CSE 571 - Robotics
Open-Ended Project

1 Suggested Project Ideas

1.1 State Estimation and Navigation

- Use a depth or color camera to detect and track an object moving quickly, e.g. a ball bouncing off the ground or a wall. This makes for a good implementation of a combination of particle and Kalman filters.
- Localize with a depth camera (e.g. with a particle filter) in a given floor plan. We have existing laser maps and floor plans of the Allen Center.
- Create an exploration strategy for finding new views of unseen areas of a 2D/3D map. Investigate reconstruction and rendering techniques for such maps. Given an existing map, figure out how to perform probabilistic localization and filtering.
- Develop a navigation system for a powered wheelchair. This involves mounting a sensor on the wheelchair and developing a localization framework so that the wheelchair can autonomously navigate given a map.
- Develop techniques for active learning that take the most *informative* actions during filtering.
- Leverage the latest off the shelf segmentation and tracking models to develop better methods for tracking and pose estimation.
- Investigate whether segmentation methods can help improve visual SLAM methods that can operate without LIDAR/SONAR.
- Implement techniques for *semantic search* - not just navigating an environment, but finding a particular object. You can try using reinforcement learning algorithms for this.

1.2 Human-Robot Interaction

- Have a robot follow a person. Additionally, using a second camera/Kinect, you could enable a person to point at objects or goal locations that the robot needs to plan to navigate to.
- Implement visual servoing on a manipulator. This involves perception, planning and control. This video shows a preliminary implementation in a food manipulation task. Servoing has to be safe and robust to movements in the person's face to feed accurately and be robust to face rotations and partial face occlusions.

- Implement a simple human-robot handover system on a manipulator. The robot has to track the person, potentially predict where the person will move to and plan accordingly to receive the object from the person's hand while ensuring that there are no collisions and the motion is smooth. This involves perception and planning.
- Develop techniques for forecasting human behavior and predicting pedestrian behavior from existing pedestrian behavior datasets. Try using the latest multi-modal generative models for this.

1.3 Manipulation

- Implement a simple pick and place system on a manipulator. The robot can pick up an object from a table and place it at a different location. This involves some minor perception for object detection, grasp planning and motion planning.
- Implement a fast local collision avoidance system (in simulation) that reacts to moving objects and people in the scene. There are multiple methods to speed up the computation (including GPU-based techniques). Find a related paper here.
- Create a benchmark of motion planning algorithms on the manipulator (mainly in simulation). There are many new motion planning algorithms (BIT*, RRT*, TrajOpt, CHOMP, etc.) that can be implemented. OMPL has multiple benchmarking tools available. You can also consider creating your own benchmarking environments of varying difficulty (for instance, in terms of clutter in the manipulator's environment).
- Implement a handover task between two arms of a robot. This can involve picking up an object with one hand and transferring it to another before placing the object at a different location.
- Implement a teleoperation system for a robot manipulator. This can involve a virtual reality controller or some other input device.
- Leverage data to learn good heuristics for search to speed it up, or better sampling distributions for planning.
- Develop techniques for planning that are able to react to changing environments and obstacles by replanning appropriately. Good sources of algorithms to start from are LPA* and D* covered in the lecture.

1.4 Reinforcement Learning

- OpenAI Gym has several robotics and continuous control environments implemented using physics simulators (MuJoCo, Box2D). You can design a new reinforcement learning algorithm (or re-implement an existing one) for these environments. Other robots may also have open-source MuJoCo simulation code that you can use, e.g. the bipedal robot Cassie.
- Deep Q-Networks (DQNs) have been successfully applied to Atari games. Implement the DQN for a simple reinforcement learning task, e.g. cartpole or double pendulum. You can even look at some of the Atari games or other similar game worlds, or try learning directly from pixels rather than the physical state.
- Develop a system to collect teleoperated demonstrations from a human demonstrator in simulation and train offline RL or imitation learning methods using this data

- Develop a technique for safe exploration in reinforcement learning that enforces safety on exploring and interacting with a human, potentially by detecting their position and safely avoiding them.
- Develop reactive planning methods that are able to dynamically avoid obstacles that may arise during execution using model based RL methods like MPPI
- Train memory based reinforcement learning algorithms to explore the environment and find particular objects.
- Train algorithms in simulation that can keep learning behaviors without resets, by resetting themselves.
- Try using large-scale generative models to generate 3-D meshes and content for simulation and then train agents in these simulation environments.
- Implement state-of-the-art exploration methods, such as random network distillation or Go-Explore for robotics problems and show how well they can do at robotics manipulation problems.
- Train hierarchical RL algorithms on problems like multi-object rearrangement and placement, and push the limits of the multi-step tasks that you can solve
- Experiment with more expressive policy distributions such as diffusion models, autoregressive models and flow-based models for imitation learning or reinforcement.
- Develop techniques for inferring rewards from demonstrations/videos by using off-the-shelf and open-world object trackers.