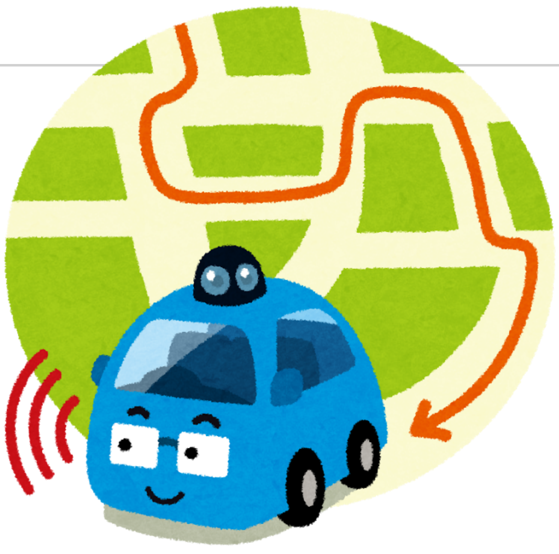
PythonRobotics



Linux_CI passing

MacOS_CI passing

build passing

docs failing

build passing

codecov 90%

code quality: python A+

codefactor A

total lines 32.9K

Python codes for robotics algorithm.

Table of Contents

What is this?

This is a Python code collection of robotics algorithms.

Features:

1. Easy to read for understanding each algorithm's basic idea.
2. Widely used and practical algorithms are selected.
3. Minimum dependency.

See this paper for more details:

- [1808.10703] PythonRobotics: a Python code collection of robotics algorithms (BibTeX)

Requirements

For running each sample code:

- Python 3.9.x
- numpy
- scipy
- matplotlib
- pandas
- cvxpy

For development:

- pytest (for unit tests)
- pytest-xdist (for parallel unit tests)
- mypy (for type check)
- Sphinx (for document generation)
- pycodestyle (for code style check)

Documentation

This README only shows some examples of this project.

If you are interested in other examples or mathematical backgrounds of each algorithm,

You can check the full documentation online: https://pythonrobotics.readthedocs.io/

All animation gifs are stored here: AtsushiSakai/PythonRoboticsGifs: Animation gifs of PythonRobotics

How to use

1. Clone this repo.

```
git clone https://github.com/AtsushiSakai/PythonRobotics.git
```

2. Install the required libraries.

using conda :

```
conda env create -f environment.yml
```

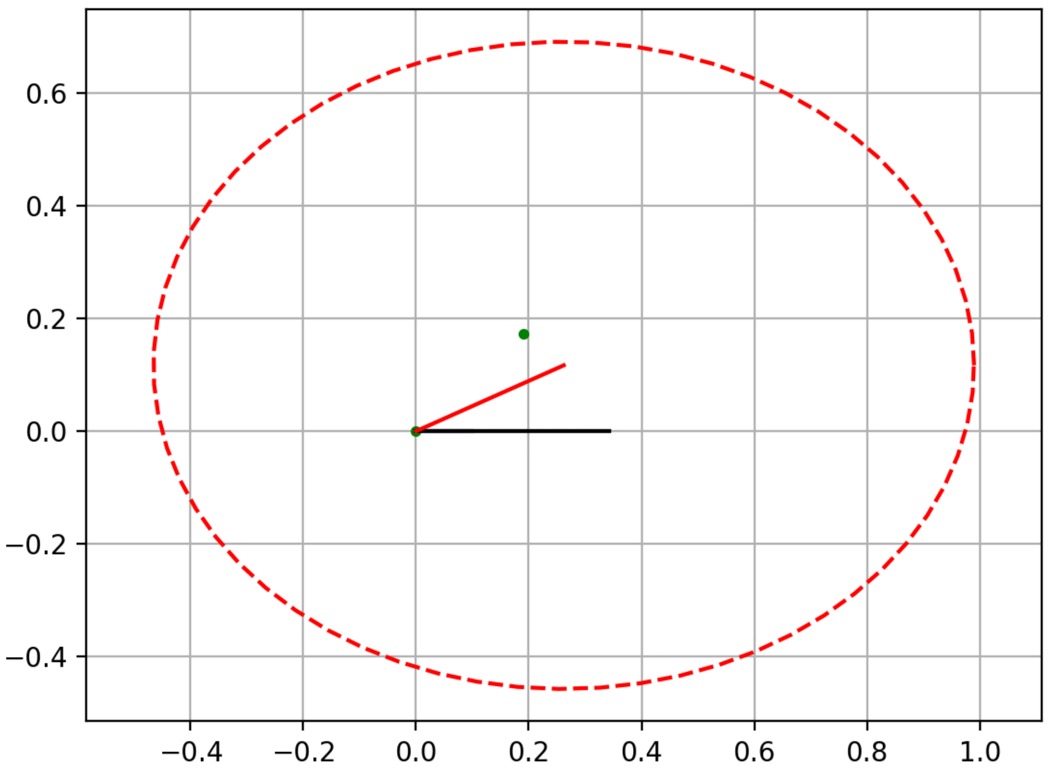
using pip :

```
pip install -r requirements.txt
```

3. Execute python script in each directory.
4. Add star to this repo if you like it 😃.

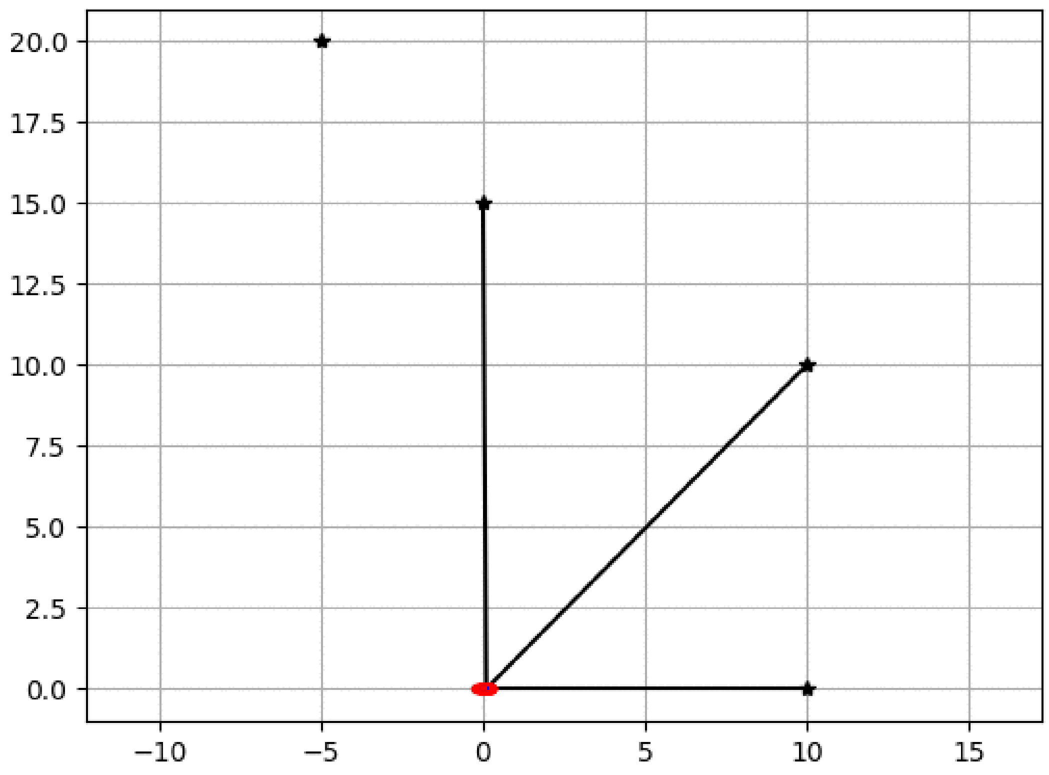
Localization

Extended Kalman Filter localization



Documentation: Notebook

Particle filter localization



This is a sensor fusion localization with Particle Filter(PF).

The blue line is true trajectory, the black line is dead reckoning trajectory,

and the red line is an estimated trajectory with PF.

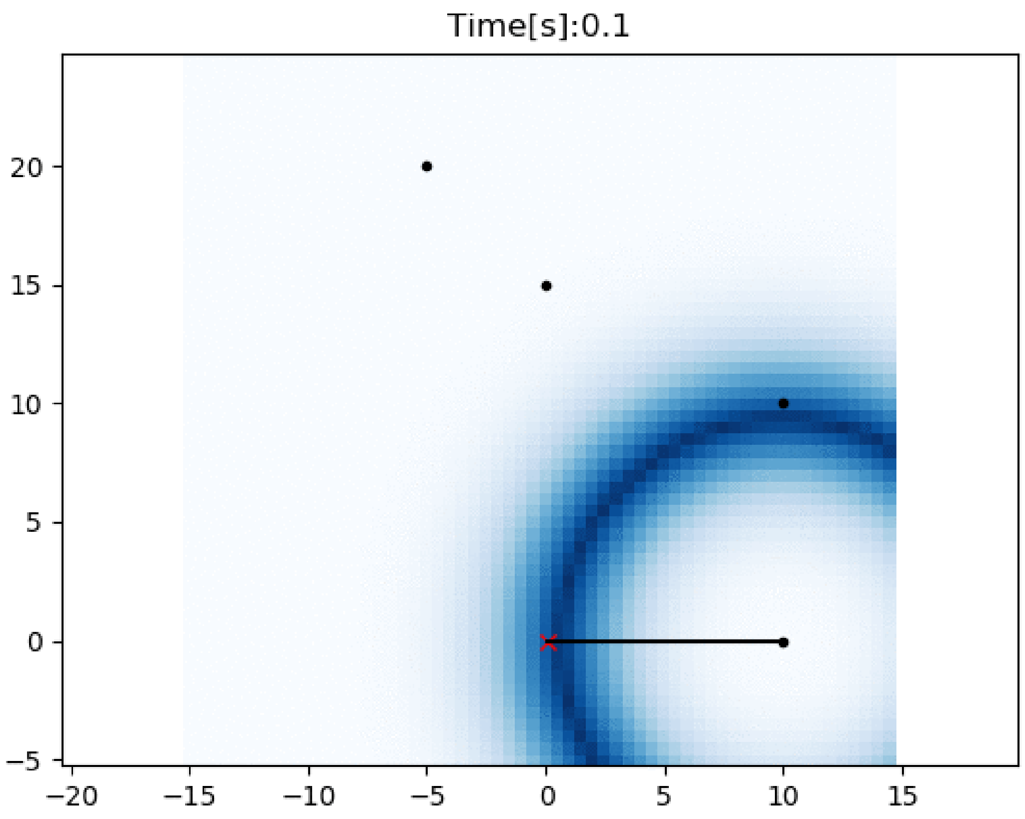
It is assumed that the robot can measure a distance from landmarks (RFID).

These measurements are used for PF localization.

Ref:

- PROBABILISTIC ROBOTICS

Histogram filter localization



This is a 2D localization example with Histogram filter.

The red cross is true position, black points are RFID positions.

The blue grid shows a position probability of histogram filter.

In this simulation, x,y are unknown, yaw is known.

The filter integrates speed input and range observations from RFID for localization.

Initial position is not needed.

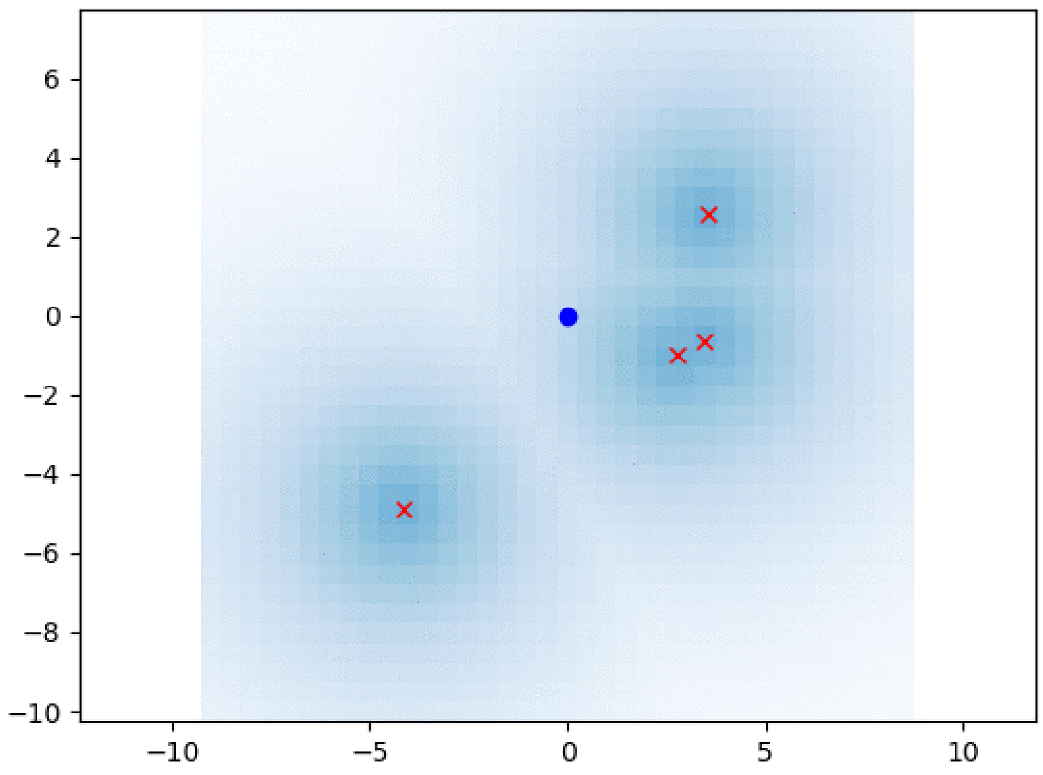
Ref:

- PROBABILISTIC ROBOTICS

Mapping

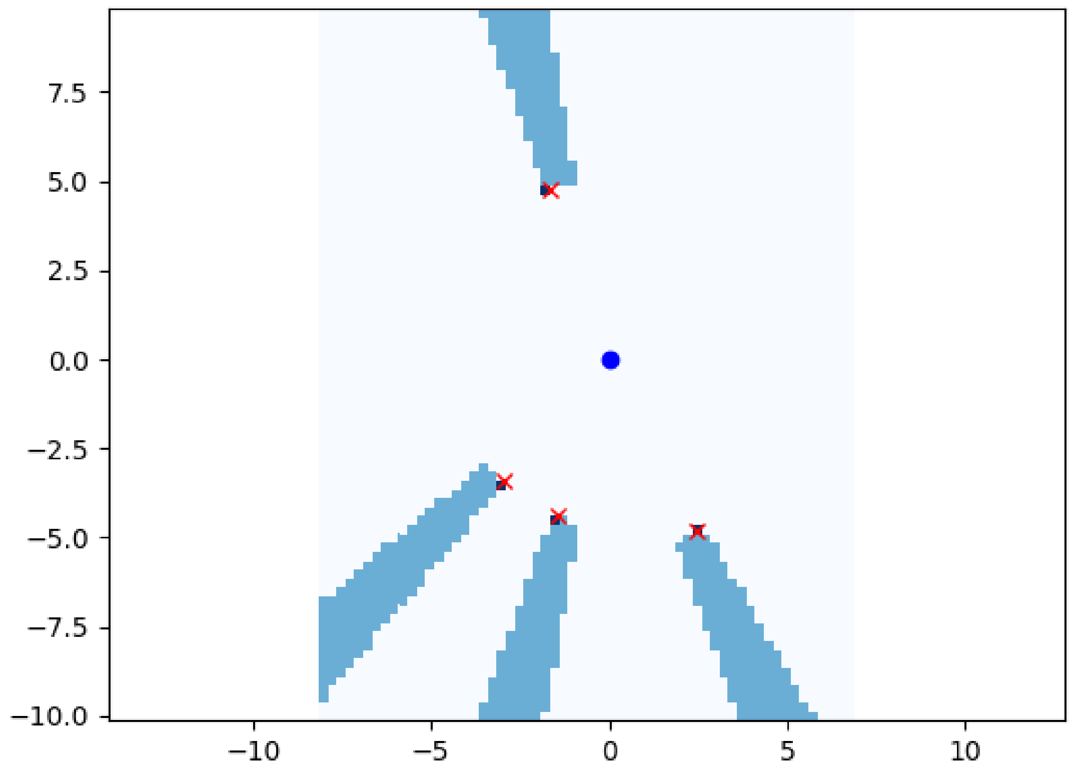
Gaussian grid map

This is a 2D Gaussian grid mapping example.



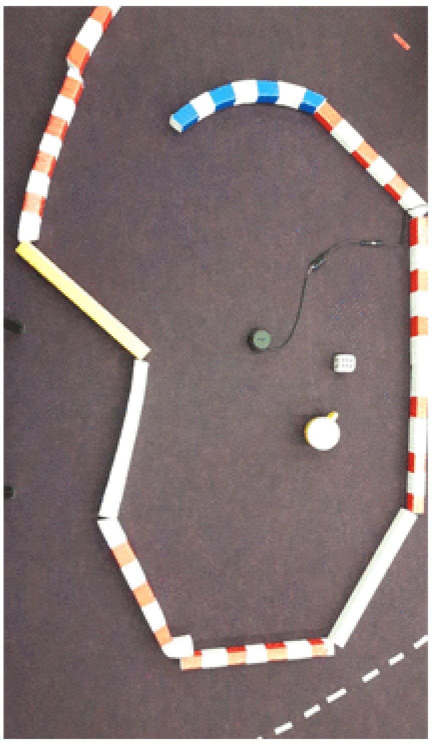
Ray casting grid map

This is a 2D ray casting grid mapping example.



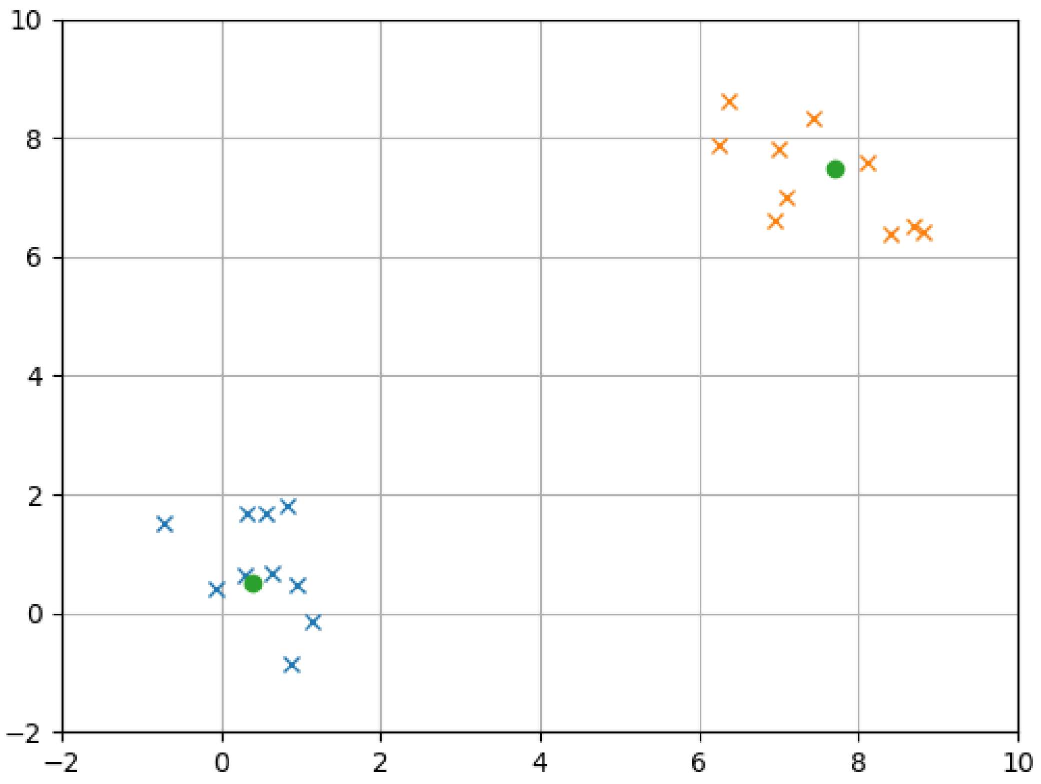
Lidar to grid map

This example shows how to convert a 2D range measurement to a grid map.



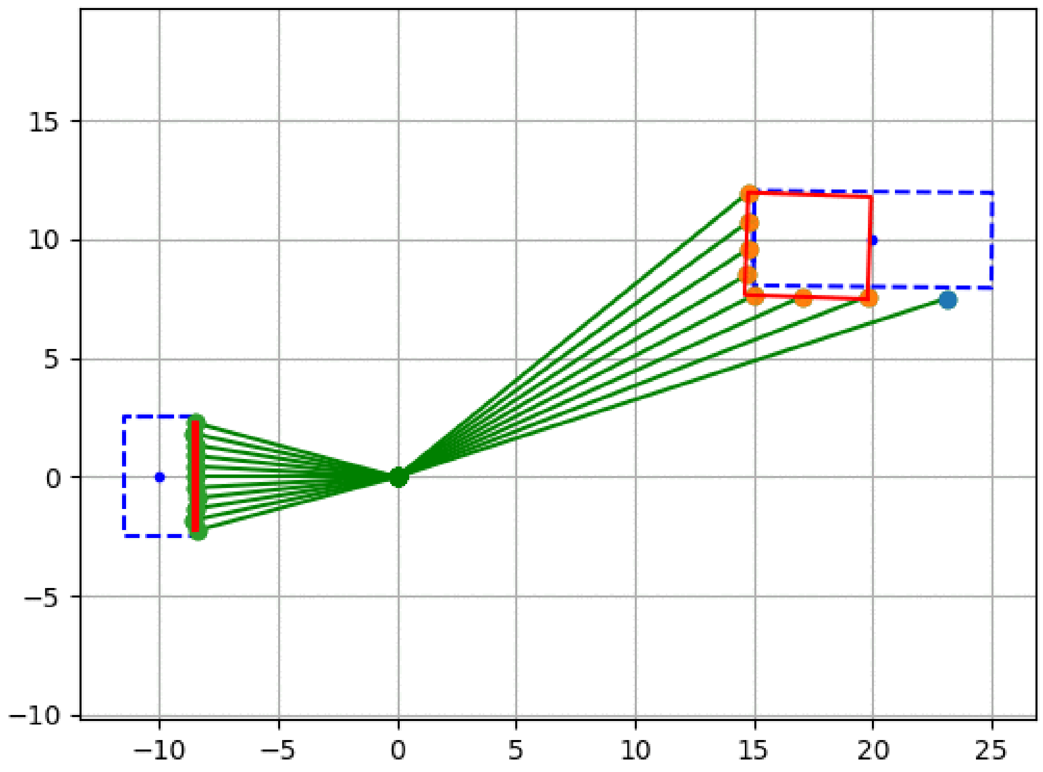
k-means object clustering

This is a 2D object clustering with k-means algorithm.



Rectangle fitting

This is a 2D rectangle fitting for vehicle detection.



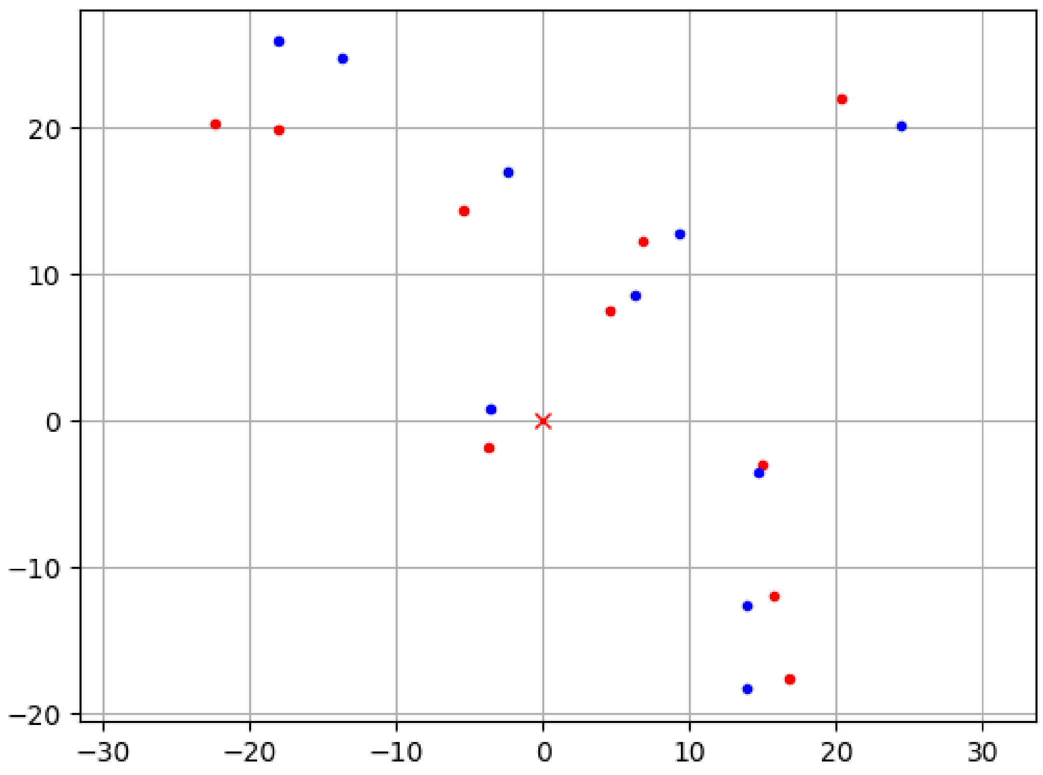
SLAM

Simultaneous Localization and Mapping(SLAM) examples

Iterative Closest Point (ICP) Matching

This is a 2D ICP matching example with singular value decomposition.

It can calculate a rotation matrix, and a translation vector between points and points.



Ref:

- Introduction to Mobile Robotics: Iterative Closest Point Algorithm

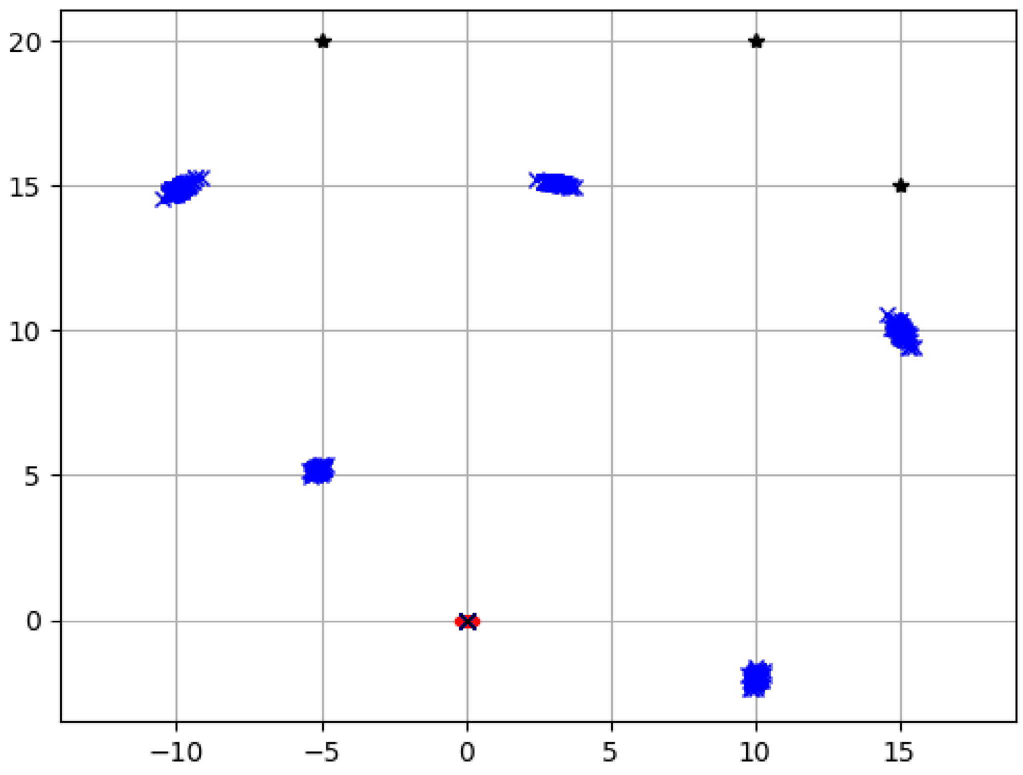
FastSLAM 1.0

This is a feature based SLAM example using FastSLAM 1.0.

The blue line is ground truth, the black line is dead reckoning, the red line is the estimated trajectory with FastSLAM.

The red points are particles of FastSLAM.

Black points are landmarks, blue crosses are estimated landmark positions by FastSLAM.



Ref:

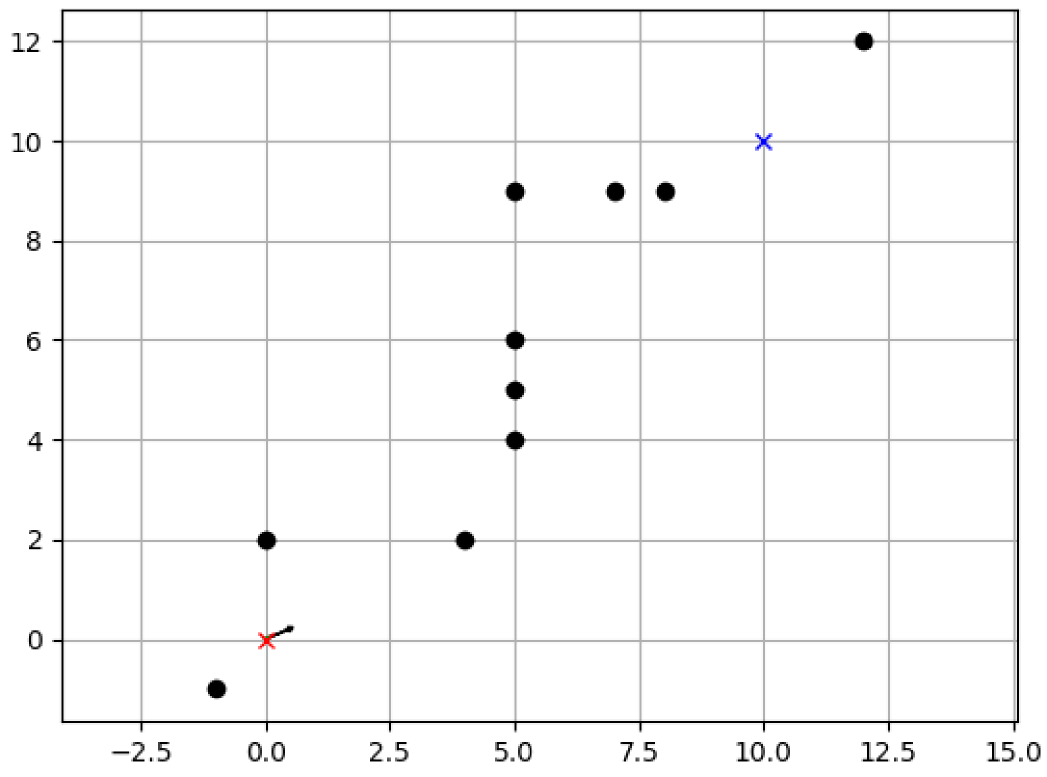
- PROBABILISTIC ROBOTICS
- SLAM simulations by Tim Bailey

Path Planning

Dynamic Window Approach

This is a 2D navigation sample code with Dynamic Window Approach.

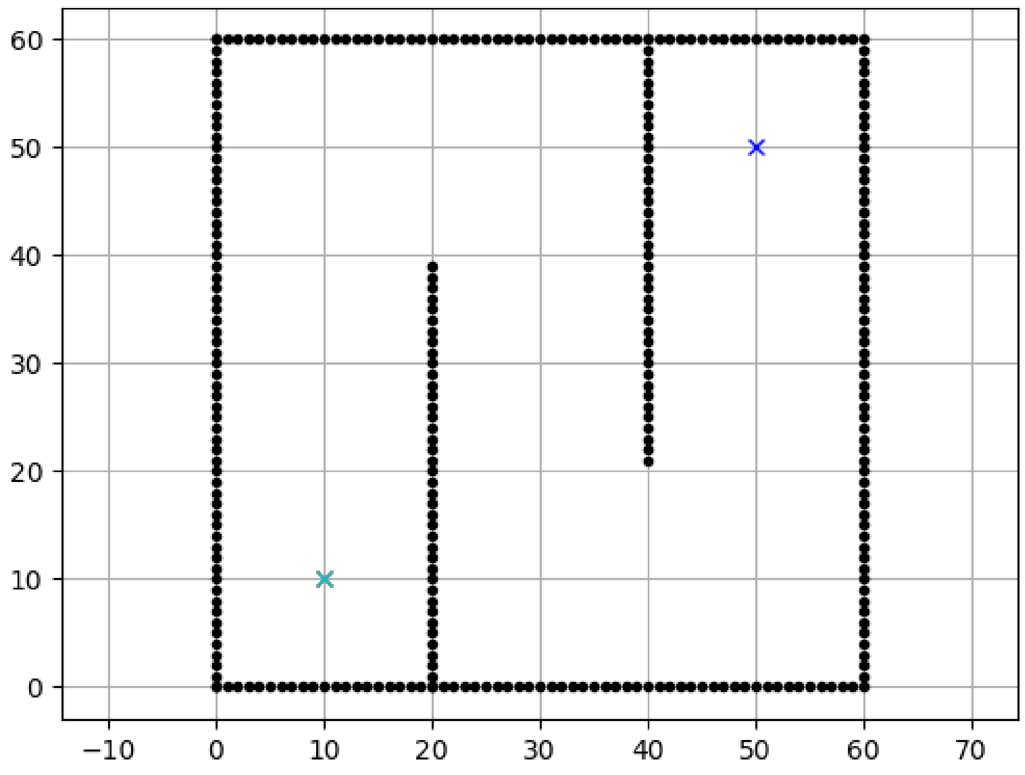
- The Dynamic Window Approach to Collision Avoidance



Grid based search

Dijkstra algorithm

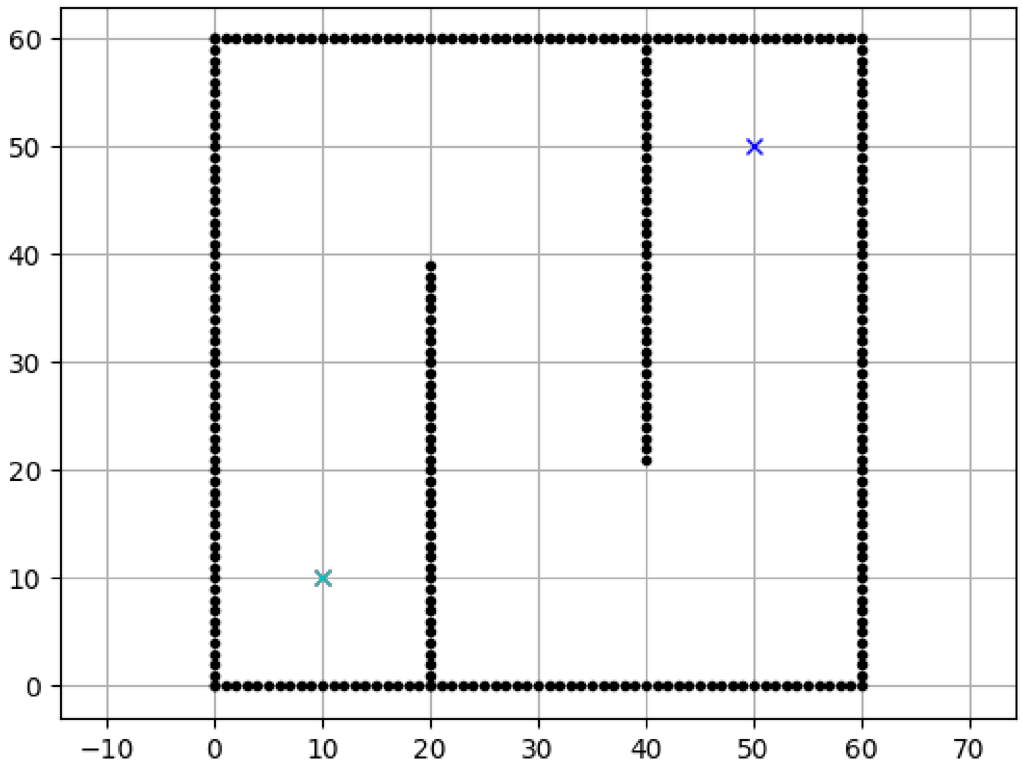
This is a 2D grid based the shortest path planning with Dijkstra's algorithm.



In the animation, cyan points are searched nodes.

A* algorithm

This is a 2D grid based the shortest path planning with A star algorithm.

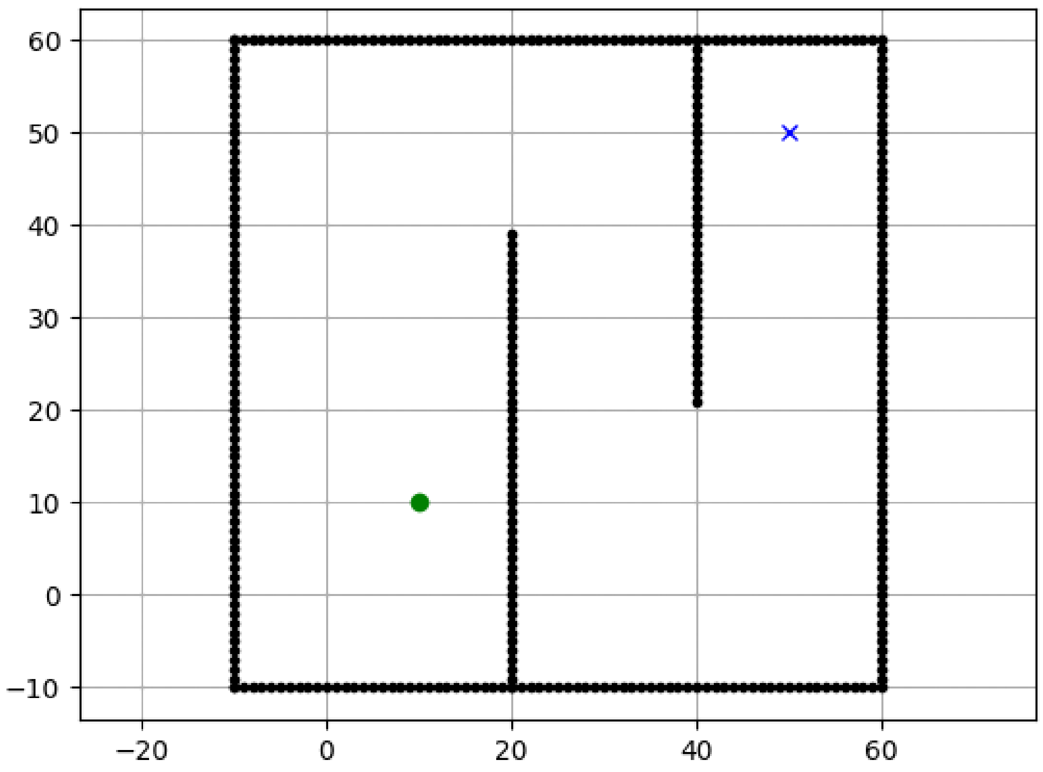


In the animation, cyan points are searched nodes.

Its heuristic is 2D Euclid distance.

D* algorithm

This is a 2D grid based the shortest path planning with D star algorithm.



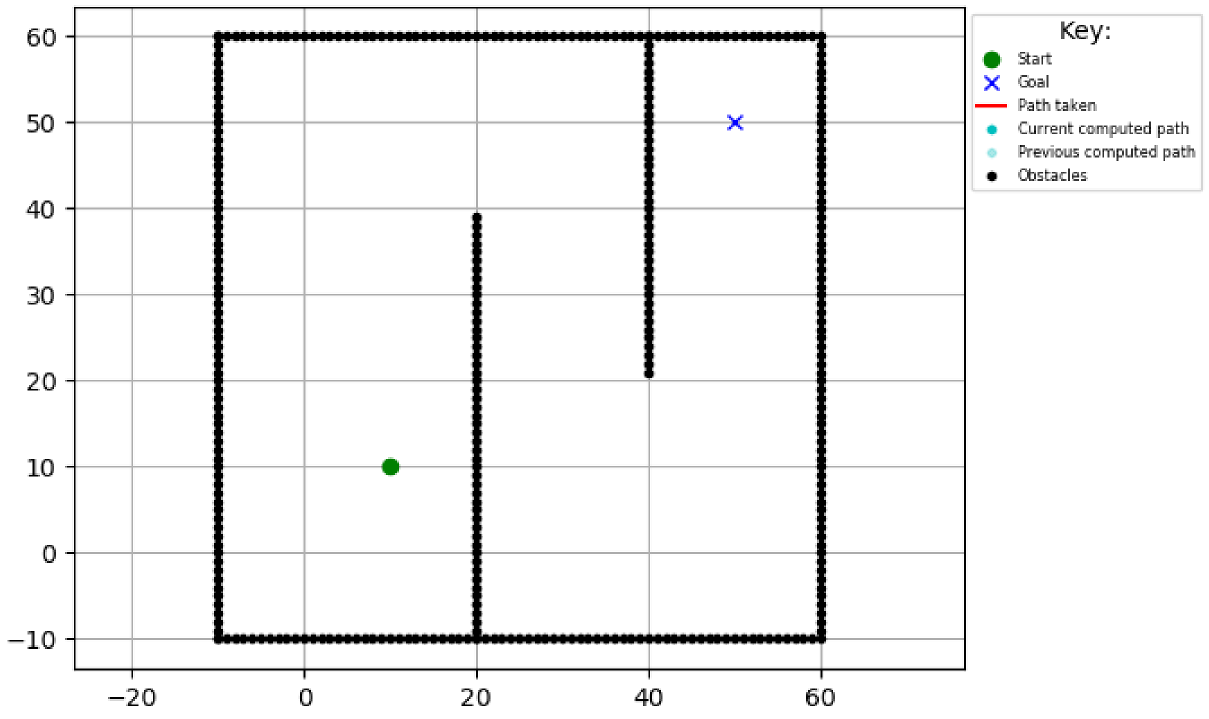
The animation shows a robot finding its path avoiding an obstacle using the D* search algorithm.

Ref:

- D* Algorithm Wikipedia

D* Lite algorithm

This algorithm finds the shortest path between two points while rerouting when obstacles are discovered. It has been implemented here for a 2D grid.



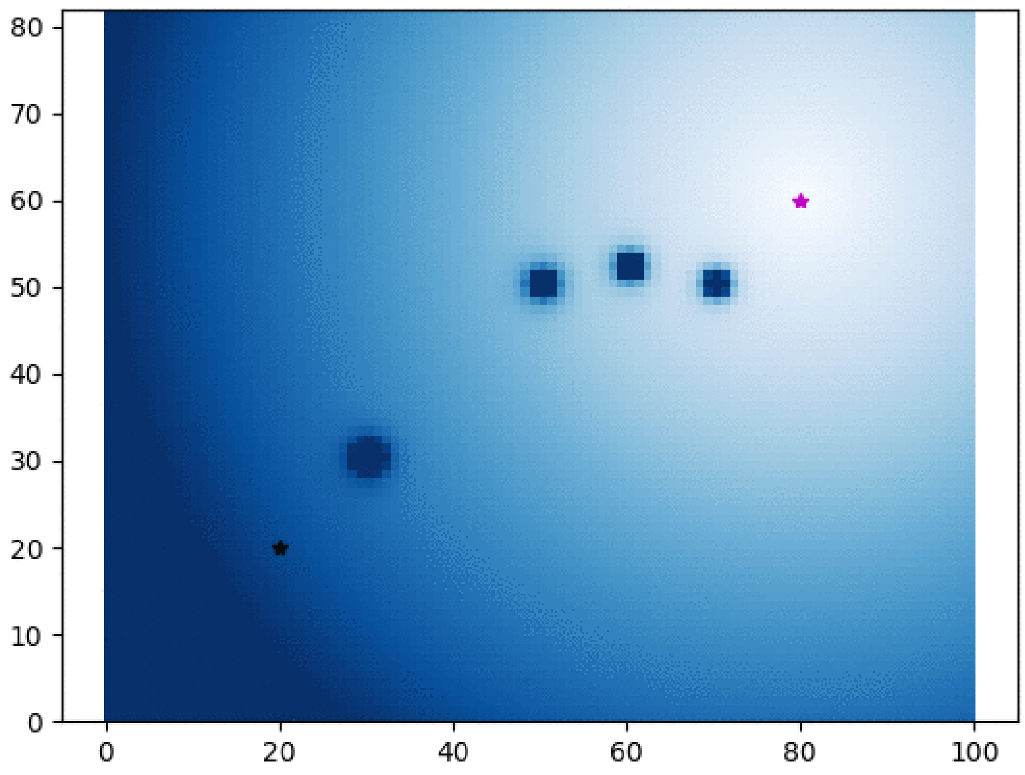
The animation shows a robot finding its path and rerouting to avoid obstacles as they are discovered using the D* Lite search algorithm.

Refs:

- D* Lite
- Improved Fast Replanning for Robot Navigation in Unknown Terrain

Potential Field algorithm

This is a 2D grid based path planning with Potential Field algorithm.



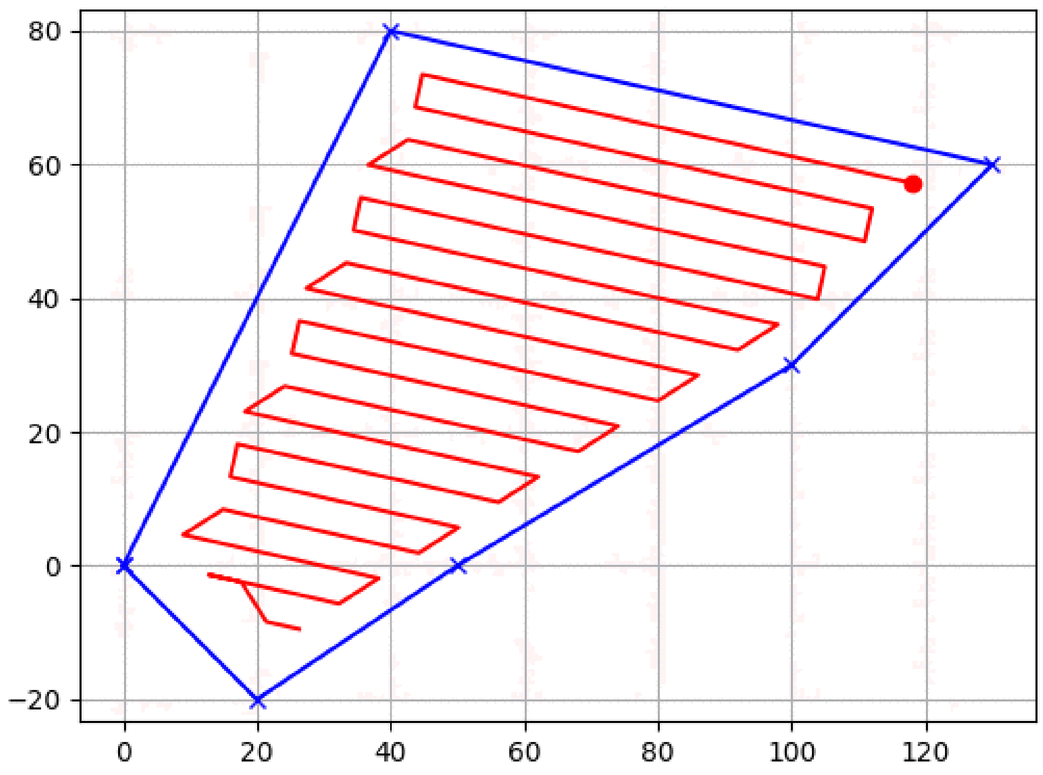
In the animation, the blue heat map shows potential value on each grid.

Ref:

- Robotic Motion Planning:Potential Functions

Grid based coverage path planning

This is a 2D grid based coverage path planning simulation.



State Lattice Planning

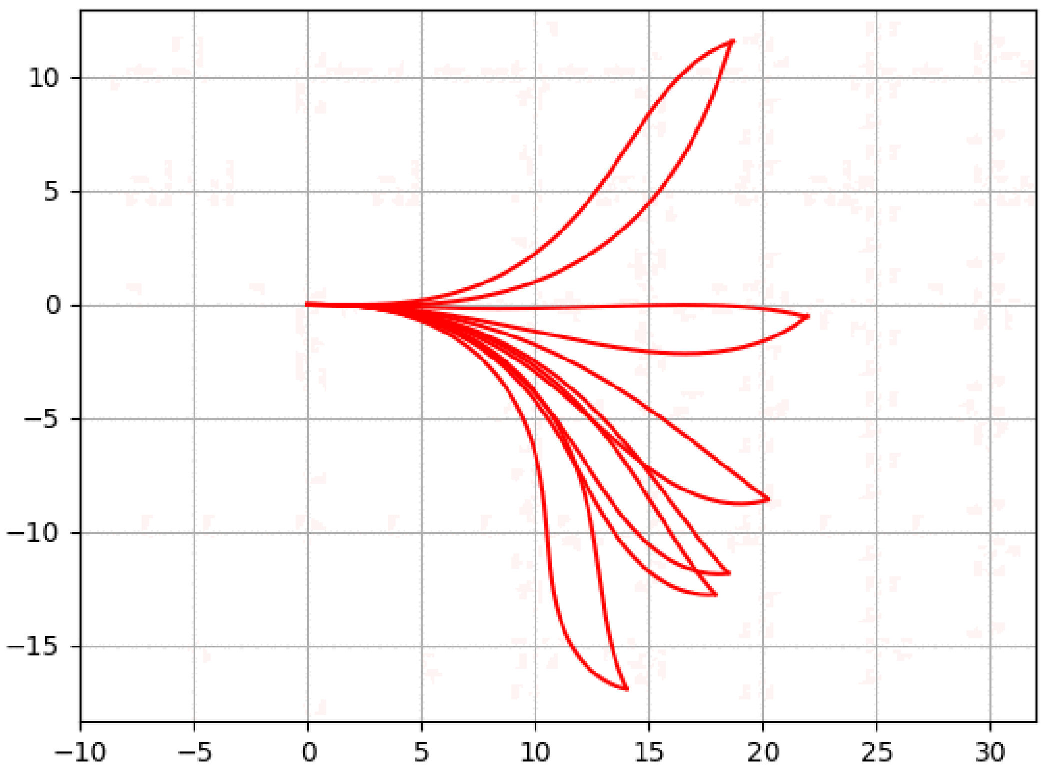
This script is a path planning code with state lattice planning.

This code uses the model predictive trajectory generator to solve boundary problem.

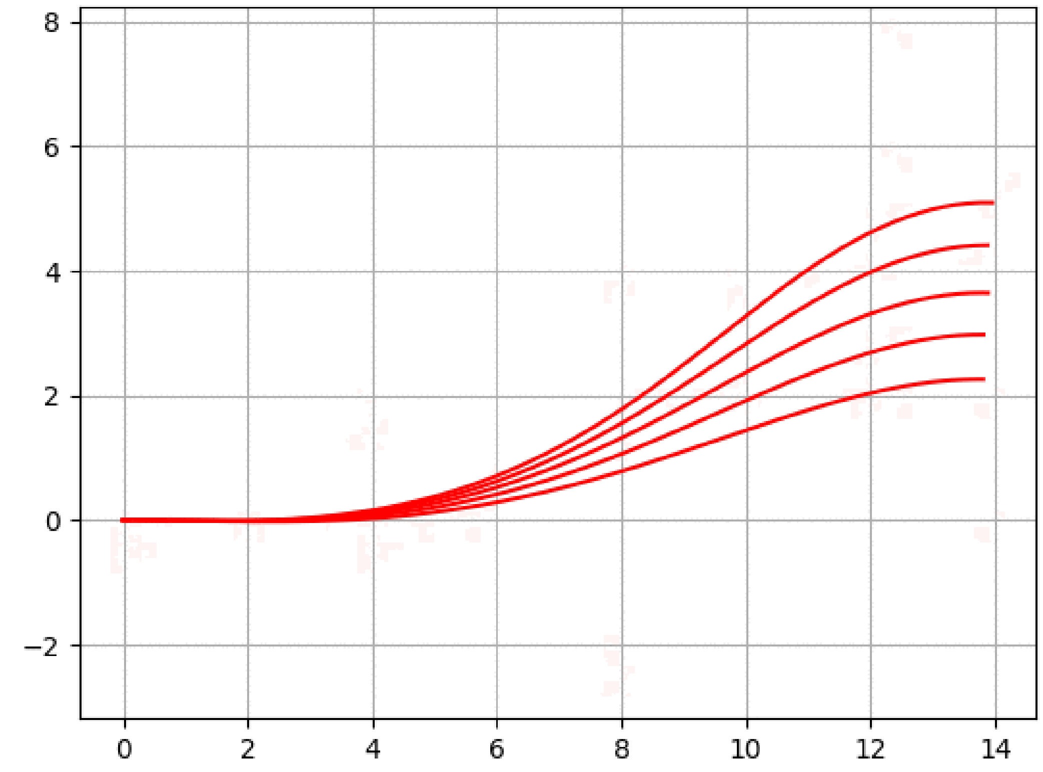
Ref:

- Optimal rough terrain trajectory generation for wheeled mobile robots
- State Space Sampling of Feasible Motions for High-Performance Mobile Robot Navigation in Complex Environments

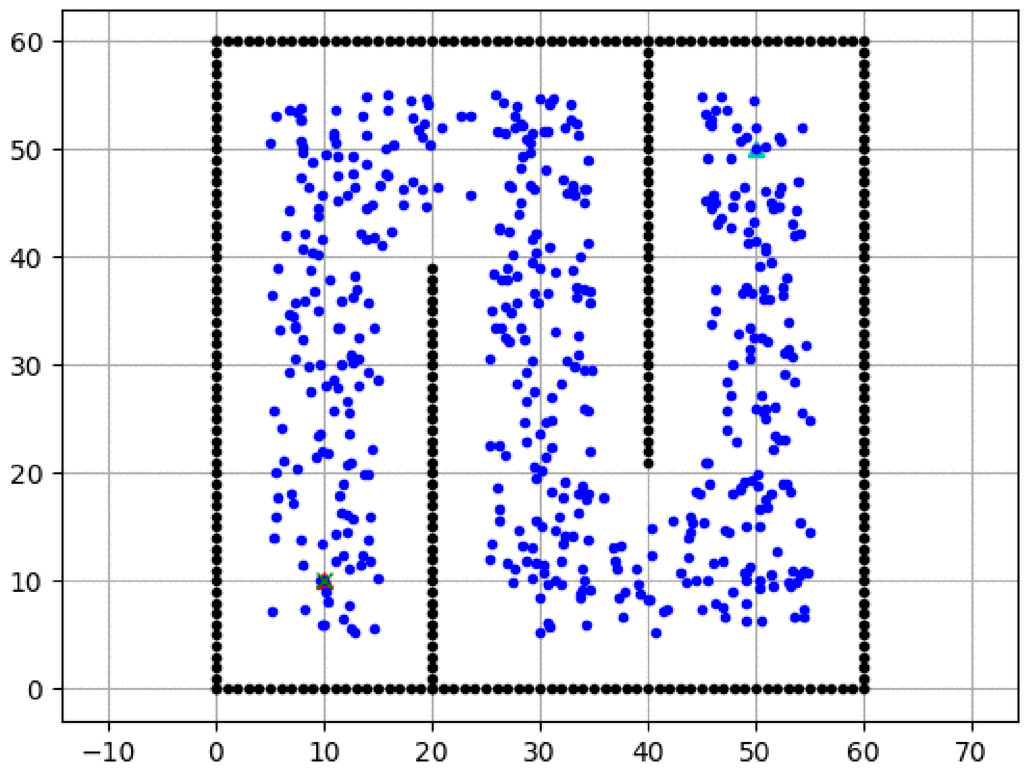
Biased polar sampling



Lane sampling



Probabilistic Road-Map (PRM) planning



This PRM planner uses Dijkstra method for graph search.

In the animation, blue points are sampled points,

Cyan crosses means searched points with Dijkstra method,

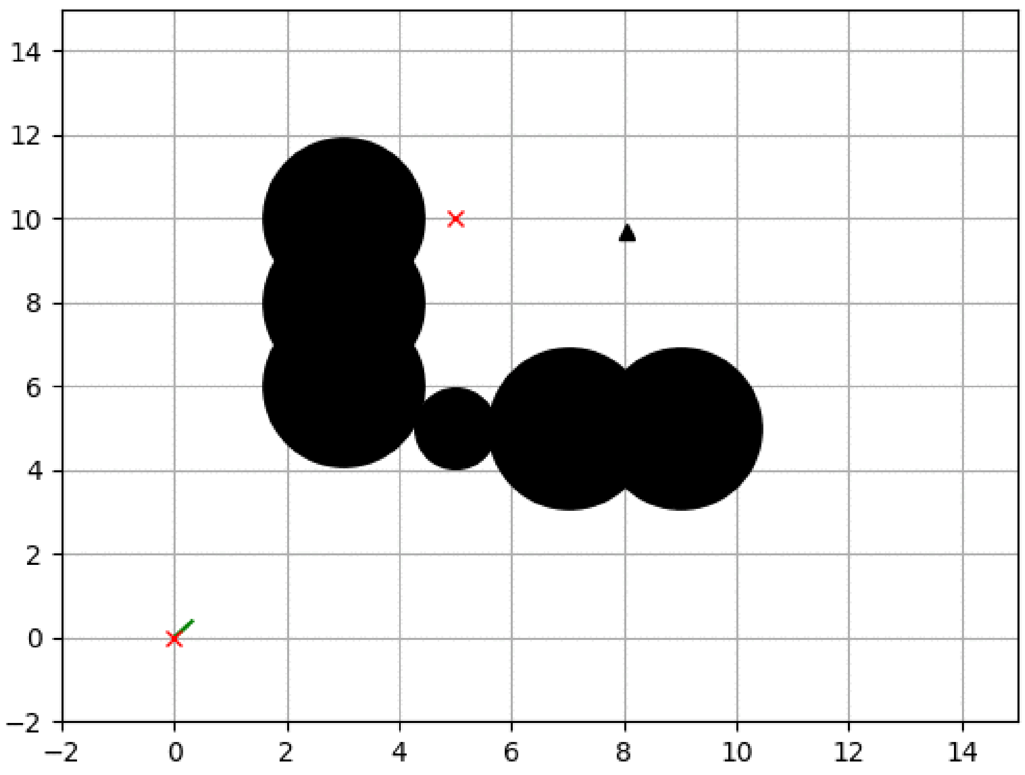
The red line is the final path of PRM.

Ref:

- Probabilistic roadmap - Wikipedia

Rapidly-Exploring Random Trees (RRT)

RRT*



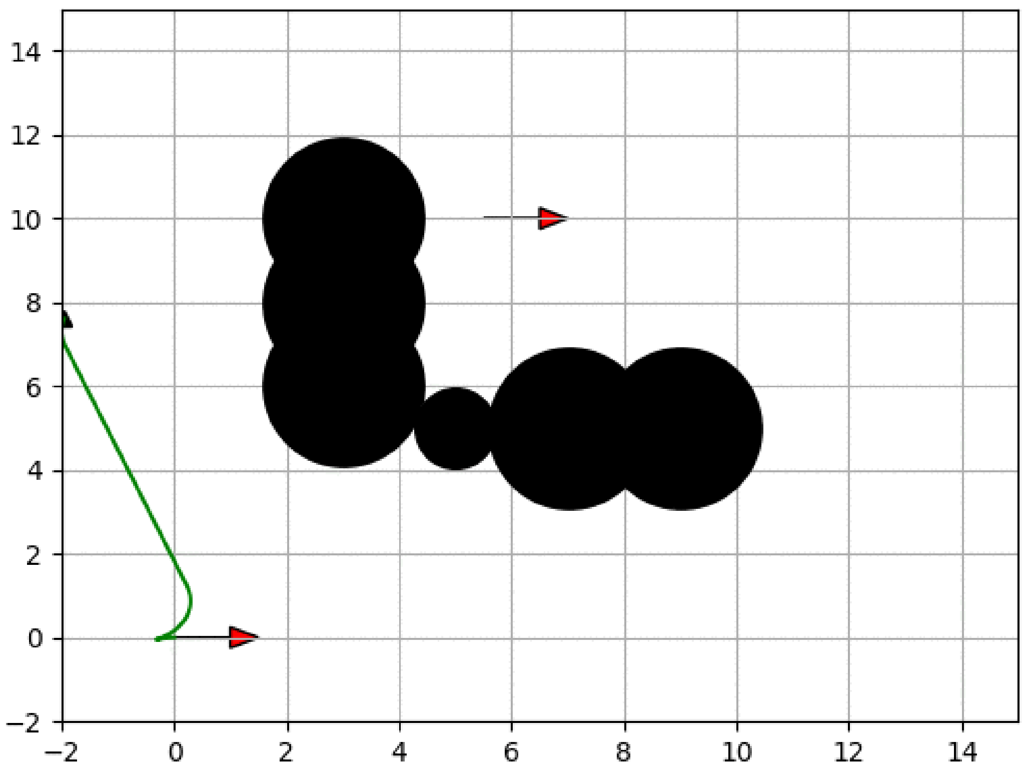
This is a path planning code with RRT*

Black circles are obstacles, green line is a searched tree, red crosses are start and goal positions.

Ref:

- Incremental Sampling-based Algorithms for Optimal Motion Planning
- Sampling-based Algorithms for Optimal Motion Planning

RRT* with reeds-shepp path



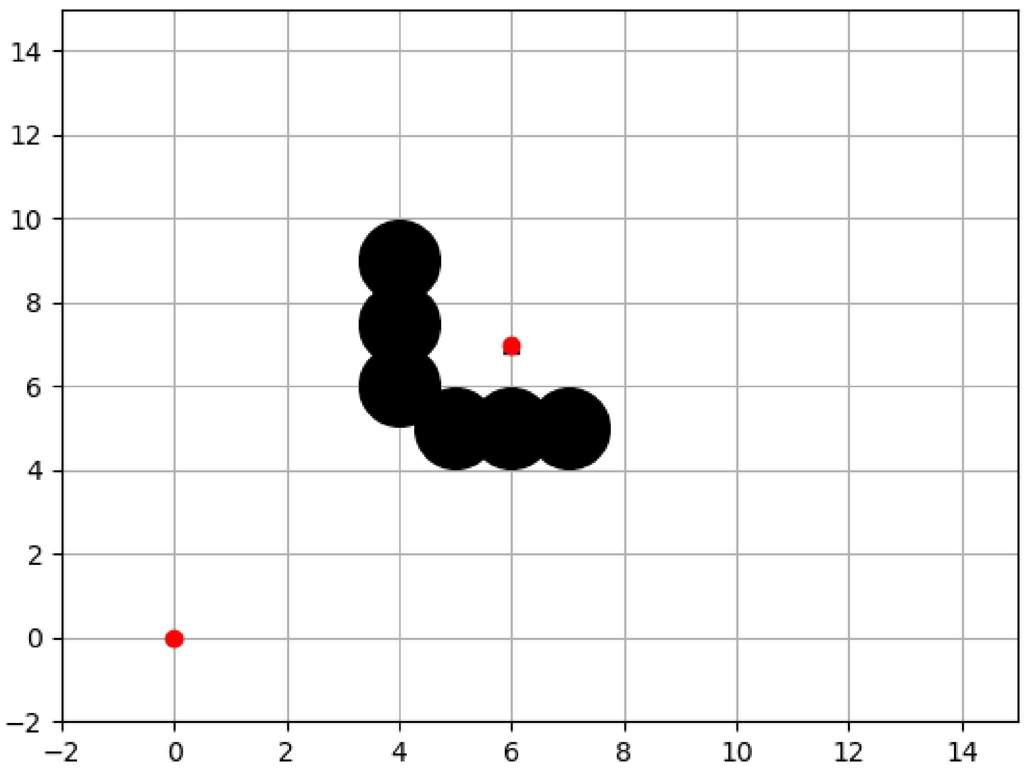
)

Path planning for a car robot with RRT* and reeds shepp path planner.

LQR-RRT*

This is a path planning simulation with LQR-RRT*.

A double integrator motion model is used for LQR local planner.

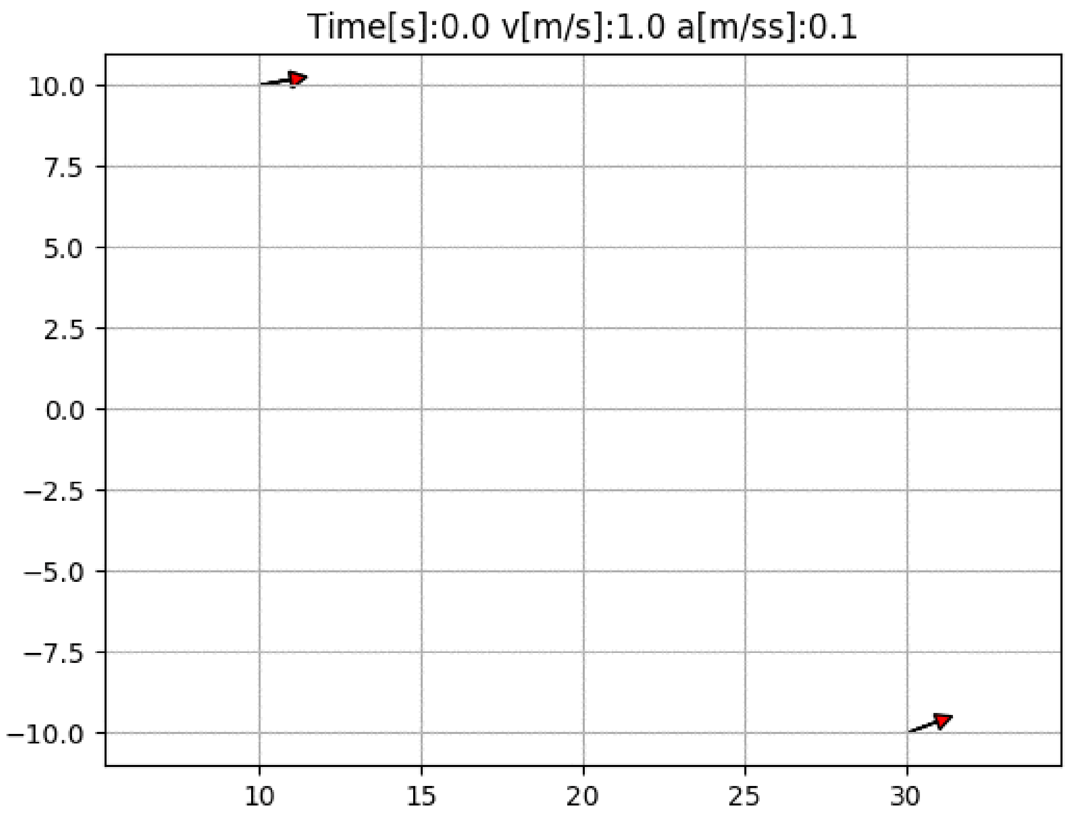


Ref:

- LQR-RRT*: Optimal Sampling-Based Motion Planning with Automatically Derived Extension Heuristics
- MahanFathi/LQR-RRTstar: LQR-RRT* method is used for random motion planning of a simple pendulum in its phase plot

Quintic polynomials planning

Motion planning with quintic polynomials.



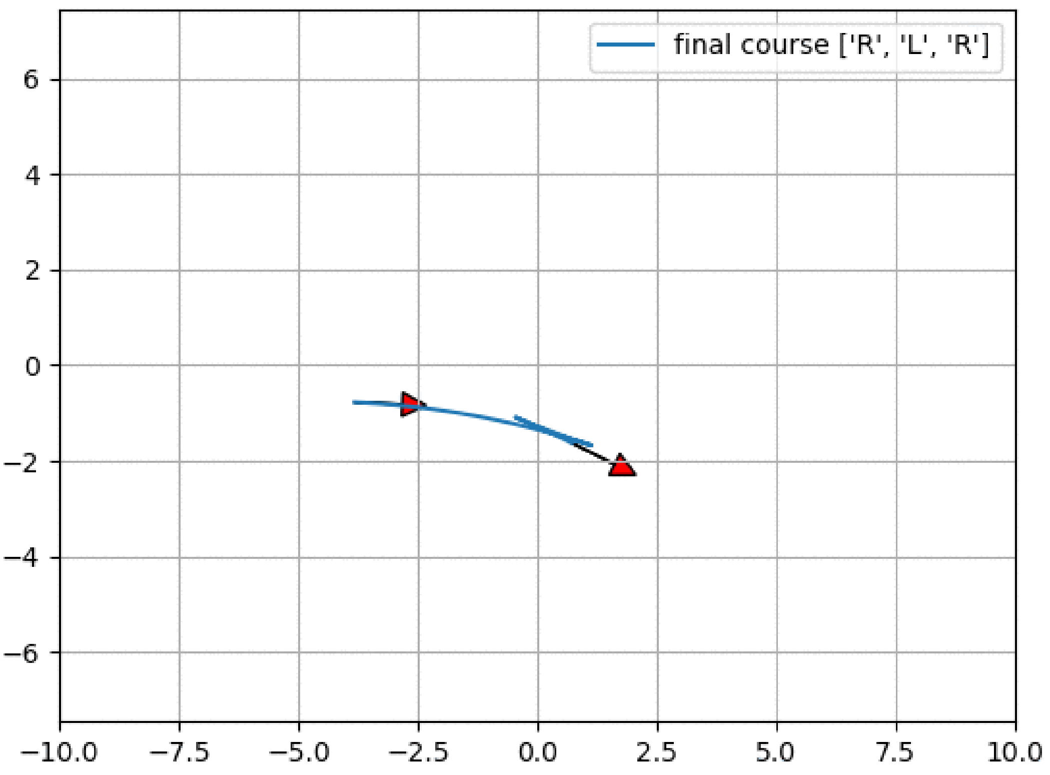
It can calculate a 2D path, velocity, and acceleration profile based on quintic polynomials.

Ref:

- Local Path Planning And Motion Control For Agv In Positioning

Reeds Shepp planning

A sample code with Reeds Shepp path planning.

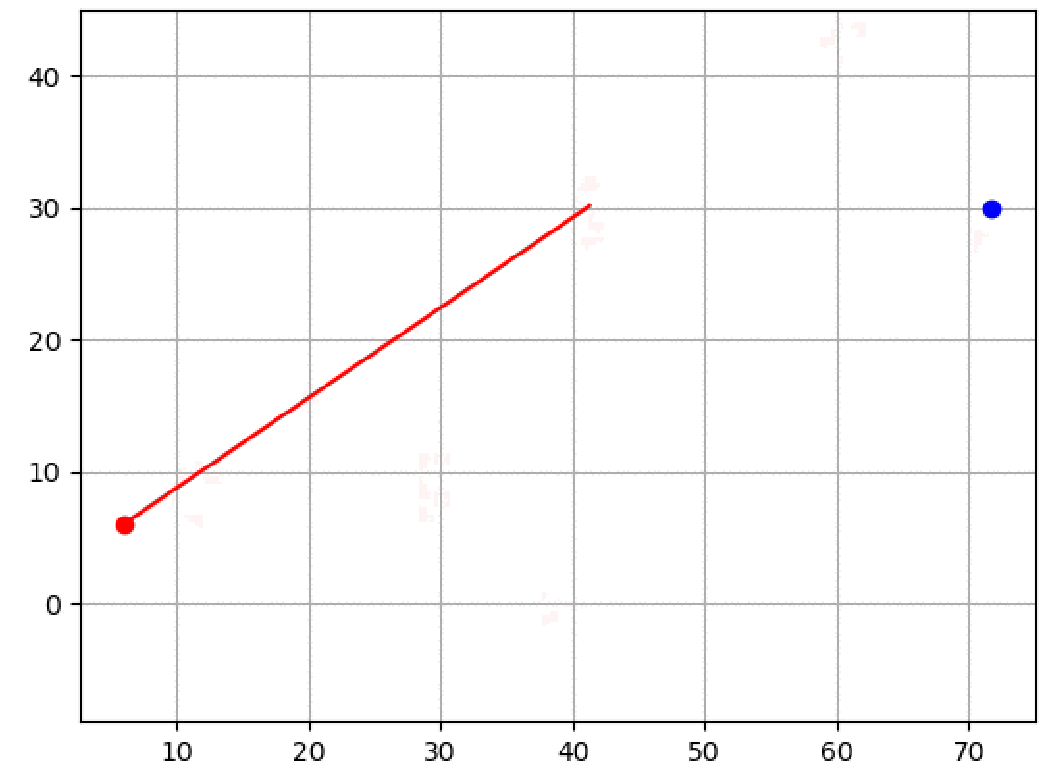


Ref:

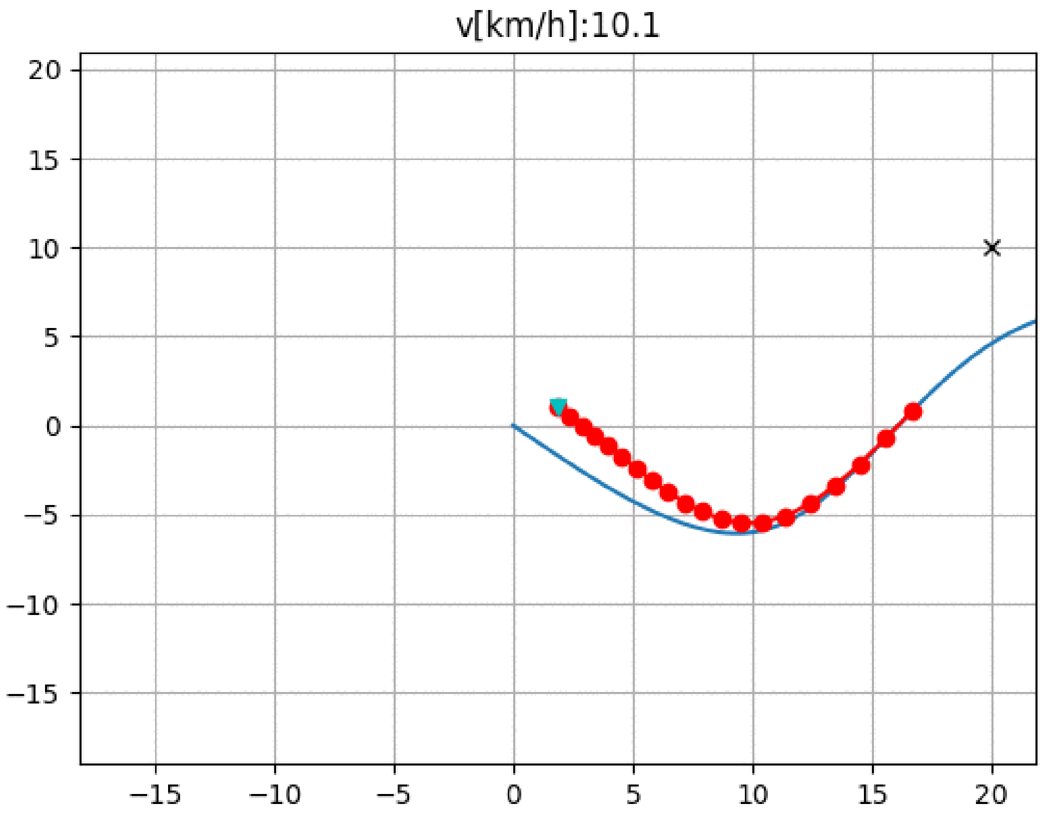
- 15.3.2 Reeds-Shepp Curves
- optimal paths for a car that goes both forwards and backwards
- ghliu/pyReedsShepp: Implementation of Reeds Shepp curve.

LQR based path planning

A sample code using LQR based path planning for double integrator model.



Optimal Trajectory in a Frenet Frame



This is optimal trajectory generation in a Frenet Frame.

The cyan line is the target course and black crosses are obstacles.

The red line is the predicted path.

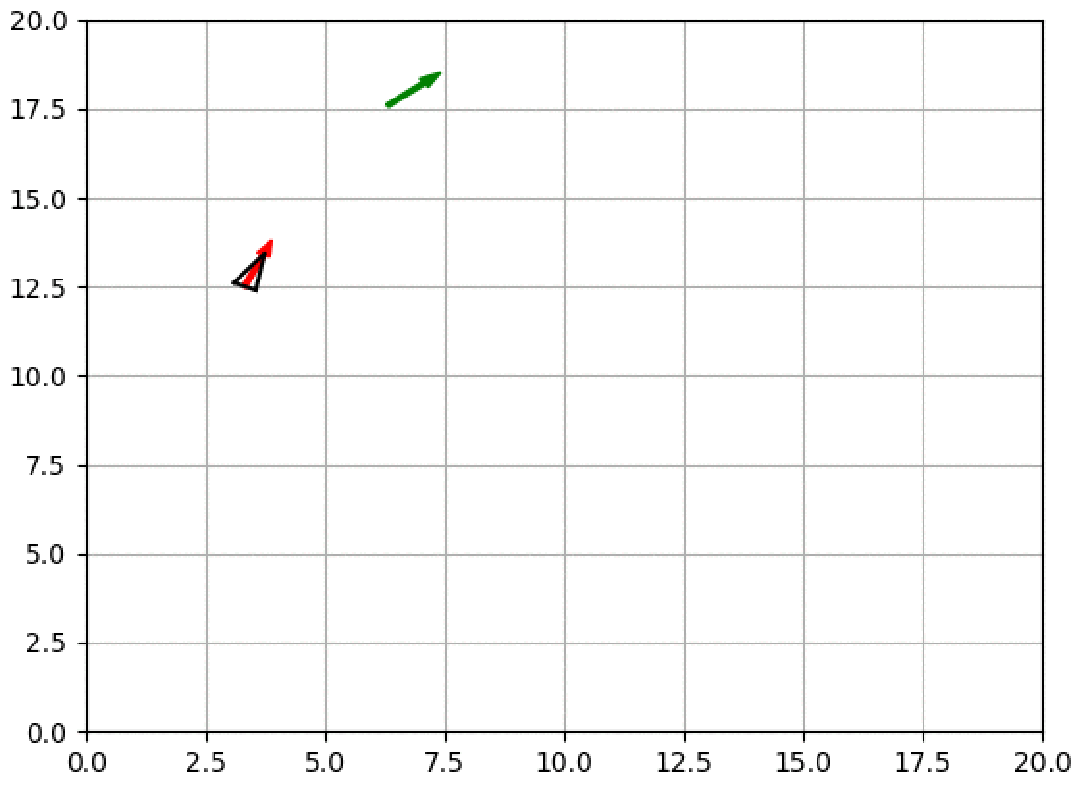
Ref:

- Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame
- Optimal trajectory generation for dynamic street scenarios in a Frenet Frame

Path Tracking

move to a pose control

This is a simulation of moving to a pose control

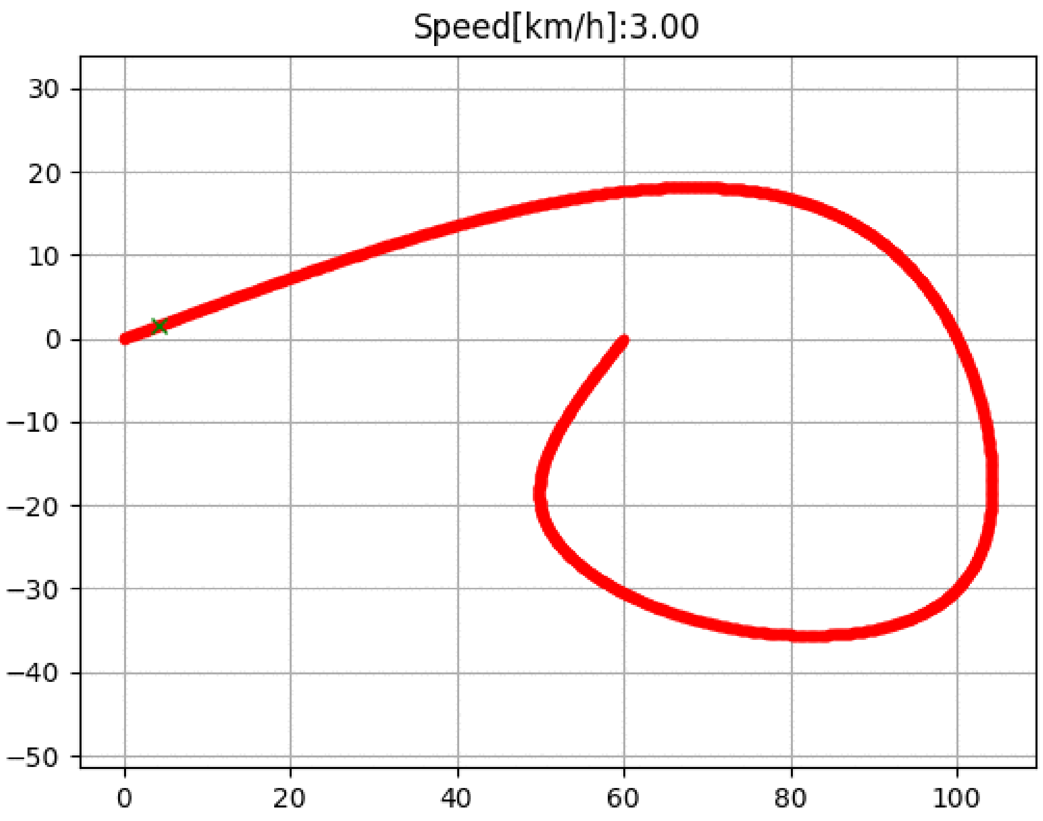


Ref:

- P. I. Corke, "Robotics, Vision and Control" | SpringerLink p102

Stanley control

Path tracking simulation with Stanley steering control and PID speed control.

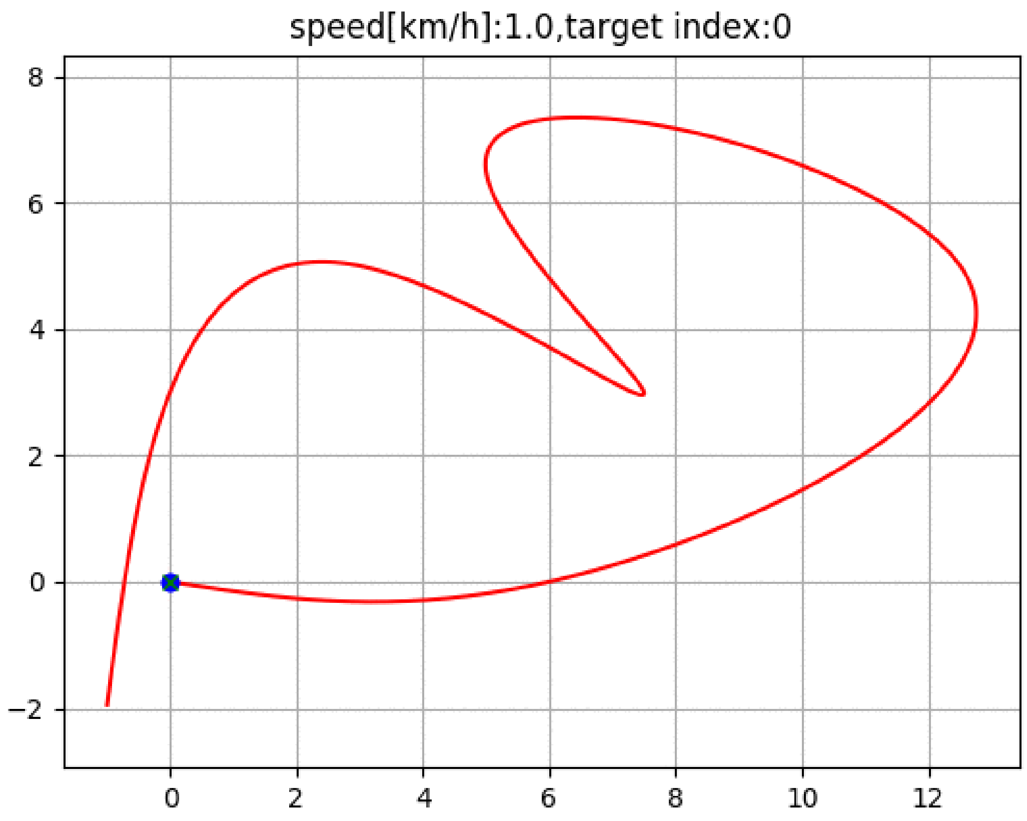


Ref:

- Stanley: The robot that won the DARPA grand challenge
- Automatic Steering Methods for Autonomous Automobile Path Tracking

Rear wheel feedback control

Path tracking simulation with rear wheel feedback steering control and PID speed control.

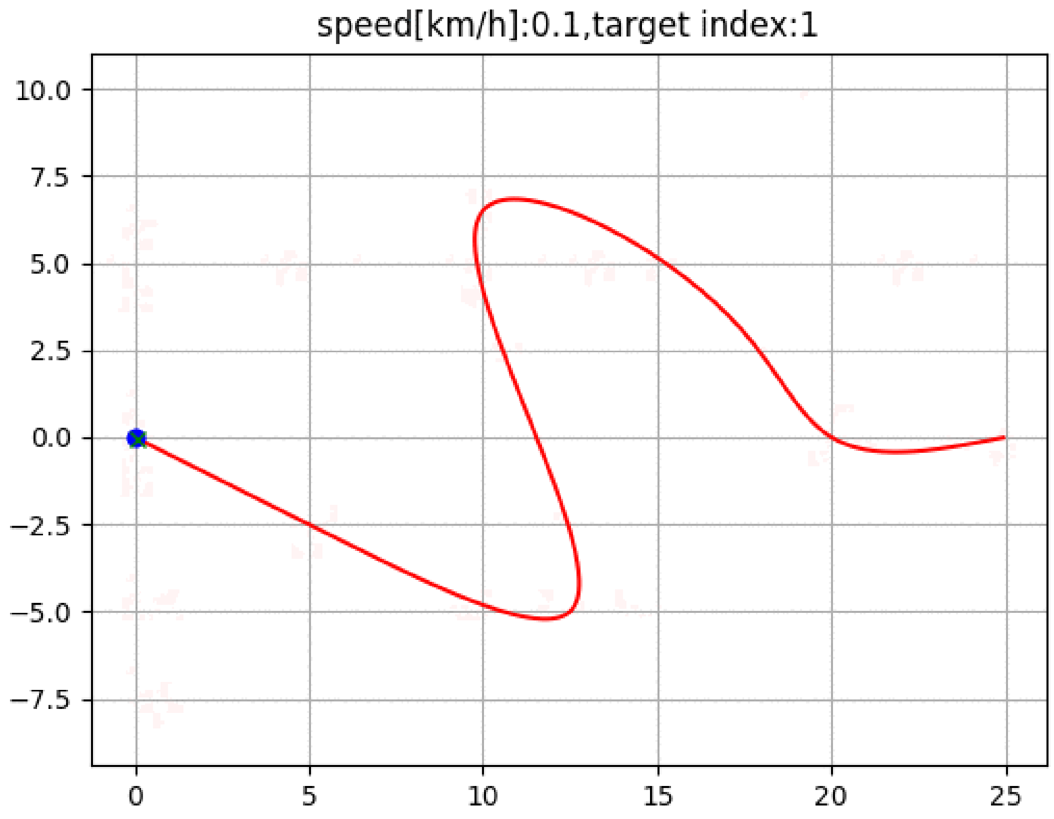


Ref:

- A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles

Linear–quadratic regulator (LQR) speed and steering control

Path tracking simulation with LQR speed and steering control.

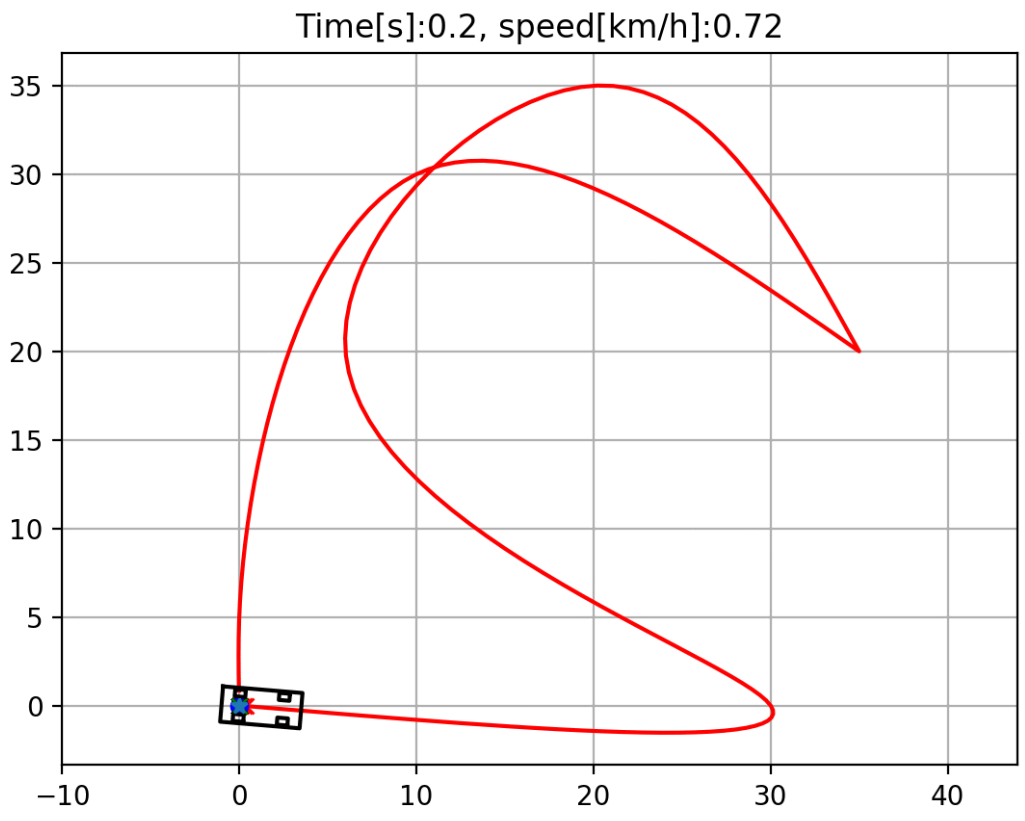


Ref:

- Towards fully autonomous driving: Systems and algorithms - IEEE Conference Publication

Model predictive speed and steering control

Path tracking simulation with iterative linear model predictive speed and steering control.

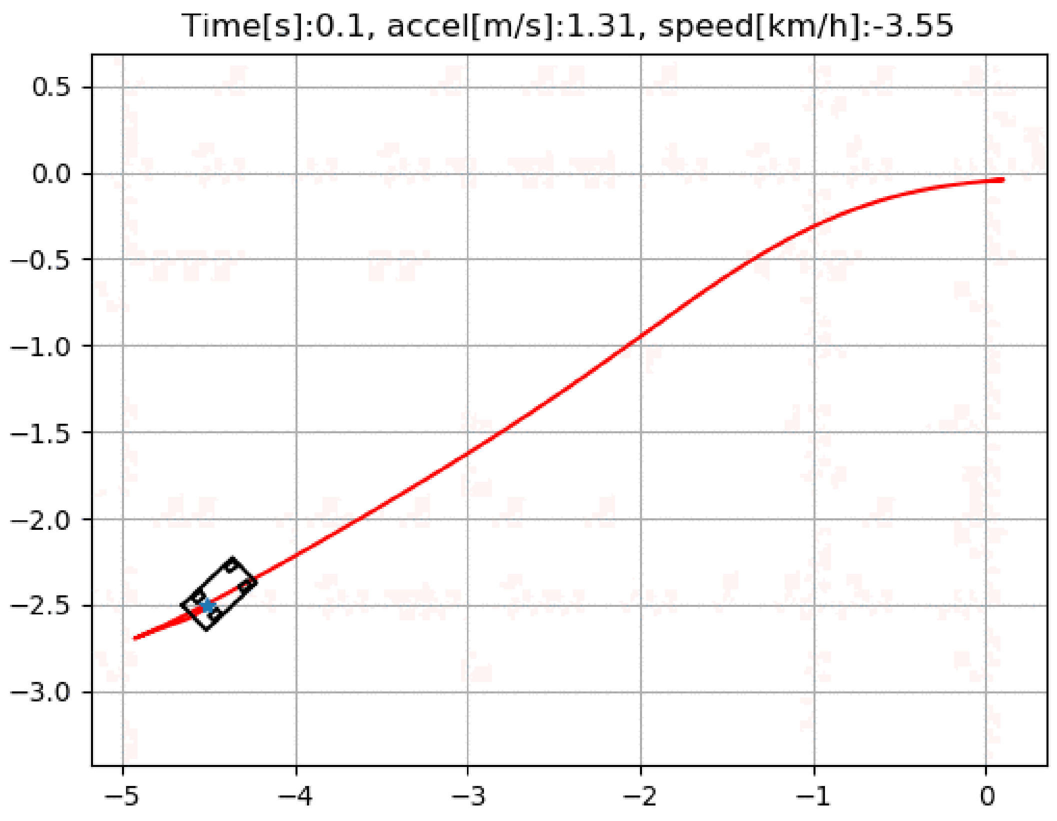


Ref:

- notebook
- Real-time Model Predictive Control (MPC), ACADO, Python | Work-is-Playing

Nonlinear Model predictive control with C-GMRES

A motion planning and path tracking simulation with NMPC of C-GMRES



Ref:

- notebook

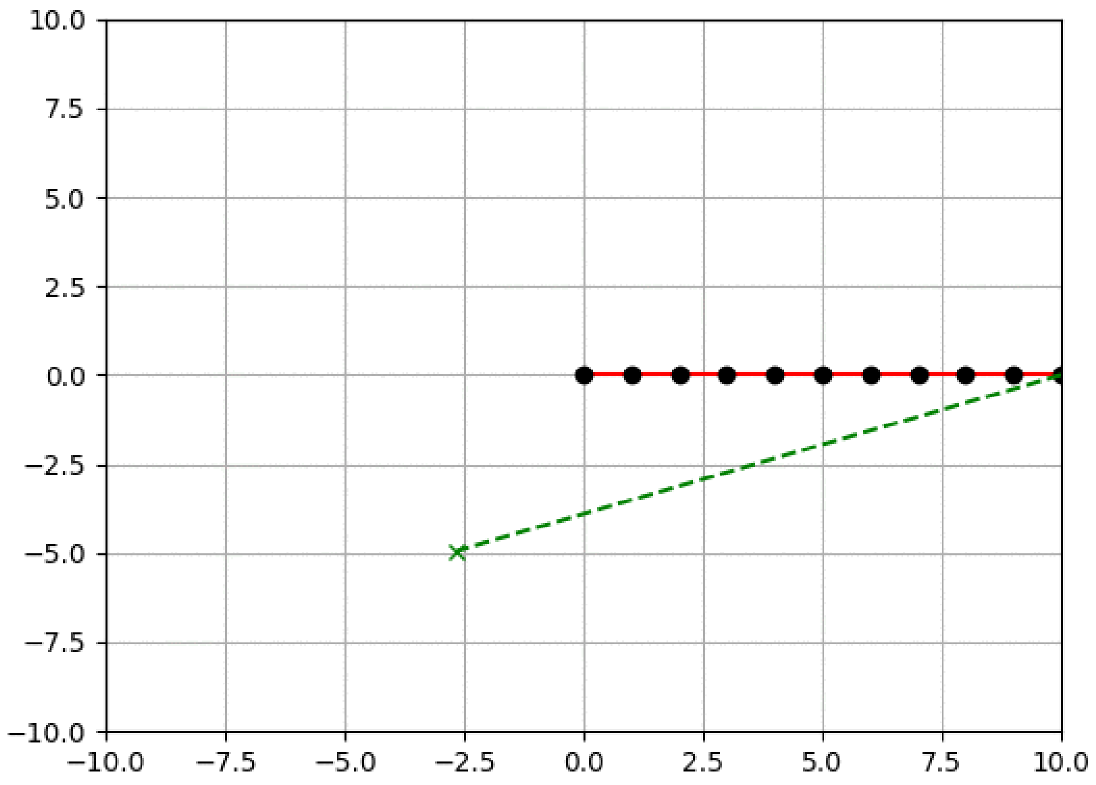
Arm Navigation

N joint arm to point control

N joint arm to a point control simulation.

This is an interactive simulation.

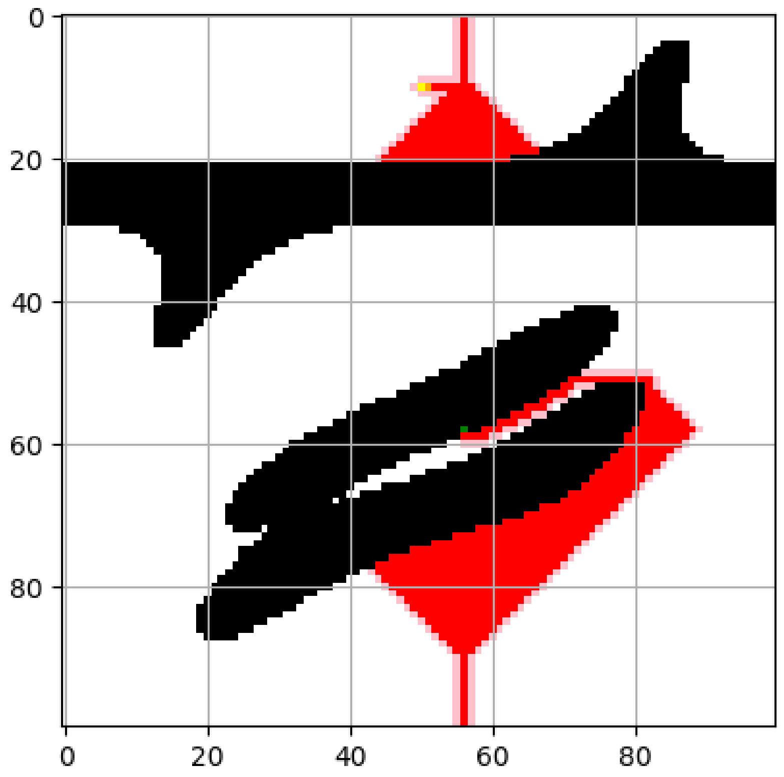
You can set the goal position of the end effector with left-click on the plotting area.



In this simulation N = 10, however, you can change it.

Arm navigation with obstacle avoidance

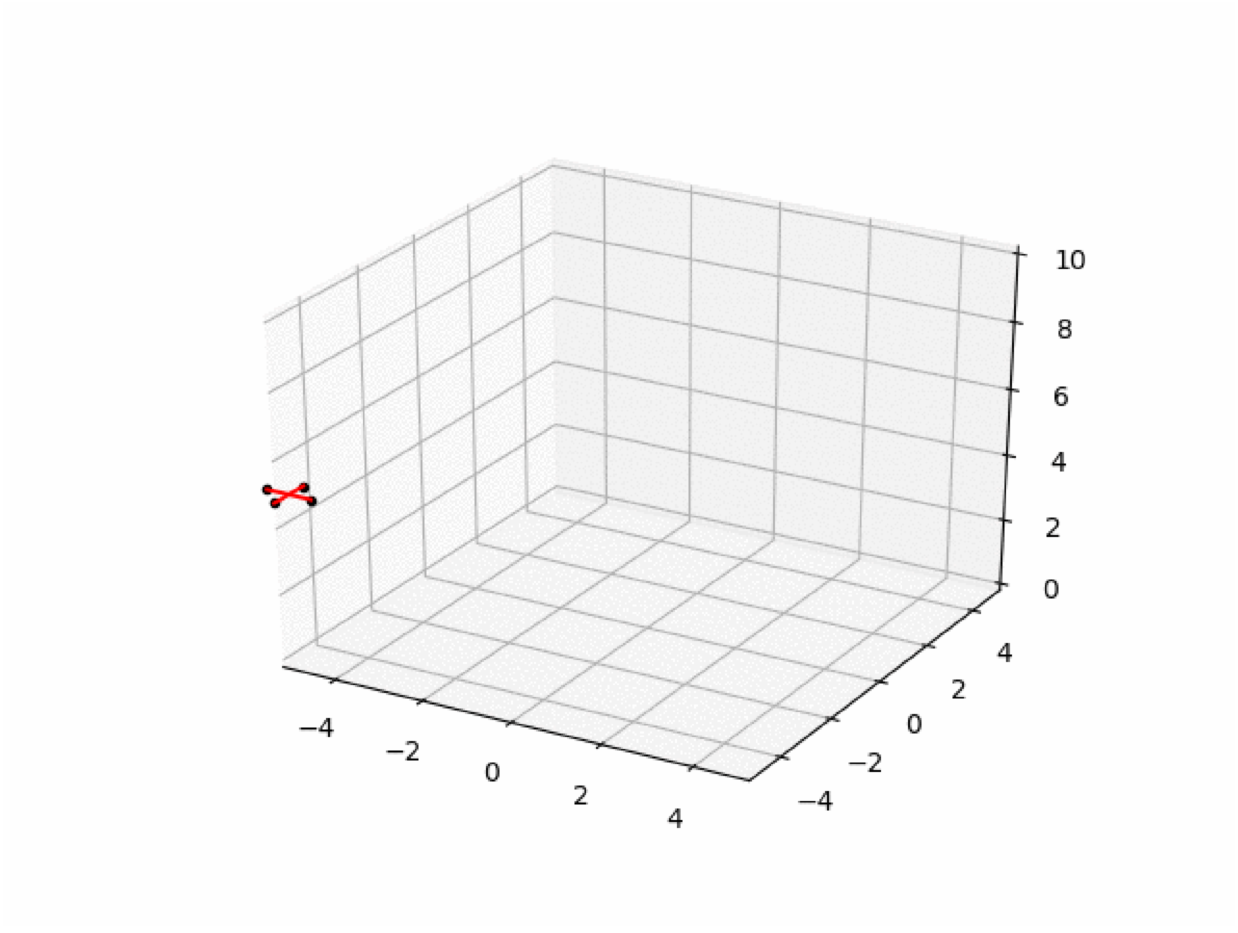
Arm navigation with obstacle avoidance simulation.



Aerial Navigation

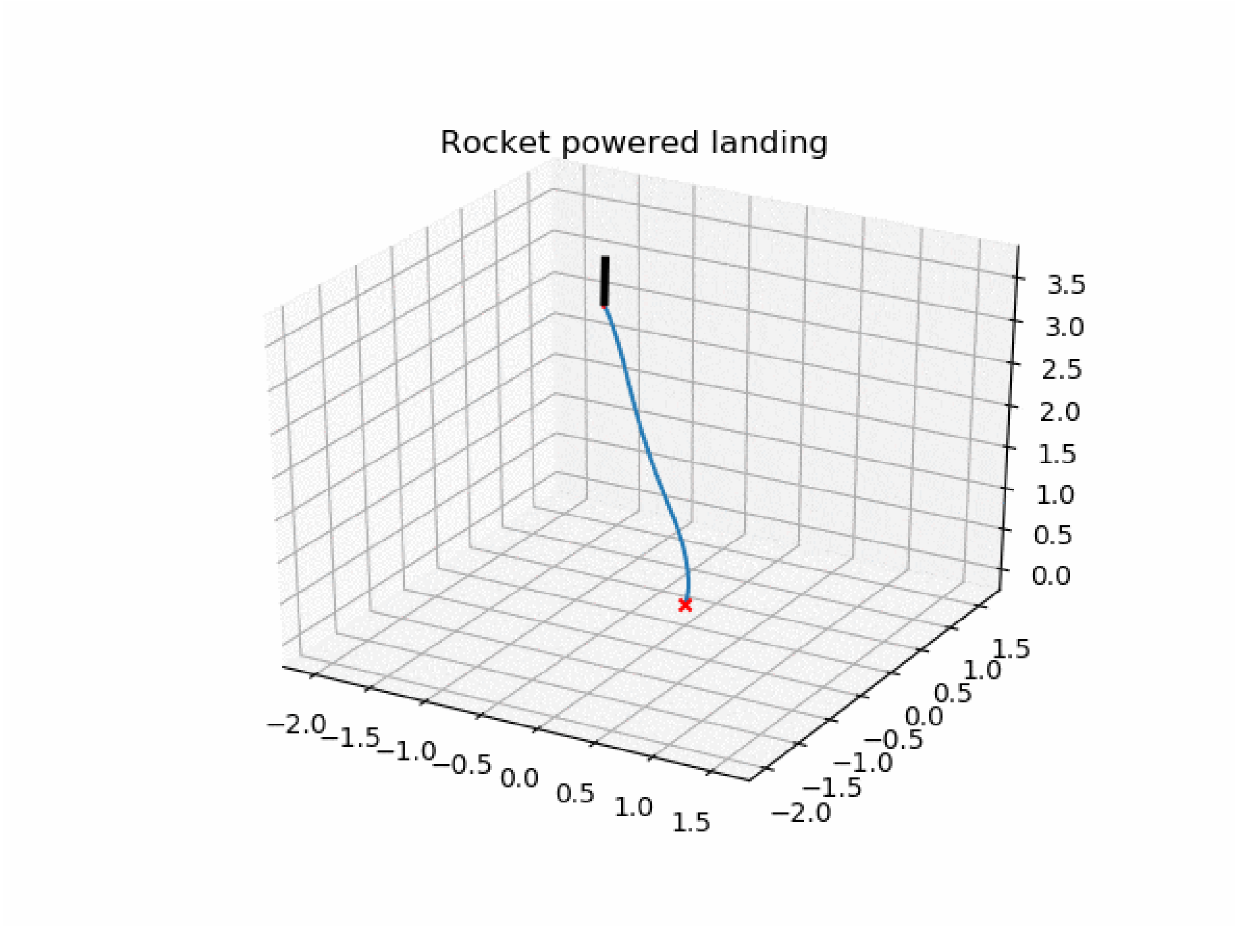
drone 3d trajectory following

This is a 3d trajectory following simulation for a quadrotor.



rocket powered landing

This is a 3d trajectory generation simulation for a rocket powered landing.



Ref:

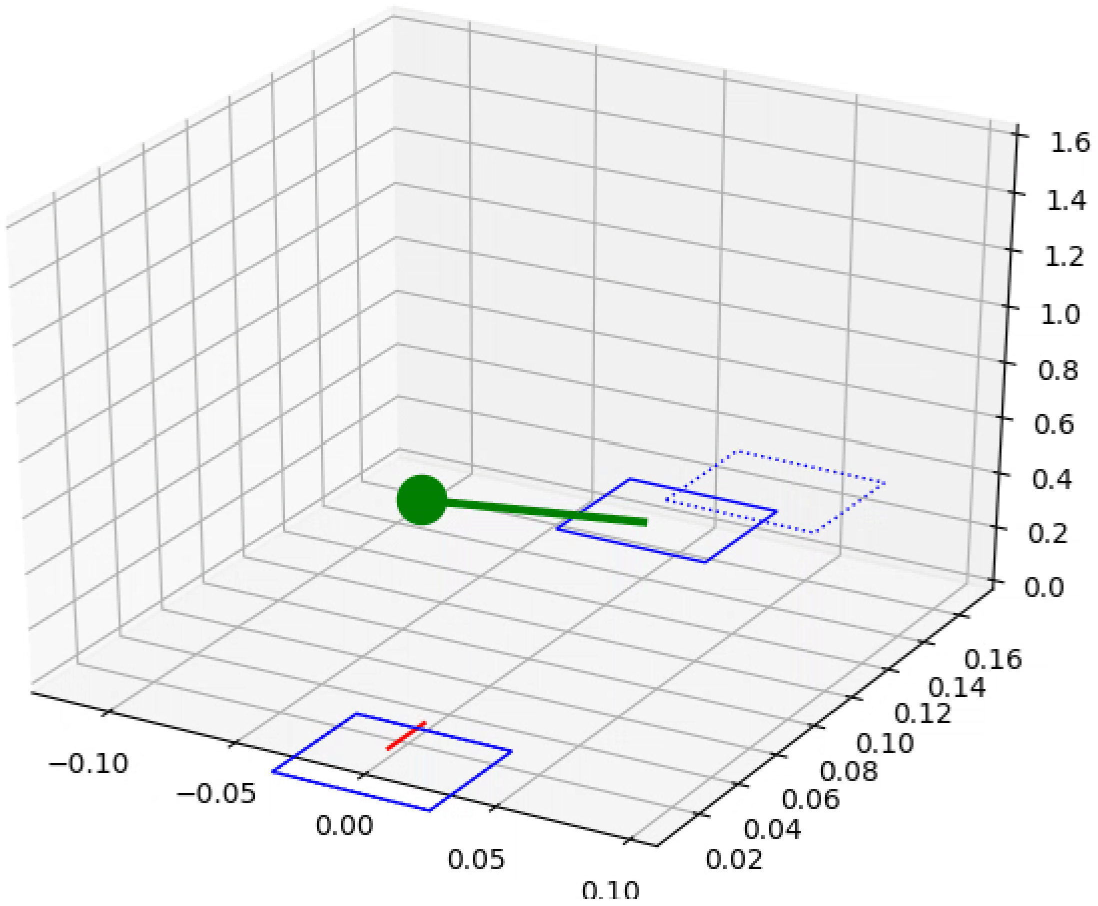
- notebook

Bipedal

bipedal planner with inverted pendulum

This is a bipedal planner for modifying footsteps for an inverted pendulum.

You can set the footsteps, and the planner will modify those automatically.



License

MIT

Use-case

If this project helps your robotics project, please let me know with creating an issue.

Your robot's video, which is using PythonRobotics, is very welcome!!

This is a list of other user's comment and references:users_comments

Contribution

Any contribution is welcome!!

Citing

If you use this project's code for your academic work, we encourage you to cite our papers

If you use this project's code in industry, we'd love to hear from you as well; feel free to reach out to the developers directly.

Support

If you or your company would like to support this project, please consider:

- Sponsor @AtsushiSakai on GitHub Sponsors
- Become a backer or sponsor on Patreon
- One-time donation via PayPal

If you would like to support us in some other way, please contact with creating an issue.

Sponsors

JetBrains

They are providing a free license of their IDEs for this OSS development.

Authors

- Contributors to AtsushiSakai/PythonRobotics