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

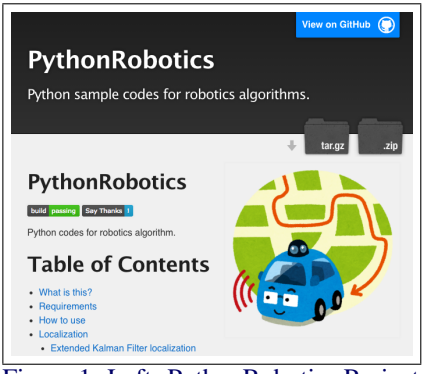
Abstract

This paper describes an Open Source Software (OSS) project: PythonRobotics[**?**]~~.~~: ~~This is~~ a Python code collection of robotics algorithms~~,~~. The focus of the project ~~especially focusing on~~ is on autonomous navigation~~.~~, ~~It aims beginners of robotics to understand basic ideas of each algorithm.~~ with the principle goal being to provide beginners with the tools necessary to understand it. 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~~ has minimal dependencies, ~~It also provides~~ and includes intuitive animations ~~to understand the behavior of the simulation~~.

1 Introduction

In recent years, autonomous navigation technologies have received ~~a~~ huge attention in many fields. ~~For example,~~ Such fields include: autonomous driving[**?**], drone flight navigation, and other transportation systems.INCLUDE ADDITIONAL REFERENCES IF YOU HAVE THEM

An autonomous navigation system is ~~a system~~ one that can move an object to a goal, potentially ~~for~~ over long periods of time, without any external control by a human operator. ~~The~~ Such a system requires a wide range of technologies~~.~~ : It needs to know where it is (localization), ~~where it is safe~~have knowledge of its environment (mapping), know where and how to ~~go~~move (path planning), and how to control its motion with its actuators (path following). The autonomous system would not work correctly if any of these technologies is missing.



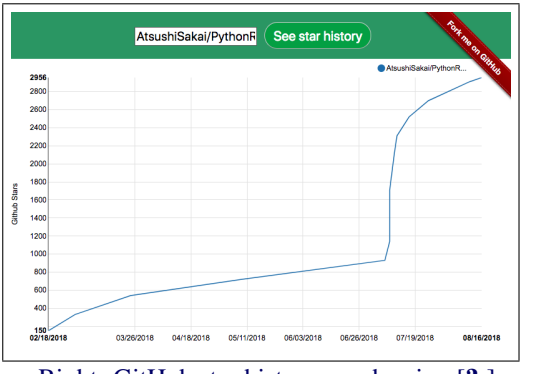


Figure 1: Left: PythonRobotics Project page, Right: GitHub star history graph using [**?**]

Educational materials are becoming more and more important for future developers to learn basic autonomous navigation technologies. This is ~~B~~because these ~~autonomous~~ technologies need different ~~technological~~technical skillsets ~~backgrounds~~. ~~such as~~ Such skillsets include: linear algebra, statistics, probability theory, optimization theory, control theory, and sensing physics, ~~etc.~~to name but a few. ~~It needs a lot of time to be familiar with these technological areas.~~These skillsets can take a long time to learn, let alone become comfortable with. Therefore, good educational ~~materials~~resources for learning basic autonomous navigation technologies are needed. PythonRobotics aims to be one such resource.

YOU PROBABLY DON'T WANT TO JUST REPEAT THE ABSTRACT. THE FOLLOWING PARAGRAPH SHOULD BE REWRITTEN OR REMOVED. 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. It aims beginners of robotics to understand basic ideas of each algorithm. 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 ~~check the~~review 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. ~~its~~The repository structure and some ~~technological~~technical backgrounds and simulation results are provided in Section 4. Conclusions are drawn and ~~some future works are drawn~~thoughts for future work are presented in Section 5. Section 6 ~~shows acknowledgements for all~~acknowledges contributors and ~~all~~ 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~~s~~ readers to learn state-of-the-art autonomous navigation technologies. However, it might be difficult for beginners to understand the basic ideas of the technologies and algorithms~~,~~ because the~~se~~ papers don't include implementation examples.

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

Like the academic literature referenced, ~~T~~these lecture notes also help~~s~~ readers to learn ~~the~~ 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 implementation examples of robotics algorithms.~~, limited available examples of actual implementation of algorithms discussed might impede a beginner's ability to learn and use the material.

Robot Operating System (ROS) is a middle-ware for robotics software development[**?**][**?**]. ROS initially released in 2007, and it ~~became a~~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 the Dynamic Window Approach, etc[**?**]. Many autonomous navigation packages using ROS are also ~~opened as an OSS~~open-sourced.

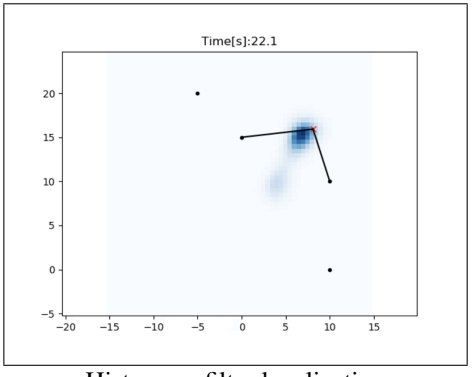
~~The~~ ROS packages ~~are~~can also be useful to learn autonomous navigation algorithms~~.~~, ~~H~~however, ~~it might has an issue, because of lacking of documentation about its basic algorithm~~the documentation is inconsistent and, at times, it can be difficult to find implementation details about the algorithms used. ~~Those~~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 ~~of~~for autonomous navigation and autonomous driving [**?**]. ~~This~~These course~~s~~ provide~~s~~ not only technical lectures, but also programming homework using Python. ~~its~~This leads to a great learning strategy ~~is great~~ for beginners to understand ~~its~~the technical background required for autonomous systems.

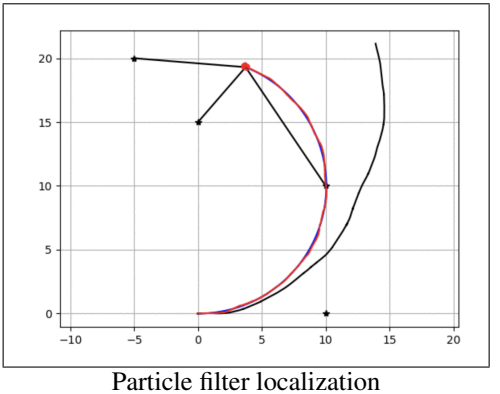
The PythonRobotics project seeks to extend Udacity's approach by giving beginners the necessary tools to understand and make further study into the methods of autonomous navigation.~~has a similar concept of the Udacity's course. It aims beginners of robotics to understand basic technical ideas of each autonomous navigation algorithm using Python.~~ The details of concepts of this OSS project will be described in the next section.

3 Philosophy

In this section, the philosophy of this project is described. It based on three main philosophies.



Histogram filter localization



Particle filter localization

Figure 2: Localization simulation results

3.1 Readability

The first one is that the code have to be easy to read for understanding each algorithm. This project aims beginners of robotics to understand basic ideas of each algorithm. Python[**?**] programming language is adopted in this project because of its simplicity and it allows us to focus on algorithm itself. Python has great libraries for matrix operation, mathematical and scientific operation, and visualization. These libraries also allows us to focus on algorithm itself.

3.2 Practicality

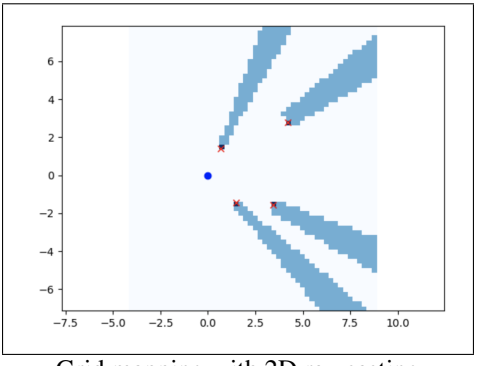
The second one is that implemented algorithms have to be practical and widely used in both academia and industry. For example, Kalman filters and particle filter for localization, grid mapping for mapping, dynamic programming based approaches and sampling based approaches for path planning, and optimal control based approach for path tracking.

3.3 Minimum dependency

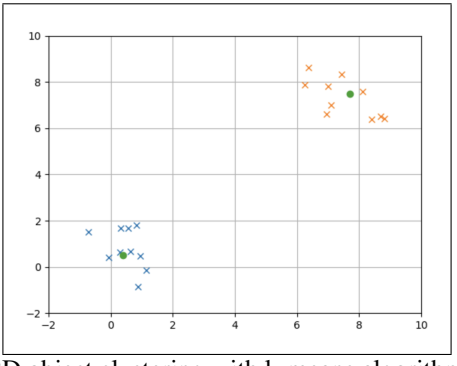
The last philosophy is minimum dependency. It allows us to run code samples easily and to convert the Python codes to other programming languages such as C++, Java for practical usage. Each sample code only depends some modules on Python3 as bellow.

- numpy[**?**] for matrix operation
- scipy[**?**] for mathematics, science, and engineering computing
- matplotlib[**?**] for visualization
- pandas[**?**] for data import
- cvxpy[**?**] for convex optimization

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



Grid mapping with 2D ray casting



2D object clustering with k-means algorithm

Figure 3: Mapping simulation results

4 Repository structure

In this section, we briefly described the repository structure.

This repository has five directories which corresponding to five technical categories 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.

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

4.1 Localization

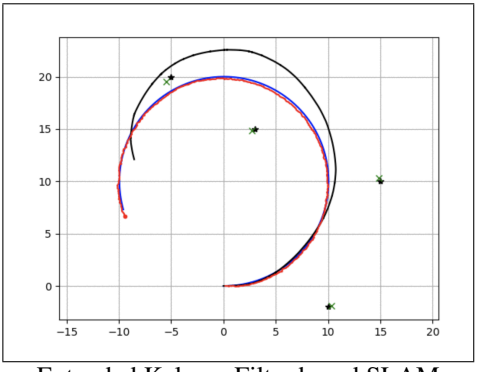
Localization is the ability of a robot to know its position and orientation with sensors such as GNSS and IMU. In localization, Bayesian filters such as Kalman filters, histogram filter, and particle filter are widely used[**?**]. Fig.2 shows localization simulations using histogram filter and particle filter.

4.2 Mapping

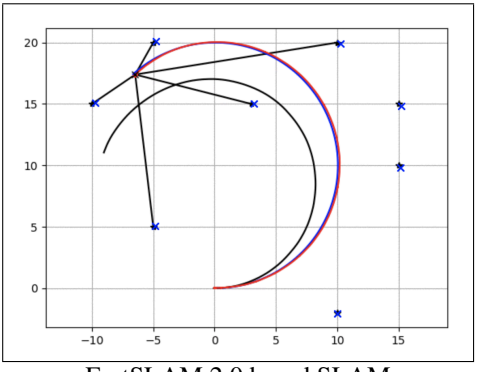
Mapping is the ability of a robot to understand its surroundings with external sensors such as LIDAR and camera. Robots have to recognize the position and shape of obstacles to avoid them. In mapping, grid mapping and machine learning algorithms are widely used[**?**][**?**]. Fig.3 shows mapping simulation 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. 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. 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[**?**].

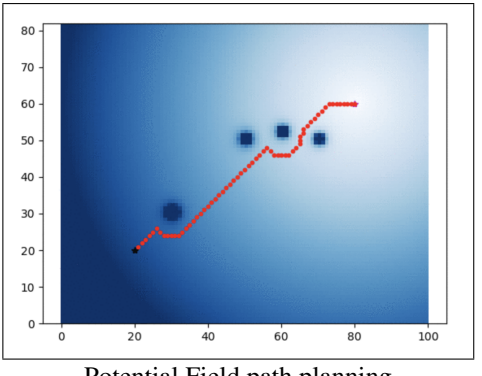


Extended Kalman Filter based SLAM

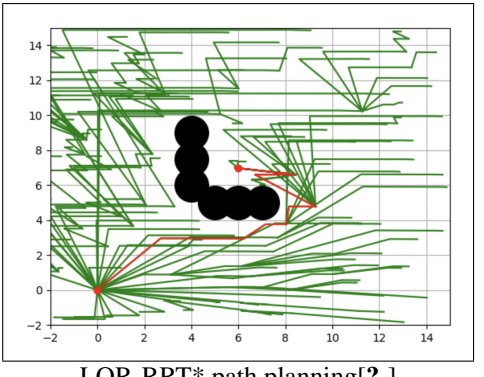


FastSLAM 2.0 based SLAM

Figure 4: SLAM simulation results



Potential Field path planning



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

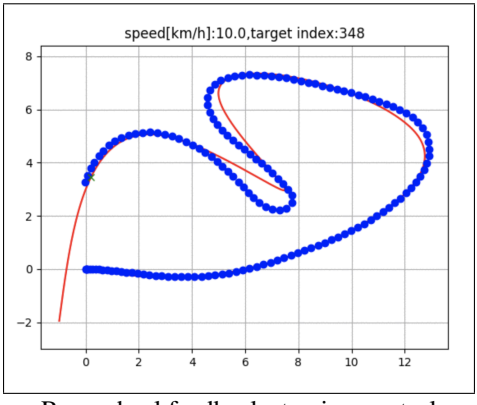
Figure 5: Path planning simulation results

4.4 Path planning

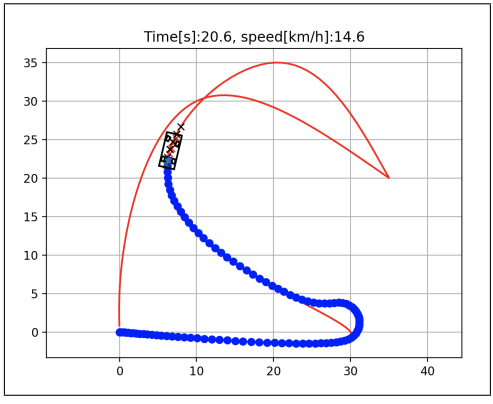
Path planning is the ability of a robot to search feasible and efficient path to the goal. The path have to satisfy some constraints based on movement model and obstacle positions, and optimize some objective functions such as time to goal and distance to obstacle etc. In path planning, dynamic programming 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 ability of a robot to follow the reference path generated by path planning algorithms. The role of the path tracking controller is to stabilize to the reference path or trajectory which has 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[**?**].



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. 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 works of this project are as followed:

- Adding technical and mathematical documentation using Jupyter notebook[?].
- Adding simple image processing samples for autonomous navigation only using OpenCV[?].
- Adding simple multi-robots simulations for autonomous navigation.

If readers were 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.