John Allard and Alex Rich
Harvey Mudd College
Summer 2014 REU

Using the 3D Localization Program

1 Locations

The 3D Localization programs are located at `github.com/jhallard/3DLocalization`.
Some example Robot Control Programs are located at `github.com/aarich/ROS-Controllers`.

2 Quick Start

See the next section on how to visualize what's happening. To run the program using the already present database of images, open three terminal windows (or tabs):

1. `roscore`
2. `cd` into `3DLocalization/Localization/build/` If needed, compile the code using the command `make`. If this is done already, run

   ```
   ./devel/lib/localization/3DLocalization 2ndFloorSprague
   ```

3. `cd` into `3DLocalization/Localization/RobotTest/build/`. Again, `make` as necessary. Run `./robot 2ndFloorSprague`.
4. Once you allow these programs to run, they should stop. The first will tell you to press enter, and the second will say "Done Loading Images." You should press enter on the first, then press enter on the second. This initializes the handshake.

The program runs, displaying the virtual robot image and the top match if found. Debug statements are printed, and can be turned off in `ProgramIO.cpp`.

3 Visualization

All visualizers are in the `3DLocalization/Localization/src/GUI/` folder.

3.1 MapViewer

Now you need to decide what environment you want to run the map viewer program in. If you want to run it in the 2nd Floor Sprague environment, you can type `./MapViewer` and the program will start.

If you wish to run it with another 3D model, you will need to type the following into the terminal `./MapViewer ModelName OBJ_FILE` where `OBJ_FILE` is the name of your 3D model file inside the your model directory.

When the program is started, the map will open up. Use the `[W A S D]` keys to navigate the environment, and the up and down arrow keys to change your z height. You can press the spacebar at any time to save a picture of your current view in the environment. Press the escape key to leave the program.

If this program is run while the localization program is running, the particles will be automatically drawn and updated in the environment in real time.

3.2 MATLAB

This is run after the localization is done. Run `weightovertime` to get a graph of the top weight over the iterations as well as a graph of the average of the top 20 particles.

3.3 Meta

In here you can run `python createTrace.py` or `python WeightOverTime.py`. The first animates the path of the localization guess, and can be run at any time during or after the localization. The second graphs the same as the MATLAB, but simpler.

3.4 PCLViewer

Run `./pcl_viewer ModelName` to view a PCL visualization of the particles. Each particle is colored according to its weight. Only one particle is shown even if multiple particles exist at a certain spot. If running in real time, press "n" to update the scene.

3.5 PyViewer

This is located in `GUI/PyViewer/` Simply run `python Viewer.py`. To adjust the number of particles displayed, click on the top section of the visualizer. Clicking towards the right increases the number of particles shown, while clicking towards the left decreases it. To close the window, click on the lower portion of the window.

4 PreLocalization

We will refer to the object file map as "ModelName." You should decide on a model name for your map, then be consistent. Changing the ModelName will mean that the program won't be able to find files. Open object in MeshLab, select "Remove Unreferenced Vertices," then export as obj file.

4.1 Rendering Images with the Perspective Generator

This program serves the purpose of taking the 3D model and generating a large library of 2D images from the model. These images are then processed for computer vision related features, for use during the localization attempt. This program will take about 5-20 minutes to run depending on the size of the 3D model, the speed of your computer, and the density of the images rendered from the environment.

4.1.1 Program Set-Up

First, you need to make sure that you have created a folder under the `/3DLocalizaton/Data/ModelData/` directory. This folder that you've made must contain all of the 3D model data for the environment that you wish to localize the actor in. For an example of this, look at either the `/Modeldata/2ndFloorSprague/` or `/ModelData/BirchLab/` folders. The perspective generator will need to use this folder to load the model data that it needs to render images.

The final thing that you need to do before you start the program is to make an input file for this program. Create a text file inside of the `/PreLocalization/PerspectiveGenerator/Inputfiles/` directory. This text file must specify the following information, line by line.

- Name of the folder that contains your model data inside of the `/3DLocalizaton/Data/ModelData/` directory.
- Name of the `Wavefront.obj` file inside of the directory listed above.
- `[x1 x2 y1 y2 z gd dtheta]`

An example input file can be seen inside of the `/3DLocalization/PreLocalization/PerspectiveGenerator/Inputfiles` directory. The 3rd line defines the x and y bounds of the map, the z height that images should be rendered from, how dense you would like the grid to be that the images are rendered from, and the theta (in degrees) that you would like the camera to rotate between consecutive image renders.

4.1.2 Running the Program

To start the program, open up the terminal and navigate to the `/3DLocalization/PreLocalization/PerspectiveGenerator/build` directory. Now type `cmake ..` followed by `make`. If an error appears in either of these steps please visit the `PerspectiveGenerator` section of the Code Documentation document to look for common errors and fixes. Next, type the following:
`./PerspectiveGenerator ModelName`.

The program will now open up a viewer of the map, and will wait for you to click anywhere in the window. When you do click, it will begin to go through all of the points defined on your grid and render a [600x600] jpeg image. These images will be names according to their location in the environment and will be stored inside of the `/3DLocalization/Data/FeatureData/` directory. When the program stops moving around from location to location inside the map it will be done and you can close the window.

4.2 Creating the Database of Features

Run CreateDB: `./CreateDB ModelName`. CreateDB does the following:

1. Reads images from directory in `Data/RenderedImages`.
2. Computes various features, including.
3. SURF Descriptors.
4. Average Pixel Sum images of size 50×50.
5. Above/Below images of same size.
6. Save images to corresponding folder.
7. Both descriptors are saved in yaml files.
8. Images are stored to gsimages and bwimages.

5 Localization

Once the database has been created, simply run the MCL program with the name of the file:

```
./devel/lib/localization/3DLocalization ModelName
```

The program will initially load all available images and their descriptors, which may take some time. When it is finished, it will wait for the user key press before continuing. At this point, you should start the robot program (see below), then press enter to continue the Localization program. This allows the two programs to have a handshake, then set up all ROS publishers and subscribers.

While 3DLocalization runs, you can run the MapViewer, PyViewer, and PCL Viewer to get a sense of where the robot thinks it is.

6 Robot Controllers

6.1 Using an established ROS Controller

The ROS Controllers are located at `github.com/aarich/ROS-Controllers`.

Connect to respective robots (depending on which ROS Controller you are using), then run the ROS-Controller program after running the main MCL Program, as instructed above. For the established controllers, use the following drivers to connect:

- iRobot Create: `irobot_mudd.py`
- ARDrone: `ardrone_autonomy`

6.2 Making your own ROS Controller

Start with one of the example ROS Controllers, then modify it to fit your needs!