CSE 571 – Robotics
Open-Ended Project

1 Overview

1. The projects will be done in teams of 2-3 and will account for 50% of your grade.
2. Project can be investigating any question related to robotics, including but not limited to object recognition, pose estimation, localization and mapping, imitation learning, reinforcement learning etc.
3. Timeline and deliverables:
   - Project Proposal, due **April 25**;
   - Midterm Report, due **May 16**;
   - Poster Presentation, **June 5**;
   - Final Report, due **June 7**.
4. Hints
   - Try to identify a concrete and well-defined problem. You only have 8 weeks!
   - Come up with a timeline (as detailed as possible) for milestones you would like to achieve (e.g. finish data collection, implement algorithm). We will look for the timeline in both proposal and midterm report.
   - Don't worry too much about the performance. Summarize *what you have learned* in the final report.
   - Be creative and have fun!

2 Some Project Ideas

We encourage you to come up with your own project ideas, but if you have no idea where to start, here are some suggestions.

2.1 2D Navigation

Navigation is a fundamental problem in robotics. Use the knowledge you learned from this class, including but not limited to localization, mapping, planning and control to write an autonomous agent that can navigate in complex environments.

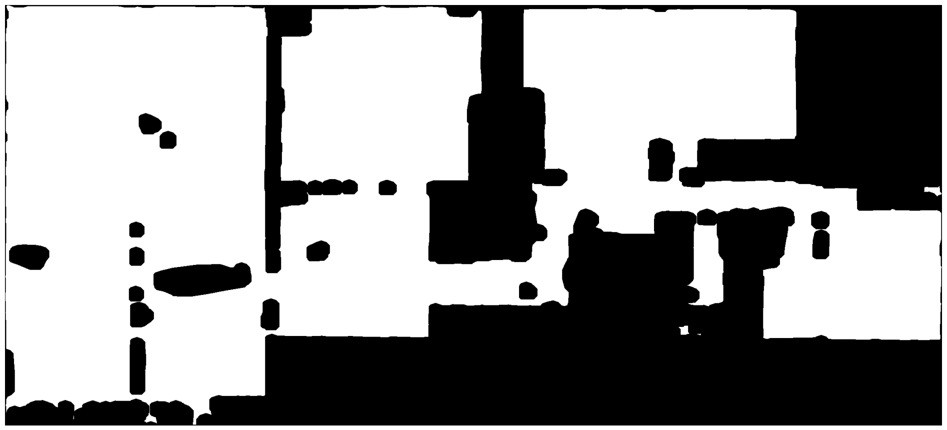

Figure 1: Maps for 2D Navigation

We provide you with 96 occupancy maps from real world environments. For learning-based methods, you can use 80 maps for training and 16 maps for testing. The dataset is available at `https://courses.cs.washington.edu/courses/cse571/22sp/project/2d_nav.zip`. Fig. 1 shows a sample map. We have also provided you with a set of utilities to read maps, generate laser scans and visualize results. Feel free to modify the code and adapt it to suit your needs. Check out `README.md` in the code directory for more information. We recommend setting up a virtual environment using `conda` to run the code, as you did in the homework.

Below are some example questions to investigate. Again, you don't have to follow these. Feel free to come up with your own questions.

1. Implement your favorite SLAM algorithms (e.g. FastSLAM, LOAM). Compare and analyze their performance under different scenarios. To make the problem more interesting, you can
   - Make symmetrical maps by concatenating an existing map with a flipped version of itself.
   - Randomly teleport the robot to a new location after a certain amount of time.
2. Train a state estimator which predicts the robot's state given an occupancy map of the environment and laser scan measurements. Compare different representations of the robot's state, e.g. continuous location plus heading vs heat map over discretized location and heading. To make the problem more challenging, can you train a model that generalize across different robot embodiments? For example, instead of a point robot with a single laser scanner, you can construct a stick robot with two laser scanners on its endpoints, or any robot that you like.
3. Implement a classical planner such as A* or RRT* that plans a path from a start location to a goal location. Generate some trajectories using the classical planner and train a learned planner that mimics the behavior of the classical planner. What are the advantages and disadvantages of classical vs learned planners? How well does the learned planner generalize to environments not in the training data?

2.2 Grasp Pose Evaluation

Grasping is one of the most important tasks in robot manipulation. We can formulate the grasping task as predicting the 6-DoF (degrees of freedom) pose (3D translation + 3D rotation) of the gripper before the gripper closes its fingers to grasp the target object (see Fig. 2 for an illustration).

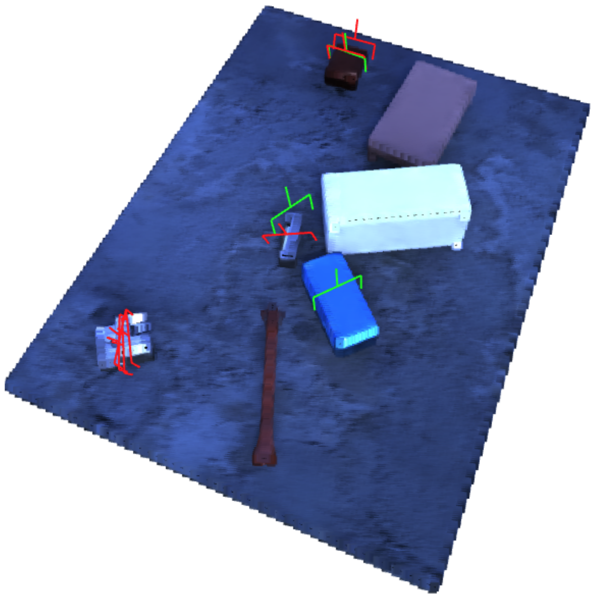

Figure 2: Grasps visualization. Green are successful grasps and red are failed grasps.

We provide an annotated dataset of grasps for you to experiment with. See `https://courses.cs.washington.edu/courses/cse571/22sp/project/grasping.zip`. The dataset contains 1200 scenes: 1000 for training, 200 for evaluation. Each scene has an RGB image, a colored point cloud and a set of labeled grasp pose proposals, as well as camera extrinsics (transformation from robot (world) frame to camera frame) and intrinsics. Your task is to use any method to distinguish the grasps poses that succeed from the ones that fail (the ground truths are evaluated using physics simulation). We have also provided some utility functions for you to load and visualize the data. Checkout `README.md` in the code directory.

Here some questions to investigate:

1. There are 100 evaluation scenes where the objects are drawn from categories not in the training set. Can your method generalize to these novel objects?

2. Modern vision-language models (VLMs) are powerful tools for open-world vision tasks, but do they work in robotics scenarios? Is it possible to come up with a clever prompting strategy to use VLMs to classify grasps? Checkout SoM and MOKA for examples of designing visual prompts for VLMs.