Weekly Progress 1/25/2017

January 30, 2017

1 Introduction

Created scripts to load datasets from UTIAS Multi-Robot Cooperative Localization and Mapping Datasets for simulations to verify our EFK works for a single robot localization. Later, we will expand to multirobots cooperative localization.

2 Applied Data Sets

There are 5 files needed for Robot1:

Landmark.dat, the locations of all 15 landmarks

# Subject #	x [m]	y [m]	x std-dev [m]	y std-dev [m]
6	5.70928255	4.96404466	0.00027464	0.00041465
7	5.25292609	5.53656921	0.00011889	0.00035386

`Robot1_Groundtruth.dat`, the records of robot1's locations and orientations during the time

# Time [s]	x [m]	y [m]	orientation [rad]
1248272263.325	3.57323240	-3.33283870	2.34080000
1248272263.346	3.57322470	-3.33283680	2.34070000

`Robot1_Measurements.dat`, the records of robot1's measurment of the distances and angle of the objects robot detected

# Time [s]	x [m]	y [m]	orientation [rad]
1248272263.325	3.57323240	-3.33283870	2.34080000
1248272263.346	3.57322470	-3.33283680	2.34070000

`Robot1_Odometry.dat`, the records of robot1's speed and angular velocity during the time

```
# Time [s]    forward velocity [m/s]    angular velocity[rad/s]
1248272272.841    0.074     0.229
1248272272.852    0.074     0.229
```

`Barcode.dat`: translate between the barcodes the robot detect and the actual objects

```
# Subject #    Barcode #
  1       5
  2      14
  3      41
```

3 Mathematical Model

There are two parts for EKF, state update and measurement update

State update:

$s_k = f(s_{k-1}, u) = [s_{k-1}[1] + deltaT * i[1] * cos(s_{k-1}[3]);$
$s_{k-1}[2] + deltaT * i[1] * sin(s_{k-1}[3]);$
$s_{k-1}[3] + i[2] * deltaT]$

Measurement Update:
$z_k = h(s, l) = [norm(l_k - s_k);$
$atan((l_k[2] - s_k[2])/(l_k[1] - s_k[1])) - s_k[3]];$

Variables: state of the robot(s): $s[1]$ is x-axis location, $s[2]$ is y-axis location and $s[3]$ is orientation

input(u): [1] is velocity, $u[2]$is angular velocity

landmark(l): $l[1]$ is x-axis location, $l[2]$ is y-axis location

measurement(z): $z[1]$ is the distance between the robot and the landmark, $z[2]$ is the bearing

$\frac{du}{dt}$

4 Program Skeleton

So far, there are 3 script files:
`git@git.uclalemur.com:billyskc/Localization_sim_py.git`

`loc_sim.py`: this will give us the estimations of robot locations given odometry and measurement data.

`comparison.py`: this will give us the groudtruth of the robot location within the time interval that match the localization performed by `loc_sim.py`.

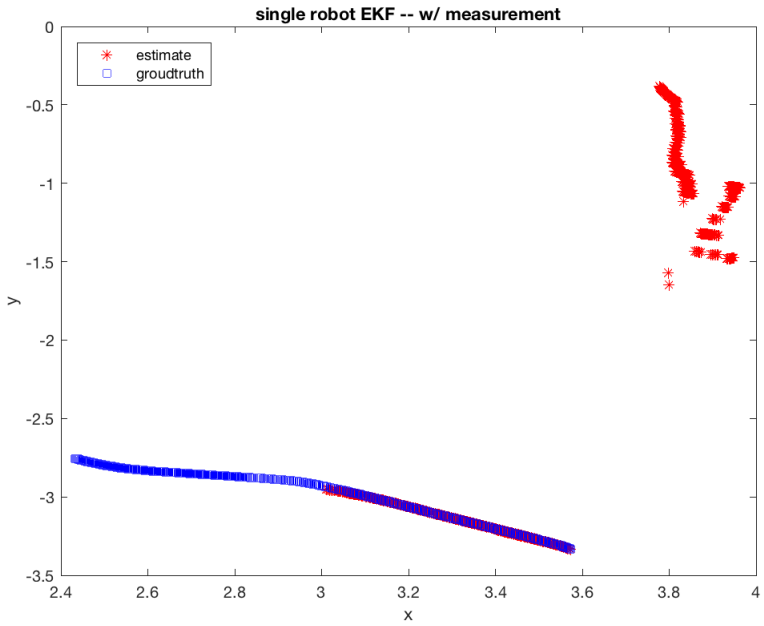
`comparison_graph.m`: this will give us the graph of the robot's actual locations and estimate location in 2D.

5 Progress and Results

Past few week, we create a single matlab script to do the simulation. Because of the sizes of all these data files, it took a long time (1-2 mins) for each run, which makes it difficult to debug. More importantly, Octave graphic toolkit doesn't work well on my laptop. My laptop kept crashing even with gnuplot.

We decide to switch to python and only use matlab for graphing purpose. Now, these scripts are compiled much faster and crashing issue is solved as well.

Here is the plot for 1000 time step after Robot1 start to move:



When time around t= 1248272284.671, estimations no longer stay near groudtruth. There are several measurements have been taken already.

6 Next Week Plan

Keep debugging