PythonRobotics: a Python code collection of robotics algorithms

Atsushi Sakai
https://atsushisakai.github.io/
University of California, Berkeley

Daniel Ingram
ingramds@appstate.edu
Appalachian State University

Joseph Dinius
jdinius@inviarobotics.com
inVia Robotics, Inc.

Karan Chawla
karan@skyryse.com
Skyryse Inc.

Antonin Raffin
antonin.raffin@ensta-paristech.fr
ENSTA ParisTech

Alexis Paques
Alexis@unmanned.life
Unmanned Life

Abstract

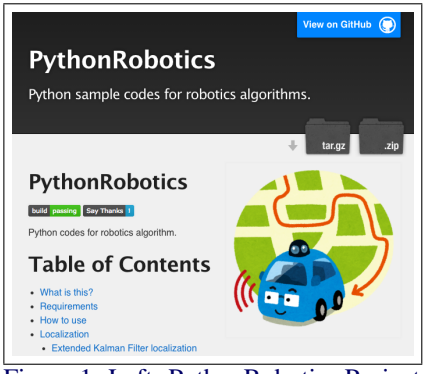
This paper describes an Open Source Software (OSS) project: PythonRobotics[**?**]. This is a collection of robotics algorithms implemented in the Python programming language. The focus of the project is on autonomous navigation, and the goal is for beginners in robotics to understand the basic ideas behind each algorithm. In this project, the algorithms which are practical and widely used in both academia and industry are selected. Each sample code is written in Python3 and only depends on some standard modules for readability and ease of use. Each algorithm is written in Python3 and only depends on some common modules for readability, portability and ease of use. It includes intuitive animations to understand the behavior of the simulation.

1 Introduction

In recent years, autonomous navigation technologies have received huge attention in many fields. Such fields include, autonomous driving[**?**], drone flight navigation, and other transportation systems.

An autonomous navigation system is a system that can move to a goal over long periods of time without any external control by a operator. The system requires a wide range of technologies: It needs to know where it is (localization), where it is safe (mapping), where and how to move (path planning), and how to control its motion (path following). The autonomous system would not work correctly if any of these technologies is missing.

Educational materials are becoming more and more important for future developers to learn basic autonomous navigation technologies. Because these autonomous technologies need different technological skill sets such as: linear algebra, statistics, probability theory,



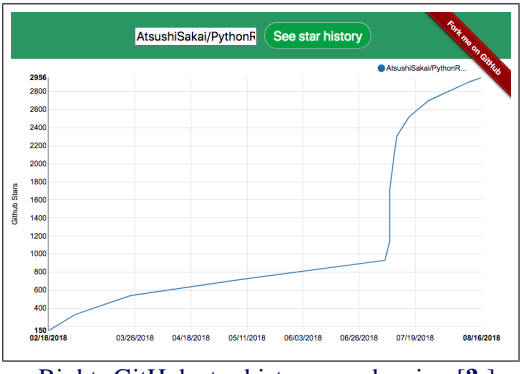


Figure 1: Left: PythonRobotics Project page, Right: GitHub star history graph using [**?**]

optimization theory, control theory, and sensing physics etc. It needs a lot of time to be familiar with these technological areas. Therefore, good educational resources for learning basic autonomous navigation technologies are needed. Our project which is described in this paper aims to be one such resource.

In this paper, an Open Source Software(OSS) project: PythonRobotics[**?**] is described. This project provides a code collection of robotics algorithms, especially focusing on autonomous navigation. The principle goal is to provide beginners with the tools necessary to understand it. It is written in Python[**?**] under MIT license[**?**]. It has a lot of simulation animations that shows behaviors of each algorithm. It helps for learners to understand its fundamental ideas. The readers can find the animations in the project page `https://atsushisakai.github.io/PythonRobotics/`. The left figure in Fig.1 shows the front image of the page. All source codes of this project are provided at `https://github.com/AtsushiSakai/PythonRobotics`.

This project was started from Mar. 21 2016 as a self-learning project. It already has over 3000 stars on GitHub, and the right figure in Fig.1 shows the history graph of the star counts using [**?**].

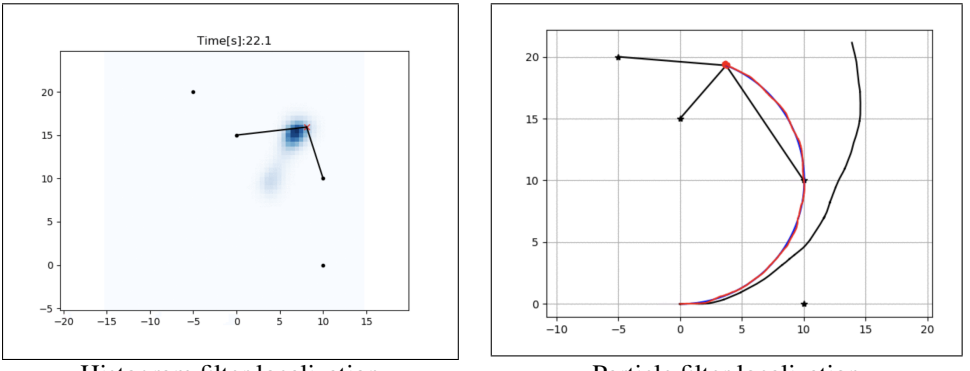
This paper is organized as follows: Section 2 reviews the related works. The philosophy of this project is presented in Section 3. The repository structure and some technical backgrounds and simulation results are provided in Section 4. Conclusions are drawn and some future works are presented in Section 5. Section 6 acknowledges for contributors and supporters.

2 Related works

There are already some great educational materials for learning autonomous navigation technologies.

S. Thrun et al. wrote a great textbook "Probabilistic robotics" which is a bible of localization, mapping, and Simultaneous Localization And Mapping (SLAM) for mobile robotics[**?**]. E. Frazzoli et al. wrote a great survey paper about path planning and control techniques for autonomous driving [**?**]. G. Bautista et al. wrote a survey paper focusing on path planning for automated vehicles[**?**]. J. Levinson wrote an overview paper about systems and algorithms towards fully autonomous driving[**?**].

These papers help readers to learn state-of-the-art autonomous navigation technologies.



Histogram filter localization Particle filter localization

Figure 2: Localization simulation results

However, it might be difficult for beginners to understand the basic ideas of the technologies and algorithms because the papers don't include implementation examples.

Many universities provide great classes to learn robotics and share their lecture notes. For example, University of Freiburg provides an introduction class to mobile robotics[**?**]. Swiss Federal Institute of Technology in Zurich (ETH in Zurich) also provides a class about autonomous mobile robot[**?**] and robotics programming with Robot Operating System(ROS)[**?**].

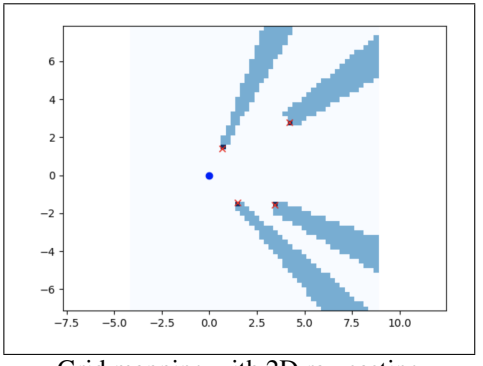
Like the academic literature referenced, these lecture notes also help readers to learn autonomous navigation technologies. However, it might be also difficult for beginners of robotics to understand the basic ideas of the technologies and algorithms if the reader is not a student of the class, because these lecture notes doesn't include actual implementation examples of robotics algorithms.

Robot Operating System (ROS) is a middle-ware for robotics software development[**?**][**?**]. ROS initially released in 2007, and it has become the defacto standard platform in robotics software development. ROS includes basic navigation software such as the Adaptive Monte Carlo Localization (AMCL) package and the local path planner based on Dynamic Window Approach, etc[**?**]. Many autonomous navigation packages using ROS are also open-sourced.

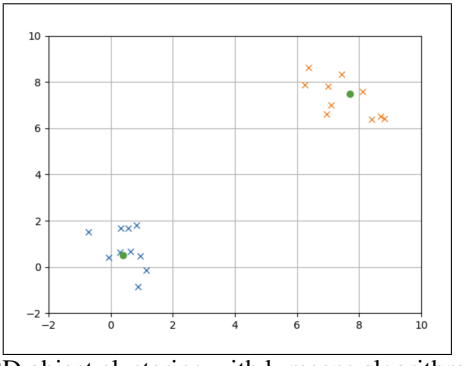
ROS packages can also be useful to learn autonomous navigation algorithms, however the documentation is inconsistent and, at times, it can be difficult to find implementation details about the algorithms used. ROS packages are usually written in C++ because the focus is computational efficiency and not readability, which makes learning from the code harder.

Udacity Inc, which is an online learning company founded by S. Thrun, et al, is providing great online courses for autonomous navigation and autonomous driving [**?**]. These courses provide not only technical lectures, but also programming homework using Python. This leads to a great learning strategy for beginners to understand the technical background required for autonomous systems.

The PythonRobotics project seeks to extend Udacity's approach by giving beginners the necessary tools to understand and make further study into the methods of autonomous navigation. The details of concepts of this OSS project will be described in the next section.



Grid mapping with 2D ray casting



2D object clustering with k-means algorithm

Figure 3: Mapping simulation results

3 Philosophy

~~In this section, the philosophy of this project is described. It~~The PythonRobotics project is based on three main philosophies. Each of which will be discussed separately in this section.

I'M NOT SURE THAT THESE NEED TO BE SUBSECTIONS. YOU COULD PROBABLY JUST USE BULLET POINTS

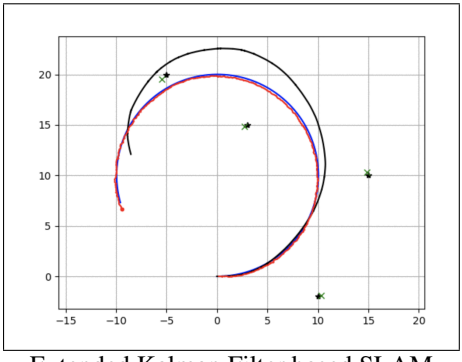
3.1 Readability

The first ~~one~~philosophy is that the code ~~have~~has to be easy to read. ~~for understanding each algorithm. This project aims beginners of robotics to understand basic ideas of each algorithm~~If the code is not easy to read, it would be difficult to achieve our goal of allowing beginners to understand the algorithms. ~~Python programming language is adopted in this project because of its simplicity and it allows us to focus on algorithm itself.~~THIS SENTENCE DOESN'T MAKE ANY SENSE HERE. PYTHON IS NOT MORE OR LESS EASY TO READ THAN ANY OTHER LANGUAGE. I'D SUGGEST REMOVING THE SENTENCE ENTIRELY. Python has great libraries for matrix operation, mathematical and scientific operation, and visualization~~.~~, which makes code more readable because such operations don't need to be reimplemented. Having the core Python packages available allows the user to focus on the algorithms, rather than the implementations. ~~These libraries also allows us to focus on algorithm itself.~~

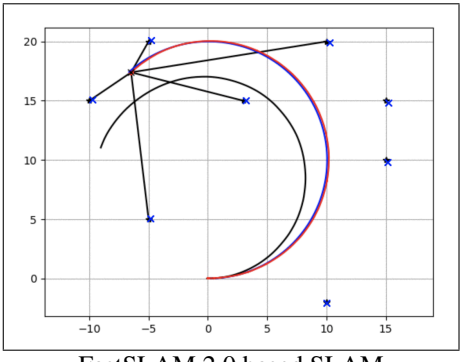
3.2 ~~Practicality~~

3.3 Relevance

The second ~~one~~philosophy is that implemented algorithms have to be ~~practical and widely used in~~relevant to both academia and industry. YOU MIGHT WANT TO MENTION HERE THAT MANY OF THE ALGORITHMS IMPLEMENTED REFERENCE A PARTICULAR ACADEMIC OR INDUSTRIAL REFERENCE IN THE COMMENTS SECTION. ~~For example, Kalman filters and particle filter for localization, grid mapping for mapping, dynamic programming based approaches and sampling based approaches for path planning, and optimal control based approach for path tracking.~~ THIS WOULD BE A GOOD PLACE TO PUT



Extended Kalman Filter based SLAM



FastSLAM 2.0 based SLAM

Figure 4: SLAM simulation results

A SUMMARY TABLE OR BULLET POINTS WITH SOME EXAMPLES OF RELEVANT ALGORITHMS THAT HAVE BEEN IMPLEMENTED

3.4 ~~Minimum dependency~~

3.5 Minimal dependencies

The last philosophy is ~~minimum dependency~~minimal dependencies. ~~It~~Having few external dependencies allows us to run code samples easily and to convert them ~~Python codes~~ to other programming languages, such as C++,~~or~~ Java, for ~~practical usage~~other applications. Each sample code only depends some modules on Python3 as bellow.

- numpy[**?**] for matrix and vector operations
- scipy[**?**] for mathematics~~,~~al and scien~~ce~~tific~~, and engineering~~ computing
- matplotlib[**?**] for plotting and visualization
- pandas[**?**] for data import and manipulation
- cvxpy[**?**] for convex optimization

These modules are OSS and ~~could~~can also be used for free. ~~This repository software~~PythonRo doesn't depend on any commercial software.

4 Repository structure

~~In this section, we briefly described the repository structure.~~THIS SENTENCE IS REDUNDANT

~~This repository has five directories which~~There are five directories, each one corresponding to ~~five~~a different technical categor~~ies~~y in autonomous navigation: localization, mapping, SLAM, path planning, and path tracking. There is one sub-directory per algorithm, which has a sample code ~~using different algorithms~~implementing and testing the algorithm.

In the following subsections, some algorithms and some simulation examples in each category are described.

4.1 Localization

Localization is the ability of a robot to know its position and orientation with sensors such as GNSS and IMU DEFINE THESE ACRONYMS! EVERY READER MAY NOT BE FAMILIAR WITH INERTIAL SENSING. In localization, Bayesian filters such as Kalman filters, histogram filter, and particle filter are widely used[**?**]. Fig.2 shows localization simulations using histogram filter and particle filter.

4.2 Mapping

Mapping is the ability of a robot to understand its surroundings with external sensors such as LIDAR and camera. Robots have to recognize the position and shape of obstacles to avoid them. In mapping, grid mapping and machine learning algorithms are widely used[**?**][**?**]. Fig.3 shows mapping simulation results using Grid mapping with 2D ray casting and 2D object clustering with k-means algorithm.

4.3 SLAM

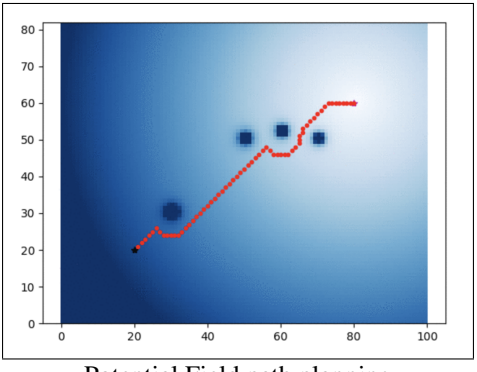
Simultaneous Localization and Mapping (SLAM) is an ability to estimate the pose of a robot and the map of the environment at the same time. The SLAM problem is hard to solve, because a map is needed for localization and ~~a~~ good localization is needed for mapping~~, which is a kind of chicken and egg problem~~. In this way, SLAM is often said to be similar to a "chicken-and-egg" problem. Popular SLAM solution methods include the extended Kalman filter, particle filter, and Fast SLAM algorithm[**?**]. Fig.4 shows SLAM simulation results using Extended Kalman Filter and results using FastSLAM2.0[**?**].

4.4 Path planning

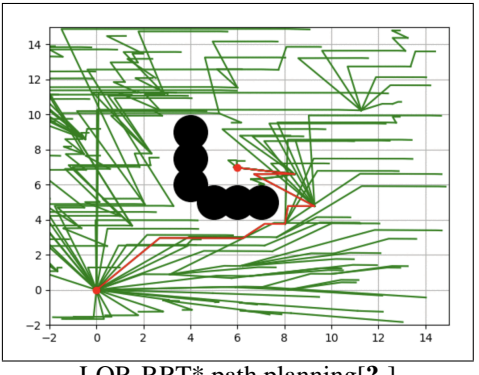
Path planning is the ability of a robot to find an allowable path from some starting point to a desired final point. This typically means ~~to~~ searching over feasible ~~and efficient~~ paths to the goal and choosing the "best" one. The best path ha~~ve~~s to satisfy some constraints based on the robot's ~~movement~~motion model and obstacle positions~~,~~. By "best" path, it is typically meant that the path with highest reward (or lowest cost), with respect to some objective function, is chosen. ~~and optimize some objective functions such as~~Some example objective functions include time to goal and distance to obstacle ~~etc~~. In path planning, dynamic programming based approaches and sampling based approaches are widely used[**?**]. Fig.5 shows simulation results of potential field path planning and LQR-RRT* path planning[**?**].

4.5 Path tracking

Path tracking is ~~an~~the ability of a robot to follow the reference path generated by a path planner~~ing algorithms~~. The role of the path tracking controller is to ~~stabilize to the reference path or trajectory which has~~follow a reference trajectory while simultaneously stabilizing the robot. The path tracking controller may need to account for modeling error and other forms of uncertainty. In path tracking, feedback control techniques and optimization based control techniques are widely used[**?**]. Fig.6 shows simulations using rear wheel feedback steering control and PID speed control, and iterative linear model predictive path tracking control[**?**].

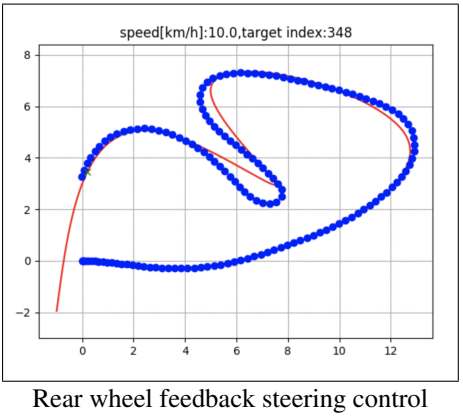


Potential Field path planning

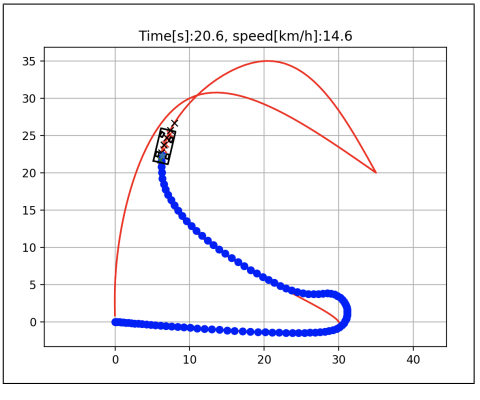


LQR-RRT* path planning[**?**]

Figure 5: Path planning simulation results



Rear wheel feedback steering control and PID speed control [**?**]



Iterative linear model predictive control

Figure 6: Path tracking simulation results

5 Conclusion and future work

In this paper, we introduced PythonRobotics, a code collection of robotics algorithms, with a focus on autonomous navigation. ~~especially for autonomous navigation.~~REDUNDANT Related works of this project, some key ideas about this OSS project, and brief structure of this repository and some simulation results were described.

The future work~~s of~~for this project ~~are~~is as follow~~ed~~s:

- Adding technical and mathematical documentation using Jupyter notebook[**?**].
- Adding image processing samples for autonomous navigation ~~only~~ using OpenCV[**?**].
- Adding simple multi-robot~~s~~DON'T NEED THE s HERE simulations for autonomous navigation.
- Adding a comparison to find which algorithm fits ~~the~~ best for a specific application.

If readers ~~were~~are interested in these future projects, contributions are welcome. Readers can also support this project financially via Patreon[**?**].

6 Acknowledgments

We appreciate all contributors: Atsushi Sakai[**?**], Daniel Ingram[**?**], Joe Dinius[**?**], Karan Chala[**?**], Antonin RAFFIN[**?**], and Alexis Paques[**?**]. We also appreciate all robotics lovers who give a star to this repository in GitHub.