Scientific Experimentation and Evaluation

Assignment 7

Chaitanya Hebbal
José Carlos Mayoral Baños

November 20, 2016

1 Task Description

Using a modified LEGO robot from the experiment *"Manual motion observation"* a new set of poses will be acquired via AICISS lab optical localization system, then the data will be analyzed to observe the variation on the poses and look at the error distributions in order to observe the variation on the measures and look at the error distributions.

The layout of this experiment consists on:

1. Five different curves movements are going to be measured: Straight line and 4 arcs (2 left and 2 right).

2. The Device Under Test is a modified LEGO robot from the experiment *"Manual motion observation"*.

2 AICISS Lab Optical Localization System

A optical-based localization system will be used in this experiment. From [1] the optical systems have several advantage to take into account:

- Less susceptible to noise from the environment.
- Optical tracking does not suffer from drift problems.
- Optical tracking allows for many objects to be tracked simultaneously.
- Interaction devices can be lightweight and wireless.

Therefore, at AICISS Lab there is already an "Optical Localization System" which uses several cameras in order to track pre-defined markers, such as the one shown at figure 1. By adding multiple cameras to the tracker system the precision of each pose increases and the system becomes more accurate. The AICISS system uses 3 cameras and a single workstation manages it. The workspace area sizes 7.3 meter width and 3.5 height.

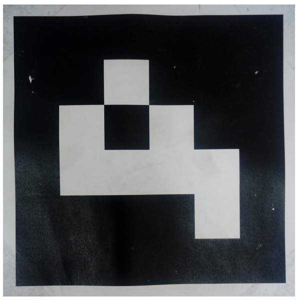

Figure 1: AICISS Marker

Speaking about software, it has some components that need to be run in the workstation for the tracking system. It is important to mention that the output data of the system is a pose array with time stamps therefore the system provides additional data in comparison with manual measurements process.

- 'camera_server': Run the cameras.
- 'robot_tracker': Run the tracker service.
- 'apt-robot >> your_file.log': In case a log file which stores all the data is needed.
- '': Open a visualization interface.

3 Experiment setup

3.1 Device Under Test

A tracker have been added to the top of the robot in order to make able the system to track it, the robot configuration is shown at figure 2.

3.2 Measuring Method

The measuring method for each curve consists on:

1. The component camera_server and robot_tracker must be running.
2. Two light sensors are used to mark the points in order to get the pose of the robot.

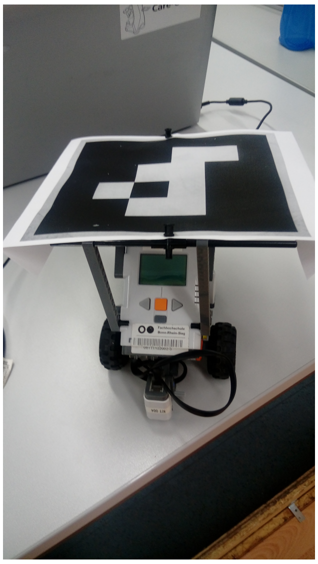

Figure 2: LEGO Nxt Robot

3. One example of the marker is shown at image 3.
4. After localizing the robot, the apt-robot component runs before the robot starts movement.
5. Click robot left button to start robot movement.
6. After the robot stops, the apt-robot component is stopped.
7. The robot location changes by hand to the initial pose.
8. Repeat 40 times steps from 4 to 7.

3.3 What difficulties are expected?

- Other tracker in the arena must be hidden.
- The initial pose markers don't guarantee the same initial pose.
- Tracker can loses the robot for instances then the data is not noise-free.

3.4 Data Processing

The raw measurements are shown at figure **??**. This figure doesn't show any kind of data processing so it makes able to see some noisy measurements that don't relies to any of the five curves. Also the figure shows an unexpected problem with a measurement

Figure 3: One mark per laser is marked on the floor with a pencil to start at the same pose

method which is that the curve were acquired in two different days then the initial marker were lost for the second day. In the figure, straight and slightleft curve were taken in the first day, the rest on the second one.

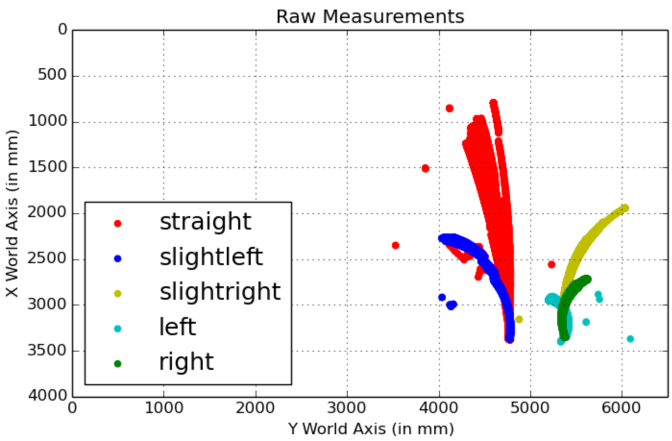

Figure 4: Raw Acquired Data

All the data has different starting point due to the fact that even using a start pose markers it still not be so accurate and the number of samples for each measurement are not the same. So for every curve there are 40 possible initial poses then:

$$\bar{p_0} = \frac{1}{n}\sum_{i=1}^{n} p_{i_0}$$

where $n = 1, 2, \cdots 40$ and i_0 is the first pose of each measurement.

Then this calculated initial pose was substracted to all the poses from the files.

$$\hat{p_m} = p_m - \bar{p_0}$$

Additionally the sizes of the measurements files were trim to the minimum length of all of the measurements. The result project all measurements to the origin of a frame 5.

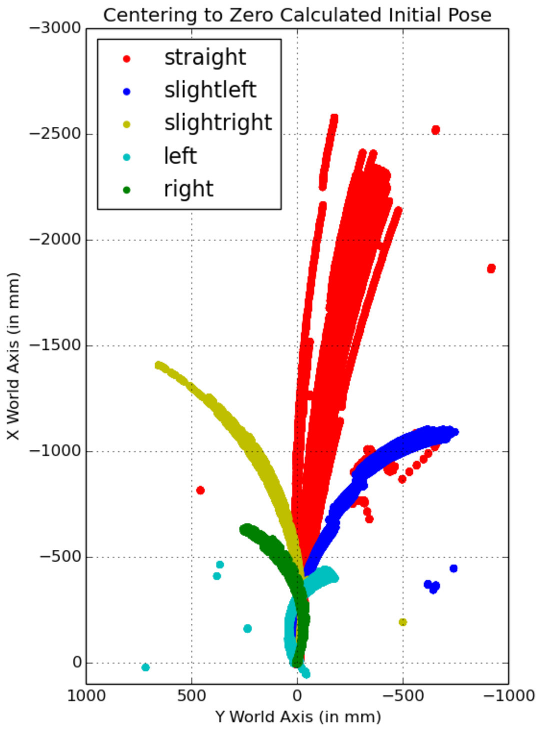

Figure 5: CenterData

Since the cameras in the lab are not caliberated properly there were some errors in our last weeks measurements.so we conducted the experiments again to get a new set of measurements.All the measurements were taken within radius of one camera so that errors due to camera caliberation do not affect our measurements. The new set of measurements are as shown in Figure:9

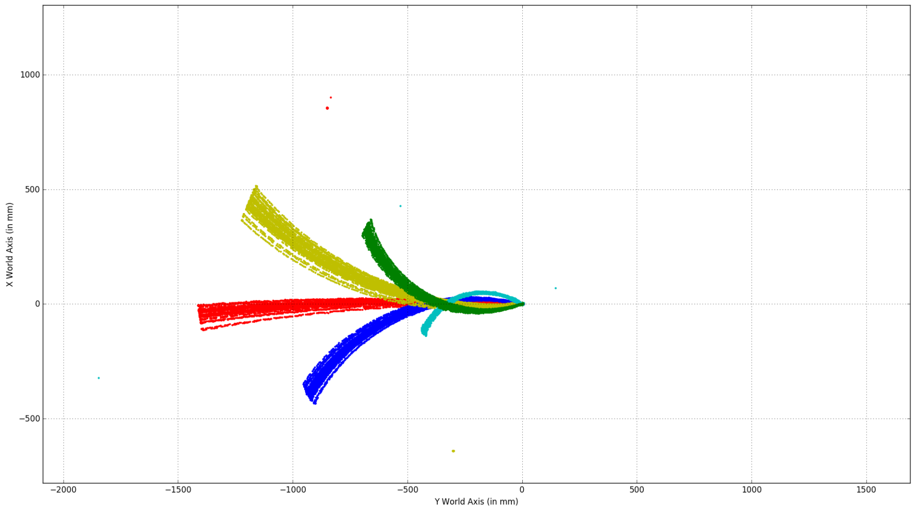

Figure 6: New set of measurements

3.5 Motion Model

Algorithm motion_model_velocity(x_t, u_t, x_{t-1})**:**

$$\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$

$$x^* = \frac{x + x'}{2} + \mu(y - y')$$

$$y^* = \frac{y + y'}{2} + \mu(x' - x)$$

$$r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$$

$$\Delta\theta = \mathrm{atan2}(y' - y^*, x' - x^*) - \mathrm{atan2}(y - y^*, x - x^*)$$

$$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$$

$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$

$$\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$$

return $\mathbf{prob}(v - \hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \mathbf{prob}(\omega - \hat{\omega}, \alpha_3|v| + \alpha_4|\omega|)$
$\quad \cdot \mathbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

Figure 7: Velocity motion model [3]

We use the velocity motion model as shown in Figure:7 According to this model, the probability of a robots pose at time t given a motion command and the pose at time $t1$ is given as in [2]

$$P(x_t|x_{t-1}, u_t) = N(0, \alpha_1 v^2 + \alpha_2 \omega^2) \cdot N(0, \alpha_3 v^2 + \alpha_4 \omega^2) \cdot N(0, \alpha_5 v^2 + \alpha_6 \omega^2)$$

where v and ω are commanded linear and angular velocities and the three normal distributions are evaluated at points $v - \hat{v}, \omega - \hat{\omega}$ and γ

We estimate the motion parameters $\alpha_1 ... \alpha_6$ using the maximum likelihood approach.

We made use of the code provided in the sample solution [2] to estimate the motion parameters $\alpha_1 ... \alpha_6$

The values obtained are as follows:
α_1 = 9.36575693e-06
α_2 = 8.53840417e-06
α_3 = 9.22593223e-06
α_4 = 6.02069757e-06
α_5 = 2.54877909e-06
α_6 = 8.97275720e-06

Once we have estimated the values of alphas, we sample data using following algorithm as shown in Figure:8

Algorithm sample_motion_model_velocity(u_t, x_{t-1}**):**

$\hat{v} = v + \mathbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$
$\hat{\omega} = \omega + \mathbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$
$\hat{\gamma} = \mathbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$
$x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}}\, sin(\theta + \hat{\omega}\Delta t)$
$y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}}\, cos(\theta + \hat{\omega}\Delta t)$
$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$
return $x_t = (x', y', \theta')^T$

Figure 8: Sample motion model velocity [3]

3.6 Results

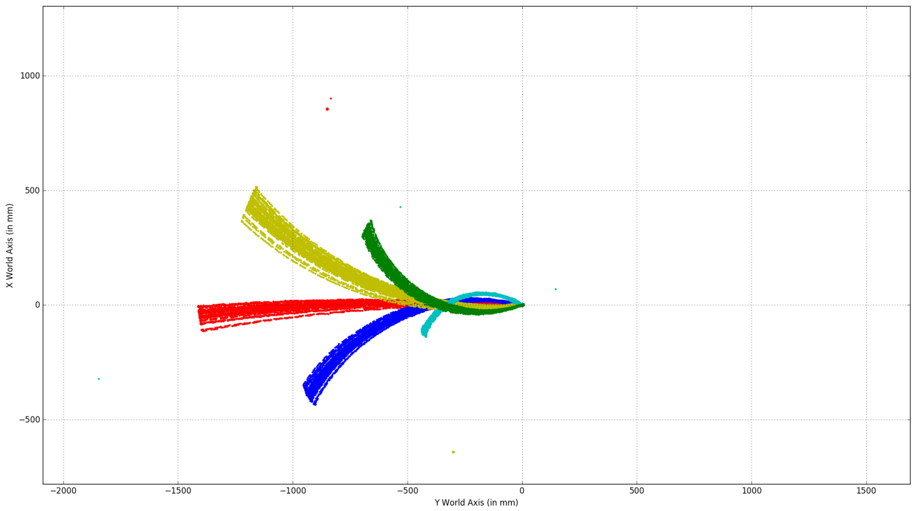

Figure 9: Original Data

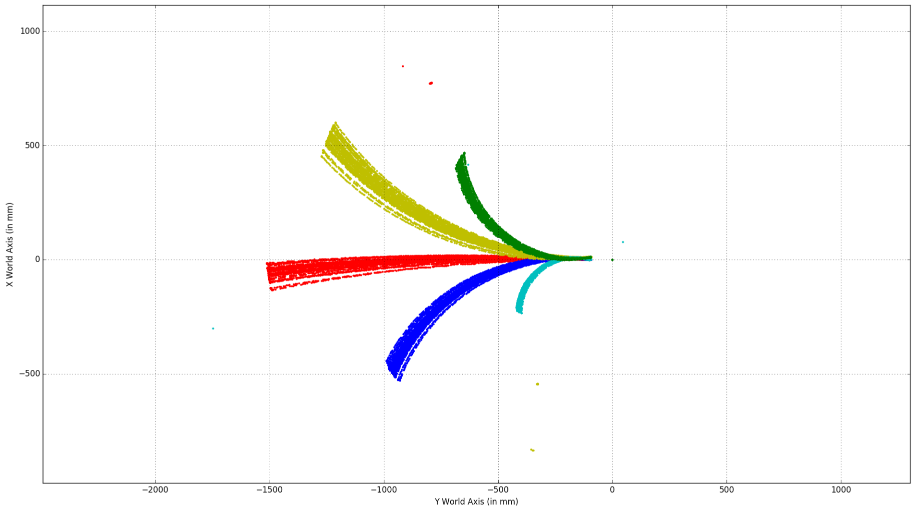

Figure 10: Sampled Data

From figures9 and 10 it is clear that the motion model perfectly fits the observed data for the estimated values of alpha.

References

[1] Optical Tracking Explanation: Link

[2] Motion Model Parameter Estimation:Sample Solution

[3] Probablistic Motion Models by Wolfram Burgard, Cyrill Stachniss,Maren Bennewitz, Kai Arras: Link

4 Appendix

4.1 Parameters used to drive the robot

Table 1: Parameters Used

	Arc radius	Angle	Distance	Track Width
Steep left arc	40	90		12
Steep right arc	40	90		12
Soft left arc		120	150	12
Soft right arc		120	150	12
Straight	160	90		12

4.2 Sampling code

```
def sample_motion_model_velocity(ut, x_prevt, alphas, delta_t):
    var1 = alphas[0]*(ut[0]*ut[0]) + alphas[1]*(ut[1]*ut[1])
    var2 = alphas[2]*(ut[0]*ut[0]) + alphas[3]*(ut[1]*ut[1])
    var3 = alphas[4]*(ut[0]*ut[0]) + alphas[5]*(ut[1]*ut[1])
    v_hat = ut[0] + np.random.normal(0,var1)
    omega_hat = ut[1] + np.random.normal(0,var2)
    gamma_hat = 0#np.random.normal(0,var3)
    K = (v_hat/omega_hat)
    xt = x_prevt[0] - K * np.sin(x_prevt[2]) + K * np.sin(x_prevt[2] + (omega_hat
    yt = x_prevt[1] + K * np.cos(x_prevt[2]) - K * np.cos(x_prevt[2] + (omega_hat
    theta = x_prevt[2] + (omega_hat*delta_t) +(gamma_hat*delta_t)

    return np.array([xt,yt,theta])
```