CS-8803 Special Topic: Machine Learning for Robotics

Homework

Support: cedric.pradalier@georgiatech-metz.fr
Office hours: 10:00 to 18:00.

Step 0: Setup

For this homework, and many of the coming ones, a basic workspace will be made available in `/cs-share/pradalier/cs8804_students_ws`
In this case, we will be working on using linear regression and in general least-square minimisation to find the ground plane in a 3D point cloud. This point cloud is provided by the simulated kinect on the V-REP robot under the topic /vrep/depthSensor.

Start by copying the floor_plane_regression and floor_plane_ceres ROS packages from the `cs8804_students_ws/src` directory. Make sure everything is working by compiling them with catkin_make.

The packages are fully configured to find the libraries they need. You don't need to touch the CmakeLists.txt of package.xml files. Just edit the src/floor_plane_regression.cpp in each package. You should read the whole file to understand what's going on, but your contributions must be inserted between the "`TODO START`" tag and the "`END OF TODO`" tag.

Step 1: linear regression using Eigen

Eigen is a math library providing all the functionalities required for matrix/vector manipulation, solving linear systems and more. We will use this library to estimate the most likely plane from the kinect point-cloud. For this step, you need to modify the `floor_plane_regression` package.

There are several points to pay attention to when solving this problem:

- First the point-cloud must be projected in the robot frame. ROS TF library is designed to take care of all the change of frame required in a robotic application. You will have to use it to project the point-cloud. The code for this part is already written, but you need to change the parameters to make it work.
- Before doing the linear regression, we may want to filter out some 3D points. A basic piece of code is already provided, eliminating potential bogus points and points too far from the sensor. You need to tweak the parameters to find which points we need to consider.
- For each new point-cloud, you need to update the linear regression problem and solve it. This requires updating the matrices that have been prepared for you and then calling the right solver from Eigen. All the links you need are provided on Piazza. Use the documentation wisely.

Step 2: Non-linear regression

CERES is the non-linear optimizer developed by Google to solve, in particular, bundle adjustment problems. A non-linear solver is obviously an overkill to solve the ground-plane extraction problem. However, this provide a simple problem to instantiate CERES and a comparison between non-linear optimisation, and a more efficient linear solution.
For this step, you need to modify the floor_plane_ceres package.
Most of the CERES setup is already prepared for you but you need to provide two elements:

- The error functor: CERES builds its optimisation problem by accumulating error terms which describe the discrepancy between a model and an observation. You need to modify the PlaneError functor to make it return the correct residual error. This functor has a complex templated form because it is also used to build automatically the differential or Jacobian of the error term.
- The error functor instantiation: we need to create a error term for each considered point in the point-cloud. This part is left to your implementation.

The CERES optimizer comes with a very clear and nice tutorial so you should read its documentation before conducting the above modifications. The link is provided on Piazza.

Step 3: Evaluation

Once the two packages are running, you should be able to see their predicted ground plane in Rviz. You should also be able to drive the robot around and see how the estimation is behaving.

Here are a few questions to evaluate on the two systems:

- Computation time and load: what are the typical values, does it depend on the environment, how much variation do you observe.
- Precision: in the simulation environment, we have a perfect setup and a perfect knowledge of the real ground plane, which should be $z = 0.0$. Evaluate how precise both approach can be, how sensitive they are to the objects in the environment.
- Robustness: the non-linear solver has an option -robustify. Is this helping near obstacles? Propose ideas (no implementation) on how to solve these issues.

Submit the ros packages (git, tar.gz or zip) and a small report focusing on quantitative evaluation by email by next Wednesday, 10:00 to cedric.pradalier@georgiatech-metz.fr (don't forget to mention you group name and the group members)