m^{-1}\,s^{-2}$
voltage	$kg\,m^{2}\,s^{-3}\,A^{-1}$

Table 9.4: Phys's substring assumptions

G Database Schema for Developer Study of Annotation Burden



H List of Open-source Systems Analyzed

SYSTEM URL
https://github.com/0x2aff/kinova_ros
https://github.com/130s/moveit-1
https://github.com/130s/moveit
https://github.com/130s/rqt_marble
https://github.com/1487quantum/ae-scoot
https://github.com/1487quantum/jaguar-bot
https://github.com/1487quantum/le-buggy
https://github.com/1997alireza/Robotic-Project1
https://github.com/2016UAVClass/Simulation
https://github.com/20chase/quadrotor_simulation
https://github.com/218Drone/px4_ros_control
https://github.com/23pointsNorth/depth_tools
https://github.com/2russellsmith/Final470Pacman
https://github.com/3drobotics/PX4Firmware
https://github.com/40323250/v-rep_edu-version
https://github.com/491734045/micros_cv_detection
https://github.com/5yler/mavpro
https://github.com/5yler/xv_11_laser_driver
https://github.com/85pando/youbot_kn
https://github.com/ACRTUDelft/ACRTUDelft
https://github.com/ACarfi/SOFAR
https://github.com/AGNC-Lab/Quad-UI
https://github.com/AGNC-Lab/Quad
https://github.com/AGV-IIT-KGP/eklavya-2015
https://github.com/AGV-IIT-KGP/eklavya-ros-pkg
https://github.com/AGV-IIT-KGP/freezing-batman
https://github.com/AIRLab-POLIMI/BasketBotGame
https://github.com/AIRLab-POLIMI/BasketBot
https://github.com/AIRLab-POLIMI/C-SLAM
https://github.com/AIRLab-POLIMI/ROAMFREE
https://github.com/AIRLab-POLIMI/Rob-E-firmware
https://github.com/AIRLab-POLIMI/Robocom-firmware
https://github.com/AIRLab-POLIMI/Triskar3
https://github.com/AIRLab-POLIMI/TriskarPixy
https://github.com/AIRLab-POLIMI/iDrive
https://github.com/AIRLab-POLIMI/laser_odometry

https://github.com/AIRLab-POLIMI/quadrivio

https://github.com/AIRLab-POLIMI/robots-common

https://github.com/AIRLab-POLIMI/xsens_driver_airlab

https://github.com/AIS-Bonn/humanoid_op_ros

https://github.com/AIS-Bonn/mrs_laser_map

https://github.com/AR2A/imu-minimu-arduino

https://github.com/ARTenshi/JUSTINA

https://github.com/ARezaK/ros

https://github.com/ATLFlight/ros-examples

https://github.com/AUV-IITK/auv

https://github.com/AalborgUniversity-RoboticSurgeryGroup/davinci_joystick

https://github.com/AaronMR/AaronMR_Robotic_Stack

https://github.com/AaronMR/JKU_Robotic_Stack

https://github.com/AaronMR/Learning_ROS_for_Robotics_Programming_2nd_edition

https://github.com/AaronMR/Learning_ROS_for_Robotics_Programming

https://github.com/AbdealiJK/quadrotor_control

https://github.com/Agrim9/MSI_2016-17_OnBoardCode

https://github.com/Aharobot/albany-ros-pkg

https://github.com/Aharobot/dukedusty2

https://github.com/Aharobot/seabee3-ros-pkg

https://github.com/Aharobot/summit-xl-ros-stack

https://github.com/AhmedAnsariIIT/iitmabhiyanros

https://github.com/AhmedElsayedHamouda/imu

https://github.com/Akindart/Mapping-project.-Final-Battle-

https://github.com/Akindart/rest

https://github.com/AkshayHinduja/Velodyne_Odom

https://github.com/AkshayaRaj/sedna

https://github.com/Alabamajack/Garfield

https://github.com/Alasknnj/IC

https://github.com/Alec0425/test

https://github.com/AleksAttanasio/Telerobotics

https://github.com/AleksAttanasio/TurtleBotFollower

https://github.com/AlessandroFaralli/nostop_rviz_visual

https://github.com/AlessioTonioni/Autonomous-Flight-ROS

https://github.com/AlexReimann/downsampler

https://github.com/AlexSwam/ros-slam

https://github.com/AlexisTM/flyingros

https://github.com/AlfaroP/isr-uc-ros-pkg

https://github.com/Alianghey/stanford-ros-pkg https://github.com/Allopart/door_detection_smr

https://github.com/Allopart/table_cleaner_ur10

https://github.com/AloshkaD/human_machine_interface

https://github.com/Alphaproxima/multiple_ardrone

https://github.com/AlvarHHM/tum_simulator

https://github.com/AlysonQ/fake_laser

https://github.com/AlysonQ/rplidar_ros

https://github.com/Amarjyotismruti/Manipulator_teleoperation

https://github.com/Amarjyotismruti/pcl_object_tracking

https://github.com/AmmarkoV/AmmarServer

https://github.com/AndLydakis/Sek_Slam

https://github.com/Andersw88/aw_drone

https://github.com/AndreaLampart/dvs-monochrome

https://github.com/AndreasAZiegler/MFfCUAVSLAM

https://github.com/AndreasAZiegler/ROTI3DBFUWBAV

https://github.com/AndrewADykman/NeuroMobile

https://github.com/AndrewADykman/Robot-Algs

https://github.com/AndrewChubatiuk/ROS-UAV

https://github.com/AngelTianYu/micros_mlog

https://github.com/Ansrala/Delta_Project_3

https://github.com/AriYu/roomba_gesture_ctrl

https://github.com/AriYu/roomba_robot

https://github.com/AriYu/ros_waypoint_generator

https://github.com/Aridane/underwater_map_construction

https://github.com/Arkapravo/laser_obstacle_avoidance_morse

https://github.com/Arkapravo/laser_obstacle_avoidance_pr2

https://github.com/Arkapravo/test_move_base

https://github.com/Arlind1992/TesiPlanner

https://github.com/ArpanHalder/tu-darmstadt-ros-pkg

https://github.com/Arrowana/my_pcl_tutorial

https://github.com/ArthurVal/Reconstruction_Stage_RF_OR_AV

https://github.com/ArthurVal/riddle_ork

https://github.com/ArticulatedSocialAgentsPlatform/UTRobotics

https://github.com/AscendNTNU/Sensor-Fusion-Benchmark

https://github.com/AscendNTNU/line_counter_gui

https://github.com/Ashay-Lokhande/3DScanning_ROS

https://github.com/Ashkan372/TeleCAMP-Controller

https://github.com/Ashkan372/TeleCAMP-Translator

https://github.com/Ashkgp/TravisCI

https://github.com/Asiron/image_restamp

https://github.com/Asiron/nao_pose_controller

https://github.com/Asiron/trajectory_gen

https://github.com/Askodan/localAStar

https://github.com/AssistiveRoboticsUNH/asd_pomdp

https://github.com/AssistiveRoboticsUNH/course_projects_spring_2016

https://github.com/AssistiveRoboticsUNH/myo_nao

https://github.com/AssistiveRoboticsUNH/naoqi

https://github.com/AssistiveRoboticsUNH/unh_pioneer

https://github.com/Astor-Bizard/Joystick_a4h

https://github.com/Aswath93/On-Board

https://github.com/Athria/turtlebot-ros

https://github.com/Atom-machinerule/OpenQbo

https://github.com/Atom-machinerule/ua-ros-pkg

https://github.com/AtsushiSakai/jsk_visualization_packages

https://github.com/Audeliano/filtro_particulas_au

https://github.com/Audeliano/filtro_particulas_basico_fixo

https://github.com/Audeliano/filtro_particulas_kld

https://github.com/Audeliano/filtro_particulas_sakldmcl

https://github.com/Audeliano/filtro_particulas_samcl

https://github.com/Audeliano/map_server_modif

https://github.com/Audeliano/robotino_catkin_expertinos

https://github.com/Audeliano/rosaria-hydro-ros-pkg

https://github.com/AurelienRoy/ardupilot_sitl_gazebo_plugin

https://github.com/AutoModality/vis_pose_test

https://github.com/AutoModelCar/model_car

https://github.com/AutoPark/Hector_SLAM

https://github.com/AutoPark/Local_Planners

https://github.com/AutoPark/Nav_Stack

https://github.com/AutoPark/Odometry

https://github.com/AutoRally/autorally

https://github.com/AutonomyLab/ardrone_autonomy

https://github.com/AutonomyLab/bebop_autonomy

https://github.com/AutonomyLab/create_autonomy

https://github.com/Azimath/ucf-gatefinder

https://github.com/Azimath/ucf-gazebo-translator

https://github.com/B0gdar/ISE_implementation

https://github.com/BARTLABMU/robot_state

https://github.com/BCLab-UNM/ALSA-ROS

https://github.com/BCLab-UNM/Swarmathon-ROS

https://github.com/BGUPioneer/mobile-robot

https://github.com/BGodefroyFR/KTH-labs

https://github.com/BJU-Robot-Team/Bruin-2-Master

https://github.com/BOTAsys/botasys_force_torque_sensor

https://github.com/BYEONGGILYOO/ros-indigo-joystick

https://github.com/BYEONGGILYOO/ros-indigo-remote_controller

https://github.com/BYUMarsRover/onboard

https://github.com/BYUMarsRover/rostesting

https://github.com/Bardo91/DroneApplications

https://github.com/Bardo91/ROS-CATEC

https://github.com/BayronP/ROS_Hand_Tracker

https://github.com/Beautiful-Flowers/opencv_localization

https://github.com/Beautiful-Flowers/ras_cv_meta

https://github.com/Beautiful-Flowers/ras_path_planning

https://github.com/Beck-Sisyphus/src

https://github.com/BenBBear/CS291-Introduction-to-Robotics-Assignment3

https://github.com/BenedicteLC/MapGenerator

https://github.com/BenedicteLC/RL_Lab_Controllers

https://github.com/BenjaminMGreen/our_scanner

https://github.com/Benjaminmar8/toyota_local_planner

https://github.com/BennyRe/ROS-LfD-LAT

https://github.com/Benson516/ROS_Practice

https://github.com/Beryl-bingqi/footstep_dynamic_planner

https://github.com/Bind3rB3njamin/arduino_brushless

https://github.com/Birkehoj/frobit_lego_transporter

https://github.com/Black-Devil/YouBot_RW

https://github.com/BlackMamba591/PCL_demos

https://github.com/BlackMamba591/Thesis-Gists

https://github.com/BlackMamba591/ardrone_command

https://github.com/BlackMamba591/object_recognition

https://github.com/BlackSlashProd/RoboticBugAlgo

https://github.com/BlazingForests/realsense_camera

https://github.com/BlueCabbage/LearningROSforRoboticsProgramming

https://github.com/BlueWhaleRobot/dso_ros

https://github.com/BlueWhaleRobot/xqserial_server

https://github.com/Boberito25/ButlerBot

https://github.com/Bobeye/pydybot

https://github.com/BossKwei/naive_mapping https://github.com/BossKwei/naive_tank

https://github.com/BrennoCaldato/Create_PointCloud-ROS

https://github.com/BrettHemes/arm_navigation

https://github.com/BrettKnadle/CS491_Autonomous_Buggy

https://github.com/BritskNguyen/rosbuildWS

https://github.com/BruinBear/agitr

https://github.com/BruinBear/minimuv5_pi_imu_publisher

https://github.com/BrunoDatoMeneses/Transport-Network-and-Baxter

https://github.com/BryanLeong/hmapping

https://github.com/Bugro/SLAM

https://github.com/CARMinesDouai/MultiRobotExplorationPackages

https://github.com/CARMinesDouai/turtlebot_car

https://github.com/CBJamo/swiftnav_piksi

https://github.com/CDibris/Quad_Swarm

https://github.com/CIR-KIT-Unit03/cirkit_unit03_robot

https://github.com/CIR-KIT/TC2016_for_thirdrobot

https://github.com/CIR-KIT/fifth_robot_pkg

https://github.com/CIR-KIT/fourth_robot_pkg

https://github.com/CIR-KIT/human_detector

https://github.com/CIR-KIT/remote_monitor

https://github.com/CIR-KIT/steer_drive_ros

https://github.com/CIR-KIT/third_robot_pkg

https://github.com/CIR-KIT/waypoint_manager

https://github.com/CLDrone/ardrone_marker_tracking

https://github.com/CLDrone/body_axis_controller

https://github.com/CLDrone/body_axis_keyboard_control

https://github.com/CLDrone/body_axis_velocity_controller

https://github.com/CLDrone/body_axis_velocity_keyboard_control

https://github.com/CLDrone/marker_tracker

https://github.com/CLDrone/moving-the-car

https://github.com/CLDrone/px4_offboard_position_control

https://github.com/CLDrone/px4_offboard_velocity_control

https://github.com/CLDrone/rotors_simulator_with_drcsim_env

https://github.com/CLDrone/velocity_marker_tracker

https://github.com/CMU-Robotics-Club/Colony

https://github.com/CMUBOOST/BOOST1

https://github.com/CMUBOOST/BOOST3

https://github.com/CMUBOOST/BOOST_Stalker

https://github.com/CMobley7/Loam_Matlab_SMP https://github.com/CNMSwarmTeam/CNM-ST-MASTER

https://github.com/CNURobotics/chris_create_navigation

https://github.com/CNURobotics/chris_ros_create

https://github.com/CNURobotics/flexible_navigation

https://github.com/CNURobotics/simple_message_to_tf

https://github.com/CNURobotics/stamped_cmd_vel_mux

https://github.com/CNURobotics/stamped_velocity_smoother

https://github.com/COHRINT/cops_and_robots

https://github.com/CPFL/Autoware

https://github.com/CS6751/Jarvis

https://github.com/CSULAQuadcopter/eagle_one_test

https://github.com/CSUN-SERL/husky-support-platform

https://github.com/CSUN-SERL/lcar-bot

https://github.com/CSUN-SERL/nao-robo-leader

https://github.com/CTTC/ROS_TurtleBot_Fun

https://github.com/CU-Robotics-Institute/a-very-brief-ros-tutorial

https://github.com/CURG/graspit_scene_builder

https://github.com/CURG/relationship_detector

https://github.com/CURG/staubli_tx60

https://github.com/CalPolyRobotics/IGVC-old

https://github.com/CambridgeAUV/software_v2

https://github.com/CanyonWind/Robotics

https://github.com/CaptD/Coverage_Planning

https://github.com/CaptD/blaser

https://github.com/CaptD/elp_stereo_camera

https://github.com/CaptD/tag_hu

https://github.com/Captain456/Racecar_Controller

https://github.com/CarlosAriel/ROS-ARDRUINO

https://github.com/CarlosPosse/Magabot

https://github.com/CarlosUrteaga/Robotica

https://github.com/CasperPlatform/casper_docker_rpi_environment

https://github.com/CathIAS/iarc_archive

https://github.com/CentralLabFacilities/m3meka

https://github.com/CentralLabFacilities/m3ros_control

https://github.com/CentralLabFacilities/meka-ros-pkg

https://github.com/CentralLabFacilities/simple_robot_gaze_tools

https://github.com/CentralLabFacilities/tobi_picknplace

https://github.com/CentralLabFacilities/tobi_ros4rsb

https://github.com/CentroEPiaggio/force-torque-sensor https://github.com/CentroEPiaggio/kuka-lwr

https://github.com/CentroEPiaggio/pacman-DR54

https://github.com/CesMak/Roomba_521

https://github.com/ChallenHB/OctoSlam

https://github.com/Chaniro/ps_ardrone

https://github.com/Chengxi9/laser_scan

https://github.com/ChielBruin/tomatenplukkers

https://github.com/ChingHengWang/NextBot

https://github.com/ChingHengWang/Scripts

https://github.com/ChingHengWang/andbot_simulator

https://github.com/ChingHengWang/andbot_tool

https://github.com/ChingHengWang/hyperlync_robot

https://github.com/ChingHengWang/hyperlync_simulator_2

https://github.com/ChingHengWang/ir_avoidance_simulator

https://github.com/ChingHengWang/metal0

https://github.com/ChingHengWang/mobile_mimo

https://github.com/ChingHengWang/nextbot_simulator

https://github.com/ChingHengWang/pcl_pkg

https://github.com/ChingHengWang/ros_tutorials

https://github.com/ChingHengWang/rplidar_verify

https://github.com/ChingHengWang/zach_gazebo_nav

https://github.com/Chpark/itomp_ca_planner

https://github.com/ChrisBove/RBE-3002-ROSbert

https://github.com/ChrisBove/RBE_510_ROS

https://github.com/ChrisR48/arduino_sensor_driver

https://github.com/ChristianLiinHansen/ISA

https://github.com/ChuChuIgbokwe/Indoor-Navigation-with-a-Quadcopter

https://github.com/Chuanbo/navigation_empty_map

https://github.com/ChuanzhengLi/AE_598_Sys_Arc

https://github.com/ChunkyBART/Mytest

https://github.com/ChunkyBART/gitUploadAuto2015

https://github.com/CityU-MBE/SickScan3D

https://github.com/ClarenceZSK/Root-catkin_ws

https://github.com/ClementLeBihan/CelluleFlexible

https://github.com/ClintonPeterson/netft

https://github.com/CoffeRobot/amazon_challenge_bt

https://github.com/CoffeRobot/amazon_challenge

https://github.com/CoffeRobot/fato

https://github.com/Cognitive-Robotics-Lab/TurtleRPI https://github.com/CokieForever/LCCP-Turtlebot

https://github.com/ColumnRobotics/anchor

https://github.com/ColumnRobotics/column

https://github.com/Comusicart/seekAndHide

https://github.com/Cornell-RPAL/occam

https://github.com/CornerOfSkyline/firmware_dev

https://github.com/CoroBot/CoroBot_ROS_Stack

https://github.com/Corpse89Grinder/ar_tracker

https://github.com/Corpse89Grinder/hostess_robot

https://github.com/Cory-Simon/learn_ros

https://github.com/CptDD/BaxterBartender

https://github.com/CreateUNSW/UGV_ROS

https://github.com/CreedyNZ/jimbob_ros

https://github.com/CrocInc/uav-croc-contest-2013

https://github.com/CumulonimbusLtd/phidgets_interface_kit

https://github.com/CumulonimbusLtd/phidgets_motion_control

https://github.com/Cyberbully/COMP3431-ass1

https://github.com/Cybonic/fsr_hunter

https://github.com/D0nPiano/DasAuTU

https://github.com/DAInamite/srf_serial

https://github.com/DAInamite/uav_object_localisation

https://github.com/DAInamite/uav_position_controller

https://github.com/DD2425-2015-Group7/catkin_ws

https://github.com/DD2425-group-5/utilities

https://github.com/DD2425-group-5/vision

https://github.com/DLu/navigation_layers

https://github.com/DLu/path_planning_metrics

https://github.com/DLu/plugin_local_planner

https://github.com/DLu/simple_local_planner

https://github.com/DNLD2017/turret_control

https://github.com/DNXSR/navigation

https://github.com/DaReis/sensor_tools

https://github.com/Daedalus359/mrp_kmb172

https://github.com/DaikiMaekawa/quadrotor_moveit_nav

https://github.com/DakotaNelson/robo-games

https://github.com/DaniSagan/youbot-xsens-controller

https://github.com/DanielDuecker/HippoC

https://github.com/Danny2036/MobileRobotics

https://github.com/DavidB-PAL/reem_tabletop_manipulation_launch https://github.com/DavidHtx/DI_Fontys_AR_Multirobot

https://github.com/DavidLevayer/robAIR

https://github.com/DavidZizu/Robot_Paths

https://github.com/DavidZizu/turtlebot_route_planning

https://github.com/DavideACucci/roamvo

https://github.com/DavyNeven/vision_ardrone

https://github.com/DeaSoftware/AirKobra

https://github.com/DecosRobotics/DecosRoboticsLab

https://github.com/DecosRobotics/eco_common

https://github.com/DedaleTSP/ddale_ardrone_nav

https://github.com/DeepBlue14/android_turtlebot_controller

https://github.com/DeepBlue14/laser_detection

https://github.com/DeepBlue14/ros_ip_transform

https://github.com/DeepBlue14/rqt_ide

https://github.com/DeepBlue14/usar_system

https://github.com/DennisMelamed/bug2_create

https://github.com/DevasenaInupakutika/assn2

https://github.com/DevasenaInupakutika/knowrobcloud

https://github.com/DinnerHowe/xbot_navigation

https://github.com/Djeef/ardrone_resizer

https://github.com/DogamusMaximus/hekateros-master

https://github.com/DongHoonPark/IS2016_project_hw3

https://github.com/DongHoonPark/IS2016_project_hw4

https://github.com/DongHoonPark/hexapod_wii

https://github.com/DrJulia/arvc-umh-ros

https://github.com/Dragneil/Simple-Robot-Control-and-Using-tf-in-ROS

https://github.com/DrawZeroPoint/kinect2_to_laserscan

https://github.com/DrawZeroPoint/move_base

https://github.com/DreamtaleCore/ROS-Nodes-in-Dji-Summer-Camp

https://github.com/DreamtaleCore/myRosPackage

https://github.com/Drone2/ros_drone

https://github.com/Dronecode/sitl_gazebo

https://github.com/DsLiner/obstacle_recognizer

https://github.com/Duceux/gd_semnav

https://github.com/Dulluhan/Snowbots_Demo

https://github.com/Durant35/laser_scan_matcher

https://github.com/EECS-376-Group-Gamma/Mobile-Robotics

https://github.com/EESC-LabRoM/AMR_FirstSteps

https://github.com/EESC-LabRoM/labrom_mav_landmark https://github.com/EESC-LabRoM/labrom_mav

https://github.com/EESC-LabRoM/labrom_optical_flow

https://github.com/EESC-LabRoM/labrom_perception

https://github.com/EI2012zyq/vanadium-ros-pkg

https://github.com/ENSTABretagneRobotics/Brest2016

https://github.com/ENSTABretagneRobotics/sbg_ros_driver

https://github.com/ESE519/MyLidar-2013

https://github.com/ESE519/MyLidar

https://github.com/EagleKnights/EK_AutoNOMOS

https://github.com/EagleKnights/SDI-11911

https://github.com/EagleKnights/ek-ssl

https://github.com/EduFill/components

https://github.com/EduFill/hbrs-ros-pkg

https://github.com/EduFill/nxt-stack

https://github.com/EduPonz/RobotProgramming

https://github.com/EduardoFF/pioneer_idsia

https://github.com/Eduzc07/CopterRos

https://github.com/ElCapitan2/Frontier_Navigation

https://github.com/ElizabethGallardo/PRC

https://github.com/EmaroLab/kinect_calibration

https://github.com/EmbeddedLab/ros_work

https://github.com/Energid/ActinROS

https://github.com/EnovaROBOTICS/Mini-Lab

https://github.com/EnstaBretagneClubRobo/enstaB-ros

https://github.com/EoinLikeOwen/ros_project

https://github.com/EricFalkenberg/robotics_project

https://github.com/Exception4U/IIITH_RTABMAP

https://github.com/Exception4U/exploration_for_care

https://github.com/Expertinos/curso_ROS_UNIFEI

https://github.com/Expertinos/ros-indigo-robotino

https://github.com/Eysiger/image_fitter

https://github.com/Eysiger/map_fitter

https://github.com/Ezzence/slam_ros

https://github.com/F34140r/visualization-userfriendly

https://github.com/FOXTTER/quad_cam

https://github.com/FRC900/2016VisionCode

https://github.com/FSGRIS/roomba

https://github.com/FSGRIS/snacbot

https://github.com/FactoryOfTheFuture/Youbot https://github.com/FaedDroneLogistics/FAED-Project

https://github.com/FalkorSystems/hector_gazebo

https://github.com/FalkorSystems/hector_localization

https://github.com/FalkorSystems/hector_quadrotor

https://github.com/FedericoPecora/lucia-winter-school-2014

https://github.com/Felicien93/ucl_drone

https://github.com/FergusSwarmathon/BClab-UNM-swarmathon-ros

https://github.com/FetchRobot-SZU/fetch_pr4

https://github.com/FetchRobot-SZU/fetch_pr4

https://github.com/Flos/ros-kitti

https://github.com/Flos/ros-ladybug

https://github.com/Flos/ros-openni2_laserscan

https://github.com/Flystix/FroboMind

https://github.com/For-motion-capture-platform/UAV_mani_pixhawk

https://github.com/FrancoNegri/RoboticaMovil2016

https://github.com/Fraunhofer-IIS/gnss

https://github.com/FredrikTheSwaglord/SnakeBae

https://github.com/FreemooVR/FreemooVR

https://github.com/Fromandto/using_markers

https://github.com/GAVLab/novatel

https://github.com/GCRobotics/GCRepo

https://github.com/GKIFreiburg/gki_3dnav

https://github.com/GKIFreiburg/gki_navigation_apps

https://github.com/GKIFreiburg/gki_navigation

https://github.com/GKIFreiburg/gki_pr2_symbolic_planning

https://github.com/GKIFreiburg/gki_robots

https://github.com/GKIFreiburg/gki_sensors

https://github.com/GKIFreiburg/pr2_tidyup

https://github.com/GT-RAIL/carl_navigation

https://github.com/GT-RAIL/carl_safety

https://github.com/GT-RAIL/nimbus_bot

https://github.com/GT-RAIL/rail_ceiling

https://github.com/GT-RAIL/rail_pick_and_place

https://github.com/GT-RAIL/robot_pose_publisher

https://github.com/GT-RAIL/spatial_temporal_learning

https://github.com/GTRIInternship2016/WaterGun2016

https://github.com/Gallard88/sunfish_control

https://github.com/Garfield753/roscpprtimulib

https://github.com/GaryJin/Navigation3D

https://github.com/Gastd/indoor_localization

https://github.com/Gastd/memsense_nano_imu

https://github.com/Gastd/msl_skycrane_gazebo_plugins

https://github.com/Gastd/novatel_gps

https://github.com/Gastd/p3at_tutorial

https://github.com/Georacer/last_letter

https://github.com/GertKanter/quaternion_demo

https://github.com/Ghinda94/ROS_drone_hand_control

https://github.com/GiovanniBalestrieri/Thor

https://github.com/GiovanniBalestrieri/bin

https://github.com/GrenobleRoboticLab/swex

https://github.com/Griger/TSI

https://github.com/Group3TAS/Group_3

https://github.com/Group3TAS/Matthaeus

https://github.com/Group3TAS/andre

https://github.com/GroupProjectMultiRobotLocalization/existing-code

https://github.com/GuidoManfredi/essential

https://github.com/GuillemSebastianFletes/laser_treshoold

https://github.com/HALLAB-Halifax/ardrone_apps

https://github.com/HANDS-FREE/PaperSimu_XYi_WS

https://github.com/HANDS-FREE/ROS_DEMO

https://github.com/HLP-R/hlpr_manipulation

https://github.com/HWiese1980/StereovisionTPR_Control

https://github.com/HackInstitute/ros-hackathon

https://github.com/HackInventOrg/px4

https://github.com/HackRoboy/TheGreatGuessingGame

https://github.com/Hacks4ROS/h4r_bosch_bno055_uart

https://github.com/Hacks4ROS/h4r_ev3_ctrl

https://github.com/HailStorm32/Q.bo_stacks

https://github.com/HappyGuyNCKU/ros_image_pipeline

https://github.com/HappyKoyo/move_mini

https://github.com/HarishKarunakaran/ATLASControl

https://github.com/Harleen-Hanspal/CameraLocalisationTestBed

https://github.com/Heverton29/CoopPkg

https://github.com/Hiroki-Goto/GPS_positioning

https://github.com/Hiwr/hiwr_camera_controller

https://github.com/HnYitc/fMoterControl

https://github.com/HoriSun/inhans

https://github.com/HovakimyanResearch/cf_ros https://github.com/HovakimyanResearch/crazyflie-demos

https://github.com/Hujianjun1992/code

https://github.com/HumaRobotics/ros-indigo-qbo-packages

https://github.com/HumanoidRobotics/pr2_example_ws

https://github.com/Humhu/argus_utils

https://github.com/Humhu/argus

https://github.com/Humhu/simu

https://github.com/HurinHall/ros_pioneer_ws

https://github.com/Hurisa/masters_dissertation

https://github.com/HutEight/ariac_time_estimator

https://github.com/HutEight/davinci_wsn

https://github.com/ICSL-hanyang/drone_simulator

https://github.com/ICSL-hanyang/solar_sys_formation

https://github.com/IDSCETHZurich/gajanLocal

https://github.com/IDSCETHZurich/re_trajectory-generator

https://github.com/IDSCETHZurich/simplePendulum

https://github.com/IMNOTAROBOT/REPO

https://github.com/IMNOTAROBOT/TesisMCC

https://github.com/INCF/MUSIC

https://github.com/IRLL/NurseryRover

https://github.com/ISBaxterGroup/lis-ros-pkg

https://github.com/ISIR-SYROCO/Model_Free_Control_Operationnel

https://github.com/ISIR-SYROCO/fri_examples

https://github.com/ISIR-SYROCO/kuka_component

https://github.com/ISIR-SYROCO/kuka_fri_grav_comp

https://github.com/IVLABS/decision_making

https://github.com/IanAlbuquerque/exploration_robot_ros

https://github.com/IchheisseJoe/tf2hz

https://github.com/IfabotTUD/Ifabot-ROS

https://github.com/IkeLevens/drogon_interface

https://github.com/IkerZamora/ros_erle_spider

https://github.com/IllinoisRoboticsInSpace/FSM

https://github.com/IllinoisRoboticsInSpace/IRIS-2017

https://github.com/IllinoisRoboticsInSpace/IRIS-6-2016

https://github.com/IllinoisRoboticsInSpace/IRIS_V_control

https://github.com/ImanSharabati/amcl_modified

https://github.com/ImmersiveSystems/net-robo

https://github.com/ImmortalGhost/ForROS

https://github.com/Imperoli/SPQR_at_work https://github.com/Imperoli/depth_to_LaserScan

https://github.com/Imperoli/gradient_based_navigation

https://github.com/Imperoli/rockin_at_work_software

https://github.com/InesDominguesFI/rwsfi2016_idomingues

https://github.com/Innovation-Cell/RobotPoseEkf

https://github.com/Innovation-Cell/bumblebee-ros-driver

https://github.com/Innovation-Cell/final_code_RISE

https://github.com/Interbotix/ros_hros5

https://github.com/IntoRobot/Rosserial4IntoRobot

https://github.com/Isaac25silva/Driver-UM6

https://github.com/Isaac25silva/Driver-UM7

https://github.com/Isaac25silva/Qlearning-Humanoid

https://github.com/IuAyala/Mastering_ROS

https://github.com/J-Pai/ram_uwsim

https://github.com/JBot/smart-robotics-ros-pkg

https://github.com/JBot/smart-ros-pkg

https://github.com/JBot/smartfr-ros-pkg

https://github.com/JJJJJJack/glviewer

https://github.com/JJJJJJack/trajectory_generate

https://github.com/JJJVic/Epsilon_376

https://github.com/JKWalleiee/P3DX_Nodes_Basic

https://github.com/JRikken/youbot_markers

https://github.com/JXS2012/QuadController

https://github.com/JXS2012/iCreateNavigationController

https://github.com/JXS2012/sandbox

https://github.com/JackieJ/WartHog

https://github.com/Jae-hyun/beginner_tutorials

https://github.com/Jae-hyun/tilting_hokuyo

https://github.com/Jaeyoung-Lim/inrol_fixedtool

https://github.com/Jaeyoung-Lim/modudculab_ros

https://github.com/Jailander/fremen

https://github.com/JamesDavis1829/HAL-UnmannedGroundVehicle

https://github.com/JaneSheard/roco504

https://github.com/JawaJack/prairiedog

https://github.com/JayYangSS/ROS-PointGrayDriver

https://github.com/JdeRobot/JdeRobot

https://github.com/JdeRobot/ThirdParty

https://github.com/JeffsanC/mavros

https://github.com/JeffsanC/uavs

https://github.com/JeffsanC/viconxbee

https://github.com/JenJenChung/nn_control_input_filter

https://github.com/JenJenChung/pioneer_gazebo_ros

https://github.com/JenJenChung/pioneer_test

https://github.com/JenniferBuehler/common-sensors

https://github.com/JenniferBuehler/gazebo-pkgs

https://github.com/JenniferBuehler/general-message-pkgs

https://github.com/JenniferBuehler/moveit-pkgs

https://github.com/JenniferNist/Wearloc

https://github.com/Jeoker/balala

https://github.com/JesseNolan/Differential_Drive_Robot

https://github.com/JhonPB/Chasing_robot

https://github.com/JhonPB/Quaternion

https://github.com/JhonPB/Sonar_3

https://github.com/JhonPB/go2point..

https://github.com/JhonPB/obstacle_avoidance

https://github.com/JimmyDaSilva/find_n_track_objects

https://github.com/JimmyDaSilva/lwr_pick_n_place

https://github.com/Jin-Linhao/bob_perception

https://github.com/JinqiaoShi/cap

https://github.com/JinqiaoShi/demo_lidar_condensed

https://github.com/JinqiaoShi/laser3D

https://github.com/JinqiaoShi/laser_sweep_with_guidance

https://github.com/JinqiaoShi/laser_sweep

https://github.com/JinqiaoShi/px4_offboard

https://github.com/Jister/ROS_sonar_node

https://github.com/Jister/ardrone_control

https://github.com/Jister/bridge_ws

https://github.com/Jister/laserscan

https://github.com/Jister/offboard_simulation

https://github.com/Jister/stereo_camera

https://github.com/Jister/test_vicon

https://github.com/Jmeyer1292/delaunay

https://github.com/Jmeyer1292/trainingv2

https://github.com/Jntzko/fix_haschke

https://github.com/JoSungUk/My_1st_ROScode

https://github.com/JoanaS/rwsfi2016_jsantos

https://github.com/Joao-M-Almeida/HumanRobotInteraction

https://github.com/Joaouzinhu/carrot_chasing https://github.com/JoeyS7/HamsterCraft

https://github.com/Johan944/Need4Stek

https://github.com/JohanVer/bosch_hackathon

https://github.com/JohannesBeck/catkin

https://github.com/Johnson11/Arduino-Ros

https://github.com/Joncy/turtlesim_position_controller

https://github.com/Jonmg/youbot_main_control

https://github.com/JoonasMelin/tree_segment

https://github.com/JorgeArino/ar_navigation

https://github.com/JoseAvalos/baxter_Object-Clasification

https://github.com/JoshuaEbenezer/ardugaz

https://github.com/JulienDufour/velodyne_tracking

https://github.com/JuliousHurtado/pionner3dx

https://github.com/Junchi-Liang/baxter_training_generation

https://github.com/JuneJulyAugust/ardrone-qut-cyphy

https://github.com/JuneKim/asulada

https://github.com/Justin-Kuehn/cp-capstone-autogolfcart

https://github.com/Juvygelbard/InstallScript

https://github.com/Jyrks/visual_servoing

https://github.com/K-Mladen/motionPlanning

https://github.com/K-Mladen/umkc-robotics-legacy-IMU-code-legacy

https://github.com/KIT-MRT/stargazer_ros

https://github.com/KKAndersen/ROSminiproject

https://github.com/KM-RoBoTa/pleurobot_ros_pkg

https://github.com/KRSSG/belief_state

https://github.com/KTH-AEROWORKS/sml_stable_code

https://github.com/KTH-AEROWORKS/sml_under_development

https://github.com/KTH-RAS-4/wheatley

https://github.com/KTH-RAS-HT14-G9/controllers

https://github.com/KTH-RAS-HT14-G9/robot_ai

https://github.com/KTH-RAS/RAS2016

https://github.com/KTH-RAS/ras_lab1

https://github.com/KTH-RAS/ras_maze

https://github.com/KadynCBR/durian_track_and_homing

https://github.com/KahnShi/qp_solver

https://github.com/KamalDSOberoi/INSA

https://github.com/Kamaros/me547

https://github.com/Karsten1987/my_ros2_demos

https://github.com/Karsten1987/ros_protobuf https://github.com/Karsten1987/sot_fcl_distance_computation

https://github.com/KastB/tas_car

https://github.com/Kazaroni/ros_phidgets_jade

https://github.com/Keerthikan/ROVI2

https://github.com/KenmeiFusamae/motoman_project

https://github.com/KenmeiFusamae/motoman

https://github.com/KentaKubota/road_surface_recognition

https://github.com/KeoChi/robot_move

https://github.com/Kevin315/DexterousHand

https://github.com/KevinOchs/hexapod_ros

https://github.com/Kigs-mx/Detection_system_not_superficial_victims

https://github.com/Kinovarobotics/kinova-ros

https://github.com/KitKat7/MPU6050

https://github.com/Konakona333/Go2Point

https://github.com/Koniyasu/kamui_minimum

https://github.com/Koniyasu/kamui_operatorstation_minimum

https://github.com/Kuba22/saas_tutorial

https://github.com/KumarRobotics/cam_imu_sync

https://github.com/KumarRobotics/flea3

https://github.com/KumarRobotics/flir_gige

https://github.com/KumarRobotics/gps-tools

https://github.com/KumarRobotics/imu_3dm_gx4

https://github.com/KumarRobotics/imu_vn_100

https://github.com/KumarRobotics/kr_attitude_eskf

https://github.com/KumarRobotics/kr_utils

https://github.com/KumarRobotics/ublox

https://github.com/KumarRobotics/velodyne_puck

https://github.com/KumarRobotics/vicon

https://github.com/Kyle-ak/RGB-D-SLAM-v1

https://github.com/KyriacosShiarli/rrt_learn_catkin

https://github.com/LARS-robotics/lars-ros

https://github.com/LCAD-UFES/carmen_lcad

https://github.com/LCAD-UFES/kinnect_mapper

https://github.com/LCAD-UFES/localization

https://github.com/LCAD-UFES/probabilistic-robotics

https://github.com/LRMPUT/PUTSLAM

https://github.com/LUHbots/luh_youbot_os

https://github.com/Lab-RoCoCo/fps_mapper

https://github.com/Lab-RoCoCo/thin_drivers https://github.com/LaboratoireCosmerTOULON/ros_vs_turtlebot_catenary

https://github.com/LaboratoireCosmerTOULON/uwvs_osl

https://github.com/Lakshadeep/ROS_robotics_simulations

https://github.com/LeavingTheFlatland/ros-vrep-tangentbug-with-waypoints

https://github.com/Lembed/ROS-FreeRTOS-STM32

https://github.com/Lenskiy/Unmanned-ground-vehicle

https://github.com/Leone9689/ORB_SLAM2

https://github.com/Leone9689/listener

https://github.com/Leslie-Fang/laser_receiver1

https://github.com/Lewthie/quadcontrol

https://github.com/LiftnLearn/se-lab

https://github.com/LiliMeng/ParticleFilter

https://github.com/Limpinho0/bilibot-ros-pkg

https://github.com/Lincoln97/avoidance

https://github.com/Lincoln97/sonar

https://github.com/Lindenroth/kinematics

https://github.com/Lindenroth/rosRepo

https://github.com/Linh-Tran/DDSA_ROS

https://github.com/LitterBot2017/MellEObstacle

https://github.com/LitterBot2017/planning

https://github.com/LiuZhongSIA/Firmware_July2016

https://github.com/LiujiangYan/control_toolbox

https://github.com/LofaroLabs/POLARIS

https://github.com/LokeZhou/patrol_robot

https://github.com/Lolu28/interactive_object_recognition

https://github.com/Lolu28/particle_filter

https://github.com/LoongWang/SP_AUV

https://github.com/LoongWang/Update-Versions

https://github.com/LordBismaya/DPPTAM

https://github.com/LordBismaya/FrenchVanilla

https://github.com/LordBismaya/coke_slam

https://github.com/LordBismaya/pointgrey_camera_drivers

https://github.com/LordBismaya/roboteq

https://github.com/LorenzoMorandi/Robot_Controller

https://github.com/Lothar94/tecnicas-sistemas-inteligentes

https://github.com/LouLinear/APC_vision

https://github.com/Low-Cost-UW-Manipulation-iAUV/imu_positioning

https://github.com/LoweDavince/roshydro

https://github.com/LowellMakes/hydrogen

https://github.com/LucBanda/piros

https://github.com/LucaGemma87/GEAM

https://github.com/LucaGemma87/force_torque_sensor_controller

https://github.com/LucaGemma87/gazebo_plugins

https://github.com/Luis93A/cyton1500

https://github.com/LuisServin/workspace

https://github.com/Lukx19/NDT-scanmatching

https://github.com/Lukx19/ndt_gslam_attachments

https://github.com/Lupasic/Plastun-Gazebo-ROS-URL

https://github.com/Lupasic/ROS-Converters-to-txt

https://github.com/MASKOR/maskor_thermal_mapper

https://github.com/MChemodanov/gazebo_inverted_pendulum

https://github.com/MISTLab/ROSBuzz

https://github.com/MJohnson459/SpareParts

https://github.com/MPC-Berkeley/barc

https://github.com/MRASL/mrasl_guidance_ros

https://github.com/MRPT/mrpt

https://github.com/MRSDTeamI/bud-e

https://github.com/MST-Robotics/Enterprise_IGVC

https://github.com/MST-Robotics/IGVC_master

https://github.com/MST-Robotics/IGVC

https://github.com/MST-Robotics/Jomegatron_IGVC

https://github.com/MST-Robotics/MST_ROS_Catkin_Packages

https://github.com/MTU-Autobot/line-tracking

https://github.com/MTirtowidjojo/lfd_actions

https://github.com/MTyk/PbD

https://github.com/MTyk/hippo

https://github.com/MaXvanHeLL/LSD-SLAM

https://github.com/MacAnthony144/proyecto01Robotica

https://github.com/Majunchong/arduino_motor_control

https://github.com/Maksemilian/MY_ROS

https://github.com/MalcolmMielle/ROS_driver_differential_drive

https://github.com/MalcolmMielle/Stalker

https://github.com/MalcolmMielle/Tobot

https://github.com/MalcolmMielle/ros_open_tld_3d

https://github.com/MalcolmMielle/tobot_goal

https://github.com/ManashRaja/agv_simulation

https://github.com/MangoMangoDevelopment/neptune

https://github.com/ManickYoj/robo_team_charlie https://github.com/ManolisCh/experiment3_mi

https://github.com/ManolisCh/pioneer_p3dx

https://github.com/Maplenormandy/sshbot

https://github.com/MaralAfris/wasp-challenge-lth-team2

https://github.com/MarcoMura85/VisualHatpic

https://github.com/MarcoMura85/rollingTest

https://github.com/Marcus-Zhu/bwi_pcl_detection

https://github.com/MarioAlexis/odm

https://github.com/MarkoObrvan/detection_module

https://github.com/MartienLagerweij/aidu

https://github.com/Martin-Oehler/compliant_control

https://github.com/MarvinElz/quadrotor_dynamics

https://github.com/MarvinElz/quadrotor_rc_delay

https://github.com/MarvinElz/quadrotor_sim_control

https://github.com/MarxBingen/TurtleBot-Soccer

https://github.com/Marz6759/Project-Anthrax

https://github.com/MatejBartosovic/mrvk_project

https://github.com/MatoMA/youbot_joy_arm

https://github.com/MatoMA/youbot_pick_and_place

https://github.com/MatsunoLab/beego-ros-pkg

https://github.com/MattDerry/model_predictive_navigation

https://github.com/MatthewKimball/Autonomous-Door-Opening-Project

https://github.com/MatthewKimball/Kinova-MICO-Arm-Simulation-Model

https://github.com/MauricioQJ25/BioToretto

https://github.com/MauricioQJ25/my_programs

https://github.com/MayankB11/agv_task1

https://github.com/McKracken/kmi_ros

https://github.com/MechanicalBulldawgs/aries

https://github.com/Mennat-Ullah/Co-Swarm15

https://github.com/Messikommer/obstacle_detection

https://github.com/MichalDobis/adis16350

https://github.com/MichelaA/experiment_wp2

https://github.com/MikeFink/learning_ps3joy

https://github.com/Minipada/xpider

https://github.com/MinorRoboticsTeam4/Cobot_ROS

https://github.com/MirkoFerrati/aviz

https://github.com/MirkoFerrati/nav_msgs_2_turtle_pose

https://github.com/MirkoFerrati/speed_control

https://github.com/Mishalassif/gazebo_hydrodynamics https://github.com/MobileRobotics-Ulaval/CollaborativeDroneLocalization

https://github.com/MobileRobotics-Ulaval/husky-trainer

https://github.com/MobileRobots/ros-arnl

https://github.com/Mobots/mobots

https://github.com/MonicaArias/rescuer_sim

https://github.com/MonicaArias/voronoi_navigation

https://github.com/MoonTTMM/rgbdslam

https://github.com/MozhiJiawei/Ardrone_2

https://github.com/MozhiJiawei/Ardrone_L-H

https://github.com/Mr-Anderson/mesh_mapper

https://github.com/Mr-Anderson/turtlebot_360

https://github.com/MrXuC/strage

https://github.com/Myzhar/RoboController-ROS

https://github.com/Myzhar/ros_imu_covariance_calculator

https://github.com/Myzhar/ros_inemo_m1_imu

https://github.com/NCharabaruk/phidgets

https://github.com/NEU-TEAM/drv_package

https://github.com/NEU-TEAM/rva_package

https://github.com/NIFTi-Fraunhofer/nifti_uav

https://github.com/NREC/camera-drivers

https://github.com/NTNU-MCS/CS_Saucer_ROS

https://github.com/NaioTechnologies/simulatoz

https://github.com/NamikiHiroaki/ros_test

https://github.com/Nancyborg/Tests

https://github.com/Nanda-Kishore-V/gazebo-sim-traj-plan

https://github.com/Nantrobot/nantrobot-rpi

https://github.com/Near32/ROSnodes

https://github.com/Nerei/pcl_old_repo

https://github.com/Newhandnew/ROS-QTTurtleControl

https://github.com/NicoChou/homemade_stereo_obstacle_detection

https://github.com/NicolasBernard456/Planification

https://github.com/NicolasBernard456/SImu_robot_holonome

https://github.com/NightFury37/eklavya-4.0

https://github.com/NightGenie/epuck_consensus

https://github.com/NikoGiovannini/imu_calibration

https://github.com/NikoGiovannini/nostop_arduino

https://github.com/NikoGiovannini/nostop_fake_imu

https://github.com/NikoGiovannini/nostop_imu_reading

https://github.com/NikosCho/thesis_polymechanon_vision https://github.com/Nishida-Lab/TC2015

https://github.com/Niyati3/IRG-kukayoubot

https://github.com/NonatoLagunas/robot_service_manager

https://github.com/Nrikolo/X8_AutoPilot

https://github.com/Nsteel/Apollo_13

https://github.com/NunoDuarte/QuadsSimulator

https://github.com/OAkyildiz/src_capstone

https://github.com/OMMatte/robotics-lab-1.2-1.5-motor_control

https://github.com/OSLL/slam-constructor

https://github.com/OSLL/steel-lemon-0.1

https://github.com/OSLL/tiny-slam-ros-cpp

https://github.com/OSURoboticsClub/Rover2016

https://github.com/OSUrobotics/barrett_wam_grasp_capture_host

https://github.com/OSUrobotics/long-term-mapping

https://github.com/OSUrobotics/privacy-interfaces

https://github.com/OSUrobotics/ros-3d-interaction

https://github.com/OSUrobotics/ros-head-tracking

https://github.com/OTL/light_curtain

https://github.com/OTL/oculus

https://github.com/OTL/ros_book_programs

https://github.com/Octanis1/Octanis1-ROS

https://github.com/OctoMap/octomap_mapping

https://github.com/OctoMap/octomap_ros

https://github.com/Oflameo/APPPRS

https://github.com/OlenkaKa/DCL_ROSIntegration

https://github.com/Olin-FunRobo/FunRobo-GPSViewer

https://github.com/Olin-FunRobo/FunRobo-Simulator

https://github.com/OpenQbo/qbo_arduqbo

https://github.com/OptoForce/etherdaq_ros

https://github.com/OrebroUniversity/yumi_demos

https://github.com/Owensb/SmartWatchROS

https://github.com/PIBAUR/Plaisir

https://github.com/PMarcL/custom_ros_publishers

https://github.com/PR2/pr2_controllers

https://github.com/PR2/pr2_ethercat_drivers

https://github.com/PR2/pr2_robot.git

https://github.com/PR2/pr2_web_apps

https://github.com/PUTvision/ROS-labbot

https://github.com/PWESiberiaBear/movingCatchRobot https://github.com/PWESiberiaBear/study_ROS

https://github.com/PX4/Firmware

https://github.com/PX4/uvc_ros_driver

https://github.com/PacktPublishing/Effective-Robotics-Programming-with-ROS

https://github.com/PalouseRobosub/robosub_simulator

https://github.com/PalouseRobosub/robosub

https://github.com/ParitoshKelkar/ros_dev

https://github.com/Parrot-Developers/slamdunk_obstacle_avoidance_example

https://github.com/Parrot-Developers/slamdunk_ros

https://github.com/PascalBauer/Notre-bon-plaisir-Pascal-Bauer

https://github.com/Patrick2990/masterTank

https://github.com/PaulBouchier/truck

https://github.com/PaulBouchier/uvc_aggregator

https://github.com/PaulGrovesAtUNM/ECE595HW1

https://github.com/PaulWells/autonomous-quadcopter

https://github.com/PawelBurchard/NSwR

https://github.com/Pei-Yachao/My_Packages

https://github.com/Pei-Yachao/Other_Packages

https://github.com/PeiMengxin/dlib_track

https://github.com/PenguinCheng/multiDrone-ROS

https://github.com/PerryLiustc/x3_gimbal_track

https://github.com/PeterMitrano/nav_points

https://github.com/PeterMitrano/recovery_supervisor

https://github.com/PeterMitrano/vector_v1

https://github.com/Phenomics-KSU/htp_auto

https://github.com/Phorce/Owl

https://github.com/Pietrohl/higp-px4

https://github.com/Pitt-RAS/iarc7_motion

https://github.com/Pitt-RAS/iarc7_sensors

https://github.com/PixR2/visual_altimeter

https://github.com/PlasticDuck/RosProject

https://github.com/Pleiades-Spiri/spiri_go

https://github.com/Pleiades-Spiri/spiri_simulator

https://github.com/Po-Jen/NTU_BeeBot

https://github.com/Pontusun/CIS700_Team2

https://github.com/PositronicsLab/humanoid_catching

https://github.com/PositronicsLab/kinematics_cache

https://github.com/PrasadNR/ardupilot-rover-plugin

https://github.com/Prier/vision_dummy

https://github.com/ProjectARCeth/arc_state_estimation

https://github.com/ProjectARCeth/arc_tools

https://github.com/ProjectARCeth/erod_control

https://github.com/ProjectARCeth/pure_pursuit_controller

https://github.com/ProjectArtemis/aprilslam

https://github.com/ProjectArtemis/tegra_stereo

https://github.com/ProjectArtemis/vision_pipeline

https://github.com/ProjectX2014/Lindsey

https://github.com/ProjektPrzejsciowy/pionier_key

https://github.com/ProjektPrzejsciowy/projekt_przejsciowy

https://github.com/ProrokWielki/Projekt-Zespolowy

https://github.com/Puhapig/UGBots_ROS

https://github.com/PyroTeam/robocup-pkg

https://github.com/PyroTeam/robotino-pkg

https://github.com/QinZiwen/turtlebot2_roam

https://github.com/QinZiwen/turtlebot_k2_exploration_3d

https://github.com/QiwenZhang/gvg

https://github.com/QuentinLab/QboStuff

https://github.com/Quinn-Donnelly/pacman

https://github.com/Quizzarex/Bachelor2013

https://github.com/Quizzarex/LIDAR

https://github.com/RANHAOHR/EECS476_final_project

https://github.com/RANHAOHR/EECS476

https://github.com/RANHAOHR/SWARM_CONTROL

https://github.com/RANHAOHR/Tool_tracking

https://github.com/RANHAOHR/ps7-8

https://github.com/RAS2015-GROUP5/ras_localization

https://github.com/RC4Group1/ResearchCamp4

https://github.com/RCP1/evidence_navigation

https://github.com/RCPRG-ros-pkg/barrett_hand

https://github.com/RCPRG-ros-pkg/common_controllers

https://github.com/RCPRG-ros-pkg/ec_drivers

https://github.com/RCPRG-ros-pkg/elektron_ballcollector

https://github.com/RCPRG-ros-pkg/elektron_emor

https://github.com/RCPRG-ros-pkg/elektron_kinectbot

https://github.com/RCPRG-ros-pkg/irp6_robot

https://github.com/RCPRG-ros-pkg/lwr_hardware

https://github.com/RCPRG-ros-pkg/orocos_controllers

https://github.com/RCPRG-ros-pkg/pthead

https://github.com/RCPRG-ros-pkg/rys_robot

https://github.com/RCTemp/RCT_Source

https://github.com/RIVeR-Lab/aero_srr_13

https://github.com/RIVeR-Lab/aero_srr_14_old

https://github.com/RIVeR-Lab/aero_srr_14

https://github.com/RIVeR-Lab/aero

https://github.com/RIVeR-Lab/computer_sensors

https://github.com/RIVeR-Lab/kvh

https://github.com/RIVeR-Lab/manzanita_mk3

https://github.com/RIVeR-Lab/multisense_ros

https://github.com/RIVeR-Lab/orientus_driver

https://github.com/RIVeR-Lab/oryx_2012

https://github.com/RIVeR-Lab/walrus

https://github.com/RMonica/ros_kinfu

https://github.com/ROBOTIS-GIT/ROBOTIS-Framework

https://github.com/ROBOTIS-GIT/ROBOTIS-THORMANG-MPC

https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver

https://github.com/ROBOTIS-Kayman/ROBOTIS-OP

https://github.com/ROBOTIS-OP/robotis_op_ros_control

https://github.com/RPLtemp/auto_trax

https://github.com/RRT-Team/free_laser

https://github.com/RSPIBot/RSPiBot_Laser_node

https://github.com/RachaelT/UTDchess-RospyXbee

https://github.com/RaduAlexandru/Visual-Slam-Lab

https://github.com/RafBerkvens/shimmer

https://github.com/RafaelMarquesRodrigues/SORS_Application

https://github.com/Raffa87/Human_tracker

https://github.com/Rahulrkr1996/Encoder_IGVC_2015

https://github.com/Rajesh-Ru/rokyo_robotics_proj

https://github.com/Ram-Z/as_assignments

https://github.com/RandomEvilHero/this_robot

https://github.com/RaulPL/golem_navigation

https://github.com/RaulPL/navigation_services

https://github.com/RaulPL/patrol_filter

https://github.com/Razlaw/laser_cam

https://github.com/RedwanNewaz/ActiveSLAM

https://github.com/RedwanNewaz/hextree_ros

https://github.com/Renda110/AGVC

https://github.com/Rentier/ros-panopticon https://github.com/ResByte/3D-mapping

https://github.com/ResByte/graph_slam

https://github.com/Rick-Kota/tutorial_sim

https://github.com/RizHuang/draw_airline

https://github.com/RobDos/Collector

https://github.com/Roberto-Vega/seguimiento-cara

https://github.com/RoblabWhGe/thermo_3d_fusion

https://github.com/RoboCupTeam-TUGraz/tedusar_manipulation

https://github.com/RoboGroup3/hand_follower

https://github.com/RoboGroup3/path_planning

https://github.com/RoboGroup3/transforms

https://github.com/RoboHow/pr2_sot_integration

https://github.com/RoboJackets/igvc-software

https://github.com/RoboJackets/roboracing-software

https://github.com/RoboSec/ros_rs_sensor_board

https://github.com/Roboauto/Udacity3

https://github.com/Roboluv-TW/pcl_cv_ws

https://github.com/Roboluv-TW/rqt_multiplot_plugin_ws

https://github.com/RobospectEU/robospect_planner

https://github.com/RobotControlTechnologies/ros_function_module

https://github.com/RobotJustina/Toretto

https://github.com/RobotMa/MYO-TurtleBot

https://github.com/RobotMa/ur5_catheter_tracking

https://github.com/RobotRose/rose_arm_controller

https://github.com/RobotRose/rose_luctor

https://github.com/RobotRose/rose_navigation

https://github.com/RobotRose/rose_qt_ui

https://github.com/RobotRose/rose_ui_support

https://github.com/RobotRose/rose_utils

https://github.com/Roboterbastler/known_map_localization

https://github.com/RoboticSwarmControl/Swarmathon2016

https://github.com/Robotica-ule/MYRABot

https://github.com/Robotica2016/pioneer_control

https://github.com/RoboticaAI/tutorial_ros

https://github.com/Robotics-DAI-FMFI-UK/smely-zajko-ros

https://github.com/Robotics-UniBG/MarkerBasedNavigation_Orocos

https://github.com/RoboticsClubatUCF/ucf_igvc

https://github.com/RobotnikAutomation/agvs_sim

https://github.com/RobotnikAutomation/agvs https://github.com/RobotnikAutomation/fotonic_3dcamera

https://github.com/RobotnikAutomation/guardian_robot

https://github.com/RobotnikAutomation/mico_common

https://github.com/RobotnikAutomation/nav200_laser

https://github.com/RobotnikAutomation/omni_drive_controller

https://github.com/RobotnikAutomation/robot_localization_utils

https://github.com/RobotnikAutomation/robotnik_arduimu

https://github.com/RobotnikAutomation/robotnik_dpro_controller

https://github.com/RobotnikAutomation/robotnik_movebase_planner

https://github.com/RobotnikAutomation/robotnik_ptu

https://github.com/RobotnikAutomation/robotnik_purepursuit_planner

https://github.com/RobotnikAutomation/robotnik_test_robot

https://github.com/RobotnikAutomation/summit_xl_common

https://github.com/RobotnikAutomation/summit_xl_our_common

https://github.com/RobotnikAutomation/summit_xl_sim

https://github.com/RobotnikAutomation/vulcano_common

https://github.com/RobotnikAutomation/vulcano_sim

https://github.com/Robots-de-Rescate/Kauil_ROS

https://github.com/Roboy/DarkRoom

https://github.com/Roboy/roboy_simulation

https://github.com/RobustFieldAutonomyLab/eth_icp_mapper_indigo

https://github.com/RobustFieldAutonomyLab/explo_turtlebot_dev

https://github.com/RobustFieldAutonomyLab/loam

https://github.com/RobustFieldAutonomyLab/turtlebot_exploration_3d

https://github.com/RomMarie/open_rm

https://github.com/RommelLayco/ROS-orchard-project

https://github.com/RoseTeam/Futurakart

https://github.com/RoseTeam/ros-rosebot

https://github.com/RossHanson/paper-robot

https://github.com/Russell91/USCAerialRobotics

https://github.com/Ry0/multi_kinect

https://github.com/RyanLowerr/crawler_ros

https://github.com/RyanLowerr/okq1_ros

https://github.com/RyodoTanaka/gim30

https://github.com/RyodoTanaka/lrf_scan

https://github.com/RyuYamamoto/ChoreonoidRosBridge

https://github.com/RyuYamamoto/quadrotor_ros_sample

https://github.com/SDSMT-CSC464-F15/landingpad

https://github.com/SDU-Robotics/fmapps_archive https://github.com/SDU-Robotics/fmcomponent_archive

https://github.com/SHTseng/renbo_whole_body_plan

https://github.com/SIA-UAVGP/px4_code

https://github.com/SICKAG/sick_visionary_t

https://github.com/SIRSIIT/soma_ur5

https://github.com/SJTU-Multicopter/ardrone_mavros

https://github.com/SJTU-Multicopter/ardrone_station

https://github.com/SNU-Sigma/ros_ahrs

https://github.com/SOSVR/SOSVR2016

https://github.com/SOSVR/base_code

https://github.com/SPARbot/turtlebot_navigation

https://github.com/SQRSMN/RIC

https://github.com/SSL-Roots/roots_2dnav_pid

https://github.com/SSL-Roots/ssl_roots_ai

https://github.com/SSL-Roots/world_observer

https://github.com/SUTURO/euroc_perception

https://github.com/SUTURO/suturo_manipulation

https://github.com/SV-ROS/srrc_img2dir

https://github.com/Sabeehulhassan/Manyears

https://github.com/SaidBenaissa/Ekf_Slam_Khep

https://github.com/SakshiAgarwal/Task-1

https://github.com/SakshiAgarwal/linefollower.pid

https://github.com/Salvagm/Computer-Vision

https://github.com/SalvoVirga/iiwa_stack

https://github.com/SawYer-Robotics/ls01c

https://github.com/SawYer-Robotics/roch_project

https://github.com/SawYer-Robotics/roch_robot

https://github.com/SawYer-Robotics/roch

https://github.com/SaxionLED/skynav

https://github.com/SaxionLED/x80sv

https://github.com/Scalevo/TCP_Server

https://github.com/Scalevo/scalaser

https://github.com/ScazLab/baxter_collaboration

https://github.com/ScazLab/baxter_tictactoe

https://github.com/Scheik/ROS-Workspace

https://github.com/SeRViCE-Lab/ensenso

https://github.com/SeRViCE-Lab/p3-dx

https://github.com/SeamusJohnston/roBoat

https://github.com/Seanmatthews/rowboat1 https://github.com/Secret-Asian-Man/REU_Swarmie

https://github.com/SelmaKchir/BugAlgorithms

https://github.com/SemRoCo/giskard_examples

https://github.com/Semiustus/csula_swarm_map

https://github.com/SenorRobot/SenDes

https://github.com/Sergeus/HCR

https://github.com/SergioGarG/ISVLAMAV

https://github.com/SergioGarG/IndExMAV

https://github.com/ShanmukhaManoj11/AMAV-Vision

https://github.com/ShaoTziYang/HKUST-ML3-14

https://github.com/ShaoTziYang/ML3-14

https://github.com/ShaowuYang/idSLAM

https://github.com/ShapeCompletion3D/graspit_shape_completion

https://github.com/ShaunHoward/spars_rover

https://github.com/ShaunKarran/dragonpi

https://github.com/Shedino/RoverHighLevel

https://github.com/Shedino/SherpaHighLevel

https://github.com/Shengdong/servo_motor_rs232_test

https://github.com/ShiyuanChen/Contact_Localization

https://github.com/ShotaroTerato/go_straight

https://github.com/ShotaroTerato/traversal_layer

https://github.com/Shrawan99/Robot-Operating-System

https://github.com/Shuhei-YOSHIDA/anoros

https://github.com/SiegfriedNijs/Thesis

https://github.com/SimonEbner/ba

https://github.com/SimoneNardi/nostop_agent_sensor

https://github.com/SimoneNardi/nostop_agent

https://github.com/SimoneNardi/nostop_analysis

https://github.com/SimoneNardi/nostop_kinect_sensor

https://github.com/SimoneNardi/nostop_robot_localization

https://github.com/SimoneNardi/nostop_simulator

https://github.com/SithAP/obstacle_removal

https://github.com/SkrobiM/Maciej

https://github.com/Skydes/PRisme-Sim

https://github.com/SmallGreens/px4_AutonomousFlight

https://github.com/Smart-Minotaur/nxt_minotaur

https://github.com/SongM/2015

https://github.com/SorenMV/rob1b217_rpro_mini-project

https://github.com/SorenMV/rob1b217

https://github.com/SquadCollaborativeRobotics/CollectorBotReal

https://github.com/SquadCollaborativeRobotics/april_tags

https://github.com/SquadCollaborativeRobotics/global_planner_fall

https://github.com/SquadCollaborativeRobotics/global_planner

https://github.com/SquadCollaborativeRobotics/sonar_to_laserscan

https://github.com/StanislawAntol/husky_gtsam_odom_demo

https://github.com/StanleyInnovation/segway_v3_robot

https://github.com/Ste1989/MCU_ArCaRa

https://github.com/Stefan210/kinematic_calibration

https://github.com/Stefdil88/flight_controller

https://github.com/StepMan91/MsCV-UE4-Robotics_Project

https://github.com/StephMc/vrep_ros

https://github.com/SteveMacenski/ROS-Nodes-and-Development

https://github.com/Stoven-Dubois/Babybot

https://github.com/Sugiue/apriltag

https://github.com/Sukmin-Yoon/orin_irobot

https://github.com/SuperRoboticsNerds/Path_planning_real

https://github.com/SuperRoboticsNerds/manual_control

https://github.com/SuperRoboticsNerds/motors

https://github.com/SuperRoboticsNerds/ras_brain

https://github.com/SuperRoboticsNerds/ras_robot_vision

https://github.com/SuperSHODAN/RoboMap

https://github.com/SupermanPrime01/Quadcopter-Escort-2016-2017

https://github.com/SurikKrdoyan/tas_group14

https://github.com/Swarm-IITKgp/swarm_simulator

https://github.com/SyllogismRXS/syllo-gazebo

https://github.com/SyllogismRXS/syllo-ros

https://github.com/SynergicRobotics/CollabNav

https://github.com/THeK3nger/quadrotor-artag-navigation

https://github.com/THeK3nger/ros-quadrotor-obstacle-avoidance

https://github.com/TIR4N4Z0R05/Rtabmap-Odomerty-tracker

https://github.com/TNTECHARC/tntech-arc

https://github.com/TRECVT/ros_kvh1750

https://github.com/TShapinsky/ROSCock

https://github.com/TaarLab/husky_joy

https://github.com/TatsuyaAmano/pcl_human_detection

https://github.com/Team-ALFRED/alfred_ws

https://github.com/Team-Anveshak/rover-control

https://github.com/Team-Nonstop/nonstop_robots https://github.com/TeamAeolus/Offboard-Control

https://github.com/Teknoman117/linbot

https://github.com/Teknoman117/ucm-ros-pkg

https://github.com/Terabee/terarangerhub-ros

https://github.com/TheCamusean/ROS_Ardrone

https://github.com/TheDash/moveit_pr2

https://github.com/TheJacketMan/Pannello

https://github.com/TheOnlyAnodien/miniproject_ws

https://github.com/TheiaRobo/Theia

https://github.com/TheoLong/widow_mine

https://github.com/Thesane/microstrain_3dm_gx3_45

https://github.com/ThiagoAlvesLima/raspberry

https://github.com/Thomas00010111/Px4Tools

https://github.com/ThomasTimm/ur_modern_driver

https://github.com/TimSweering/test2

https://github.com/Timbabs/First-Robotics-Project

https://github.com/TimboInSpace/Armin

https://github.com/TjlHope/Rutler

https://github.com/TobiasBaer/marble_plugin

https://github.com/Tobichimaru/simulator

https://github.com/TomHulme/306-Swarm-Robotics-Project

https://github.com/Tomakko/RL_nav

https://github.com/TorstenHeverhagen/de-floribot-software

https://github.com/ToshibaRobot/pick-and-place

https://github.com/Towards-Autonomous/kitti_player

https://github.com/Towards-Autonomous/simple_loam

https://github.com/Trent0881/block-sorter-397

https://github.com/Trexter/HighlyAutonomousAerialReconnaissanceRobot

https://github.com/TsotsosLab/ps4_ros

https://github.com/TuZZiX/ariac

https://github.com/TuZZiX/baxter_moveit

https://github.com/TuZZiX/coke_fetcher

https://github.com/TuZZiX/p5_beta

https://github.com/TuZZiX/p7_beta

https://github.com/TuZZiX/p8_beta

https://github.com/TuZZiX/ros_workspace

https://github.com/TurtleZhong/Python-Learning

https://github.com/Tutorgaming/cyphy-people-mapping

https://github.com/Tutorgaming/ros-dumbobot https://github.com/Tutorgaming/ros-teleop-joy

https://github.com/TwoBit01/agve

https://github.com/Tyler-Gauch/quadroter_auto_nav

https://github.com/UAVMasterLabs/firstpackage

https://github.com/UBC-Snowbots/IARRC2010

https://github.com/UBC-Snowbots/IARRC2011

https://github.com/UBC-Snowbots/IGVC-2017

https://github.com/UBC-Snowbots/IGVC2015

https://github.com/UBC-Snowbots/Snowflake

https://github.com/UC3MSocialRobots/maggie_navigation

https://github.com/UC3MSocialRobots/people_detection_vision

https://github.com/UC3MSocialRobots/pose_tracker_stack

https://github.com/UC3MSocialRobots/vision_utils

https://github.com/UGVProject/IMU_taobao

https://github.com/UHDRobotics/NASACompetition2017

https://github.com/UM-ARM-Lab/sdf_tools

https://github.com/UM-ARM-Lab/uncertainty_planning_examples

https://github.com/UML-Robotics-Club/igvc-2012

https://github.com/UMLRoverHawks/gps_umd

https://github.com/USC-ACTLab/crazyflie_ros

https://github.com/USC-ACTLab/create2_ros

https://github.com/USTseniordesignSNOWPLOW2016/ROSCODE

https://github.com/UT-RAM/viki-modules

https://github.com/UTNuclearRoboticsPublic/contact_control

https://github.com/UTNuclearRoboticsPublic/netft_utils

https://github.com/UTNuclearRoboticsPublic/temoto_intuitive_teleoperator

https://github.com/UVa-StarLab/asctec_pos_control

https://github.com/UVa-StarLab/bebop2_controller

https://github.com/UbiCALab/cam_exploration

https://github.com/UbiquityRobotics/fiducials

https://github.com/UltharPeliki7/oduasv2017

https://github.com/UniOL-AMT/uol-robotino-ros-indigo

https://github.com/Unicornair/AUAR_Robot

https://github.com/UtahDARCLab/darc_ardrone

https://github.com/UtahDARCLab/darc_custom_joy

https://github.com/UtahDARCLab/darc_manual_fly

https://github.com/VCunhaJ/FRASIER_Robot_WPI

https://github.com/VCunhaJ/PARbot2_WPI_2015

https://github.com/VCunhaJ/Senz3D_Virtual_Fixtures https://github.com/VCunhaJ/softkinetic_interactive_Gesture_Camera

https://github.com/VRI/OpenVRI

https://github.com/VTAstrobotics/Austonomy1617

https://github.com/VahidAminZ/db_ati_sensor

https://github.com/ValentinVERGEZ/RBQT_communication

https://github.com/ValentinVERGEZ/pyro-robotino

https://github.com/VandyUAVCourse/Documents

https://github.com/Veerachart/blimp_controller

https://github.com/Vibek/Approaching-Goal-using-QR-Code

https://github.com/Vibek/Human_intention

https://github.com/ViktorWalter/MRS-OpticFlow

https://github.com/VinArt/amr-stage

https://github.com/VinArt/assignment

https://github.com/VinArt/stagetest2

https://github.com/VinArt/stagetest

https://github.com/Vincenzo88/Planner_RRT-_Robotica2

https://github.com/Vinhuit/ROS_WHEELROBOT

https://github.com/VisualComputingInstitute/ROS-laserdumper

https://github.com/Vitorluizcalmon/repositoriofinal

https://github.com/Voidminded/DQNStageROS

https://github.com/Vtn21/amr_carrot

https://github.com/Vtn21/amr_go2point

https://github.com/Vtn21/amr_obstacle

https://github.com/Vtn21/amr_race

https://github.com/WEARHAP/data-reader

https://github.com/WHCobot/Pizza_Demo

https://github.com/WPI-AIM/wpi-dvrk-ros

https://github.com/WPI-ARC/datalink_toolkit

https://github.com/WPI-ARC/drc_common

https://github.com/WPI-ARC/hubo_ros_core

https://github.com/WPI-ARC/mocap_robot_pose

https://github.com/WPI-FRASIER/FRASIER_Computer_Vision

https://github.com/WPI-FRASIER/PARbot

https://github.com/WSMao/i_robot

https://github.com/WYXF007/Halobject

https://github.com/WalkingMachine/Short-Circuit

https://github.com/WalkingMachine/sara_commun

https://github.com/WangJinxinNEU/jsonros

https://github.com/WangRobo/peter_motor_core_test https://github.com/Wangxuefeng92/nav_april_laser_odom

https://github.com/Wangxuefeng92/slam_april_laser_odom

https://github.com/Waywrong/ROS_Nav

https://github.com/Waywrong/ros_win_msg

https://github.com/WesleyYep/ROS_Testing

https://github.com/WingBot/hans_bringup

https://github.com/WingBot/pose2odom

https://github.com/Winwinindustrial/rplidar

https://github.com/WrRichy/navkite

https://github.com/Wrakor/robo-cat-and-mouse

https://github.com/WuNL/lab_workspace

https://github.com/WuNL/rob_workspace

https://github.com/X-Bar/X-Bar_stack

https://github.com/Xala/MAV

https://github.com/Xala/MEX-Quad

https://github.com/Xamla/torch-ros

https://github.com/XavierBouvard/LunarNXT

https://github.com/Xiarain/drone-control

https://github.com/Xu12/path-node

https://github.com/Xu12/path

https://github.com/XwLu/Notes

https://github.com/XxingLi/AGV_NUC

https://github.com/XxingLi/GLM_AGV

https://github.com/YZHANGFPE/pick_challenge

https://github.com/YaleIGVC/IGVC2014

https://github.com/Yang---XU/asctec_to_gazebo

https://github.com/Yeshasvitvs/Dynamixel_pantilt

https://github.com/YildizTeam/SkidSteeringNoiseModel

https://github.com/YoanSallami/head_manager

https://github.com/YoheiKakiuchi/jvrc_temp

https://github.com/YouBot-SAAS/mocap_optitrack

https://github.com/Youngdong-Clark-Son/keyboard_teleop

https://github.com/YusakuSakamoto/my_kit-C3

https://github.com/Yvaine/grid_map_avoidance

https://github.com/ZXWBOT/odom_tf_package

https://github.com/ZXWBOT/rplidar_scan

https://github.com/ZahraBoroujeni/Autos

https://github.com/ZahraBoroujeni/astar

https://github.com/ZahraBoroujeni/mpc_planner https://github.com/ZahraBoroujeni/showing_path

https://github.com/ZahraBoroujeni/spline_test

https://github.com/Zaki-/pyamr

https://github.com/ZdenekM/toad

https://github.com/Zephyrin/turtlebot_ros_hydro

https://github.com/Zhangziyang14/roman-technologies

https://github.com/ZhecanJamesWang/comprobo15

https://github.com/ZhenshengLee/zsROS

https://github.com/ZhiangChen/eecs_376_Mobile_Robotics_delta

https://github.com/ZhiangChen/eta3_spline

https://github.com/ZhiangChen/multi_motion_planning

https://github.com/ZhuYuJin/Project

https://github.com/a-price/ap_robot_utils

https://github.com/a-price/hubo_motion_ros

https://github.com/a-price/reactor_controller

https://github.com/a0919958057/MapDataReceiver

https://github.com/aa755/cfg3d

https://github.com/aamalik/gnss-master

https://github.com/aamirahmad/H-MRI_Search_and_Rescue

https://github.com/aamirahmad/cooperative_target_tracking

https://github.com/aamirahmad/read_dsr_dataset

https://github.com/aamirahmad/read_omni_dataset

https://github.com/aamirahmad/telekyb

https://github.com/aaramirezd/kinect2_network

https://github.com/aaronfostersmith/sweeper_interface

https://github.com/aaronma37/muro_lab_summer

https://github.com/aau-ros/aau_multi_robot-release

https://github.com/aau-ros/aau_multi_robot

https://github.com/ab3nd/TinyRobo

https://github.com/abdelrahman-osama/RGB-D-SLAM

https://github.com/abdullah38rcc/kfls2

https://github.com/abdullahmohiuddin/rotors_simulator

https://github.com/abencomo/AutonomousFlight

https://github.com/abhat91/ros_osx

https://github.com/abhigoudar/pixhawk_ws

https://github.com/abhinavkssk/c2_pilot_ddp

https://github.com/abhinavkssk/c2_ros

https://github.com/abhishekrajdutta/usingjoy

https://github.com/abrzozowski/ros_example_node https://github.com/abubakr007/mapping-stereo-camera-imu

https://github.com/abubeck/brics_research_camp

https://github.com/abubeck/ros_dissect

https://github.com/abuchele/robo_team_charlie_v2

https://github.com/acaldas/Footballess

https://github.com/acdb1112/Mcn-uav

https://github.com/acosgun/svn_old_ros_pkgs

https://github.com/acousticachilles/From_BitBucket_UGV_course

https://github.com/adamjardim/youbot_teleop

https://github.com/adasta/adk_linefollower

https://github.com/adasta/rosserial_cmake_test

https://github.com/adeguet1/sawDATAQSerial

https://github.com/adeguet1/sawForceDimensionSDK

https://github.com/adham-negm/ROS_SLAM

https://github.com/adi243/raha2016_slam

https://github.com/adithyamurali/cisst-ros

https://github.com/adrelino/loam_velodyne-1

https://github.com/adrianohrl/arena-unifei-hydro-ros-pkg

https://github.com/adrianohrl/mas-indigo-ros

https://github.com/adrianohrl/ros-indigo-refbox-pkg

https://github.com/adrianohrl/ros_robotino

https://github.com/adrizein/ros-supelec

https://github.com/adrizein/speech-command-robot

https://github.com/advancedroboticsaws/angel

https://github.com/adwilson10/quad_track

https://github.com/agneevguin/husqvarna_V0

https://github.com/agoila/rp-Lidar

https://github.com/agrirobo/arcsys2

https://github.com/agrirobo/raspberry_pi_driver

https://github.com/agusorte/dynamic_segmentation

https://github.com/aida147/HallwayNavigator

https://github.com/aijunbai/quadrotor_moveit

https://github.com/air-lasca/air1

https://github.com/airballking/feature_constraints_controller

https://github.com/aishwaryap/rlfd_ast3

https://github.com/aislancesar/RoboFEI-HR

https://github.com/ajayjain/husky_pursuit

https://github.com/ajayjain/nps_ws

https://github.com/ajithmaniac/fakelaser_viewer https://github.com/ajwolverton/Dronesmith-tech

https://github.com/akashgaurav/odom_base-link

https://github.com/akashgaurav/vectornav

https://github.com/akhil22/ekf_turtlebot

https://github.com/akhil22/human_prediction

https://github.com/akhil22/imu_data_to_yaw

https://github.com/akhil22/robot_localization-hydro-devel

https://github.com/akihirohh/imu_helvis

https://github.com/akshararai/HERMES_qp_robot

https://github.com/akshararai/HERMES_qp

https://github.com/akshararai/HERMES

https://github.com/aksungurlu/ros_joystick_trials

https://github.com/aladds/recycle-bot

https://github.com/alambert14/tag-the-turtle

https://github.com/alankarkotwal/urc-cnip-2015

https://github.com/alankarkotwal/urc2016

https://github.com/alba315/ardupilot

https://github.com/albertoCrive/minimalpnp_ROS

https://github.com/albertoLopezchaparro/swri-ros-pkg

https://github.com/aldarionsevero/apmcontroller

https://github.com/aleeper/cat_backend

https://github.com/aleixrr/pl_odometry

https://github.com/alex920a/Wrapper_Planner-BacherlorDegreeThesis

https://github.com/alexander-hagg/MAS

https://github.com/alexbaucom17/HuskyProject

https://github.com/alexluleme/Drone-Indoor-Navigation

https://github.com/alexsleat/projectChimaera

https://github.com/alextoind/reactive-grasping-task

https://github.com/algui91/grado_informatica_tsi_practicas

https://github.com/alishuja/range_to_exrange

https://github.com/allenc4/quadrotor_autonav

https://github.com/allenh1/Vanderbilt-ROS

https://github.com/almc/humanoid_framework

https://github.com/almc/nao_basic

https://github.com/alsaibie/PX4_Rescuebot

https://github.com/alvarovillena/velodyne_detect_person

https://github.com/amarburg/lsd_slam_conan

https://github.com/amayakal/Ur5_run

https://github.com/amazon-picking-challenge/plocka_packa https://github.com/amazon-picking-challenge/ru_pracsys

https://github.com/amazon-picking-challenge/rutgers_arm

https://github.com/amazon-picking-challenge/team_acrv

https://github.com/amazon-picking-challenge/team_cvap

https://github.com/amazon-picking-challenge/team_delft

https://github.com/amazon-picking-challenge/team_iitk_tcs

https://github.com/amelim/AR.Drone-Visual-Servo

https://github.com/amilgeorge/2015-PhotoGoalNavigation

https://github.com/amilgeorge/2015-VisualNavigation-Assignments

https://github.com/amitthobbi/thobbirepo

https://github.com/amndan/ekf_test

https://github.com/amndan/ohm_pf

https://github.com/amndan/slam_benchmarking

https://github.com/amor-ros-pkg/rosaria

https://github.com/amsully/rgb_line_navigating_robot

https://github.com/ana-GT/achq

https://github.com/ana-GT/robot_sim

https://github.com/anakinskyh/Robot

https://github.com/anand-ajmera/cyphy-vis-slam

https://github.com/anantanurag/en-route-roscon

https://github.com/ananyakka/eecs376

https://github.com/anaritaflp/usplasi_vo_evaluation

https://github.com/anbriel/GroundFirmware

https://github.com/andre-nguyen/an_mav_tools

https://github.com/andre-nguyen/an_ros_tools

https://github.com/andre-nguyen/bpvo_ros

https://github.com/andrea-nisti/AsctecInterface

https://github.com/andrea-nisti/lcm_bridge

https://github.com/andreasBihlmaier/lwr_safe_cartesian

https://github.com/andreasBihlmaier/trocar2cartesian

https://github.com/andreasBihlmaier/ur5_safe_cartesian

https://github.com/andrestoga/IntroductionToRobotics

https://github.com/andrestoga/Map_Merging

https://github.com/andriyukr/ros_optitrack

https://github.com/andschaf/GPS-sensorpod

https://github.com/andywolff/laser_flipper

https://github.com/andywolff/youbot_navigation

https://github.com/aneesh1993/Personal-Robotic-Butler

https://github.com/angetria/flir_lepton

https://github.com/angusleigh/leg_tracker_benchmarks

https://github.com/anindex/PathPlanning-ROS-Rviz

https://github.com/aniskoubaa/ROSWebServices

https://github.com/aniskoubaa/gaitech_edu

https://github.com/ankeshanand/auv-simulator

https://github.com/anosnowhong/table_mapping

https://github.com/antdroid-hexapod/antdroid

https://github.com/anthonymonori/hector-slam-wreck

https://github.com/antoniocoratelli/ros_utils

https://github.com/antonyomusabini/-TWO-EARS-_navigation

https://github.com/antunestiago/roboticappgi

https://github.com/antwonn/swarmie_cpp

https://github.com/anuppari/ardrone_follower

https://github.com/anuppari/aruco_ros

https://github.com/anuppari/homography_vsc_cl

https://github.com/anuppari/switch_vis_exp

https://github.com/anuragmakineni/laser_scanner

https://github.com/anyfilatov/ROS-course

https://github.com/aoklyunin/kukaRos

https://github.com/aolgu003/senior_design

https://github.com/aorait/beckonbotgesture

https://github.com/aorait/supergesture

https://github.com/apennisi/rgbd_person_tracking

https://github.com/apoorvcn47/GPR

https://github.com/apoorvcn47/marble_sidewalk

https://github.com/apostoee/pandora_ros_pkgs

https://github.com/appar3ntly/ros-demo-ucf

https://github.com/applededipan/Firmware1.5

https://github.com/aram-azbekyan/simple_manipulator

https://github.com/araml/robmovil

https://github.com/aransena/wip_ws

https://github.com/aranyadan/simulator_ip

https://github.com/aravindpreshant/bayesian_project

https://github.com/arbeitor/autonomous_charging

https://github.com/arc-lab/Pan-tilt-Stabilization

https://github.com/arc-lab/imu_tools

https://github.com/arcgithub/reference_finder

https://github.com/arebgun/erratic_robot

https://github.com/arg0n1s/automap

https://github.com/argenos/ros_multirobot

https://github.com/aritas1/ardrone_brown

https://github.com/arjunskumar/maze_solver

https://github.com/arjunskumar/swarm_robots

https://github.com/arnaud-ramey/followme_laser

https://github.com/arnaud-ramey/laser_random_walk

https://github.com/arnaud-ramey/multilaser_surveillance

https://github.com/arnaud-ramey/rosavatar

https://github.com/arnaud-ramey/rosmariokart

https://github.com/arnaud-ramey/roswifibot

https://github.com/arne48/arl_controllers

https://github.com/arne48/arm_navigation_hydronized

https://github.com/arne48/webots_youbot_ros_controller

https://github.com/arne48/youbot_kinect_registration

https://github.com/arpg/HAL

https://github.com/arpg/rpg-ros

https://github.com/arrayoutofbounds/se306-1

https://github.com/art32fil/laser_scan_builder

https://github.com/artificiell/qbo_angen

https://github.com/artur84/arduino_sketches

https://github.com/artur84/emotion

https://github.com/artursg/golfcart

https://github.com/arushk1/calibration

https://github.com/arwintang/pars-ros

https://github.com/asadat/nuc

https://github.com/asbroad/semantic_navigation

https://github.com/ashariati/blinker_tracking

https://github.com/ashish-kb/CNV

https://github.com/ashish-kb/ROS-cnv_quad_testcodes

https://github.com/ashish-kb/bayesian_compact_ware

https://github.com/ashish-kb/bayesian_submit_sim

https://github.com/ashokzg/billiards

https://github.com/ashokzg/cpb

https://github.com/ashraf1234/ic2020

https://github.com/ashwinreddy/maelstrom

https://github.com/asilx/rossi-demo

https://github.com/asl-beachbot/bb_controller

https://github.com/asr-ros/asr_descriptor_surface_based_recognition

https://github.com/asr-ros/asr_mild_base_fake_driving https://github.com/assiaben/gtCourses

https://github.com/astronaut71/Mapping-and-Tracking-packages

https://github.com/astronaut71/pow_analyzer

https://github.com/at-wat/hokuyo3d

https://github.com/at-wat/mcl_3dl

https://github.com/at-wat/rsj_robot_test

https://github.com/at-wat/trajectory_tracker

https://github.com/atenpas/handle_detector

https://github.com/athallas/rapp_face_detection

https://github.com/atulyashivamshree/cv_sens

https://github.com/auboliuxin/aubo_robot

https://github.com/auduchinok/trik-ros

https://github.com/augustjd/arm_navigation_msgs

https://github.com/auro666/666

https://github.com/aurone/moveit_planners_sbpl

https://github.com/aurone/moveit_scenarios

https://github.com/aurone/rcta_manipulation

https://github.com/aurone/sbpl_adaptive

https://github.com/aurone/spellbook_ros

https://github.com/aut-sepanta/Sepanta3

https://github.com/autonohm/robotworkshop

https://github.com/auviitkgp/kraken_3.0

https://github.com/auvnitrkl/tiburon

https://github.com/avansig/TeraRanger_hub-ROS-package

https://github.com/avirtuos/ros_hercules

https://github.com/avlara/Tilt-rotor_Simulator

https://github.com/avrora-robotics/ur_rangefinder_driver

https://github.com/awesomebytes/hist_user_tracking

https://github.com/awesomebytes/iri_poseslam

https://github.com/awesomebytes/p5glove_ros

https://github.com/awesomebytes/trac_ik

https://github.com/ax33/sep

https://github.com/axilos22/rescuer_project

https://github.com/ayshtmr/SmallObjectDiscovery

https://github.com/ayushgaud/mono2stereo

https://github.com/azaganidis/generalp

https://github.com/azaganidis/mutlimap_ndt

https://github.com/bIRIS-GSU/complete_test

https://github.com/babaksit/achilles

https://github.com/badrobit/Cartesian-Trajectory-Controller

https://github.com/badrobit/UGV-Controller

https://github.com/baishibona/jackal_exploration_3d

https://github.com/balakumar-s/lidarPotentialField

https://github.com/balakumar-s/p3atGazeboRos

https://github.com/balakumar-s/proximal_avoidance

https://github.com/ban-masa/test_rappelling

https://github.com/bansalvarun/swarath-PnC

https://github.com/barbara0811/subcultron-sim

https://github.com/barcelosandre/catkin_src

https://github.com/barno0695/Swarm-task

https://github.com/bartville/b_logger

https://github.com/barulicm/bassbot

https://github.com/bassie8881/ROS_stuff

https://github.com/bassie8881/ROS_uts_cas_wheelchair

https://github.com/basti35/pose_to_tf

https://github.com/batmanatucsd/blly

https://github.com/baxelrod/pr2_calibrated_ft

https://github.com/bbrieber/raconcom_env_scan

https://github.com/bbrieber/sherpa_unihb

https://github.com/bcharrow/matlab_rosbag

https://github.com/bcharrow/scarab

https://github.com/bcohen/pr2-band

https://github.com/bcoudrin/vec2odo

https://github.com/beamiter/ros_wtf

https://github.com/beaverauv/auv_mission_control

https://github.com/beckss/theWowBaggers

https://github.com/behnamasadi/mybot

https://github.com/bellenss/asctec_drivers

https://github.com/bellz867/homog_track

https://github.com/belolourenco/guardian-ros-pkg

https://github.com/bemery94/misc_funcs

https://github.com/ben-ebersole/rluob-devel

https://github.com/benadler/cognidrive_ros

https://github.com/benedictbauer/fuerte_sandbox

https://github.com/benersuay/riss_bot_teleop

https://github.com/bergercookie/slam_ws

https://github.com/beyersn/MicrogridSetup

https://github.com/bgromov/phantom_omni

https://github.com/bgromov/sensable_phantom

https://github.com/bgumodo/bgumodo-dev

https://github.com/bgumodo/bgumodo_ws

https://github.com/bgurobotics/location_controller

https://github.com/bgurobotics/map_from_position

https://github.com/bgurobotics/rosbot_path_finder

https://github.com/bhaskara/eband_planner

https://github.com/bhaskarsuper9000/land_robot

https://github.com/bhavyangupta/pandubot_wkspc

https://github.com/bigjun/ais-bonn-ros-pkg

https://github.com/billtyler/hackathon_ws

https://github.com/bing0037/state_machine

https://github.com/biobotus/rosserial

https://github.com/bioinroboticsuottawa/dextrus_ws

https://github.com/bioinroboticsuottawa/pumpkin_ws

https://github.com/birkiro/mdb_drone

https://github.com/birlrobotics/birl_baxter_controllers

https://github.com/birlrobotics/birl_baxter_demos

https://github.com/birlrobotics/birl_baxter_examples

https://github.com/birlrobotics/birl_baxter_vision

https://github.com/birlrobotics/birl_sensors

https://github.com/birlrobotics/pick_n_place_demo

https://github.com/bishwa9/dock_in_piece

https://github.com/bit-bots/bitbots_vision

https://github.com/bitathome-team/bitathome-2015W

https://github.com/bitcoinsoftware/UrbanDroneSystem

https://github.com/bitcoinsoftware/depth_algorithm

https://github.com/bj0rkedal/image_processor

https://github.com/bj0rkedal/qt_agilus_planner

https://github.com/bjfiedler/PMSR

https://github.com/bk8190/bk-ros

https://github.com/bk8190/bk_nav

https://github.com/bkap/Mobile_Robotics

https://github.com/blackhawkn7/EECS-397-Final-Project

https://github.com/blackhawkn7/ros_class

https://github.com/blackpc/stc_patrol

https://github.com/blandry/px4controller

https://github.com/blezalex/ros_hello

https://github.com/blinky-robot/waypoint_global_planner https://github.com/bloesch/MeasLoader

https://github.com/bloxidus/delta_final_project

https://github.com/bluesat/html5gui-poc

https://github.com/bluesat/mtig_driver_fork

https://github.com/bluesat/mtig_imu_driver_fork

https://github.com/bluesat/owr_software

https://github.com/bmagyar/imu_to_ground

https://github.com/bmagyar/loam_continuous

https://github.com/bobsomers/avc

https://github.com/boczekbartek/LABANRO

https://github.com/boottp/e2

https://github.com/boris-il-forte/ROSHardwareTools

https://github.com/bosch-ros-pkg/pr2-dancing-demo

https://github.com/bosch-ros-pkg/stm32

https://github.com/bpwiselybabu/drc_sensor_fusion

https://github.com/brNX/imu_offset

https://github.com/brNX/ros_mpu6050_node

https://github.com/bradleypowers/build_tools_test

https://github.com/brahmgardner/baxter_control

https://github.com/brakespear/rsa-ass2

https://github.com/brakespear/rsa-crosbot

https://github.com/breily/baxter_picking

https://github.com/brenmous/catkin_ws_tiago

https://github.com/brennerm23/LSR-TAS-GROUP-5

https://github.com/brent8149/teamMEGA

https://github.com/briantoth/BeerPongButler

https://github.com/brinkap/plesiosaur_arm_setup_tf

https://github.com/brunogouveia/intro-robotics

https://github.com/bsaund/Classes

https://github.com/bsb808/microstrain_3dm_gx5_45

https://github.com/bsgiovanini/projeto

https://github.com/buckbaskin/augmented_odom

https://github.com/budhi15/MeasureKinectFusion

https://github.com/bultu/ros_code

https://github.com/buuav/magician-slam

https://github.com/buzzer/pr2_imagination

https://github.com/buzzer/tams_pr2

https://github.com/bvobart/ES-RobotLab

https://github.com/bwi-spring-2013/segbot_apps https://github.com/byu-magicc-archive/fcu_io

https://github.com/byu-magicc/fcu_sim

https://github.com/byu-magicc/ros_copter

https://github.com/byu-magicc/ros_plane

https://github.com/bzsinger/joystick_bwi

https://github.com/cLuuLess/mae242

https://github.com/cabama/PatatitasRobot

https://github.com/callalilychen/legocar

https://github.com/callemein/Masterproef

https://github.com/calvanize/BlueBerry

https://github.com/cameron-ridgewell/bayes_team1

https://github.com/canaltinigne/ITU-Computer-Engineering

https://github.com/canyou2014/conclude

https://github.com/canyou2014/my_msf

https://github.com/caochao39/hku_m100_gazebo

https://github.com/carlesOA/ROS-system-in-a-ABB-IRB120

https://github.com/carlesOA/quadrotor_lab

https://github.com/carlosjoserg/stewart-ppcc-wrist

https://github.com/carlosmccosta/Robot-Object-Manipulation

https://github.com/carlosmccosta/laserscan_to_pointcloud

https://github.com/carlosmccosta/robot_localization_tools

https://github.com/carpe-noctem-cassel/cnc-msldriver

https://github.com/carpe-noctem-cassel/cnc-naos

https://github.com/carpe-noctem-cassel/cnc-turtlebots

https://github.com/carpe-noctem-cassel/ice

https://github.com/cartejac/maidbot_scanner

https://github.com/casimirsowinski/ROS_Control

https://github.com/casimirsowinski/robo_hand

https://github.com/cassidy-brown/eecs397_ros

https://github.com/catcatcatcatcatcatcat/rosrt

https://github.com/catppinto/robotnik_arm

https://github.com/catthomas/tour-guide-bot

https://github.com/cavedweller/mrp

https://github.com/cbegay/swarmathonNTU

https://github.com/cbourquin/CS491_AquaticRobotProject

https://github.com/cburbridge/uuisrc-ros-pkg

https://github.com/ccc395/Gazebo

https://github.com/cchart/betajinx

https://github.com/ccny-ros-pkg/cata

https://github.com/ccny-ros-pkg/ccny_gps

https://github.com/ccny-ros-pkg/mav_tools

https://github.com/ccny-ros-pkg/scan_tools

https://github.com/ccoulton/cs455

https://github.com/cdondrup/hesitation_study

https://github.com/cdrfiuba/papa

https://github.com/cedricpradalier/ros_kf_stack

https://github.com/cedricpradalier/vrep_ros_stack

https://github.com/cedricpradalier/vrep_ros_ws

https://github.com/cedricpradalier/vsv_stack

https://github.com/celibertojr/RoboFEI-TL

https://github.com/cephalsystems/drc

https://github.com/cer-nagas/roboticArm_ROS

https://github.com/cfernald/WiFi-Testbed-Robots

https://github.com/cfosco/turtlesim-rosjavasub-fosco

https://github.com/cgracia/robucar-simulation

https://github.com/chara1273/IS2016-project

https://github.com/chatrasen/Miscellaneous

https://github.com/cheehieu/deixis

https://github.com/chenshiyuhit/fly_offboard

https://github.com/chenshiyuhit/tld

https://github.com/chenxingzhe/laser_map

https://github.com/chesternimiz/micros_flocking

https://github.com/chgrand/effibot_driver

https://github.com/chgrand/velodyne_gps

https://github.com/chicagoedt/iarc_software

https://github.com/chicagoedt/revo_robot

https://github.com/chicagoedt/rmc_software

https://github.com/chinjao/ros_src

https://github.com/chiragmajithia/grid_map

https://github.com/chiragrajk/joy_arDrone

https://github.com/chiwunau/2013jskseminal

https://github.com/chiwunau/pcl_tutorials

https://github.com/chriskochunlong/robotics_camera_calibrate

https://github.com/chrisplent/ros-patroller

https://github.com/christianpfitzner/ohm_tutorial

https://github.com/christianpfitzner/thermal_workshop

https://github.com/christianrauch/ros-multiwii

https://github.com/chujong/CNMSwarmathon https://github.com/chukdor/ROS_York

https://github.com/chuthagoras/img_service

https://github.com/cipherlab-poly/crazyflie-public

https://github.com/cissovilla/ICsergio

https://github.com/cjpedc/ras_lab1_ctrl

https://github.com/cjpedc/rcv_setup_tf

https://github.com/clairedune/ros_uw_osl_nessie_vs

https://github.com/clancye/estimation_term_project_clancy

https://github.com/clearpathrobotics/declination

https://github.com/clearpathrobotics/enu

https://github.com/clearpathrobotics/hg_ros_3dm_gx4

https://github.com/clearpathrobotics/imu_compass

https://github.com/clearpathrobotics/myo_ros_windows

https://github.com/clearpathrobotics/occupancy_grid_utils

https://github.com/clearpathrobotics/vrpn_client_ros

https://github.com/clems4ever/ardrone_loclib

https://github.com/clubcapra/Ibex

https://github.com/clydemcqueen/rodeobot

https://github.com/cmaestre/adaptive_affordance_learning_actions

https://github.com/cmastalli/excavabot

https://github.com/cmdli/door_assist

https://github.com/cmdunkers/Robo_Stats

https://github.com/cmdunkers/moving_fan

https://github.com/cmdunkers/softkinetic

https://github.com/cmgladding/aslam-project

https://github.com/cmiley/OLSR_Deployment

https://github.com/cmitash/6DPose-RCNN4PCS

https://github.com/cmitash/RobotLearning

https://github.com/cmitash/ShelfDetection

https://github.com/cmlaney/BallDropperROS

https://github.com/cmsc421/ar_track_alvar

https://github.com/cmsc421/mobility_base_simulator

https://github.com/cmumrsdproject/mrsd_ros_part3

https://github.com/cnguyen1994/racecar

https://github.com/cnhzcy14/cy-ros-pkg

https://github.com/cocasema/ros-lidar-lite

https://github.com/code-iai/designator_integration

https://github.com/code-iai/iai_boxy

https://github.com/code-iai/iai_control_pkgs https://github.com/code-iai/iai_robot_drivers

https://github.com/code-iai/pico_flexx_driver

https://github.com/codehacken/Athena

https://github.com/codenotes/DiffDriveShim

https://github.com/codenotes/OptiTrack

https://github.com/codenotes/SensorLabAccelRead

https://github.com/codenotes/androidlibs2

https://github.com/codenotes/opti-triangulator

https://github.com/cogniteam/hamster_gazebo

https://github.com/cogniteam/hamster_server

https://github.com/cogsys-tuebingen/csapex_core_plugins

https://github.com/colek42/TrackIt

https://github.com/columbia-robotics-drone/drone_ws

https://github.com/comet-robot/robot2015

https://github.com/contradict/SampleReturn

https://github.com/contradict/ros-navigation

https://github.com/corobotics/corobot-kinetic

https://github.com/corobotics/corobots

https://github.com/corot/annotations_store

https://github.com/corot/thorp

https://github.com/corot/world_canvas

https://github.com/correlllab/cu-perception-manipulation-stack

https://github.com/correlllab/jaco_cu

https://github.com/cosimani/verano_tec_2014

https://github.com/costashatz/nao_dcm

https://github.com/costashatz/vrep_catkin

https://github.com/cott81/rosha

https://github.com/cottsay/control_panel_sim

https://github.com/cottsay/joy_to_twist

https://github.com/countofkrakow/Stalker_Bot

https://github.com/cpduerk/edge_following

https://github.com/cra-ros-pkg/robot_localization

https://github.com/crcdor/voice-controlled-drone

https://github.com/cretaceous-creature/jsk_mbzirc_task3

https://github.com/cretaceous-creature/mbzirc_task2

https://github.com/crisb-DUT/summerCamp

https://github.com/cristiiacob/pr2_movements

https://github.com/critcher/EZ-Kart

https://github.com/crlee/imu_cam_acquisition https://github.com/crrlab368/Gen2_Platforms_old

https://github.com/crvogt/leap_to_parrot

https://github.com/csc301/iarc

https://github.com/csc301/ycc

https://github.com/cstenico/amr

https://github.com/ct2034/auckbot

https://github.com/cubei/OculusCamViewer

https://github.com/cuixiongyi/cxy-ros

https://github.com/cunnia3/CoboticsTurtlebot

https://github.com/curtiswest/pr2_laserscan_to_toto_sonar

https://github.com/curtiswest/turtlebot_toto_sonar_gusimplewhiteboard_poster

https://github.com/cvpapero/body_motion_analysis

https://github.com/cvpapero/goal_manager

https://github.com/cvpapero/mysql_connector

https://github.com/cvpapero/okao_client_legacy

https://github.com/cvpapero/okao_client

https://github.com/cvra/goldorak

https://github.com/cwru-robotics/babs

https://github.com/cwru-robotics/cwru-eecs-275

https://github.com/cwru-robotics/cwru-ros-pkg

https://github.com/cwru-robotics/cwru_base

https://github.com/cwru-robotics/cwru_baxter

https://github.com/cwrucutter/cwrucutter_core

https://github.com/cwrucutter/snowmower_bot_drivers

https://github.com/cwrucutter/snowmower_localization

https://github.com/cwrucutter/snowmower_sim

https://github.com/cwrucutter/snowmower_steering

https://github.com/cxzhp/ratslam

https://github.com/cyb7369299/hello-world

https://github.com/cyberphantom/arl_ardrone_examples

https://github.com/dachengxiaocheng/Neo_Robot

https://github.com/daft27/r2r

https://github.com/daichi-yoshikawa/ahl_3rd_party

https://github.com/daichi-yoshikawa/ahl_manipit

https://github.com/daichi-yoshikawa/ahl_ros_pkg

https://github.com/daichi-yoshikawa/existing_packages

https://github.com/daichi-yoshikawa/work_in_ca

https://github.com/daiemna/amr_ss14

https://github.com/daivikswarup/swarm

https://github.com/daju1-ros/cv_bridge

https://github.com/daju1-ros/geometry-experimental

https://github.com/daju1-ros/image_transport

https://github.com/daju1-ros/rpg_svo

https://github.com/daju1-ros/rqt

https://github.com/daju1-ros/tf

https://github.com/damanfb/darc_saca

https://github.com/damanfb/fake_sonar

https://github.com/damanfb/hs_study

https://github.com/damanfb/odroid_lidar

https://github.com/damanfb/vrep_lidar

https://github.com/damonkohler/parsec.rosserial

https://github.com/damonkohler/parsec

https://github.com/dan-git/outdoor_bot

https://github.com/dan-git/outside

https://github.com/dani-carbonell/loam_back_and_forth

https://github.com/daniel-serrano/rotors_exercise

https://github.com/danielgeier/ml2-spiking

https://github.com/danilo-perico/RoboFEI-HT_CBR_LARS_2015

https://github.com/danilo-perico/robofei_ht_framework

https://github.com/danimtb/pioneer3at_ETSIDI

https://github.com/dankex/compv

https://github.com/danthony06/basic_controller

https://github.com/daobilige-su/loam_velodyne

https://github.com/darin-costello/kris_alder_alog

https://github.com/darin-costello/mm_apriltags_tracker

https://github.com/dario-wat/rt-stitch

https://github.com/darknight-007/NSF-Student-UAV-Competition

https://github.com/darknight-007/mavros-v0

https://github.com/darknight-007/sitl_gazebo-v0

https://github.com/darryusa/robotic

https://github.com/daslrobotics/dasl-ros-pkg

https://github.com/davetcoleman/bolt

https://github.com/davetcoleman/trep_ros

https://github.com/davheld/driving_public

https://github.com/david-alejo/arcas_ws

https://github.com/david-alejo/thermal_ws

https://github.com/daviddoria/PCLMirror

https://github.com/davidfornas/joy_controls https://github.com/davidfornas/mobman-summerschool-ros-pkg

https://github.com/davidfornas/pcl_manipulation

https://github.com/davidfornas/pm_pipeline

https://github.com/davidrotor/tesis

https://github.com/davidswords/graphslam

https://github.com/davidtvs/calibration-old

https://github.com/dawonn/MichiganTech

https://github.com/dbs0412247/image_pipeline

https://github.com/dcanu/avoid-obstacle-turtlebot

https://github.com/deanzaka/truinect

https://github.com/decoderdan/projectPhoenix

https://github.com/deebuls/RecommenderSystemInRobotics

https://github.com/dejanpan/mapping

https://github.com/dejanpan/people_detector_node

https://github.com/delftrobotics/dr_eigen

https://github.com/delftrobotics/dr_ensenso

https://github.com/demulab/object_recognizer

https://github.com/demulab/person_recognizer

https://github.com/demulab/tukuba_challenge_2014

https://github.com/denbyk/people_tracker_denbyk

https://github.com/denis-draca/localisation

https://github.com/denis-draca/wallpusher_sensor_decompress

https://github.com/denkrau/ar_nav

https://github.com/dennisss/quadctrl

https://github.com/dennisss/tansa

https://github.com/denzist/adaptive_control

https://github.com/denzist/joy_teleop

https://github.com/denzist/octomap_buggy_demo

https://github.com/denzist/robot_navigation

https://github.com/deriyuval/gazeboFilm

https://github.com/deruhat/projectksr

https://github.com/desinghkar/Object-Search-ROS

https://github.com/devmax/AutoNav

https://github.com/dgitz/icarus_rover_rc

https://github.com/dgp34/eecs397_ros_hw

https://github.com/diav79/bebop_data

https://github.com/diav79/teleop_joy

https://github.com/diav79/teleop_key

https://github.com/dieg0-/amr_ws1314_individual https://github.com/diegocepedaw/UASResearch

https://github.com/digimatronics/ros_build_packages

https://github.com/digimatronics/ros_common_messages

https://github.com/digimatronics/ros_coordinates_transform

https://github.com/diogoalmeida/sarafun_folding_assembly

https://github.com/dirkcgrunwald/ros-sdr

https://github.com/dishank-b/AGV-Controls

https://github.com/divyashah/QATL

https://github.com/dji-sdk/Guidance-SDK-ROS

https://github.com/dkanou/walkman_calibration

https://github.com/dlunni/px4_dario_git

https://github.com/dmcconachie/KinematicsToolbox

https://github.com/dmcconachie/controls_project

https://github.com/dme722/FlyNet

https://github.com/dmeltz/multy_rob

https://github.com/dmillard/ardrone_lmt

https://github.com/dmitry64/ros-treasure-hunter

https://github.com/dmr-goncalves/TRSA

https://github.com/dnovichman/ground_station

https://github.com/doebrowsk/p8_zeta

https://github.com/doebrowsk/tank-chair

https://github.com/doebrowsk/tiny_path_service

https://github.com/doebrowsk/tiny_reaction_server

https://github.com/doebrowsk/zeta_closed_loop

https://github.com/doebrowsk/zeta_open_loop

https://github.com/dolmangi/ddori

https://github.com/dolmong/BackUp

https://github.com/dolmong/LocationSystem

https://github.com/dolphin-slam/dolphin_slam

https://github.com/domingoesteban/teo_remote

https://github.com/dominiquehunziker/libav_image_transport

https://github.com/donatodipaola/people_detector

https://github.com/doomzzju/ZED_Odom_Grabber

https://github.com/dornhege/navigation_trajectory_planner

https://github.com/dortmans/ros_examples

https://github.com/dqyi11/AprilTagsTracker

https://github.com/dqyi11/ColorBlobTracker

https://github.com/dqyi11/apriltags_intrude_detector

https://github.com/dqyi11/harrt_ros

https://github.com/dqyi11/morrf_ros

https://github.com/dqyi11/multi_apriltags_tracker

https://github.com/dqyi11/multi_camera_apriltags_tracker

https://github.com/drewlatwas/myracecar

https://github.com/drfeiliu/pioneer3_control_ros

https://github.com/dronesinma/ucl_drone_2016

https://github.com/dsapandora/SIMUL_DSA

https://github.com/dseredyn/barrett_hand_sim_dart

https://github.com/dseredyn/velma_core_cs_ve_body_interface

https://github.com/dseredyn/velma_core_cs

https://github.com/dseredyn/velma_core_ve_body_re_body_interface

https://github.com/dseredyn/velma_core_ve_body

https://github.com/dseredyn/velma_low_level_safety

https://github.com/dseredyn/velma_robrex

https://github.com/dseredyn/velma_sim_gazebo

https://github.com/dseredyn/velma_task_cs_ros_interface

https://github.com/dsgou/ros_visual

https://github.com/duckietown/Software

https://github.com/duke-iml/ece490-s2016

https://github.com/dumorgan/MobRobotics_as4

https://github.com/dumorgan/exs416_as7

https://github.com/duo3d/duo3d_driver

https://github.com/duongdang/hand_eye_calibration

https://github.com/durovsky/bin_pose_emulator

https://github.com/durovsky/sef_roboter

https://github.com/dvorak0/sensor_drivers

https://github.com/dwong229/catkinpkg_plane_camera_magnet

https://github.com/dxs/octanis

https://github.com/dxydas/ros-bioloid

https://github.com/dynMapMaster/dynamicMap

https://github.com/dynMapMaster/mir_layered_costmap

https://github.com/dzenoo/hexapod

https://github.com/e-lab/turtlebot

https://github.com/eagle-deep-blue/tutorial_cmd_vel_publisher

https://github.com/eborghi10/AR.Drone-ROS

https://github.com/eborghi10/BB-8-ROS

https://github.com/eborghi10/Fetch-ROS

https://github.com/eborghi10/Fetch-ROS

https://github.com/eborghi10/Hexapod-ROS https://github.com/eborghi10/Sphero-ROS

https://github.com/eborghi10/TIAGo-ROS

https://github.com/ecomptiago/TCC

https://github.com/ecward/RCV_course

https://github.com/edgarzhavoronkov/disp_map

https://github.com/edinferno/edinferno

https://github.com/edinpeter/brokenGlasses

https://github.com/edinpeter/thrust_vis

https://github.com/eemmyy23/imu-test

https://github.com/eemmyy23/imu-trivisio

https://github.com/eemmyy23/imu_position

https://github.com/eemmyy23/odom_covariance

https://github.com/eemmyy23/rma-ros-install

https://github.com/eetuna/eet12-dvrk-ros-wsn

https://github.com/eetuna/eet12_davinci_wsn

https://github.com/efernandez/iri_navigation

https://github.com/ehuang3/drchubo

https://github.com/ehuang3/moveit_ros

https://github.com/ein57ein/ros_plugin_demo

https://github.com/einarsn/px4_firmware

https://github.com/eitan-babcock/Kingfisher_Code

https://github.com/eitan-babcock/MRSD_Kingfisher

https://github.com/ejpark0618/grp_navi

https://github.com/ekelleyv/QuadcopterMapping

https://github.com/ekumenlabs/ifc6410p_mbed_ros_demo

https://github.com/ekumenlabs/ros2_demo

https://github.com/ekuri/baxter_moveit_stomp_trac_ik_config

https://github.com/elhussieny/eyetribe

https://github.com/elicucco/sheperd

https://github.com/elijahcz/ProjectX

https://github.com/elisabetta2016/Nicola

https://github.com/elisabetta2016/Robot2016

https://github.com/elisabetta2016/donkey_rover

https://github.com/elisabetta2016/pc_maker

https://github.com/elisabetta2016/pcl_analyser

https://github.com/elisabetta2016/rover_simulator

https://github.com/elitechrome/auv_ros

https://github.com/elmoF/KUKA_youBot_ROS_VREP

https://github.com/em-er-es/rollo

https://github.com/emalbt/3_imu

https://github.com/emalbt/disney_demo

https://github.com/emalbt/reactive_grasp

https://github.com/embeddedprogrammer/soccer-sim

https://github.com/embr/cs225b

https://github.com/emreay-/Arduino-Codes

https://github.com/emreay-/agv

https://github.com/emreay-/mobile_robot

https://github.com/ems1992/Demo_Matlab_SMP

https://github.com/ems1992/SMP_DEMO_MATLAB

https://github.com/endlesswho/real-time-slam-sysu

https://github.com/enriccorona/MobileRobotics-Challenge4

https://github.com/enyen/rosProgramming

https://github.com/eog-2017/cameras

https://github.com/epezhman/cpp_ros

https://github.com/epfl-lasa/kuka-lwr-ros

https://github.com/epfl-lasa/load-share-estimation

https://github.com/epfl-lasa/net-ft-ros

https://github.com/epfl-lasa/object-impedance-control

https://github.com/epfl-lasa/robot-toolkit

https://github.com/eranda1985/Drive_NXT

https://github.com/eratner/teach-a-robot

https://github.com/eric1221bday/hallway_navigator

https://github.com/ericdanz/laser_scanner_eric

https://github.com/erichahn/vtbolt_personal

https://github.com/ericperko/cwrucam

https://github.com/ericperko/precision_navigation

https://github.com/ericperko/ros_navigation

https://github.com/ericperko/uvc_cam

https://github.com/erik-nelson/octomap_server

https://github.com/erivera1802/ArDroneThesis

https://github.com/erlerobot/gazebo_cpp_examples

https://github.com/erlerobot/lrm30_ros

https://github.com/erlerobot/precaster_CA113

https://github.com/erlerobot/precaster_RPS800

https://github.com/erlerobot/ros_erle_imu

https://github.com/eroniki/ROS-Tutorials

https://github.com/eruffaldi/aruco_ros_offline

https://github.com/eruffaldi/ros_kinect_info https://github.com/esmilg/multivac

https://github.com/esonderegger/kuka_youbot

https://github.com/esraerdem/coaches

https://github.com/ethanrublee/ecto_ros

https://github.com/ethz-ait/duo3d_ros

https://github.com/ethz-ait/duo_vio

https://github.com/ethz-asl/asctec_mav_framework

https://github.com/ethz-asl/elevation_mapping

https://github.com/ethz-asl/ethzasl_sensor_fusion

https://github.com/ethz-asl/geodetic_utils

https://github.com/ethz-asl/image_undistort

https://github.com/ethz-asl/infinitam

https://github.com/ethz-asl/kindr_ros

https://github.com/ethz-asl/kitti_to_rosbag

https://github.com/ethz-asl/laser_slam

https://github.com/ethz-asl/mav_control_rw

https://github.com/ethz-asl/mavlink_ros

https://github.com/ethz-asl/okvis_ros

https://github.com/ethz-asl/ros-message-transport

https://github.com/ethz-asl/ros_vrpn_client

https://github.com/ethz-asl/traversability_estimation

https://github.com/ethz-asl/variant

https://github.com/ethz-asl/velodyne-ros

https://github.com/ethz-asl/visensor_node

https://github.com/etsardou/intelligent_robot_systems_2016

https://github.com/evangravelle/turtlebot_competition

https://github.com/exuvo/kbot

https://github.com/eybee/heatmap

https://github.com/f-alemauro/loopCloser

https://github.com/fabianschilling/robot-vision

https://github.com/fabrizioschiano/apriltag2

https://github.com/facontidavide/ros_type_introspection

https://github.com/farhanrahman/nice

https://github.com/fariner/VisBot

https://github.com/fawkesrobotics/fawkes

https://github.com/fbalaban/tnp

https://github.com/fboris/imu_publisher

https://github.com/fcolas/nifti_ros_utils

https://github.com/fcolas/ugv_3d_navigation https://github.com/fdcqsjn/ardrone_test

https://github.com/fei-ros-pkg/tf_p2os_demo

https://github.com/feixiao5566/learngit

https://github.com/feixiao5566/ratslam_feixiao5566

https://github.com/felipepolido/gaia

https://github.com/felix-kolbe/scitos_metralabs

https://github.com/felix-kolbe/uashh-rvl-ros-pkg

https://github.com/felixendres/rgbdslam_v2

https://github.com/felixmn91/hw-interfacing

https://github.com/ferdianjovan/sqsr

https://github.com/feroze/amor-ros-pkg

https://github.com/fetchrobotics/fetch_ros

https://github.com/fetchrobotics/robot_controllers

https://github.com/fevb/DD2425_2013

https://github.com/fevb/dumbo_contact_point_estimation

https://github.com/fevb/dumbo_ft_drift_compensation

https://github.com/fevb/pr2_wrench_estimation

https://github.com/feying/tutorials

https://github.com/fferri/octomap_path_planner

https://github.com/field-robot/simple_navigation_goals

https://github.com/field-robot/slam_navigation

https://github.com/filipelbc/tangentbug

https://github.com/filipjares/mkr

https://github.com/filipnovotny/pioneer_tools

https://github.com/filipnovotny/turtlesim_pioneer

https://github.com/fionachui/PX4-Firmware_Recovery

https://github.com/fishpepper/dlib_facedetector_ros

https://github.com/fishpepper/nmpt_saliency_ros

https://github.com/fivef/kate_apps

https://github.com/fkaiser/image_imu_packer

https://github.com/fkaiser/ros_to_opencv

https://github.com/fkanehiro/choreonoid_ros_pkg

https://github.com/fl4v/lgsm_conversions

https://github.com/fl4v/manipulator_state_publisher

https://github.com/flatlevel/Luci-APM

https://github.com/flobotics/flobotics_tf2_to_joint_states

https://github.com/florianhauer/GT_mini_autonomous_car

https://github.com/florianhauer/mapnplan

https://github.com/florianhauer/msp_pr2_test https://github.com/flytbase/flytsamples

https://github.com/fmrchallenge/fmrbenchmark

https://github.com/fmrico/WaterMellon

https://github.com/fmw-hb/follow_me

https://github.com/fontysrobotics/robot_pick_place

https://github.com/foolchi/Extracting_Wall

https://github.com/formica-multiuso/vrep-ros-joypad-pack

https://github.com/forno/rosTutorials

https://github.com/fqez/common

https://github.com/fqez/kobuki_test

https://github.com/francescoriccio/sem_map_demo

https://github.com/francibon93/ar_sys

https://github.com/francibon93/ros-automatic-landing

https://github.com/francisc0garcia/education_robotics

https://github.com/francisc0garcia/suricate_robot

https://github.com/francisengelmann/FabScan

https://github.com/francoisferland/hbba_base

https://github.com/francoisferland/irl_audio

https://github.com/franzlst/PbD_FRI

https://github.com/fredrabelo/humanoids_robots

https://github.com/freefloating-gazebo/freefloating_gazebo

https://github.com/french1802/Ros_supervizer

https://github.com/freshNfunky/ros-indigo

https://github.com/frog0705/universal_robot

https://github.com/fruitdove/laser_consumer

https://github.com/fsfrk/hbrs-youbot-hackathon

https://github.com/fsteinhardt/mpu6050_serial_to_imu

https://github.com/fuhuayu/slam_apriltag_isam_single

https://github.com/fujy/ROS-Project

https://github.com/fukafuka-kobuki/rostomtsa

https://github.com/fulei623/pronto-distro

https://github.com/fulkast/RPL_WORK

https://github.com/furushchev/_OLD_ROBOTIS-OP

https://github.com/furushchev/jsk_2014w_rinko

https://github.com/furushchev/template_based_grasp_planning

https://github.com/futuhal-arifin/ardrone_swarm

https://github.com/fvantilburg/joost

https://github.com/fxia22/TurtleSeek

https://github.com/fxia22/tinker

https://github.com/g-ch/Clarence-Chan-new

https://github.com/g-ch/Clarence-Chan

https://github.com/g-ch/Clarence-mav

https://github.com/g-ch/chen_rasberry

https://github.com/g-ch/crop_station

https://github.com/g-ch/imiate_file

https://github.com/g-ch/mavros_rosberrypi

https://github.com/g-ch/qiaosui_workspace

https://github.com/g-sobral/ransac_catkin

https://github.com/g41903/PEV-MediaLab

https://github.com/gKouros/path-smoothing-ros

https://github.com/gKouros/reeds-shepp-paths-ros

https://github.com/gKouros/rsband_local_planner

https://github.com/gagolucasm/Lidar-Scan-ROS

https://github.com/gameyin/rrbot_gravity_compensation

https://github.com/gamma-programme/rospitch

https://github.com/gandhigauri/collision_avoidance

https://github.com/gandhigauri/robographers-teamg

https://github.com/ganhao89/husky_citrus_nav

https://github.com/ganterd/dissertation

https://github.com/ganweitech/beauty

https://github.com/gaochq/Gcs_mavros

https://github.com/gaowenliang/toy_code

https://github.com/garamizo/gaitest

https://github.com/garamizo/visser_power

https://github.com/gareth-cross/imu_covariance

https://github.com/gareth-cross/rviz_satellite

https://github.com/garethdicker/Firmware_Nov7

https://github.com/gary9555/ELEC6910P

https://github.com/garycheung123/myProject

https://github.com/garyservin/Proyecto-de-Sistemas-Inteligentes

https://github.com/gatsoulis/uu-planners

https://github.com/gatsoulis/uu-utils

https://github.com/gauravgardi/Quadrotor-circleeight

https://github.com/gavanderhoorn/ros_industrial_training

https://github.com/gcangussu/i3dmgx2

https://github.com/gcc-robotics/ros_arduino_gps

https://github.com/gcc-robotics/ros_arduino_imu

https://github.com/gcc-robotics/ros_lab_solution https://github.com/gcc-robotics/trash_detect

https://github.com/gcc-robotics/wall_z_software

https://github.com/gctronic/elisa3_node_cpp

https://github.com/gctronic/epuck_driver_cpp

https://github.com/gdrio/navibot

https://github.com/gds-emt/linorobot_4wd

https://github.com/gds-emt/linorobot_ackermann

https://github.com/gds-emt/linorobot

https://github.com/gds-emt/wheelchair_bump_sensors

https://github.com/gdsl/SOFTENG-306-PROJECT-1

https://github.com/geduino-foundation/geduino-ros

https://github.com/geetesh/RoboStatsEKF

https://github.com/gempio/MSci-Project

https://github.com/geoffviola/nmea

https://github.com/geoffviola/tf_to_imu

https://github.com/geoffviola/tf_to_odometry

https://github.com/gerikkub/CPRC-IGVC

https://github.com/gerkey/ros1_external_use

https://github.com/gestom/ICRAI_workshop

https://github.com/ggregory8/gg_mavlink

https://github.com/ggregory8/mavtools

https://github.com/ghanimmukhtar/collision_free_traj

https://github.com/gharghabi/PGITIC_sh

https://github.com/ghe/rosbaxter

https://github.com/ghirlekar/ros-localization

https://github.com/gibinjoezach/mrdp

https://github.com/giladh11/KOMODO_BYRG

https://github.com/gimait/RoVi2-PickHandProject

https://github.com/gimlids/BodyScanner

https://github.com/giorgiocorra/coverage_giorgio

https://github.com/giorgosera/HCR-project

https://github.com/giovanidebiagi/attitude_decision

https://github.com/girvaw/dev

https://github.com/girvaw2/Maggie-version-3

https://github.com/girvaw2/Maggie

https://github.com/gitdrrobot/drrobotV2_player

https://github.com/gitdrrobot/drrobot_jaguar4x4_player

https://github.com/gitdrrobot/jaguar4x4_2014

https://github.com/githubmsj1/BipedControl https://github.com/githubmsj1/bIped

https://github.com/giuseppelafortezza/TestTurtlebot

https://github.com/gizatt/xen_quad

https://github.com/gjain0/PathPlanning-CollisionAvoidance-and-SkeletalDetection-using-ROS

https://github.com/gjuhasz86/ros-rtm-gateway

https://github.com/gkahn13/pr2_eih_ros

https://github.com/glennliu/uavformation

https://github.com/gogojjh/maprecognition

https://github.com/golaced/px4fireware_in_vs2013

https://github.com/goncabrita/fsr_husky

https://github.com/goncabrita/hratc2014_field_trials

https://github.com/goncabrita/mine_mapping

https://github.com/goncabrita/rtk_gps

https://github.com/goneflash/vio_ros

https://github.com/gongwenbo/c270_camer

https://github.com/gonzalesMK/AMR_GonzalesMK_go2point

https://github.com/gonzalesMK/AMR_gonzales_Bugs

https://github.com/gonzalesMK/Autonomous_mapping_algorithm

https://github.com/gpldecha/kuka_planning_interface

https://github.com/gpldecha/netft_rdt_driver

https://github.com/gpldecha/peg_in_hole_project

https://github.com/gpldecha/peg_sensor

https://github.com/gpldecha/plug_sensor_models

https://github.com/gpldecha/record

https://github.com/gpldecha/turtlebot-navigation

https://github.com/graiola/pose_filter

https://github.com/graiola/wrench_filter

https://github.com/grandcat/robotics_g7

https://github.com/grassjelly/linorobot_mecanum

https://github.com/graziegrazie/my_turtlebot

https://github.com/greatforce/CAR

https://github.com/grimrpr/fu_tools

https://github.com/grimrpr/pathmessenger

https://github.com/gripsCAR/ati_netft_ros_driver

https://github.com/gripsCAR/force_torque_tools

https://github.com/gripsCAR/gazebo_ROS_pkgs

https://github.com/gripsCAR/mockup_optitrack

https://github.com/grisetti/g2o_frontend

https://github.com/groundmelon/djiros

https://github.com/groundmelon/gps_converter

https://github.com/group-8-robotics-of-destruction/s8_explorer

https://github.com/group-8-robotics-of-destruction/s8_mapper

https://github.com/group-8-robotics-of-destruction/s8_orientation

https://github.com/group-8-robotics-of-destruction/s8_pose

https://github.com/group-8-robotics-of-destruction/s8_turner

https://github.com/grvcTeam/grvcQuadrotor

https://github.com/grvcTeam/mbzirc

https://github.com/gryphonuav/deprecated_xsens_mtig

https://github.com/gsengupta2810/tactics

https://github.com/gstavrinos/jaguar_base

https://github.com/gt-ros-pkg/cpl

https://github.com/gt-ros-pkg/excel-util

https://github.com/gt-ros-pkg/hrl-assistive

https://github.com/gt-ros-pkg/hrl-pr2-behaviors

https://github.com/gt-ros-pkg/hrl-sensor-utils

https://github.com/gt-ros-pkg/hrl

https://github.com/gt-ros-pkg/humans

https://github.com/gt-ros-pkg/rim-bmw-experiments

https://github.com/gt-ros-pkg/siml

https://github.com/gtagency/buzzmobile-old

https://github.com/gtagency/buzzmobile

https://github.com/gtagency/roscorobot

https://github.com/guanlinchao/obsolete_sample_opencv-ros-webcam-capture

https://github.com/guilhermecano/Robot_Mapping_TG

https://github.com/guilhermelawless/pfuclt_omni_dataset

https://github.com/guilhermezaffari/dolphin

https://github.com/gurou/autobot

https://github.com/gus484/ros-tinkerforge_sensors

https://github.com/gustavovelascoh/vintersim

https://github.com/guyumao/SLAM_sonar

https://github.com/gyuhyeonkim/E.o.N-Drone119

https://github.com/h-kamada/jsk_2014_04_pr2_73b2

https://github.com/h-kamada/polarization

https://github.com/h2r/baxter_h2r_packages

https://github.com/h2r/ein

https://github.com/h2r/helpful_baxter

https://github.com/h3ct0r/AsctecROS

https://github.com/haiing/aiibot_bringup https://github.com/half-potato/depth-map

https://github.com/hamodp/ardrone_essentials

https://github.com/hanhu/PCL

https://github.com/hansonrobotics/HEAD

https://github.com/haplo31/Tcomp

https://github.com/haquang/Haquang-Research

https://github.com/hari-sikchi/sbplplanner

https://github.com/hariravi/HumanoidJengaTake2

https://github.com/harmishhk/hanp_humans_sim

https://github.com/harmishhk/hanp_local_planner

https://github.com/harmishhk/hanp_prediction

https://github.com/harmishhk/pr2_point_head

https://github.com/harryeakins/FREDbot

https://github.com/hasauino/rrt_exploration

https://github.com/hasbiestheim/smart_wheelchair

https://github.com/hausze1/academic_dump

https://github.com/hboortz/TurtlebotMapping

https://github.com/hbradlow/bulletsim

https://github.com/hcdth011/ROS-Hydro-SLAM

https://github.com/hchsiao/img_saver

https://github.com/hcrlab/convert_pcl

https://github.com/hcrlab/push_pull

https://github.com/hcrmines/IntBotAPC

https://github.com/hcrmines/apc2

https://github.com/hcrmines/apc

https://github.com/hcrskippi/laser_combiner

https://github.com/hdeeken/rviz_semap_plugin

https://github.com/hee502/PRGP_i90_talker

https://github.com/heiscsy/evolutus_ros_src

https://github.com/heivi/Pozyx-ROS

https://github.com/hejoseph/TTB2

https://github.com/hejoseph/TTB3

https://github.com/hejoseph/TTB

https://github.com/hengli/vmav-ros-pkg

https://github.com/henrypc6/rsn

https://github.com/henrywen2011/msf_imu_ekf

https://github.com/henrywen2011/rovio

https://github.com/heron/heron_robot

https://github.com/hershwg/point_cloud_frame_test https://github.com/heuristicus/DD2425

https://github.com/hexroytom/linemod_pose_est

https://github.com/hexroytom/newGitTest

https://github.com/heyBeaUtY/ROS_enabled_uav

https://github.com/hfarazi/RMG146

https://github.com/hhony/jetbot

https://github.com/hicopyht/plane_segment

https://github.com/himlen1990/demo

https://github.com/hippomormor/Cmis

https://github.com/hippomormor/frontier

https://github.com/hishamcg/DrdoProject

https://github.com/hiveground-ros-package/leptrino_force_torque

https://github.com/hjeldin/HILAS

https://github.com/hkaraoguz/deep_net_object_store

https://github.com/hkaraoguz/jaguarControlISL

https://github.com/hkaraoguz/semantic_data_store

https://github.com/hlkbyrm/taskHandlerISLH

https://github.com/hll4fork/Firmware-old

https://github.com/hmcRobotLab/hmc-robot-code

https://github.com/hmchung/opencv_test

https://github.com/hoangtungdinh/ros-drone

https://github.com/hobie88/tu.tino

https://github.com/hobie88/tutino.indigo

https://github.com/hoergems/ABT_CONTINUOUS

https://github.com/hoergems/abt_newt

https://github.com/hongalan/powerboard-arduino

https://github.com/hongalan/powerboard

https://github.com/hordurk/resized_image_transport

https://github.com/horgh/boldly

https://github.com/hoseongseo/bluefox

https://github.com/hoseongseo/data_logger

https://github.com/hpflatorre/diff_pi_bot

https://github.com/hrnr/robo-rescue

https://github.com/hrnr/ros-usarsim

https://github.com/hsioumin/three_nodes

https://github.com/hsmrs/mqp

https://github.com/hsnuhayato/rtm-ros-robotics

https://github.com/hualili/robotics-open_abb

https://github.com/huanglilong/mission

https://github.com/huangrui815/MonoSLAM_ARDrone

https://github.com/huliheng2011/ROS-final_project-A_QR

https://github.com/hungnguyen94/DPR4Base

https://github.com/hungnguyen94/GarbageRobot

https://github.com/hwopereis/aeroworks_quad_control

https://github.com/hx14384/textureless-detector

https://github.com/hxq2016jiayou/testgit

https://github.com/hzheng40/crazyflie_test

https://github.com/hzheng40/learning_tf

https://github.com/hzheng40/turtlebot_platform

https://github.com/hzobrist/TheResistance

https://github.com/i-v-s/gcs_agent

https://github.com/i-v-s/simco

https://github.com/iConor/learn-ros

https://github.com/iKrishneel/jsk_mbzirc_perception

https://github.com/iManbot/create_ctrl

https://github.com/ian-craig/jaco_hydro

https://github.com/ianyon/bachelors-project

https://github.com/iav-student/iav_depthimage_to_laserscan

https://github.com/ibaira/ardrone_gamepad

https://github.com/ibaranov-cp/UOR05

https://github.com/ibaranov-cp/epfl01

https://github.com/ibaranov-cp/kaust02_husky_customization

https://github.com/ibaranov-cp/ridgeback_ur10

https://github.com/ibaranov-cp/tor11

https://github.com/ichatzinikolaidis/Rula

https://github.com/icolwell/coffeebot

https://github.com/icyflame/ikat-auv-packages

https://github.com/idaohang/gps_goal_publisher

https://github.com/idaohang/gps_tf_publisher

https://github.com/idaohang/gps_umd-3

https://github.com/idmind-robotics/idmind_ros

https://github.com/ieatmosquitos/bearings_collector

https://github.com/ihmcrobotics/ihmc-open-robotics-software

https://github.com/ihrabar/zavrsni

https://github.com/iirob/ati_force_torque

https://github.com/iirob/iirob_filters

https://github.com/ijun-slee/honiden_robot

https://github.com/iliasam/OpenSimpleLIDAR https://github.com/ilsemae/sphero_ball

https://github.com/imesluh/ursula

https://github.com/imr-ctu/morbot

https://github.com/incebellipipo/kezbot

https://github.com/inesc-tec-robotics/carlos_motion

https://github.com/inesc-tec-robotics/guardian_ros_pkg

https://github.com/informatik-mannheim/Serviceroboter

https://github.com/ingeniarius-ltd/forte_rc_robot

https://github.com/ingerbha/ros_asv_system

https://github.com/ingersollc/matrix-gyroplane

https://github.com/inomuh/docs

https://github.com/inomuh/evapi_ros

https://github.com/inomuh/evarobot_simulator

https://github.com/intel-ros/realsense

https://github.com/intelligenceBGU/komodo

https://github.com/intelligenceBGU/robo-service

https://github.com/interimadd/seniorcar_src

https://github.com/interimadd/wheelcahir_ros

https://github.com/introlab/introlab-ros-pkg

https://github.com/introlab/rtabmap_ros

https://github.com/iocchi/PetriNetPlans

https://github.com/iolyp/ardrone_simulator_gazebo7

https://github.com/iolyp/tum-simulator-indigo

https://github.com/iory/Unreal-ROS-Plugin

https://github.com/iou16/3d_ogmapping

https://github.com/iou16/edogmapping

https://github.com/iou16/elevation_difference_cloud_pub

https://github.com/iou16/elevation_difference_map_localization

https://github.com/iou16/orne_waypoints_editor

https://github.com/ipa-josh/ratslam_ros

https://github.com/ipa-lth/schunk_gripper_egl90

https://github.com/ipa-lth/weiss_kms40

https://github.com/ipa-nhg/squirrel_robotino_backup

https://github.com/ipa-rmb-yp/scitos_drivers

https://github.com/ipa-svn/palcomIf

https://github.com/ipa320/accompany

https://github.com/ipa320/autopnp

https://github.com/ipa320/cob_bringup_sandbox

https://github.com/ipa320/cob_control_sandbox https://github.com/ipa320/cob_control

https://github.com/ipa320/cob_driver

https://github.com/ipa320/cob_navigation_tests

https://github.com/ipa320/ipa_canopen

https://github.com/ipa320/ipa_lcrob

https://github.com/ipa320/research-camp-5

https://github.com/ipa320/seneka_deployment_strategies

https://github.com/ipa320/seneka_deployment_unit

https://github.com/ipa320/seneka_sensor_node

https://github.com/ipa320/srs_public

https://github.com/ipab-rad/ICRIN

https://github.com/ipab-rad/Youbot-RVO

https://github.com/ipab-rad/rad_matrix

https://github.com/ipab-rad/rad_pr2_stack

https://github.com/ipab-rad/rad_youbot_stack

https://github.com/ipatient-zju/basic_algo

https://github.com/ipatient-zju/eth_ssf

https://github.com/ipatient-zju/rosbag_output

https://github.com/ipatient-zju/vo_flow

https://github.com/ipatient-zju/zmart_img_localization

https://github.com/ipeet/mte481

https://github.com/ipleiria-robotics/adv_robotics

https://github.com/ipleiria-robotics/nomad200

https://github.com/ipleiria-robotics/p3dx_apps

https://github.com/iralab/laser_merger

https://github.com/iralabdisco/ira_laser_tools

https://github.com/iralabdisco/ira_open_street_map

https://github.com/iranmansoori/lucia-winter-school-2016

https://github.com/irenesanznieto/ocular

https://github.com/ironjr/AI_DIONYSUS_4

https://github.com/ironjr/AI_DIONYSUS

https://github.com/irvs/avoidance-behavior

https://github.com/irvs/ros_tms

https://github.com/isabellezheng/Stagerobot

https://github.com/isacarnekvist/wall-a-maze

https://github.com/isadliu229/MotionC

https://github.com/ishiguroJSK/leptrino_reader

https://github.com/ishiguroJSK/vive_image_view

https://github.com/ishikawakouji/SOINN-JIT_for_ARDrone https://github.com/islamelnabarawy/ROS-Tutorial

https://github.com/islers/youbot_controllers

https://github.com/ismailr/myslam

https://github.com/isnow4ever/x50

https://github.com/issei-takahashi/PointMario

https://github.com/istinj/SLAM_srcs

https://github.com/istinj/tracker_hri

https://github.com/iti-luebeck/HANSE2012

https://github.com/iti-luebeck/gentl-tof-driver

https://github.com/ivan-vishniakou/amr_lab_test

https://github.com/ivan-vishniakou/sdp_ss_15

https://github.com/ivany4/Lunabotics_ROS

https://github.com/jackMilano/bb_to_world

https://github.com/jackhadfield/object-assembly-agent

https://github.com/jackros1022/pcl_tutorials_code

https://github.com/jacobperron/orb_slam_ros

https://github.com/jaehobang/drone_tragectory_display

https://github.com/jainajinkya/learning_joystick

https://github.com/jainak/particle_filter_SLAM

https://github.com/jake2184/Portfolio

https://github.com/jameshermus/bonusPoints

https://github.com/jamesr27/pixhawk

https://github.com/jamessteve/rwsfi2016_testeves

https://github.com/jarrettHoltz/DeltaCalibration

https://github.com/jarrettHoltz/SSLCalibration

https://github.com/jarvisschultz/objecttracker_nu

https://github.com/jarvisschultz/receding_planar_sys

https://github.com/jarvisschultz/robot_simulator

https://github.com/jas497/beta_coke_fetcher

https://github.com/jaubere/automatic-car

https://github.com/javsalgar/navigation_assistant_3dpc

https://github.com/jaychak/dronesim-

https://github.com/jayyoung/bham_seg_filter

https://github.com/jayyoung/initial_surface_view_evaluation

https://github.com/jayyoung/object_interestingness_estimator

https://github.com/jbohren-forks/pr2_doors

https://github.com/jbohren-forks/rtt_geometry

https://github.com/jcgarsan/thruster_control

https://github.com/jcpince/bobby

https://github.com/jcs04/ROS-SF

https://github.com/jdawkins/ardu_pilot

https://github.com/jdawkins/dist_control

https://github.com/jdawkins/emaxx_ros

https://github.com/jdawkins/koala-ros

https://github.com/jdawkins/ros_lcm_proxy

https://github.com/jdawkins/uav_control_ros

https://github.com/jdawkins/uav_planner

https://github.com/jdemp/agile_grasp

https://github.com/jdomin98/P1-B213-Programmin-Exam-Project

https://github.com/je310/microRobotsROS

https://github.com/jedirandy/mr-robot

https://github.com/jeguzzi/alma_robot

https://github.com/jeguzzi/ardrone_gestures_demo

https://github.com/jeguzzi/mavros_external_tracker

https://github.com/jemaloqiu/optim_planner_ros

https://github.com/jenniferdavid/cargo-ants-ros

https://github.com/jenniferdavid/hh-cargo-ants

https://github.com/jenniferdavid/path_planner

https://github.com/jenniferdavid/summer_school_jul14

https://github.com/jeskesen/gps_cartesian_grid

https://github.com/jeskesen/i2c_imu

https://github.com/jeweljames/pandabot

https://github.com/jfollestad/quadros

https://github.com/jfqiu/bankrobot

https://github.com/jfrascon/BIOMOT_EXOSKELETON_ROS

https://github.com/jfrascon/MANFRED_ROS_STACK

https://github.com/jgm88/ComputerVision2014

https://github.com/jgrogers/microstrain_mip_node

https://github.com/jgrossmann/pr2_tic_tac_toe_ws

https://github.com/jgstorms/sacar

https://github.com/jheddy/ROSGPSnode

https://github.com/jhfrost/uzliti_slam

https://github.com/jhielson/ROS_Projects

https://github.com/jhjune91/p3dx_velodyne_slam

https://github.com/jhu-asco/gazebo_rosmatlab_bridge

https://github.com/jhu-asco/gcop_ros_packages

https://github.com/jhu-dvrk/dvrk-ros

https://github.com/jhu-lcsr/lcsr_controllers https://github.com/jhu-lcsr/sp_segmenter

https://github.com/jhu-lcsr/vrep_ros_packages

https://github.com/jhu-saw/sawOptoforceSensor

https://github.com/jian-li/LineLidar

https://github.com/jiangdizhao/turtlebot_rst

https://github.com/jiawei-mo/bug

https://github.com/jiawei-mo/kinect_slam

https://github.com/jiawei-mo/macl

https://github.com/jim1949/dashgo_ws

https://github.com/jimcha21/robonaut_myo

https://github.com/jimfinnis/lightsensor_gazebo

https://github.com/jimfinnis/pioneernet

https://github.com/jimitgandhi66/Capstone_teamg_2015

https://github.com/jingego/ust_odroid_ws

https://github.com/jinpyojeon/TRIN_EARL

https://github.com/jitete/FYP_workspace

https://github.com/jitinratra/drone_project

https://github.com/jitrc/eklavya2015-ros-pkg

https://github.com/jiying61306/2016project

https://github.com/jizhang-cmu/receive_ublox

https://github.com/jizhang-cmu/receive_xsens

https://github.com/jkairborne/visual_servoing_ardrone

https://github.com/jkcooley/robotic_comprehension_of_viscocity

https://github.com/jkrs/EEC193

https://github.com/jlamyi/zumy_ros_nav

https://github.com/jlurz24/dog-walking-simulation

https://github.com/jlurz24/gazebo_utils

https://github.com/jlurz24/multi-object-position-tracker

https://github.com/jlurz24/pr2-litterbox-scooping

https://github.com/jmartinezot/Heimann-thermopile-ROS-driver

https://github.com/jmaye/dynamixel-ros

https://github.com/jmaye/mv_cameras_ros

https://github.com/jmaye/pure-pursuit-controller-ros

https://github.com/jmaye/velodyne_ros

https://github.com/jmaye/visensor-ros

https://github.com/jmcjacob/strands_movebase

https://github.com/jmessias/monarch_active_perception

https://github.com/jmirats/Show_trajectory

https://github.com/jmpechem/sm_n_vis

https://github.com/jmwest0774/mast_ros

https://github.com/joanpau/pose_twist_meskf

https://github.com/joao-avelino/bio_visual_suppression

https://github.com/joaoleao15/rwsfi2016_jleao

https://github.com/jodavaho/ros_helpers

https://github.com/joe-eklund/cs470

https://github.com/johncade/turtleSIM_NEAT

https://github.com/johnnyboss/localProjectToken

https://github.com/jokla/teamcv_ch3

https://github.com/jokla/visp_cam_vs

https://github.com/jonathan-w/RoboChair

https://github.com/jonathangingras/pcl-ppl-detector2

https://github.com/jonathankoss/EECS_376

https://github.com/jonclaassens/cri

https://github.com/jonfink/rviz

https://github.com/jonlwowski012/Cooperative_Mav_Ugv

https://github.com/jontromanab/sq_grasp

https://github.com/jordivilaseca/objects-tracker

https://github.com/jorgeazul/MobileRobot

https://github.com/jorgediosdado/simple_planner

https://github.com/jormunmor/doctorado

https://github.com/joschu/raven_vision

https://github.com/joseparnau/kview_prenodes

https://github.com/joseparnau/rosjac

https://github.com/joseparnau/workspace_transformation

https://github.com/josheddy/kalman_sense

https://github.com/joshvillbrandt/mecanumbot-ros-pkg

https://github.com/jotaemesousa/ros_gps_useful_tools

https://github.com/jpiat/scout_node

https://github.com/jpiramirez/ros_intro

https://github.com/jplhbird/asctec_mav_motion_planning

https://github.com/jplhbird/dd

https://github.com/jplhbird/oldcomputer

https://github.com/jplhbird/tlcquadrotor

https://github.com/jpmerc/perception3d

https://github.com/jpmerc/rosWS

https://github.com/jpreiss/ros_walkthrough

https://github.com/jprodriguezg/robot_cooperation_IRIS

https://github.com/jsagfr/ssmr_4wd

https://github.com/jsk-ros-pkg/jsk_common

https://github.com/jsk-ros-pkg/jsk_control

https://github.com/jsk-ros-pkg/jsk_demos

https://github.com/jsk-ros-pkg/jsk_recognition

https://github.com/jsk-ros-pkg/jsk_robot

https://github.com/jsk-ros-pkg/jsk_smart_apps

https://github.com/jsk-ros-pkg/jsk_visualization

https://github.com/jstnhuang/autocp

https://github.com/jstnhuang/rapid

https://github.com/jstnhuang/rviz_camera_publisher

https://github.com/jstnhuang/rviz_publish_shadow

https://github.com/jstnhuang/simplecp

https://github.com/jstraub/ptamRosBinaryFeatureRelocalization

https://github.com/jsully83/twist2roboteq

https://github.com/jtritell/adjustment_ui

https://github.com/jttte/lucylin0821-pr2-band

https://github.com/juauer/RO_1516

https://github.com/julianangel/EmotionBot

https://github.com/junhaoxiao/MultiSensorRatSLAM

https://github.com/junhongs/ROS_pidcalculator

https://github.com/junkilee/brown_perception

https://github.com/justinthomas/cylinder_perching_3D

https://github.com/justwillim/monitor

https://github.com/jvalinsky/jacktelop

https://github.com/jvcleave/PCL_1-3-0_Libraries_for_OF

https://github.com/jvcleave/example_PCL_1-4-0

https://github.com/jwillem/docker-ros

https://github.com/jxh576/tracked-robot

https://github.com/jysklS/jlros

https://github.com/k3174r0/ts_ros_packages

https://github.com/k4jt/ros_simple_ws

https://github.com/kadircet/roboturk

https://github.com/kajabo/trajectories

https://github.com/kalectro/pcl_groovy

https://github.com/kandithws/SeniorProject

https://github.com/kandithws/clothes_detection_egbis

https://github.com/kanster/ros_sandbox

https://github.com/kapatel/rhocode-explorer

https://github.com/kardee/DSTAR_FOR_ROS

https://github.com/kardee/Estar_with_ROS

https://github.com/kargm/morse_ros

https://github.com/karlzipser/kzpy3.0

https://github.com/kartikmohta/attitude_filter

https://github.com/kartikmohta/test-circle-ci-ros

https://github.com/kash1102/Supply-Chain-Management-System-with-Autonomous-Robots

https://github.com/katiewasnothere/model_3d_object

https://github.com/katycat5e/robotics_FP

https://github.com/kawo123/cs403-final-project

https://github.com/kckent34/cis700-vision

https://github.com/kdedakia/597-lab3

https://github.com/kdedakia/fydp

https://github.com/kdhansen/aseta_task_management

https://github.com/kdhansen/busroute

https://github.com/kdhansen/roburoc4

https://github.com/kejriwalnishant1990/bowpmap_ros-release

https://github.com/kejriwalnishant1990/bowpmap_ros

https://github.com/kekraft/contamination_stack

https://github.com/kellymorse/turtlebot_tf

https://github.com/kellysteich/remote-cavity-inspection_open

https://github.com/ken-kentan/SRC2015_Bteam_Manual

https://github.com/ken0618102000/SKHuang

https://github.com/ken0618102000/VVVO

https://github.com/ken0618102000/control_simulation

https://github.com/kendemu/followme

https://github.com/kendemu/out_navigation

https://github.com/kendemu/roomba_robot_meiji_executable_in_indigo

https://github.com/kendemu/tum_simulator_ros_ingido

https://github.com/kentegrate/alpha_autopilot

https://github.com/kentsai0319/techman_robot

https://github.com/keshinikeane/sidewalk-detector

https://github.com/kevin63857/jon_numl_catkin_ws

https://github.com/kevingkim/bimanual-catching-coman

https://github.com/kevinlichow90/robotic_projects

https://github.com/kevinlyu1/all_bayesian_src

https://github.com/kfu/metu-ros-pkg

https://github.com/kghite/FunRobo2016

https://github.com/kickhero/hello

https://github.com/kingasare/ROS-Final-Year-Project https://github.com/kintzhao/ar_localization

https://github.com/kintzhao/cv-slam_raw

https://github.com/kintzhao/cv-slam

https://github.com/kintzhao/cv_ekfslam

https://github.com/kintzhao/cvslam_gazebo

https://github.com/kintzhao/imu_serial_node

https://github.com/kintzhao/laser_slam_openSources

https://github.com/kintzhao/navigation_follow_road

https://github.com/kipage11/kerry_tools

https://github.com/kiran4399/ardrone_navmap

https://github.com/kirmani/hlpr_cadence

https://github.com/kirubeltadesse/SummerProject

https://github.com/klauskryhlmand/fuerte_workspace

https://github.com/klpanagi/rpi_hardware_interface

https://github.com/kmarkus/CompliantControllerComponent

https://github.com/kobei/hand_following

https://github.com/koellsch/youbot_mechatroniklabor

https://github.com/koenlek/topological_navigation_fuerte

https://github.com/kolasanichaitanya/alexaguidebot

https://github.com/konanrobot/agas-ros-pkg

https://github.com/konradb3/RCPRG-ros-pkg

https://github.com/kouhara3/kobuki_asmd

https://github.com/kralf/morsel-ros

https://github.com/kraudust/flight_control_project

https://github.com/krishal/beerOpener

https://github.com/krishnam94/human_tracking

https://github.com/kristofferlarsen/agilus_master_project

https://github.com/ksaiyo/RP_Lidar_ROS_SLAM

https://github.com/ksatyaki/cluster_extraction

https://github.com/ksatyaki/doro_manipulation

https://github.com/ksatyaki/ros_over_peis

https://github.com/kschwan/geomagic

https://github.com/kshitijdholakia/mobile_robot_ros

https://github.com/ksivakumar/duomlx_ros

https://github.com/kth-ros-pkg/baxter_dream_portfolio

https://github.com/kth-ros-pkg/contact_point_estimation

https://github.com/kth-ros-pkg/dumbo_apps

https://github.com/kth-ros-pkg/dumbo_driver

https://github.com/ktossell/camera_umd

https://github.com/ktossell/libuvc_ros

https://github.com/ku-ya/occ_grid_map

https://github.com/kuasha/stanley

https://github.com/kuikui1/UAV_pix

https://github.com/kuikui1/UAVdev

https://github.com/kuka-isir/ati_sensor

https://github.com/kuka-isir/cart_opt_ctrl

https://github.com/kuka-isir/depth_cam_extrinsics_calib

https://github.com/kuka-isir/kinects_human_tracking

https://github.com/kuka-isir/rtt_lwr_cart_ctrl

https://github.com/kuka-isir/rtt_lwr_controllers

https://github.com/kuka-isir/rtt_lwr

https://github.com/kuka-isir/static_laser_tools

https://github.com/kulbu/kulbabu_hardware

https://github.com/kunzel/object_search_utils

https://github.com/kuobenj/ECE-550-catkin_ws

https://github.com/kuri-kustar/aircraft_inspection

https://github.com/kuri-kustar/endoscopy_capsule-project

https://github.com/kuri-kustar/haptic_teleoperation

https://github.com/kuri-kustar/indoor_uav_control

https://github.com/kuri-kustar/kuri_mbzirc_offboard_control

https://github.com/kuri-kustar/kuri_ros_uav_commander

https://github.com/kuri-kustar/kuri_usar

https://github.com/kuri-kustar/laser_collision_detection

https://github.com/kuri-kustar/leap_control

https://github.com/kuri-kustar/multi_rotors_ftc

https://github.com/kuri-kustar/nbv_exploration

https://github.com/kuri-kustar/reemc_hri_interaction

https://github.com/kuri-kustar/ros-kuri-pkg

https://github.com/kvolle/Summer-Work-2014

https://github.com/kwokth22/ownTest

https://github.com/kyChuGit/TyphoonQ

https://github.com/kyChuGit/YuneecDroneCode

https://github.com/kylecorry31/Simple-ROS-Example

https://github.com/kyoto-u-shinobi/kamui_operator_station

https://github.com/kyoto-u-shinobi/kamui

https://github.com/l0g1x/DUO-Camera-ROS

https://github.com/l0g1x/SpringerROS_Gazebo2015

https://github.com/laas/gtp

https://github.com/laas/humanoid_walk

https://github.com/laas/move3d_ros_lib

https://github.com/laas/rviz_plugin_covariance

https://github.com/laas/toaster

https://github.com/laazizi/ros-robot

https://github.com/laboshinl/ndt_localizer

https://github.com/labust/labust-ros-pkg

https://github.com/labust/sensors-ros-pkg

https://github.com/labust/vehicles-ros-pkg

https://github.com/lagadic/demo_pioneer

https://github.com/lagadic/door_handle_detection

https://github.com/lagadic/ibvs_servo_head

https://github.com/lagadic/matlab_ros_bridge

https://github.com/lagadic/pepper_hand_pose

https://github.com/lagadic/vision_visp

https://github.com/lagadic/visp_blob_tracker

https://github.com/lagadic/visp_bridge-deprecated

https://github.com/lagadic/visp_ros

https://github.com/lagadic/vrep_ros_bridge

https://github.com/lagadic/vs_grasping_pepper

https://github.com/lakehanne/superchicko

https://github.com/lakid/random_nodes

https://github.com/lalten/tas

https://github.com/lama-imr/lama_costmap

https://github.com/lama-imr/lama_laser

https://github.com/lama-imr/lama_polygon_matcher

https://github.com/lama-imr/lama_utilities

https://github.com/landuber/wonderbot

https://github.com/lannersluc/ecto-SegFault

https://github.com/lannersluc/inverse_capability_map

https://github.com/lanyusea/4010F

https://github.com/lanyusea/dji_sdk

https://github.com/lanyusea/iarc2015

https://github.com/lanyusea/robomasters

https://github.com/laperss/wasp_search_and_rescue

https://github.com/laptopc700/kinect-turtlebot-reconstructor

https://github.com/lara-unb/aramis

https://github.com/lara-unb/porthos

https://github.com/larics/lpms_imu

https://github.com/larics/optoforce-ros-pusblisher

https://github.com/larsvens/WASP-AS-KTH-T3

https://github.com/larsys/socrob-ros-pkg

https://github.com/laughlinbarker/openrov_ros

https://github.com/lb5160482/Turtlebot-Autonomous-SLAM-and-Feature-Tracking-on-ROS

https://github.com/leeduoduo/pix_demo

https://github.com/leejang/grasping_modeling

https://github.com/lemmersj/utsa_intelligentrobotics

https://github.com/lengji22/ros_smartcar

https://github.com/lengmo714/flight-test

https://github.com/lennartclaassen/3rd-party-ressources

https://github.com/lennartclaassen/imu-processor

https://github.com/lennartclaassen/kinpro

https://github.com/lesire/mauve_sterela

https://github.com/leticiamaki/Ros_go2point

https://github.com/leticiamaki/Sonar3

https://github.com/lev2cu/IMU-

https://github.com/lev2cu/research

https://github.com/lfermin77/Exploration_Fraenkel

https://github.com/lfermin77/Exploration_Tremaux

https://github.com/lhc610github/PX4_RPI_TEST1

https://github.com/lia2790/object_tracking

https://github.com/liang-tang/pendulum

https://github.com/liangfok/controlit_configs

https://github.com/liangfok/controlit

https://github.com/liangkaiwen/baxter_project

https://github.com/libing64/att_ekf

https://github.com/libing64/pose_ekf

https://github.com/libing64/wireless_localization

https://github.com/liesrock/ft_publisher

https://github.com/lightjiang/vechicle_move_control

https://github.com/lihuang3/STONES-THROW

https://github.com/lileee/Robotics-RL

https://github.com/lilulab/crawler_core_jade

https://github.com/lilulab/opencv_template

https://github.com/limhyon/asctec_mav_framework_sandbox

https://github.com/liminglong/micros_mars_task_alloc

https://github.com/linhongbin/my_baxter

https://github.com/link-er/mobile_robots_lab_ss15 https://github.com/linux-creatures/auto_flight

https://github.com/linux-creatures/beagleus-ros

https://github.com/linux-creatures/lsd_ekf_slam

https://github.com/linuxhex/boost_serial_port

https://github.com/linuxhex/class_chassis

https://github.com/linuxhex/navigation_slam

https://github.com/linwendi/child_car

https://github.com/linwendi/four_wd_rover

https://github.com/liqiang1980/VTFS

https://github.com/lixianyu/Sinope

https://github.com/lixiao89/AFEPRC

https://github.com/lixiao89/plane_registration

https://github.com/lixinyutfd/conveyer_tracking

https://github.com/lixinyutfd/eecs_600_2016Fall

https://github.com/lixinyutfd/xinyu_tracking

https://github.com/lixinyutfd/zeta_final_project

https://github.com/lixinyutfd/zeta_project

https://github.com/liyiying/cloudrobot-semantic-map

https://github.com/liz-murphy/odometry_modifier

https://github.com/ljjyxz123/intel_ros_vm

https://github.com/ljmanso/prp

https://github.com/lkumar93/Deep_Learning_Crazyflie

https://github.com/lkumar93/Object-Tracker

https://github.com/lligen/myserial

https://github.com/lmc0709/filteringprocess

https://github.com/lmh-hml/UGV

https://github.com/lmh-hml/buggy

https://github.com/lmymapu/ROS_Hockey

https://github.com/lncd/Robots-OD

https://github.com/loganE/rvinci

https://github.com/loiannog/PTAMM_RGBD_cooperative

https://github.com/loiannog/rc_monitor

https://github.com/loiannog/state_control_vehicle

https://github.com/loicdubois/pdm_crazyflie

https://github.com/loncar-fer/formation_control

https://github.com/lorenzoriano/Monpal

https://github.com/loupdavid/crazyflie_project

https://github.com/lrocha3/mecatronica

https://github.com/lrse/floordetection

https://github.com/lrse/ros-utils

https://github.com/lrse/whycon

https://github.com/lsmigiel/anro

https://github.com/luandoan/mbzirc.old

https://github.com/luansilveira/auv_controller

https://github.com/luca-morreale/robotics-project

https://github.com/lucaghera/TestCode

https://github.com/lucanus43/WaypointNavHector

https://github.com/lucasw/simple_sim_ros

https://github.com/lucasw/vimjay

https://github.com/lucbettaieb/alg_robo_p1

https://github.com/lucbettaieb/alg_robo_p2

https://github.com/lucbettaieb/cutting_task

https://github.com/lucbettaieb/flexcraft

https://github.com/lucbettaieb/path_definer

https://github.com/lucci-spinitalia/progetto_sciami_ros

https://github.com/lucykidd/RSA

https://github.com/luis2r/learning_image_geometry

https://github.com/luisege/drrobot_jaguarV2_player

https://github.com/luisege/p2p

https://github.com/luispedraza/SPLmapping

https://github.com/lukaszmitka/joy_to_twist_msg

https://github.com/lukaszmitka/path_viz

https://github.com/lukaszmitka/roboclaw_driver

https://github.com/lukaszmitka/sensor_stick_driver

https://github.com/lukscasanova/joy2twist

https://github.com/lukscasanova/mtig_driver

https://github.com/lumeier/rosbag_remote

https://github.com/lvr-ros/mesh_navigation

https://github.com/lxd20/hololensROS

https://github.com/lxldavid91/XV-11-LIDAR-Data-Wireless-Transfer-Socket-

https://github.com/lxrobotics/lex

https://github.com/lxrobotics/rosintro

https://github.com/ly11tea/mechanum_robot

https://github.com/lycantropus/aqrabuamelu_ros

https://github.com/lycantropus/reactiveros

https://github.com/lzhzh/kaqi_driver

https://github.com/lzhzh/kaqi_gazebo_plugin

https://github.com/lzhzh/kaqi_teleop

https://github.com/m-shimizu/USARSimSampleClient

https://github.com/m-shimizu/p3at_for_ros_with_modelsdf

https://github.com/m0rph03nix/local_planner_student

https://github.com/maasLu/MA_AI_Y1_AS

https://github.com/maayanco/ROS-Room-Organizer

https://github.com/machinehead/dnepr

https://github.com/madratman/hybridplanner

https://github.com/madsbahrt/sdu-frobo-team3

https://github.com/madsciencetist/teamshield

https://github.com/maetulj/tf_tutorials

https://github.com/mafilipp/RPL_individual_2

https://github.com/magazino/pylon_camera

https://github.com/magiccjae/baxter_fistbump

https://github.com/magiccjae/gimbal_control

https://github.com/magrimm/RPL_group

https://github.com/magrimm/RPL_individual

https://github.com/maheer425/Haptics_Project

https://github.com/maheer425/Haptics_Robot_Interface

https://github.com/maheer425/baxter_ee_cart_imped

https://github.com/maheer425/catkin_ee_cart_imped

https://github.com/mahmoudabdulazim/DAQ

https://github.com/majvr93/Reactive-robot-

https://github.com/makcakoca/evarobot

https://github.com/makhaniadnan/RoNaoldo

https://github.com/malachico/lar-car-proj

https://github.com/malalves/AMR-2016

https://github.com/malalves/obstacle-avoidance

https://github.com/malalves/race

https://github.com/malasiot/clopema_certh_ros

https://github.com/mandad/moos-ivp-manda

https://github.com/mandalarobotics/m3dunit_ws

https://github.com/manhnhhdvn/RT_project

https://github.com/manhnhhdvn/wandrian_src

https://github.com/manishkjain21/Cpp_programs

https://github.com/manojngb/Crazyflie_optitrack_hover

https://github.com/manojngb/Multiple_Crazyflie_hover

https://github.com/manomitbal/A-Star_PathPlanner_ROS

https://github.com/manomitbal/Pololu_Localization

https://github.com/mantouRobot/share

https://github.com/manuelbonilla/desperate_housewife

https://github.com/manuelbonilla/dual_manipulator_control

https://github.com/manuelbonilla/force_sensor_ati

https://github.com/manuelbonilla/wg_planning

https://github.com/manujagobind/Learning-ROS

https://github.com/manujagrawal/controls-eklavya-4.0

https://github.com/manujagrawal/controls_on_beagle_bone

https://github.com/manujagrawal/mannu_local_eklavya

https://github.com/marbosjo/lemniscata_planner

https://github.com/marcelo-borghetti/sampleCode

https://github.com/marcinkaszynski/vrep-ros-plugins

https://github.com/marcmi9/ARDRONE_2.0

https://github.com/marcoarruda/dynamic_model

https://github.com/marcoarruda/ros_turtlesim_actlib

https://github.com/marcowplm/pinguiglio

https://github.com/marcozorzi/RoboticsProgrammingLaboratory

https://github.com/marcozorzi/Semester-Project

https://github.com/maria-miranda/rwsfi2016_mmiranda

https://github.com/marija185/coveragedemining

https://github.com/mario-gianni/cast

https://github.com/markch1991/SPRO3

https://github.com/markcsie/slam_project

https://github.com/marklendering/DemconRobot_robotControl

https://github.com/markrosoft/place_feature_recognition

https://github.com/mars-uoit/aar2

https://github.com/mars-uoit/aar

https://github.com/mars-uoit/ams

https://github.com/mars-uoit/omnimaxbot

https://github.com/mars-uoit/um7

https://github.com/maruthven/IRISS

https://github.com/mas-group/robocup-at-work

https://github.com/mas-group/youbot_simulation

https://github.com/masaeedu/me597

https://github.com/masamasa9841/Turtlebot_driving-along-the-wall

https://github.com/mast-demo/mast_pi_zero

https://github.com/mateus03/HANNRS

https://github.com/mathewpang/claptrap

https://github.com/matpalm/ros-hc-sr04-node

https://github.com/matpalm/ros-mpu6050-node https://github.com/mattamert/jr_hector_map_server

https://github.com/mattamert/odom_to_pose_covariance

https://github.com/mattamert/pose_to_odom

https://github.com/matthewia94/PIlot

https://github.com/matthewkirk203/ecen674trim

https://github.com/mattjrush/tankbot

https://github.com/mattncc1701/Assignment5

https://github.com/matwilso/rc_bot

https://github.com/mau25rojas/tesis-hexapodo

https://github.com/mauriliodc/kinectLaserCalibration

https://github.com/maverick-long/wheelchair_arm

https://github.com/maxbader/transitbuddy

https://github.com/maxfiedler/CVlab

https://github.com/maximest-pierre/sara_navigation_relaxed_astar

https://github.com/maxseidler/PAS

https://github.com/maxsvetlik/bwi_max

https://github.com/mbed92/laserScannerROS

https://github.com/mbriden88/COMPSCI403FinalProject

https://github.com/mcamarneiro/rwsfi2016_mcamarneiro

https://github.com/mcanter0/line_recog

https://github.com/mcast75/EARL

https://github.com/mclumd/drone_tracker_demo

https://github.com/mcoteCPR/jackal_gps_navigation

https://github.com/mcubelab/cpush

https://github.com/mcubelab/fompush

https://github.com/mdombro/CSC232

https://github.com/mdrwiega/depth_nav_tools

https://github.com/mdrwiega/tacbot

https://github.com/mdykshorn/mapBot

https://github.com/meccanoid-tx1/meccanoid-ros-packages

https://github.com/mechAndy/darc_research

https://github.com/meerasebastian/Evaluation-Project-ROS

https://github.com/meerasebastian/edge_leg_detector

https://github.com/meerasebastian/people_beginner

https://github.com/megaremi63/Betsy_setup

https://github.com/meghag/duplo_v0

https://github.com/meghag/duplo_v1

https://github.com/meghag/object_search

https://github.com/mehdish89/UR5_Cooperative_Transform https://github.com/mehdish89/legged_locomotion

https://github.com/meliaspl/clearpath-ros-pkg

https://github.com/merosss/VRepRosQuadSwarm

https://github.com/metallo25/ROS_vehicles

https://github.com/metropt/rwsfi2016_jxavier

https://github.com/mexomagno/pr2_placing

https://github.com/meysambasiri/RSBB_Complete

https://github.com/mfaessle/coax-control

https://github.com/mfcarmona/ros-isis

https://github.com/mfiore/scenarios

https://github.com/mfiore/situation_assessment

https://github.com/mfiore/supervision_system

https://github.com/mfueller/vrep_youbot_plugin

https://github.com/mgolpar/field-robotics

https://github.com/mgorski-zlatan/ROS_Map

https://github.com/miccol/Nao-Demo

https://github.com/michaellin/NeedleCalibration

https://github.com/michalek92/saas-ros

https://github.com/michalneoral/clopema_scroll_on_table

https://github.com/mickra/SegFault

https://github.com/mickra/segfault_vision

https://github.com/miguelriemoliveira/RustBot

https://github.com/miguelriemoliveira/amazon-inesc

https://github.com/miguelriemoliveira/rwsfi2016_moliveira

https://github.com/mik077/razor_imu_9dof

https://github.com/mikeferguson/robot_calibration

https://github.com/mikewrock/phdbackup

https://github.com/mikolak/irp6_sim_meng

https://github.com/milvusrobotics/mrp2_robot

https://github.com/minhnh/hbrs_courses

https://github.com/minolo/robostylus

https://github.com/mintar/imu_test

https://github.com/mircolosi/HRI_LAS

https://github.com/miriansyah/depthimage_to_laserscan

https://github.com/miriansyah/pointcloud_to_laserscan

https://github.com/mirwox/robotica16

https://github.com/mit-uav/mavros-mod

https://github.com/mitchellwills/roscpp_message_reflection

https://github.com/mizulily/daq_ft

https://github.com/mjaein/aladdin_working

https://github.com/mjaein/fri_modified

https://github.com/mjcarroll/zed_ros

https://github.com/mjg152/eecs_600_ariac_project

https://github.com/mkjaergaard/catkin_test

https://github.com/mkjaergaard/cppclient

https://github.com/mkjaergaard/geometry_experimental

https://github.com/mklingen/arm_slam_calib

https://github.com/mlab-upenn/AnytimeCPS2015

https://github.com/mlab-upenn/autobots

https://github.com/mllofriu/op2_control

https://github.com/mmsarode/ROS-Hexacopter

https://github.com/mnegretev/CRM_Material

https://github.com/mnegretev/MyDocs

https://github.com/mnegretev/RoboticsCourses

https://github.com/mohakbhardwaj/auto-park

https://github.com/mohamedalsherif/rcll_adapter

https://github.com/moicez/pentaxis

https://github.com/mongoose8/Imu_Filter

https://github.com/mongoose8/Imu_tool

https://github.com/moogle19/Robo

https://github.com/morgancormier/corobot

https://github.com/moriarty/mcr_arm_base_cartesian_control

https://github.com/morsag/across_optitrack

https://github.com/mortenege/Master-Thesis

https://github.com/mortonjt/Redblade

https://github.com/mosteo/turtlearm

https://github.com/mpatalberta/rostemperhumid

https://github.com/mpkuse/gps_process

https://github.com/mpkuse/rgbd_odometry

https://github.com/mpomarlan/moveit_puzzle_demo

https://github.com/mpp/random_navigation

https://github.com/mr11m036/mmrprojekt

https://github.com/mrath/tf_pose_rebroadcaster

https://github.com/mrclient/kuka_brazil

https://github.com/mrgransky/Autonomous-Vision-Based-Docking-Rob-work-3

https://github.com/mrinmoysarkar/ariac_competition

https://github.com/mrivi/master_thesis

https://github.com/mrkris13/quat-ekf

https://github.com/mrpt-ros-pkg/mrpt_navigation

https://github.com/mrpt-ros-pkg/mrpt_slam

https://github.com/mrpt-ros-pkg/pose_cov_ops

https://github.com/mrsd16teamd/loco_car

https://github.com/mrsolder/ROBO702_Project

https://github.com/mryellow/ros-catslam

https://github.com/mschild/rosmagnetmap

https://github.com/mschuurmans/ros_pioneer

https://github.com/mstuettgen/maskor_rover

https://github.com/msunardi/jeevesrobot

https://github.com/msy22/autosys_repo1

https://github.com/mthrok/roomba_500driver_meiji

https://github.com/mu-777/ros_blackship

https://github.com/mudrole1/BARC_Rockin

https://github.com/muhammedAlsayadi/cleaningrobot

https://github.com/muhrix/asctec_sdk3_framework

https://github.com/muhrix/robot_behaviours

https://github.com/muneebshahid/MAS-AMR-01

https://github.com/munjalbharti/PhotoGoalNavigation_2015

https://github.com/munjalbharti/VisionBasedNavigationAssignments_2015

https://github.com/mushroonhead/epochmini

https://github.com/mustafasezer/cloud_visualizer

https://github.com/mvgijssel/pas-2013

https://github.com/mwalecki/velma_head

https://github.com/mwegiere/mwegiere_ws

https://github.com/mwegiere/serwo

https://github.com/mwimble/ArduinoConroller

https://github.com/mwimble/Ianthe

https://github.com/mwimble/ewyn2

https://github.com/mwimble/jabin

https://github.com/mwimble/kaimiPi

https://github.com/mwswartwout/PS5

https://github.com/mycodeself/VAR

https://github.com/mylxiaoyi/odor_search

https://github.com/mylxiaoyi/ros_distro

https://github.com/mylxiaoyi/ros_python3

https://github.com/mylxiaoyi/ros_qt5

https://github.com/n800sau/roborep

https://github.com/nag92/Omnibot

https://github.com/nag92/Videoray-simulation-

https://github.com/nahueJ/steering_behaviors_controller

https://github.com/nao2853/move_alpha

https://github.com/naor9991/OnlineSCRAM

https://github.com/narayanaditya95/vn200

https://github.com/natashad/Max_Ros

https://github.com/nathangeorge1/beetlebot

https://github.com/native93/ptam

https://github.com/naustra/mon_package

https://github.com/navigator8972/DataWrapper

https://github.com/navigator8972/kinect_imu_tracking

https://github.com/navzam/tour_robot

https://github.com/nbanyk/toro_orchard_navigation

https://github.com/nbfigueroa/kinect-process-scene

https://github.com/nbfigueroa/kuka_interface_packages

https://github.com/nbfigueroa/task_motion_planning_cds

https://github.com/ncku-ros2-research/xenobot

https://github.com/ncos/mipt-airdrone

https://github.com/nduavlab/ros_discover

https://github.com/ne0h/hmmwv

https://github.com/neariot/fizifly

https://github.com/nearmrs/dynamic_active_constraints

https://github.com/neckutrek/hiqp

https://github.com/neemoh/orocos

https://github.com/neemoh/ros_catkin

https://github.com/nelsonn3c/monoc_slam_lsa

https://github.com/neobotix/neo_driver

https://github.com/neobotix/neo_locate_station

https://github.com/nestormh/frenetTransform

https://github.com/neu-capstone-quadcopter/monarc_tf

https://github.com/neu-capstone-quadcopter/monarc_uart_driver

https://github.com/newbie-zju/boundary_detect

https://github.com/newbie-zju/iarc_tf

https://github.com/newbie-zju/laser_detect

https://github.com/nextgeofront/pcl1

https://github.com/ngtienthanh/crops-moveit

https://github.com/nickarmstrongcrews/hl-ros-pkg

https://github.com/nickhuan/pulsedlight_driver

https://github.com/nickspeal/McGill-Robotics-AUV-Control-System https://github.com/nickur/Jaguar4x4

https://github.com/nickwalton/ros_ws

https://github.com/nicoladallago/control

https://github.com/nicolasrosa/ARteamROS

https://github.com/nikhil9/nayan_ros

https://github.com/nikhilkalige/robotis_manipulator

https://github.com/niki-s/UASS_Pioneer_Drone

https://github.com/nikolausmayer/cmake-templates

https://github.com/niliayu/pi_trees_cpp

https://github.com/niliayu/proj_2

https://github.com/niliayu/proj_3

https://github.com/niliayu/proj_7

https://github.com/niliayu/proj_8

https://github.com/niliayu/stdr_controller

https://github.com/nilsbore/cmd_amcl_viewer

https://github.com/nilsbore/rrt_planner

https://github.com/nimishkumar/lab2

https://github.com/niosus/depth_clustering

https://github.com/nirlevi5/robo-waitress

https://github.com/nischalkn/F1tenth

https://github.com/nischalkn/gpuSLAM

https://github.com/nisoT7/myROS

https://github.com/nitekrawler/robostats

https://github.com/nj1407/door_manipulation_demo

https://github.com/nj1407/elevator_press_button

https://github.com/nlyubova/orkobj_tomoveit

https://github.com/nlyubova/romeo_moveit_actions

https://github.com/nmathew87/ArdroneReporter

https://github.com/nmathew87/local_planner

https://github.com/nmathew87/spacejaunt

https://github.com/nmichael/vicon_mocap

https://github.com/nordic-robotics/nord_estimation

https://github.com/novalabs/madgwick

https://github.com/novalabs/workspace-tilty

https://github.com/npentrel/CoastLineExplorer

https://github.com/npochhi/Agv-Task

https://github.com/nqanh/my_cv_bridge

https://github.com/nqanh/pre_grasp_control

https://github.com/nrjl/volturnus

https://github.com/nshafii/inesc_robotis_arm

https://github.com/ntantisujjatham/tracking_realsense

https://github.com/ntbeyers/HuskySetup

https://github.com/ntnudavidcb/int_sys_proj

https://github.com/ntnudavidcb/project4

https://github.com/nttputus/proteus

https://github.com/ntua-cslep/cepheus_space_robot

https://github.com/ntua-cslep/space_robot

https://github.com/nubot-nudt/SICF

https://github.com/nucobot/lisa

https://github.com/nucobot/nucobot

https://github.com/nunobarcellos/Dronha

https://github.com/nunobarcellos/TeamROS

https://github.com/nuriozalp/heightController

https://github.com/nvoltex/AV

https://github.com/nvoltex/Aevehdet

https://github.com/nvoltex/arduino_node

https://github.com/nvoltex/target_tracker

https://github.com/nvtienanh/Hitechlab_humanoid

https://github.com/nwadams/snowbots

https://github.com/nwotero/ros_drivers

https://github.com/oKermorgant/ecn_sensorbased

https://github.com/oKermorgant/ecn_sonar

https://github.com/oaa3wf/Linux-data-processing

https://github.com/oddbotics/locomotion_module

https://github.com/oevsegneev/ros-dev

https://github.com/officineroboticHe/ros_robot_wandering_demo

https://github.com/ohm-ros-pkg/optris_drivers

https://github.com/okadahiroyuki/trcp-old

https://github.com/olgen2013/videoCoding

https://github.com/olinalpha/forebrain

https://github.com/olinalpha/midbrain

https://github.com/olivecroomes/BWI_Object_Avoidance

https://github.com/olivierdm/tum_ardrone

https://github.com/olmomoreno/ROS_collision_omega

https://github.com/olmomoreno/ROS_omega_telemanipulation

https://github.com/olmomoreno/ROS_using_markers_animation

https://github.com/olmomoreno/ROS_using_markers_collision

https://github.com/olpa/motion-capture

https://github.com/omercans/fp_vrep_old

https://github.com/omercans/fp_vrep

https://github.com/omercans/tas_omercan

https://github.com/omh1280/door_pcl

https://github.com/omh1280/tilt_laser

https://github.com/omichele/frame_extractor

https://github.com/onurtore/Turtlebot_Summer_Study

https://github.com/open-rdc/EKF_Localization_learning

https://github.com/open-rdc/cit_adis_imu

https://github.com/open-rdc/fulanghua_navigation

https://github.com/open-rdc/orne_navigation

https://github.com/open-rdc/orne_tools

https://github.com/open-rdc/pcl_explore_human

https://github.com/open-robot/Benewake-ROS-Package

https://github.com/open-robot/open-robot

https://github.com/openhumanoids/oh-distro

https://github.com/openrobotics/openrobotics_thunderbot

https://github.com/openspur/ypspur_ros

https://github.com/orbbec/ros_astra_camera

https://github.com/orchidproject/x264_image_transport

https://github.com/orhaneee/robotoperatingsystem

https://github.com/oriolorra/webcam_kinematics

https://github.com/orsalmon/bgumodo_arm

https://github.com/oscar-lima/little_bas

https://github.com/osrf/baxter_demos

https://github.com/osrf/rba_scripts

https://github.com/osrf/rosbag_direct_write

https://github.com/osu-uwrt/jaws-2

https://github.com/osu-uwrt/mirror-lake

https://github.com/osu-uwrt/riptide-ros

https://github.com/osudrl/rtt_common_msgs_zotac

https://github.com/otamachan/ros-indigo-gazebo7-ros-pkgs-deb

https://github.com/outlawchief/Mobile-Robots-ROS-

https://github.com/owenqyzhang/intersect_point

https://github.com/owenqyzhang/position_kalman_filter

https://github.com/owinklerhb/camera_topic_uwsim

https://github.com/oxcar103/Practicas-TSI

https://github.com/ozkuran/ITU

https://github.com/ozymandium/lse-xsens-mti https://github.com/p870668723/gmapping-ros

https://github.com/p942005405/rosss

https://github.com/pacocp/GII-UGR-TERCERO-2.CUATRIMESTRE

https://github.com/pal-robotics-graveyard/overlay

https://github.com/pal-robotics-graveyard/reem_controllers

https://github.com/pal-robotics-graveyard/reem_plugins

https://github.com/pal-robotics-graveyard/reemc_standalone

https://github.com/pal-robotics/pal_gazebo_plugins

https://github.com/pal-robotics/pal_navigation

https://github.com/pal-robotics/perception_blort

https://github.com/pal-robotics/reem_tutorials

https://github.com/pal-robotics/reemc_tutorials

https://github.com/pal-robotics/tiago_tutorials

https://github.com/palmieri/filter_laser_scan_max_range

https://github.com/pammirato/robot_control

https://github.com/pangfumin/laser_assembler

https://github.com/pangfumin/lsd-slam-catkin

https://github.com/pangfumin/rcars

https://github.com/pangfumin/sensor_fusion

https://github.com/panjekm/tradr-ugv-base

https://github.com/panx/my_lidar

https://github.com/paolostegagno/uri-soft

https://github.com/parcon/arl_ROA

https://github.com/parcon/arl_ardrone

https://github.com/parcon/arl_bag2csv

https://github.com/pardi/hexacopter_class

https://github.com/pardi/ids_viewer

https://github.com/pardi/mark_follower

https://github.com/parthamishra1996/new_agv_simulation

https://github.com/pascualy/umlaut-wayfinding-robot

https://github.com/patilnabhi/nuric_wheelchair_model_02

https://github.com/patriciammencia/RAIDER

https://github.com/paulachristiny/carrot_chase

https://github.com/paulachristiny/object_avoidance

https://github.com/paulachristiny/quaternoin

https://github.com/paulbovbel/frontier_exploration

https://github.com/paulbovbel/nav2_platform

https://github.com/paulruvolo/CompRobo15_testing

https://github.com/paulruvolo/comprobo2014 https://github.com/paulyang1990/my_VINS

https://github.com/pauvsi/pauvsi_vio

https://github.com/pbouffard/ccny-ros-pkg-pbouffard

https://github.com/pcchenxi/Pluto_stuff

https://github.com/pcchenxi/RCV_velodyne_classification

https://github.com/pcchenxi/cloud_img_mapper

https://github.com/pcchenxi/fused_terrain_classifier

https://github.com/pcchenxi/geometric_terrain_classifier

https://github.com/pcchenxi/mrs_slam_kthversion

https://github.com/pcchenxi/pluto_calibration

https://github.com/pcchenxi/terrain_classifier

https://github.com/pcchenxi/terrain_feature_fuser

https://github.com/pcchenxi/terrain

https://github.com/pciavald/openimmersion

https://github.com/pdrews/ros_kf_control

https://github.com/peci1/ros_packages

https://github.com/pedro-abreu/quadcopter

https://github.com/pedropgusmao/posewithcovstamped2tf

https://github.com/pedrorangell/drone-load-transportation

https://github.com/peetahzee/telepresence

https://github.com/peipei2015/3d_model

https://github.com/pengatseu/robocup

https://github.com/penghou620/turtlebot_simulator_rviz

https://github.com/pengtang/pars_ros

https://github.com/pepon1/saastutorial

https://github.com/percipioxyz/camport_ros

https://github.com/perezsolerj/ExperimentMeasurer

https://github.com/perezsolerj/benchmark_software_launcher

https://github.com/perezsolerj/pipefollowing

https://github.com/perezsolerj/uwsimbenchmarks

https://github.com/perimosocordiae/manifold_mapping

https://github.com/personalrobotics/or_owd_controller

https://github.com/personalrobotics/or_rviz

https://github.com/personalrobotics/owd

https://github.com/pet1330/circleIDTracker

https://github.com/peteflorence/memory_visualizer

https://github.com/peteflorence/motion-primitives

https://github.com/peteflorence/quadrotor_polynomial_planning

https://github.com/peterkty/pr2-roslcm-bridge https://github.com/peterkty/pr2_force_torque_tools

https://github.com/peterweissig/ros_octomap

https://github.com/peterweissig/ros_pcdfilter

https://github.com/pgigioli/darknet_ros_custom

https://github.com/pgigioli/depth2cam

https://github.com/pgigioli/face_tracker

https://github.com/pgigioli/turtlebot_demos

https://github.com/philko4711/ohm_gazebo

https://github.com/phmaho/ROS_packages_pHossbach

https://github.com/photoneo/phoxi_camera

https://github.com/phuicy/ssc_ardrone_simulator

https://github.com/piappl/ros-jaus-bridge

https://github.com/pico737/robomasters_ros

https://github.com/pierg/wasp_cht2_catkin

https://github.com/pierrePch/projet-blue

https://github.com/pierrelux/aquaviz

https://github.com/pierriko/morse-ros

https://github.com/pigheadx/grouper_lidar

https://github.com/pilif-pilif/magnetometer

https://github.com/piliwilliam0306/RS485

https://github.com/piliwilliam0306/andbot0

https://github.com/piliwilliam0306/andbot_base

https://github.com/piliwilliam0306/andbot_gazebo

https://github.com/piliwilliam0306/andbot_pkg

https://github.com/piliwilliam0306/learning_nav

https://github.com/piliwilliam0306/mbed-hello

https://github.com/piliwilliam0306/mega_base_ultrasonic

https://github.com/piliwilliam0306/metal1

https://github.com/piliwilliam0306/willdroid

https://github.com/pinkedge/robot_manipulation

https://github.com/pires9/node-master

https://github.com/pires9/race_point

https://github.com/pixhawk/pixhawk-kinect-pkg

https://github.com/pjpeng/thinair

https://github.com/pkaveti784/AeroTracker

https://github.com/pkok/pololu_imu

https://github.com/plancherb1/6.141-All-Code

https://github.com/plepej/iri_3d_navigation

https://github.com/pleschinski/SSUAV-ADC-Monitor https://github.com/plusangel/radiationUAV

https://github.com/plynn17/Aruco_move_ros_pkg

https://github.com/po1/lasermux

https://github.com/poinu/RPL

https://github.com/pomerlef/ptu_laser_assembler

https://github.com/potaopereira/sml_code

https://github.com/ppp2006/runbot_number0

https://github.com/prabhu-dev/LineDetectorLengthFix

https://github.com/pragmaticTNT/autolab

https://github.com/praij90/steered_wheel_base_controller

https://github.com/prajankya/AgriMapper

https://github.com/prajankya/Lidar-Robot

https://github.com/praman2s/aiciss_logger

https://github.com/praman2s/youbot_wrench_controller

https://github.com/prarobo/cooprov

https://github.com/prasadvagdargi/optoforce_ros

https://github.com/prashanta-gyawali/pr2_climb_ladder

https://github.com/prashanthr05/pan-tilt-tracker

https://github.com/praveenv4k/ros-vrep

https://github.com/pravinas/aircraft_cabling_cv

https://github.com/pregier/peter_simulation

https://github.com/prguimaraes/cursoRos

https://github.com/procopiostein/leader

https://github.com/procopiostein/ompl_planner_base

https://github.com/progtologist/fake_gravity_ft

https://github.com/prone2drift/COMP3431-Assignment1

https://github.com/pronobis/rocs-ros

https://github.com/proxemics2014/ross_gcer_study

https://github.com/pryre/breadcrumb

https://github.com/pryre/mavel

https://github.com/pryre/transform_rebroadcaster

https://github.com/pscho/phantomx_arm

https://github.com/pses-mephobia/mephobia-sensors

https://github.com/psh117/thormang_mobile

https://github.com/psodhi/multiagent_boundary_mapping

https://github.com/psoetens/pcl-svn

https://github.com/pt07/asterx1_node

https://github.com/ptrb/ipa_loc_feature_slam

https://github.com/pulver22/ardrone_autonomous_flight https://github.com/pulver22/uga_tum_ardrone

https://github.com/purcola/density_filter

https://github.com/purplenavi/FSR13Group2

https://github.com/purukaushik/puru_ros_basics

https://github.com/pusnik/robot-human-follower

https://github.com/pvanagtmaal/PluC

https://github.com/pvarin/Olin_AR_Drone

https://github.com/pwec/husar_sandbox

https://github.com/pxlong/interactive_segmentation_textured_groovy

https://github.com/pxlong/richard_sandbox

https://github.com/pynpyn/Robotic-Planning

https://github.com/qfyhaha/fly_shape

https://github.com/qianyizh/StanfordPCL

https://github.com/qintony/odom_translation

https://github.com/qintony/tag_detector

https://github.com/qjones81/johnny5

https://github.com/qjones81/victor

https://github.com/qqfly/arc

https://github.com/qqtk/cra_robot_localization

https://github.com/qqtk/key_teleop

https://github.com/qqtk/xdrive_odom

https://github.com/qqtk/xnet_gyro_madgwick

https://github.com/qqtk/xnet_imu_rtqfusion

https://github.com/quadrotor-IITKgp/ark_stateestimation

https://github.com/quattroteama/Odometry-Publisher

https://github.com/querquer/ardrone.swp

https://github.com/r3n33/lidar-lite-ros-scanner-driver

https://github.com/r5cop/jaguar_bringup

https://github.com/raadal/RUDA

https://github.com/raaslab/Exploration

https://github.com/raaslab/publish_stampedtransform

https://github.com/rab-bit/obsavoid

https://github.com/radhen/learning_moveit_jaco

https://github.com/radioactive1014/svn

https://github.com/rafaelrgb/ros_oop_example

https://github.com/rafaelrgb/sensor_pid

https://github.com/rafaelrgb/trabalho_final

https://github.com/rafafigueroa/cws

https://github.com/rafonki/Tidying-Robot

https://github.com/raggyftw/B216-Rob1-miniproject
https://github.com/raggyftw/guidebot
https://github.com/raharjaliu/TheCodingGame
https://github.com/rahul003/catkin_ros
https://github.com/rahul30694/samudraAUV
https://github.com/ramshaw888/COMP3431-ass2
https://github.com/randomed/CRF
https://github.com/ranjanashish/air_lab_iitm
https://github.com/raouiyounes/ph
https://github.com/raouldc/softeng306-Rosville
https://github.com/rapp-project/rapp-platform
https://github.com/rapp-project/rapp-robot-nao
https://github.com/raptort3000/ublox_catkin
https://github.com/ras-sight/hratc2017_entry_template
https://github.com/ras-sight/hratc2017_framework
https://github.com/ras14-group2/point_follow_controller
https://github.com/ras14-group2/primesense_pkgs
https://github.com/ras14-group2/robo_ctrl
https://github.com/ras14-group2/robo_imu
https://github.com/ratmcu/doa
https://github.com/raucha/chairbot
https://github.com/raucha/raucha_utils
https://github.com/raultron/ardrone_velocity_ekf
https://github.com/raultron/robotino-ros-pkg
https://github.com/rayontheon/TXAP
https://github.com/razinanu/ottocar
https://github.com/rbaldwin7/rhocode
https://github.com/rbart/guide-dog
https://github.com/rbautista11/PathFinder
https://github.com/rbonghi/discrete_controller
https://github.com/rbtying/Companion-Cube
https://github.com/rcxking/rpi_robotics_work
https://github.com/rdelfin/cyberphysical-robot-car
https://github.com/rdelfin/robotics_spring_ros_kinect
https://github.com/rdeliallisi/wall-follower
https://github.com/rdrive/isr_m2_driver
https://github.com/red-itmo/box_finder
https://github.com/red-itmo/navigation_step
https://github.com/redbaron148/kipr_open_09

https://github.com/redbaron148/siue_coax_dev

https://github.com/redbaron148/this_dev

https://github.com/redfulvio/phasespace_description

https://github.com/redheli/ethercat

https://github.com/reem-education/stacks

https://github.com/renatogg/gary

https://github.com/rezeck/Xperia-ROS

https://github.com/rflmota/Magabot-ROS-Package

https://github.com/rhogroup/simple

https://github.com/rhololkeolke/EECS-376-Alpha

https://github.com/rhololkeolke/eecs_600_robot_project1

https://github.com/rhopfer/rtairos

https://github.com/rhuincalef/TKinect-2016

https://github.com/rhuitl/slam_gmapping

https://github.com/rhuitl/slam_karto

https://github.com/ricardojmf/ChemestryROBO

https://github.com/ricfehr3/blue2drone

https://github.com/rickeywang/me780-project

https://github.com/ridgeback/ridgeback_robot

https://github.com/riemervdzee/Flarb

https://github.com/rileyBloomfield/lunatronCatkin

https://github.com/riptideas/uUUV

https://github.com/riscmaster/risc_maap

https://github.com/rishabhagarwal880/RoboMuse-4.0

https://github.com/ritwik1993/youbot_499

https://github.com/riverstrom/grull_verdino_simulator_gazebo

https://github.com/riverstrom/move_base_3D

https://github.com/rizar/rossvs

https://github.com/rkobbo15/miniproject

https://github.com/rkoyama1623/ros_sample_pkgs

https://github.com/rkyuan/BABSLAM

https://github.com/rkyuan/ROS-class

https://github.com/rlalleme/prolog-interface

https://github.com/rlklaser/lrm_carina

https://github.com/rll/berkeley_demos

https://github.com/rll/graveyard

https://github.com/rll/raven_2

https://github.com/rm32/Diss

https://github.com/rmellema/PAS-Sokkers

https://github.com/rmihalyi/bumblebee

https://github.com/rmolin88/robotBrain

https://github.com/roadnarrows-robotics/hekateros

https://github.com/roadnarrows-robotics/jenny

https://github.com/roadnarrows-robotics/kuon

https://github.com/roadnarrows-robotics/laelaps

https://github.com/roadnarrows-robotics/pan_tilt

https://github.com/rob5a/2015_initial_workspace

https://github.com/robertjacobs/ros-test

https://github.com/robertsatya/ros-bot

https://github.com/robinJKU/hackathon2013EntryAssignment

https://github.com/robinJKU/mobrob_robin

https://github.com/robinJKU/omnirob_robin

https://github.com/robinJKU/ros_common_robin

https://github.com/robinJKU/tools_robin

https://github.com/robinJKU/wheeled_robin_simulator

https://github.com/robinzhoucmu/MLab_EXP

https://github.com/roboFriend1977/ra_hri

https://github.com/robocomp/robocomp-ursus-rockin

https://github.com/robofisshy/follow_ros

https://github.com/robofisshy/follow_trunk

https://github.com/robofisshy/turtle_control

https://github.com/robofit/ar-table-itable

https://github.com/robofit/ar-table-pr2

https://github.com/robofit/but_robot_demos

https://github.com/robofit/but_sensor_fusion

https://github.com/robosavvy/vive_ros

https://github.com/roboskel/RoboCoffee

https://github.com/roboskel/SekApps

https://github.com/robotambassador/robot-ambassadors

https://github.com/robotica-ifce/robot_arduino_driver

https://github.com/robotican/lizi_robot

https://github.com/robotican/robotican

https://github.com/robotics-at-maryland/qubo

https://github.com/robotics-in-concert/rocon_ensemble

https://github.com/robotics-in-concert/rocon_motion_pipeline

https://github.com/robotics-upo/arduimu_v3

https://github.com/robotics-upo/raposa_driver

https://github.com/robotics-upo/siar_packages https://github.com/robotics-upo/teresa-wsbs

https://github.com/robotics-upo/upo_robot_navigation

https://github.com/roboticslab-fr/rplidar-turtlebot2

https://github.com/roboticslab-uc3m/teo_robot

https://github.com/robotlinker/radoe

https://github.com/robotlinker/robotlinker_core

https://github.com/robotology-playground/idynutils

https://github.com/robotology-playground/kvh-yarp-devices

https://github.com/robotology/OpenSoT

https://github.com/robotology/wysiwyd

https://github.com/robotology/yarp

https://github.com/robotpilot/myahrs_driver

https://github.com/robz/igvc2013_backup

https://github.com/rockin-robot-challenge/at_home_rsbb_comm_ros

https://github.com/rockin-robot-challenge/rockinYouBot

https://github.com/rodschulz/pr2_grasping_OLD

https://github.com/rodschulz/pr2_grasping

https://github.com/rohanbhargava11/spiri_navigation-1

https://github.com/rohitrajskk/test_vision

https://github.com/rolling-robot/adafruit_imu

https://github.com/romainreignier/rtimulib_ros

https://github.com/romanganchin/TowerDefense

https://github.com/romulortr/motion_control

https://github.com/ron1818/Singaboat_RobotX2016

https://github.com/rondell/Cognitive-Robotics-Project

https://github.com/rorromr/bender_test

https://github.com/ros-controls/ros_controllers

https://github.com/ros-drivers/omron

https://github.com/ros-drivers/pointgrey_camera_driver

https://github.com/ros-drivers/rosserial-experimental

https://github.com/ros-drivers/um6

https://github.com/ros-gbp/pcl-fuerte-release_defunct

https://github.com/ros-geographic-info/geographic_info

https://github.com/ros-industrial-consortium/fermi

https://github.com/ros-industrial-consortium/reuleaux

https://github.com/ros-industrial/industrial_training

https://github.com/ros-industrial/robot_movement_interface

https://github.com/ros-industrial/robotiq

https://github.com/ros-interactive-manipulation/pr2_object_manipulation https://github.com/ros-naoqi/naoqi_driver

https://github.com/ros-perception/graft

https://github.com/ros-perception/image_common

https://github.com/ros-perception/imu_pipeline

https://github.com/ros-perception/laser_filters

https://github.com/ros-perception/laser_proc

https://github.com/ros-perception/pcl-fuerte

https://github.com/ros-perception/vision_opencv

https://github.com/ros-pkg-git/kinect

https://github.com/ros-planning/map_store

https://github.com/ros-planning/moveit_core

https://github.com/ros-planning/moveit_tutorials

https://github.com/ros-planning/navigation_experimental

https://github.com/ros-planning/navigation_tutorials

https://github.com/ros-teleop/twist_mux

https://github.com/ros-visualization/interactive_marker_twist_server

https://github.com/ros-visualization/rqt_common_plugins

https://github.com/ros/common_msgs

https://github.com/ros/common_tutorials

https://github.com/ros/geometry2

https://github.com/ros/geometry_tutorials

https://github.com/ros/geometry

https://github.com/ros2/demos

https://github.com/ros2/turtlebot2_demo

https://github.com/rosindex-attic/code-tum-git-mapping

https://github.com/rosskidson/ScaViSLAM_ros

https://github.com/rosskidson/ros_tools

https://github.com/rosskidson/visual_odometry

https://github.com/roussePaul/SML

https://github.com/rowoflo/grits_ros

https://github.com/roxcarpio/dynamic_fiducial

https://github.com/rpng/img_imu_record

https://github.com/rqou/prlite-pc

https://github.com/rrg-polito/mono-slam

https://github.com/rrg-polito/rrg-polito-ros-pkg

https://github.com/rs1990/darc_crazyflie

https://github.com/rsait/rsait_public_packages

https://github.com/rsbGroup1/frobo_rsd

https://github.com/rsbGroup1/rc_rsd

https://github.com/rsd15gr3/rsd3_ROS_stack

https://github.com/rshnn/baxter_planning

https://github.com/rst-tu-dortmund/pxpincher_ros

https://github.com/rst-tu-dortmund/teb_local_planner

https://github.com/rt-net/rt_usb_9axisimu_driver

https://github.com/rty2357/gnd_laserscan_plot

https://github.com/rty2357/gnd_ls_coordtf

https://github.com/rudasi/ethercat_hardware

https://github.com/ruipimentelfigueiredo/quadrotor

https://github.com/ruipiresc/arena

https://github.com/rukiasan15/Rescate-de-mu-eca

https://github.com/rushipatel/tube_polishing

https://github.com/ruthvik92/OpenCV-ROS-Hydro-

https://github.com/rwatso12/novatelROS

https://github.com/ryandavid/avc-corvette

https://github.com/ryanluna/moveit_r2

https://github.com/ryanluna/r2_planning

https://github.com/rych-uch/PR2Misc

https://github.com/rynderman/tactile_sensor

https://github.com/ryotasuzuki0919/practice

https://github.com/s-j-b/perceptual_ice_model

https://github.com/s894330/kr7150_robot

https://github.com/safari-k/dog_picker

https://github.com/safrimus/capstone-project

https://github.com/sahandy/crops_vision_toolbox

https://github.com/sahiljuneja/Image-Aquisition-ROS

https://github.com/sailuo/XV11-LIDAR

https://github.com/saiprasannasp/octagon

https://github.com/sakthiram/CSE190_CompRobotics

https://github.com/sambishop/aav-ros-car

https://github.com/sammarden/ros-hydro

https://github.com/samoooop/PID-Controller

https://github.com/sandeepmanandhargithub/ROS-Turtlebot2

https://github.com/santhan-k/SOFTENG_306_Group3_Project_1

https://github.com/santosj/SpatioTemporalExploration

https://github.com/saszaz/alvar_ptgrey

https://github.com/saszaz/boeing-project

https://github.com/sathya1995/dynamics_ws

https://github.com/satyeshmundra/eklavya_localization https://github.com/saucompeng/rosIndigo

https://github.com/sauronalexander/heel_toe_planner

https://github.com/savik28/my_ros_mujoco_tests

https://github.com/sbpl/exploration_map

https://github.com/sbpl/melvin

https://github.com/sbpl/sbpl_geometry_utils

https://github.com/sbpl/uavros

https://github.com/sbragagnolo/xsens

https://github.com/sbrodeur/ros-icreate-bbb

https://github.com/sc071139/ROS_Mecanum_Node

https://github.com/schdomin/cds

https://github.com/schdomin/vi_mapper

https://github.com/scheideman/rpi_robot

https://github.com/schizzz8/scan_parser

https://github.com/schultza/hokuyo_test

https://github.com/scockrell/kinect2

https://github.com/scockrell/kinect3

https://github.com/scottmishra/creative_interactive_gesture_camera

https://github.com/sdas-cecsc/CollaborativeLocalization_3D_UAV

https://github.com/sdipendra/ros-projects

https://github.com/sdsmt-robotics/akcermann-planner

https://github.com/sdsuvei/navigation_test

https://github.com/sebasgm85/new_slam_karto

https://github.com/sefyas/tangible

https://github.com/segwayrmp/segway_rmp

https://github.com/sehomi/AUTMAV_GS

https://github.com/sendtooscar/ariaClientDriver

https://github.com/sentervip/rvio

https://github.com/sevenbitbyte/ros_gps_hacks

https://github.com/sevenbitbyte/waas

https://github.com/severin-lemaignan/dorothy

https://github.com/severin-lemaignan/ros-qml-plugin

https://github.com/severin-lemaignan/zoo-map-maker

https://github.com/sfchik/subscriber_basics

https://github.com/sfotiadis/hdetect

https://github.com/sgXmachina/musicBots

https://github.com/sgabello1/Blob-detector

https://github.com/sgabello1/Look3DROS

https://github.com/sgabello1/SAR

https://github.com/sgang007/codeSnippets

https://github.com/shadow-robot/sr-taco

https://github.com/shadow-robot/sr_core

https://github.com/shadowkun/rviz4ogre2

https://github.com/shahsk/stochastic-receding-horizon-control

https://github.com/shangl/iros2016

https://github.com/shangl/uibk_moveit_tools

https://github.com/shanjit/ros_turtlebot

https://github.com/shanwu12/DJIsummerCamp

https://github.com/shanwu12/my_design

https://github.com/shaykalyan/SE306-ROS

https://github.com/shehzi001/amr-ss

https://github.com/shehzi001/amr-stage-clone-test

https://github.com/shehzi001/binary_bitbots

https://github.com/shehzi001/ros_foundation_course

https://github.com/shengwen1997/mapviz_mission_planner

https://github.com/shihyuan/samples

https://github.com/shinyTyrannosaurus/mindQuad

https://github.com/shivamvats/costmap_2d

https://github.com/shivareddy37/Baxter_playing_pool

https://github.com/shobhit6993/egraphs-with-dmp

https://github.com/shobhit6993/kinect_demonstrate

https://github.com/shotahirama/my_ros_practice

https://github.com/shpower/HuskyLwa

https://github.com/shsiung/16720Proj

https://github.com/shyamalschandra/drcsim

https://github.com/siddhu95/fused_leg_detection

https://github.com/sigproc/robotic_surgery

https://github.com/sikang/laser_image_rect

https://github.com/sikang/laser_pose_estimator

https://github.com/sikang/nanoplus

https://github.com/silgon/pioneer3at

https://github.com/silverbullet1/bbauv

https://github.com/silvercup011/team2aDanny

https://github.com/simharry3/duckietown

https://github.com/simonleonard/gazebo_soft

https://github.com/simonwang2015/eecs_376_s16

https://github.com/simubhangu/marker_pose_detection

https://github.com/simubhangu/pal_vision_segmentation https://github.com/sina-cb/nested_amcl

https://github.com/sipimars/sipimars_ros

https://github.com/sivteck/kai

https://github.com/sjager/Fanboat

https://github.com/sjriek/MinorAR_2016

https://github.com/sjwaller/ros-qp4

https://github.com/skasperski/drivers

https://github.com/skasperski/navigation_2d

https://github.com/skeel3r/exercise1

https://github.com/skeel3r/lab1

https://github.com/skohlbr/hector_sick_robot_day_2014

https://github.com/skohlbr/simple_2wd_robot

https://github.com/skorbiz/epuck_driver

https://github.com/sksavant/analyze_pc

https://github.com/skuba-athome/door_detection

https://github.com/skuba-athome/gesture_detection

https://github.com/skuba-athome/object_perception

https://github.com/skybotix/ros_coax

https://github.com/skyslimit/delta

https://github.com/slamPlus/laser_nodelet

https://github.com/slesinger/robik

https://github.com/sloumotion/cs_merge

https://github.com/slzeng/Robocapstone_CLEANUP-Planner

https://github.com/smARTLab-liv/mitro

https://github.com/smARTLab-liv/smartlabatwork-release

https://github.com/smart-robotics-team/common-smart-team-ros-pkg

https://github.com/smart-robotics-team/eurobot-ros-pkg

https://github.com/smd-cvlab-devel/mocap_kalman

https://github.com/smd-ros-devel/control_panel

https://github.com/smd-ros-devel/diff_drive_controller

https://github.com/smd-ros-devel/joint_state_aggregator

https://github.com/smd-ros-devel/roboteq_nxtgen_controller

https://github.com/smd-ros-devel/smd_ardrone2

https://github.com/smd-ros-devel/yei_tss_usb

https://github.com/smilingpaul/marhes-ros-pkg

https://github.com/smjro/ar_hand_matching

https://github.com/snagged/missioncontrol

https://github.com/snehaljain018/waveslab

https://github.com/snowburnerx/MegaBlocks https://github.com/snowhong/node

https://github.com/snowhong/stuart_kit

https://github.com/snydergo/RobotSoccer_TheFirstOrder

https://github.com/solbach/bumblebee2

https://github.com/songrotek/CLDrone-Autonomous-Quadrotor-Simulation-Research-Platform

https://github.com/sonia-auv/provider_dvl

https://github.com/sonia-auv/rqt_sonia_plugins

https://github.com/soomy/lunokhod

https://github.com/soravux/ros4mat

https://github.com/sourav-jena/ROS-Lab

https://github.com/sourishg/jackal-navigation

https://github.com/spacemanspiff0211/normal_extraction

https://github.com/spalanza/riskrrt_ros

https://github.com/spartangt117/EMARO

https://github.com/spayne/sigevo_champ

https://github.com/spc418-ZC/SPC418-Fall-2016

https://github.com/speckdavid/shakey2016

https://github.com/spencer-project/spencer_people_tracking

https://github.com/spiralray/nhk2015_back_ros

https://github.com/spiralray/robotx-ouxt

https://github.com/spiralray/stm32f4_template

https://github.com/spiralray/stm32f4discovery_dualshock3

https://github.com/spmaniato/cs2024_ros_cpp_project

https://github.com/spoday/IntelligentRobots

https://github.com/spohorec/flyingcar_ros

https://github.com/sputnick1124/irols

https://github.com/squirrel-project/squirrel_calibration

https://github.com/squirrel-project/squirrel_robotino

https://github.com/sradmard/Turtlebot_Gazebo_Simulator

https://github.com/srath287/ASL_MULTIMAPPING_WORKSTATION

https://github.com/srazojr/design-challenge

https://github.com/sri-robotics/segwayrmp

https://github.com/srikanthmalla/auto_landing

https://github.com/srikanthmalla/boiler_gazebo

https://github.com/srl-freiburg/pedsim_ros

https://github.com/srmanikandasriram/multi-agent-robot-test-platform

https://github.com/srrcboy/gazebo_snippets

https://github.com/srsidd/CIS700_Squirtle

https://github.com/srv/auv_framework

https://github.com/srv/imu_odometer

https://github.com/srv/loop_closing_detector

https://github.com/srv/memsense_imu

https://github.com/srv/merbots_ibvs

https://github.com/srv/miniking_ros

https://github.com/srv/object_detection

https://github.com/srv/pattern_pose_estimation

https://github.com/srv/pose_twist_meskf_ros

https://github.com/srv/srv_tools

https://github.com/srv/usbl_position

https://github.com/ssafarik/ImageAnalysis

https://github.com/ssddttaa/RosTriangulation

https://github.com/sshuh/hs2test

https://github.com/sshuh/rover

https://github.com/ssriramana93/aslam_demo

https://github.com/ssriramana93/aslam_ws

https://github.com/ssriramana93/bnr_thesis

https://github.com/ssriramana93/graph_search_sim

https://github.com/ssrselvamraju/PR2RA-pool

https://github.com/stanislas-brossette/cloud-treatment-ecto

https://github.com/start-jsk/rtmros_common

https://github.com/start-jsk/rtmros_gazebo

https://github.com/stateSpaceRobotics/aries2015

https://github.com/stdr-simulator-ros-pkg/stdr_simulator

https://github.com/steevo87/thermalvis

https://github.com/stefanbo92/Visual-Odometry

https://github.com/stephane-caron/manipulation_markers

https://github.com/stereolabs/zed-ros-wrapper

https://github.com/sterlingm/ramp

https://github.com/sterlingm/uncc_robo_lab

https://github.com/stevendaniluk/ghost

https://github.com/stevespiss/ardrone_gesture_control

https://github.com/stevespiss/tactile_feedback_quadcopter_ros

https://github.com/stonier/cost_map

https://github.com/stonier/sophus_ros_toolkit

https://github.com/stoplime/common_cv

https://github.com/stormtiti/carmen

https://github.com/stormtiti/hokuyo_ust_10lx

https://github.com/stormtiti/imu_node

https://github.com/stormtiti/range_sensor_layer

https://github.com/strands-project/aaf_deployment

https://github.com/strands-project/data_compression

https://github.com/strands-project/scitos_apps

https://github.com/strands-project/strands_3d_mapping

https://github.com/strands-project/strands_exploration

https://github.com/strands-project/strands_hri

https://github.com/strands-project/strands_perception_people

https://github.com/strands-project/strands_recovery_behaviours

https://github.com/straszheim/mephitis

https://github.com/strawdiving/crop_protection_1017

https://github.com/strawlab/image_transport_plugins

https://github.com/strymj/marker_detection

https://github.com/strymj/marker_path_planning

https://github.com/strzepek/Cubelet

https://github.com/stschubert/H-RoC

https://github.com/stsssts/ff_test

https://github.com/stsssts/smtu_uauv

https://github.com/sud0301/roasted_ros_codes

https://github.com/sujiwo/robocar

https://github.com/sukibean/receive_image

https://github.com/sunbibei/dragon_rl

https://github.com/sundar2044/RTB

https://github.com/sundar2044/rviz_sat

https://github.com/sungjik/my_personal_robotic_companion

https://github.com/sunnydayw/imu_lsm9ds1

https://github.com/sunzuolei/rosss2015

https://github.com/superjax/cv3_bridge

https://github.com/superjax/tinyIMU_relay

https://github.com/supermandugi/virtualFence

https://github.com/supermaro84/sraluch_ros

https://github.com/surya2191997/agv_task

https://github.com/sushilparti/multi-robot-coverage-ros

https://github.com/suturo16/manipulation

https://github.com/suturo16/perception

https://github.com/svyatoslavdm/hardware

https://github.com/svyatoslavdm/operation_panel

https://github.com/swarmlab/swarmlabatwork

https://github.com/swege/uni-teamarbeit-laserscanner https://github.com/swhart115/pronto_translators

https://github.com/swift-nav/ros_rover

https://github.com/swri-robotics/mapviz

https://github.com/syed911/pcl_pipeline

https://github.com/syl072604/zhibao

https://github.com/sylviadai/ME212_SiyuDai

https://github.com/synapticon/youbot_demo_code

https://github.com/syskaseb/magisterium

https://github.com/sysuyedong/cart

https://github.com/syywh/icp_mapper

https://github.com/syywh/my_ethzasl_icp_mapper

https://github.com/syywh/new_realtime_mapping

https://github.com/syywh/show_loop

https://github.com/t1ina2003/vrep_plugin_ros

https://github.com/tacman-fp7/controlledSlip

https://github.com/takahashi-e6/pan_servo_slam

https://github.com/talkingrobots/GeoCompROS

https://github.com/talkingrobots/NIFTi_OCU

https://github.com/tallerorg/finder_v2

https://github.com/tanmayshankar/hexcopter_autonomy

https://github.com/tanmayshankar/robot_base

https://github.com/tanmayshankar/visual_collision_avoidance

https://github.com/tarquasso/softroboticfish6

https://github.com/tas-car-05/tas_car_05

https://github.com/tas-group-01/my_repository

https://github.com/tas-group-01/tas_car_01

https://github.com/tas-group-04/tas_car_04

https://github.com/tas-group-07/tas-group-07

https://github.com/tas-group-09/tas_car_09

https://github.com/tas1516-group-01/tas1516_car_01

https://github.com/tas1516-group-04/tas1516_car_04

https://github.com/tas1516-group-08/tas1516_car_08

https://github.com/tas1516-group-10/tas1516_car_10

https://github.com/tas1516-group-12/tas1516_car_12

https://github.com/tas1516-group-14/tas_pcs

https://github.com/taurob/taurobtrackerapi

https://github.com/tayyab-naseer/person_following

https://github.com/tbuchman/AdvRobotics

https://github.com/tdenewiler/os5000

https://github.com/team-diana/io-adc

https://github.com/team-diana/io-imu-filter-madgwick

https://github.com/team-diana/io-pr2-robot

https://github.com/team-vigir/vigir_footstep_planning_basics

https://github.com/team-vigir/vigir_lidar_proc

https://github.com/team-vigir/vigir_object_template_manager

https://github.com/team-vigir/vigir_ocs_common

https://github.com/team-vigir/vigir_rviz

https://github.com/teamZeta/zeta

https://github.com/teamrecon/test_drone

https://github.com/techtron16/Anchovy

https://github.com/techtron16/Cornell-Research

https://github.com/techtron16/ModularPerception

https://github.com/tecnalia-medical-robotics/wrench_marker

https://github.com/teddy92/weed_detection

https://github.com/teddyort/icarus

https://github.com/thaeds/BalanceBot

https://github.com/thassa2s/NegarNemati_NiranjanDeshpande_TeenaHassan

https://github.com/thegratefuldawg0520/MobileMapper

https://github.com/theprovidencebreaker/coax_simple_control

https://github.com/theroboticsheep/uvc_camera

https://github.com/thesidjway/Controls-Eklavya4.0-PC

https://github.com/thesidjway/Eklavya5-Control

https://github.com/thiagohomem/RoboFEI-HT_Demo

https://github.com/thieri/toady

https://github.com/thinclab/ardrone_thinc

https://github.com/thirdarm/moveit_interface

https://github.com/thirdarm/thirdarm_controller

https://github.com/thobotics/turtlebot_mpepc

https://github.com/thomas-moulard/pcl-deb

https://github.com/thomas3016/lane_center_keeping

https://github.com/thor-mang/biped_state_estimator

https://github.com/thor-mang/object_template_alignment_plugin

https://github.com/thor-mang/object_template_alignment_server

https://github.com/thor-mang/thor_mang_common

https://github.com/thushv89/deepRLNav

https://github.com/tiagopms/pacman-behavior

https://github.com/tiancovici/WaterBot

https://github.com/tianwailaike/DJI_CHALLENGE_2016 https://github.com/tiberiusferreira/VilmaProject

https://github.com/tidota/uav_practice-Nov-2016

https://github.com/tik0/robocup-viz

https://github.com/tik0/twb-viz

https://github.com/tim-tindell/MILES

https://github.com/tinkerfuroc/tk2_vision

https://github.com/tinokl/alfred

https://github.com/tk0khmhm/knm_tiny_power

https://github.com/tk0khmhm/rp_mx28

https://github.com/tk0khmhm/senior_mapping

https://github.com/tk0khmhm/trajectory_generation

https://github.com/tklaassen/JoystickTurtle5k

https://github.com/tklaassen/Turtle5k_Tim

https://github.com/tlund80/MARVIN

https://github.com/tlund80/RoboVision3D

https://github.com/tnn9/nurse_robot

https://github.com/toborguru/beagleboard_navstack

https://github.com/toddsampson/vendbot

https://github.com/todivasudha/Monocular-VO

https://github.com/tomas-xu/LabVIEW-ROS-Ubuntu

https://github.com/tomas789/tonav

https://github.com/tomemelko/cwru-ros-pkg-hydro-delta

https://github.com/tomlarkworthy/USB-robot-arm

https://github.com/tomlarkworthy/linuxCNC_ROS

https://github.com/tommises/ros-tf2-issue

https://github.com/tonybaltovski/ros_arduino

https://github.com/tork-a/dynpick_driver

https://github.com/touching-foots-huskie/lascer_avoidance

https://github.com/towardthesea/BaxterOculus

https://github.com/towardthesea/kh3ROS

https://github.com/trainman419/Senior-Project

https://github.com/trainman419/dagny_robomagellan

https://github.com/trainman419/dagny_ros

https://github.com/trainman419/rqt_gps_mission

https://github.com/trancept/ROS-Autonomous-Quadcopter-Flight

https://github.com/tranphitien/rosprojects

https://github.com/travisrideout/Rover5_ROS

https://github.com/trgtylcnky/icp_laser

https://github.com/trgtylcnky/laser2map

https://github.com/trhermans/AffLearning

https://github.com/trhermans/bosch_image_proc_pbuffer_fork

https://github.com/trigrass2/Multi-Robot_Hector_Mapping

https://github.com/trih/src_master

https://github.com/trih/src_robot

https://github.com/tristanbell/nao-autism

https://github.com/trorornmn/quadruped_gait

https://github.com/trthanhquang/dynamic_navigation

https://github.com/truecarfield/Automatic-Landing-Control-of-a-Quadrotor-UAV

https://github.com/tsnowak/cwru_376_tsn11

https://github.com/tttor/crim-flapping-bot

https://github.com/tttor/trui-bot-prj

https://github.com/tu-darmstadt-ros-pkg/affw_control

https://github.com/tu-darmstadt-ros-pkg/blob_tools

https://github.com/tu-darmstadt-ros-pkg/grid_map_navigation_planner

https://github.com/tu-darmstadt-ros-pkg/grid_map_proc

https://github.com/tu-darmstadt-ros-pkg/hector_calibration

https://github.com/tu-darmstadt-ros-pkg/hector_camera_control

https://github.com/tu-darmstadt-ros-pkg/hector_handle_detector_tools

https://github.com/tu-darmstadt-ros-pkg/hector_laserscan_to_pointcloud

https://github.com/tu-darmstadt-ros-pkg/hector_mapping2

https://github.com/tu-darmstadt-ros-pkg/hector_move_base_navigation

https://github.com/tu-darmstadt-ros-pkg/hector_mpu6050_imu_converter

https://github.com/tu-darmstadt-ros-pkg/hector_mpu6050_imu_driver

https://github.com/tu-darmstadt-ros-pkg/hector_observation_planner

https://github.com/tu-darmstadt-ros-pkg/hector_perception

https://github.com/tu-darmstadt-ros-pkg/hector_teleop

https://github.com/tu-darmstadt-ros-pkg/hector_tracker_affw

https://github.com/tu-darmstadt-ros-pkg/hector_vision

https://github.com/tu-darmstadt-ros-pkg/hector_visualization

https://github.com/tu-darmstadt-ros-pkg/hector_worldmodel

https://github.com/tu-darmstadt-ros-pkg/navigation_collision_checker

https://github.com/tu-darmstadt-ros-pkg/topic_field_tools

https://github.com/tu-darmstadt-ros-pkg/topic_proxy

https://github.com/tu-darmstadt-ros-pkg/uxvcos

https://github.com/tu-darmstadt-ros-pkg/vehicle_controller

https://github.com/tu-rbo/omip

https://github.com/tud-pses/PSES-Basis

https://github.com/tue-robotics-graveyard/amigo_head_ref https://github.com/tue-robotics-graveyard/amigo_whole_body_controller

https://github.com/tue-robotics-graveyard/fast_simulator2

https://github.com/tue-robotics-graveyard/whole_body_planner

https://github.com/tue-robotics/ed_localization

https://github.com/tue-robotics/ed_sensor_integration

https://github.com/tue-robotics/fast_simulator

https://github.com/tue-robotics/head_ref

https://github.com/tue-robotics/rtt_control_components

https://github.com/tuloski/swm_ros_interface

https://github.com/tumbili/PX4_ROS_shared_lib

https://github.com/tumbili/PX4_mavros

https://github.com/turgaysenlet/carry

https://github.com/turtlebot/turtlebot_apps

https://github.com/tut-yury/tut_ibx0020

https://github.com/tuw-cpsg/dynamic_mapping

https://github.com/tuw-robotics/tuw_gazebo

https://github.com/tuw-robotics/tuw_marker_filter

https://github.com/tuw-robotics/tuw_path_planning

https://github.com/twelsche/inverse_capability_3dmap

https://github.com/twelsche/pr2_arm_base_control

https://github.com/twelsche/pr2_matlab_interface

https://github.com/twighk/ROS-Pi3

https://github.com/txlin/cfly_coordinator

https://github.com/tylkonieona/elektron_balls

https://github.com/tysik/mrop

https://github.com/tysik/mtracker

https://github.com/tysik/obstacle_detector

https://github.com/tyunist/segway_v3

https://github.com/ubi-agni/tactile_toolbox

https://github.com/ucd-robotics/AR.Drone.2.0

https://github.com/ucdart/UCD-UAV

https://github.com/ucfroboticsclub/ucf_robotics

https://github.com/uchile-robotics/ros_workshop

https://github.com/uf-aggregator/AggreGatorRMC

https://github.com/uf-mil/Navigator

https://github.com/uf-mil/PropaGator

https://github.com/uf-mil/Sub8

https://github.com/uf-mil/SubjuGator

https://github.com/uf-mil/hardware-common https://github.com/uf-mil/rawgps-tools

https://github.com/uf-mil/software-common

https://github.com/uji-ros-pkg/underwater_simulation

https://github.com/um10kh/baxter-project

https://github.com/umair4848/RM_SS13_CTC_Ashfaq_Tahir

https://github.com/umdrobotics/Rendezvous

https://github.com/uml-robotics/gradient_map_server

https://github.com/uml-robotics/object_separator

https://github.com/uml-robotics/race_solutions

https://github.com/uml-robotics/tango_ros_ndk

https://github.com/unboundedrobotics/ubr1_preview

https://github.com/unboundedrobotics/ubr_calibration

https://github.com/unl-nimbus-lab/phriky-units

https://github.com/unreason/doomba

https://github.com/unt-robotics/eddiebot_navigation

https://github.com/unusual-thoughts/ros-pixart

https://github.com/uobirlab/RobotButler

https://github.com/uobirlab/jaguar

https://github.com/uos/curious_table_explorer

https://github.com/uos/perception_ar_kinect

https://github.com/uos/semantic_object_maps

https://github.com/uos/sick_tim

https://github.com/uos/uos_active_perception

https://github.com/uos/uos_rotunit

https://github.com/uos/uos_tools

https://github.com/upcomingGit/StageRosKinect

https://github.com/usc-clmc/roscpp_utilities

https://github.com/usc-clmc/usc-clmc-ros-pkg

https://github.com/uscauv-legacy/old-uscauv-ros-pkg

https://github.com/uscresl/laserscan_filter

https://github.com/uscresl/zed_cv_driver

https://github.com/uscrs-art/beohawk-ros

https://github.com/usnistgov/el-robotics-core

https://github.com/ut-ims-robotics/youbot_manual_operation

https://github.com/ut-ras/IGVC2013

https://github.com/ut-ras/RegionV2014

https://github.com/utari/UTARI_ROSTutorials

https://github.com/utexas-air-fri/ardrone_fly

https://github.com/utexas-bwi/bwi_experimental https://github.com/utexas-bwi/segbot_arm

https://github.com/utexas-bwi/segbot

https://github.com/utiasASRL/trex

https://github.com/uts-magic-lab/pr2_pointing

https://github.com/uts-magic-lab/slam_glass

https://github.com/uw-biorobotics/raven2

https://github.com/uwindsor-turtlebot/robot_driver

https://github.com/uwmarsroverteam/Mars-Rover

https://github.com/uzh-rpg/rpg_dvs_ros

https://github.com/v2-mini/v2mini_robot

https://github.com/v4r-tuwien/v4r_ros

https://github.com/vaibhavsaxena11/coord_keyboard

https://github.com/vaibhavsaxena11/motor_drive

https://github.com/van17061/imu_subscriber

https://github.com/vanurag/Kinect-Calibration-Tools

https://github.com/vbillys/icp_test

https://github.com/vbillys/velo_mapping

https://github.com/vbillys/velo_scanmatch

https://github.com/vczhou/learning_object_dynamics

https://github.com/vdiluoffo/gazebo_ros2_pkgs

https://github.com/vdutor/project-rsc

https://github.com/vegardsl/MobileAutonomousRobot

https://github.com/vegardsl/ros_mar

https://github.com/veltrop/veltrop_ros_pkg

https://github.com/velveteenrobot/fydp_demo

https://github.com/velveteenrobot/me597_lab3

https://github.com/velveteenrobot/turtlebot_example_lab_2

https://github.com/veniora/se306p1

https://github.com/venkatrn/ltm

https://github.com/venkatrn/monolithic_pr2_planner

https://github.com/versatran01/usb_camera

https://github.com/vertcli/robotic_learning

https://github.com/vfdev-5/Catkin_ws_cv_tests

https://github.com/victor-unsw/Jaco-Robot

https://github.com/victorkhoo/NewP3AT

https://github.com/vidits-kth/SAR-Autonomy

https://github.com/vigneshba/Drone_TLD

https://github.com/vigneshba/Stereo-Drone_TLD

https://github.com/vigneshrajaponnambalam/Risk-RRT-Social-Navigation

https://github.com/vikiboy/AGV_Localization
https://github.com/vinaykumarhs2020/lidar_publisher
https://github.com/vincentwenxuan/catkin_ws_git
https://github.com/viron11111/anglerfish
https://github.com/viron11111/robot_setup_tf
https://github.com/vishnu291093/ROS_Summer2016
https://github.com/vishnumuthu/darkROS
https://github.com/vishu2287/turtlebot-slam-group6
https://github.com/vislab-tecnico-lisboa/HumanAwareness
https://github.com/vislab-tecnico-lisboa/ardrone_gazebo
https://github.com/vislab-tecnico-lisboa/foveated_stereo_ros
https://github.com/vislab-tecnico-lisboa/grasping
https://github.com/vislab-tecnico-lisboa/move_robot
https://github.com/vislab-tecnico-lisboa/vizzy
https://github.com/vislab-tecnico-lisboa/yarp-with-moveit
https://github.com/vivekprayakarao/AMAV
https://github.com/vivekprayakarao/lidar_node
https://github.com/vivicampo21/Cuadricoptero_ROS
https://github.com/vkrishnakanth/ar_recog
https://github.com/vladestivillcastro/Reproducibility
https://github.com/void32/frobit
https://github.com/vortexntnu/hw_interface
https://github.com/vortexntnu/rov-control
https://github.com/vortexntnu/uranus_dp
https://github.com/vpersaud/trunk-master
https://github.com/vscroll/sitl_gazebo_2
https://github.com/vvanirudh/tum_ardrone_iitb
https://github.com/walaankit/consensus_position
https://github.com/walchko/ahrs
https://github.com/walchko/ros_soccer
https://github.com/walchko/scout
https://github.com/walchko/wiic_twist
https://github.com/walk1ndude/navpts
https://github.com/wallarelvo/rvo2
https://github.com/wallarelvo/tca
https://github.com/waltYeh/flight-strategy
https://github.com/wambot/sdl_viewer
https://github.com/wangcong386/robotino
https://github.com/wangwei19900806/github-program

https://github.com/warcraft23/MobileRobots

https://github.com/warlockhjn/ROS_SLAM_XDD

https://github.com/watermelon1995/fyp2016

https://github.com/watertown/pff-node-ws

https://github.com/wcaarls/ur_arm

https://github.com/wdecre/kuka-robot-hardware

https://github.com/weichnn/AvoidancebyProjection

https://github.com/weiin/transporter-project

https://github.com/weiweihuang/Pathplanning

https://github.com/weiweikong/aruco_ros_landing

https://github.com/weiweikong/px4_velocity_control_keyboard

https://github.com/wengsaltedfish/camera_info_test

https://github.com/wengsaltedfish/serial_mavlink

https://github.com/wengsaltedfish/visoRos

https://github.com/wenhust/imlidar_driver

https://github.com/wennycooper/base_local_planner

https://github.com/wennycooper/mybot_autodocking

https://github.com/wennycooper/mybot_followme

https://github.com/wennycooper/xu3_nav

https://github.com/wercool/valter

https://github.com/wert23239/MHacks216

https://github.com/westpoint-robotics/iggy

https://github.com/westpoint-robotics/mavros_pose_control

https://github.com/westpoint-robotics/usma_optitrack

https://github.com/wfearn/ros-morrf-project

https://github.com/wfriedl/IGVC_Scipio_Software

https://github.com/wg-perception/linemod

https://github.com/wg-perception/reconstruction

https://github.com/wher0001/sparton_turtle

https://github.com/whoenig/ros-clang-instrumentation

https://github.com/will-andrew/couch_driver

https://github.com/will-zegers/Robotics291

https://github.com/willdzeng/ARTI4

https://github.com/williamalu/bravobot

https://github.com/wilson-ko/agile_demo

https://github.com/windbicycle/oru-ros-pkg

https://github.com/wine3603/motion_detection

https://github.com/wine3603/scanline-slam

https://github.com/wine3603/scanline_grouping https://github.com/withniu/ncvrl_ros

https://github.com/wjwwood/3d_teleop

https://github.com/wjwwood/open-robotics-platform

https://github.com/wliu88/rail_agile_grasp

https://github.com/wliu88/turtlebot_avoidance

https://github.com/wliu88/turtlebot_crossing

https://github.com/wmarshall484/Mesh-making-pipeline

https://github.com/won13y/mavros_test

https://github.com/woshigeshabiaaa/auto_takeoff

https://github.com/woshigeshabiaaa/hector_quadrotor_gazebo_plugins

https://github.com/woshigeshabiaaa/hector_works

https://github.com/woshigeshabiaaa/parallel_uav

https://github.com/woshigeshabiaaa/rotors_gazebo_plugins

https://github.com/wrousseau/doctor-drone

https://github.com/wsnewman/cwru-ros-pkg-hydro-wsn

https://github.com/wsnewman/learning_ros

https://github.com/wsnewman/rethink_cwru

https://github.com/wuconnie/SVODepth

https://github.com/wuconnie/SVO_Demo

https://github.com/wuhy08/GLEWproject

https://github.com/wwyiyi/bebop_image_pkg

https://github.com/wyqsnddd/pr2ForceTorqueReader

https://github.com/wystephen/IMU_NAVIGATION

https://github.com/wystephen/M_SLAM

https://github.com/wystephen/tf_to_pose

https://github.com/x-logan/uav_control

https://github.com/x007dwd/visca_ros

https://github.com/xd1313113/AutonomousRobot

https://github.com/xdiegool/sjtu_simulator

https://github.com/xdmeng09/Firmware_new_Aug2016

https://github.com/xenron/sandbox-github-clone

https://github.com/xiaopenghuang/Gazebo_summit_robot

https://github.com/xihiro96/myCRCL

https://github.com/xiruzhu/BUGS_417

https://github.com/xkingkiller/Map_Optimizer

https://github.com/xm-project/xm_velocity_smoother

https://github.com/xpecttrum/fsr_backpack

https://github.com/xperroni/InterestPointCorrelationSupportMaterials

https://github.com/xperroni/yamabros

https://github.com/xpharry/DaVinci_Robot

https://github.com/xpharry/ROS-learning

https://github.com/xpharry/SelectPublisherBugFix

https://github.com/xpharry/SwarmRoboticsResearch

https://github.com/xpharry/alpha_ps8

https://github.com/xpharry/eecs-397-f15

https://github.com/xuefengchang/micROS-SARRT-GlobalPlanner

https://github.com/xunkai55-ood/Laser-Refinery

https://github.com/xuqingwenkk/ROS-beginner

https://github.com/xuxie1031/Pattern-Recognition-tum_ardrone

https://github.com/xwu4lab/jaco_husky_demo

https://github.com/xwu4lab/jaco_moveit

https://github.com/xwu4lab/some_small_tools

https://github.com/xxj79/cwru_davinci

https://github.com/xxj79/davinci_calibration

https://github.com/xxzc/hector_quadrotor_tutorial

https://github.com/yaleinve/sailbot

https://github.com/yang85yang/getYawfromQuaternions

https://github.com/yang85yang/navigation_loop

https://github.com/yang85yang/odom_publisher

https://github.com/yang85yang/simple_navigation_goals-show_picture-

https://github.com/yangfuyuan/Okrobot

https://github.com/yangjian940712/serial_port

https://github.com/yangmeitang/stereocam_traffic_cone_locator

https://github.com/yanik-porto/TurtleSwarmMapper

https://github.com/yannrs/Project1

https://github.com/yasu80/pcl_hemisphere_tracker

https://github.com/yawei1/frontier_extraction

https://github.com/yazdani/cram_sar_mission

https://github.com/yazdani/hc_sherpa_spatial_relations

https://github.com/yazdani/iai_rescue_mission

https://github.com/yazdani/old_pkgs

https://github.com/yazdani/sar_mission

https://github.com/ych09041/ee206akitchen

https://github.com/yfZhong/visual_tracking

https://github.com/yhao0410/hokuyo

https://github.com/yhao0410/sensortest

https://github.com/yhb852948970/ST_UAV_Phase1

https://github.com/yhb852948970/ST_UAV_Phase2 https://github.com/yincanben/turtlebot_wander

https://github.com/yingjh/IARC_13

https://github.com/yitong-quan/master

https://github.com/yl573/CAUV_software_v2

https://github.com/yoneken/gazebo_ros_wheel

https://github.com/yoneken/measure_message_passing

https://github.com/yoshimalucky/pcl_ros_outdoor

https://github.com/yoshito-n-students/bno055_usb_stick

https://github.com/youBot-DHBW-Karlsruhe/grab_demo

https://github.com/youbot-hackathon/youbot-apps

https://github.com/youbot/youbot-ros-pkg

https://github.com/yournjell/Catheter

https://github.com/yowlings/rocwang_server_robot

https://github.com/yowlings/xbot

https://github.com/ypei92/cs393r_autonomous_robot_code

https://github.com/ysonggit/gazebo_simple

https://github.com/yswhynot/AprilTag-ROS-package-for-V-REP

https://github.com/yswhynot/Bookle

https://github.com/yswhynot/MultiCameraBot

https://github.com/yuchenqiaqia/uavgp_vision

https://github.com/yuconglin/ardrone_rrt_avoid

https://github.com/yuhangc/ford_project

https://github.com/yujinrobot/kobuki-x

https://github.com/yujinrobot/kobuki_desktop

https://github.com/yujinrobot/kobuki

https://github.com/yujinrobot/minsu_pcl

https://github.com/yujinrobot/rplidar_extras

https://github.com/yujinrobot/yujin_ocs

https://github.com/yukkysaito/autoware_thesis

https://github.com/yukkysaito/hokuyo3d_pointcloud2

https://github.com/yukkysaito/hokuyo3d_pointcloud_pointcloud2

https://github.com/yuyuyu00/CartoInterface

https://github.com/yuzhangbit/vrep-package

https://github.com/yzrobot/explore

https://github.com/zakharov/BRICS_RN

https://github.com/zalessio/rpg_mpc

https://github.com/zamora18/robot-soccer

https://github.com/zamxt9/Main-Codes

https://github.com/zamxt9/spauv

https://github.com/zastrix/ros-drivers-um7
https://github.com/zbr3550/FW-StallLanding-Test
https://github.com/zchenpds/turtlebot_test
https://github.com/zeabusTeam/modbus_ascii_master
https://github.com/zengzhen/ros_zhen
https://github.com/zephirefaith/spencer_jeeves_version
https://github.com/zeus3000/PathPlannerModule
https://github.com/zhangaigh/rovio-standalone
https://github.com/zhangshouqi/cooperation
https://github.com/zheng-rong/matrix_camera
https://github.com/zhenrychen/MaidbotLaserscan
https://github.com/zhiyuhuo/hri_stack
https://github.com/zhiyuhuo/kimi_robot
https://github.com/zhoujoey/base2odom
https://github.com/zhoujoey/cartographer_movebase_tested
https://github.com/zhoujoey/hcrdemo
https://github.com/zhoujoey/ros_newlife
https://github.com/zhoujoey/turtlebot_cartographer_movebase
https://github.com/ziox/leonardo
https://github.com/zohannn/aros_moveit_planner
https://github.com/zohannn/aros_moveit_tutorials
https://github.com/zohannn/motion_manager
https://github.com/zphilip/KinectOpenGL1
https://github.com/zukoo/MScProject
https://github.com/zurich-eye/ze_oss
https://github.com/zyms5244/aksrobot_edge_clean
https://jpwco@bitbucket.org/udg_cirs/ladybug3_ros_pkg.git

Table 9.5: Open source systems analyzed in § 6.