I).My algorithm

CSE276A Hw3 Yunhai Han.

i). My algorithm.

a). ~~First~~ First, let me write down the Gaussian probability distribution for multivariate systems:

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp[-\tfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)].$$

Where, $x \in \mathbb{R}^n$, $\mu \in \mathbb{R}^n$, $\Sigma \in \mathbb{R}^{n\times n}$.

$\because \Sigma$ is a covariance matrix, so Σ is a positive ~~se~~ semidefinite matrix.

$\therefore -\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu) \le 0 \quad \forall x, \mu \in \mathbb{R}^n$

$\therefore$ I ~~can~~ can conclude that ~~when~~ $x=\mu$ is the most possible solution ($p(\mu)$ is the largest value of $p(x)$).

Σ introduces uncertainty into our systems and if Σ is $0_{n\times n}$, the system ~~beco~~ becomes a deterministic system, which means $x=\mu$ is a 100% correct solution. However, due to the complex dynamics in the real world, we must use ~~a~~ Σ matrix to represent these uncertainty.

b). Second, for our homework, ~~the~~ we have 12 landmarks and each landmark has its x, y coordinates in the world frame. Besides, we need to use x^*, y^*, θ^* to represent the position and orientation. So, we totally have $12\times2+3 = 27$ variables $\rightarrow x \in \mathbb{R}^{27}$, $\mu \in \mathbb{R}^{27}$, $\Sigma \in \mathbb{R}^{27\times27}$.

For motion model, as we have used many times, I write down here:

$\tan\gamma = \frac{L}{V}\cdot\dot\theta \quad \therefore \dot\theta = \frac{V}{L}\cdot\tan\gamma$. ($V$ is velocity of picar, L is the wheel base, γ is the steering angle).

In this case, we just want picar to run in a circle and a "8" trajectory. So we can define V and γ beforehand and when picar is running, they will not change. (And this can guarantee the trajectory is ~~as~~ what as the title required) $\rightarrow V = V_0, \gamma = \gamma_0$, $V_0, \gamma_0 \in \mathbb{R}$.

$\therefore \dot\theta = \frac{V}{L}\cdot\tan\gamma$ is a constant $\therefore \theta_t = \theta_{t-1} + \dot\theta\cdot\Delta t = \theta_{t-1} + \frac{V}{L}\tan\gamma\cdot\Delta t$.

$\because \dot x = V\cos\theta$. $\dot y = V\cdot\sin\theta$. $\therefore x_t = x_{t-1} + \dot x\cdot\Delta t \quad y_{(t)} = y_{t-1} + \dot y\cdot\Delta t$.

In the above ~~two~~ two lines, I discretize the system since I need to run it on a robot.

$$\begin{cases} \theta_t = \theta_{t-1} + \frac{V}{L}\cdot\tan\gamma\cdot\Delta t \\ x_t = x_{t-1} + \dot x\Delta t = x_{t-1} + V\cdot\cos\theta\cdot\Delta t = x_{t-1} + V\cdot\cos(\theta_{t-1} + \frac{V}{L}\cdot\tan\gamma\Delta t)\cdot\Delta t \\ y_t = y_{t-1} + \dot y\Delta t = y_{t-1} + V\cdot\sin\theta\cdot\Delta t = y_{t-1} + V\cdot\sin(\theta_{t-1} + \frac{V}{L}\cdot\tan\gamma\Delta t)\cdot\Delta t. \end{cases}$$

It can be easily said that, the motion model is nonlinear for x and y.
So in order to use Kalman filter, I need to linearize these ~~two~~ two variables.
Suppose. $x(t) = f(x_{t-1}, u_t)$. here, $x(t)$ and $x(t-1)$ are both state vectors $\in R^{27}$
As I said above, $u(t)$ includes v and γ. (constants).

$$\frac{\partial \hat{x}(t)}{\partial \hat{x}(t-1)} = 1, \quad \frac{\partial \hat{x}(t)}{\partial \hat{\theta}(t-1)} = v\cdot(-\sin(\theta_{t-1}))\,\Delta t\cdot \dot{\theta}_{t-1} = v\,\Delta t\cdot(-\sin(\theta_{t-1}))\cdot\frac{v}{L}\tan\gamma$$

$$\frac{\partial \hat{x}(t)}{\partial \hat{y}_{t-1}} = 0.$$

$$\frac{\partial \hat{\theta}(t)}{\partial \hat{\theta}(t-1)} = 1, \quad \frac{\partial \hat{\theta}(t)}{\partial \hat{x}(t-1)} = 0, \quad \frac{\partial \hat{\theta}(t)}{\partial \hat{y}(t-1)} = 0.$$

$$\frac{\partial \hat{y}(t)}{\partial \hat{y}(t-1)} = 1, \quad \frac{\partial \hat{y}(t)}{\partial \hat{\theta}(t-1)} = v\cdot\Delta t\cdot\cos(\theta_{t-1})\cdot\frac{v}{L}\tan\gamma, \quad \frac{\partial \hat{y}(t)}{\partial \hat{x}(t-1)} = 0.$$

$\therefore\ g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$.

$$\therefore\ g(u_t, \mu_{t-1}) \to \begin{cases} \mu_{\theta t} = \mu_{\theta t-1} + \frac{v}{L}\tan\gamma\cdot\Delta t \\ \mu_{xt} = \mu_{xt-1} + v\cdot\cos\mu_{\theta t-1}\cdot\Delta t \\ \mu_{yt} = \mu_{yt-1} + v\cdot\sin\mu_{\theta t-1}\cdot\Delta t. \end{cases}$$

$\mu_{x^1 t} = \mu_{x^1 t-1}$, $\mu_{x^2 t} = \mu_{x^2 t-1}$ ...
$\mu_{y^1 t} = \mu_{y^1 t-1}$, $\mu_{y^2 t} = \mu_{y^2 t-1}$
$x^1, y^1, x^2, y^2 \cdots x^{12}, y^{12}$

$$G(t) = \begin{bmatrix} \underbrace{\begin{matrix} 1 & 0 & v\Delta t\cdot\frac{v}{L}\tan\gamma(-\sin(\theta_{t-1})) \\ 0 & 1 & v\Delta t\cdot\frac{v}{L}\tan\gamma\cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{matrix}}_{3\times 3} & 0 \\ 0 & \underbrace{\begin{matrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{matrix}}_{24\times 24 \to \text{Identity matrix}} \end{bmatrix}$$

! Because the motion model will not affect the location of landmarks.

$\therefore\ g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$.

For the ~~prediction step~~ motion updates in EKF filter, we have:

$\bar{\mu}_t = g(u_t, \mu_{t-1})$

$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$, R_t is a covariance matrix modifing uncertainy in the motion model.

For the same reason (the motion model will not affect the location of landmarks).

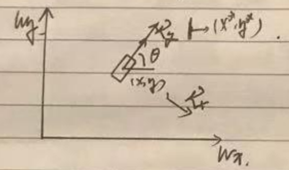
The R_t matrix has the same configuration as $G_t(\cdot)$.

$$R(t) = \begin{bmatrix} R_{3\times3} & O_{3\times 2N} \\ O_{2N\times3} & O_{2N\times2N} \end{bmatrix}.$$

In the case, we need to determine what values should be filled in $R_{3\times3}$, which are closely related to the hardwares of our robot (It is not good).

(2). Third, we need to ~~d~~ deal with the measurement model.

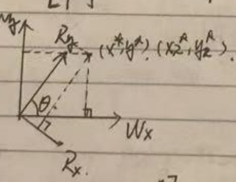
I draw a diagram to show the measurement model:



x^*, y^* denote the position of a landmark.
x, y, θ denote the position and orientation of our robot.
All x^*, y^*, x, y, θ are in the world coordinate.

Then, I use x_r^*, y_r^* to denote the position of a landmark in the robot coordinate which ~~co~~ corresponds to x^*, y^*.

$$\therefore \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = [R \ t] \begin{bmatrix} x_R^* \\ y_R^* \\ 1 \end{bmatrix} \rightarrow \text{we need to figure out what are } R, t.$$



$$\therefore y^* = (y_R^* - x_R^* \cdot \tan(\tfrac{\pi}{2} - \theta)) \cdot \sin\theta.$$

$$= \sin\theta \, y_R^* - x_R^* \cdot \cos\theta.$$

$$x^* = (y_R^* - x_R^* \cdot \tan(\tfrac{\pi}{2} - \theta)) \cdot \cos\theta + \frac{x}{\sin\theta}.$$

$$= \cos\theta \, y_R^* - x_R^* \cdot \frac{\cos^2\theta}{\sin\theta} + \frac{x}{\sin\theta}.$$

$$= \cos\theta \, y_R^* - x_R^* \frac{1 - \sin^2\theta}{\sin\theta} + \frac{x}{\sin\theta}$$

$$= x_R^* \cdot \sin\theta + \cos\theta \cdot y_R^*.$$

$$\begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta & x \\ -\cos\theta & \sin\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_R^* \\ y_R^* \\ 1 \end{bmatrix} \Longleftarrow$$

$\because t = \begin{bmatrix} x \\ y \end{bmatrix}$. (the position of robot in the world coordinate)

$\therefore$ We need to compare the x_R^*, y_R^* with the measurement $\begin{bmatrix} u \\ v \end{bmatrix}$, we ~~is~~ reverse the expression.

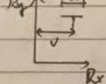
$$\therefore \begin{bmatrix} x_z^* \\ y_z^* \\ 1 \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta & x \\ -\cos\theta & \sin\theta & y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sin\theta & -\cos\theta & -x \\ \cos\theta & \sin\theta & -y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix}$$

Then, we need to find a relationship between $\begin{bmatrix} x_z^* \\ y_z^* \end{bmatrix}$ with the pixels value in the image plane.

Since we only have one camera, it is hard for us to use some algorithm to get the ~~physical~~ x_z^*, y_z^* with real units like meter, or centimeter.

Hence, I ~~therefore~~ use a trick which could greatly simplify the model.



I denote u as the width of landmarks in image plane and denote v as the difference between the center of landmark with the center of the image plane.

In common sense, if the landmark is far away from the camera, the ~~distance could be~~ width of landmark is very small:

$\alpha y_z^* = u$, α is a constant and I could use some data to find an α which fits these data best

Also, if the landmark is on the right side, the v could be larger.

$\beta x_z^* = v$, β could be calibrated, either.

$\therefore$ I build model which helps me find the relationship between the $\begin{bmatrix} u \\ v \end{bmatrix}$ in image plane and the variables $\begin{bmatrix} x \\ y \\ \theta \\ x^* \\ y^* \end{bmatrix}$ in the state vector:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sin\theta & -\cos\theta & -x \\ \cos\theta & \sin\theta & -y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \\ 1 \end{bmatrix} = h(x_t) \left(x_t = \begin{bmatrix} x \\ y \\ \theta \\ x^* \\ y^* \\ \vdots \end{bmatrix} \right)$$

It is obvious that the function is a nonlinear function, I have to linearize it:

$$u: \frac{\partial h(x_t)}{\partial x} = -\alpha, \quad \frac{\partial h(x_t)}{\partial y} = 0, \quad \frac{\partial h(x_t)}{\partial \theta} = \alpha x^* \cos\theta + \alpha y^* \sin\theta.$$

$$\frac{\partial h(x_t)}{\partial x^*} = \alpha \sin\theta \quad \frac{\partial h(x_t)}{\partial y^*} = -\alpha \cos\theta.$$

v: $\frac{\partial h(x_t)}{\partial x} = 0$, $\frac{\partial h(x_t)}{\partial y} = -\beta$, $\frac{\partial h(x_t)}{\partial \theta} = -x^* \beta \sin\theta + \beta y^* \cos\theta$

$\frac{\partial h(x_t)}{\partial x^*} = \beta \cos\theta$ $\frac{\partial h(x_t)}{\partial y^*} = \beta \sin\theta$

$\therefore h(x_t) \approx h(\bar{\mu}_t) + h'(\bar{\mu}_t)(x_t - \bar{\mu}_t)$,
$= h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)$.

$$H(t) = \begin{bmatrix} -\alpha & 0 & \alpha x^* \cos\theta + \alpha y^* \sin\theta & 0 \cdots 0 & \alpha \sin\theta & -\alpha\cos\theta & 0 \cdots 0 \\ 0 & -\beta & -x^* \beta \sin\theta + \beta y^* \cos\theta & 0 \cdots 0 & \beta\cos\theta & \beta \sin\theta & 0 \cdots 0 \end{bmatrix} \in R^{2 \times 27}$$

$\underbrace{\beta\cos\theta \quad \beta\sin\theta}_{\text{ith landmark } \binom{x^*}{y^*}}$

$\therefore$ For the measurement updates in EKF filter, we have:

$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z(t) - h(\bar{\mu}_t))$

$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$.

$z(t)$ represents the measurements $\begin{bmatrix} u \\ v \end{bmatrix}$. we obtain directly with decode function in zbar library.

$h(\bar{\mu}_t)$ represents the ideal measurement $\begin{bmatrix} u^* \\ v^* \end{bmatrix}$. we obtain based on the state variables and nonlinear model.

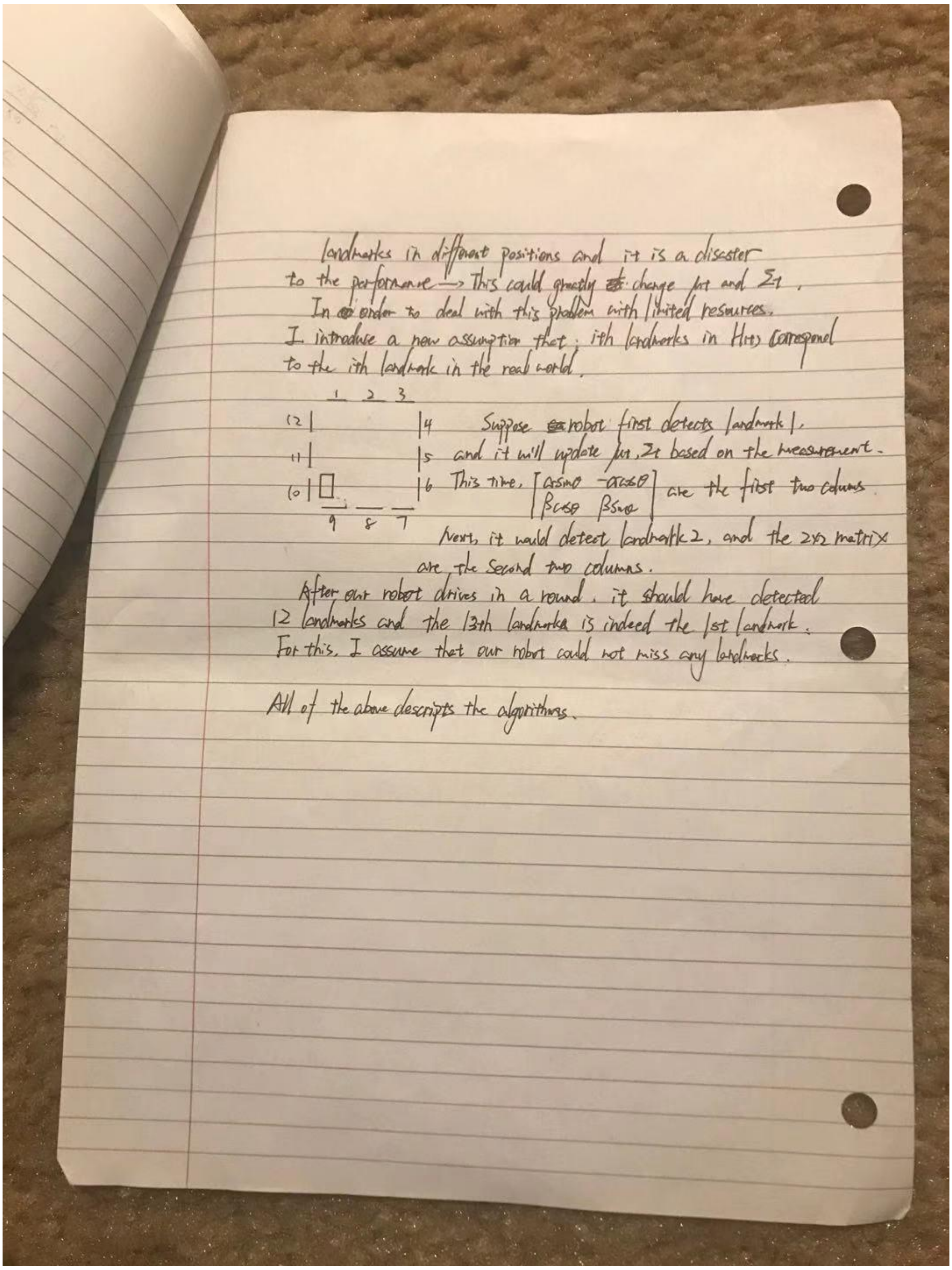
Q_t is a convariance matrix modifying uncertainty in $\begin{bmatrix} u \\ v \end{bmatrix}$, $Q_t \in R^{2 \times 2}$.

$Q_t = \begin{bmatrix} \theta_1^2 & \\ & \theta_2^2 \end{bmatrix}$, θ_1^2 and θ_2^2 are the variances for u and v, respectively.

(Under the assumption that u and v are two independent measurement)

$\rightarrow$ no correlation between them)

d). Finally, there is still one important problem: In the ederivation of $H(t)$ I use 'ith landmark' to match the measurement with one of all the landmarks in the state vectors (12). However, in the case we can not tell which one is the right one. The reason is that we only have 4 different landmarks, so there must be some landmarks which appears many times in different positions. Unfortunately, our picar can not tell the different between them. As a result, it may mismatch the same



landmarks in different positions and it is a disaster to the performance → This could greatly ~~change~~ change μ_t and Σ_t.

In ~~or~~ order to deal with this problem with limited resources. I introduce a new assumption that: ith landmarks in $H(t)$ correspond to the ith landmark in the real world.

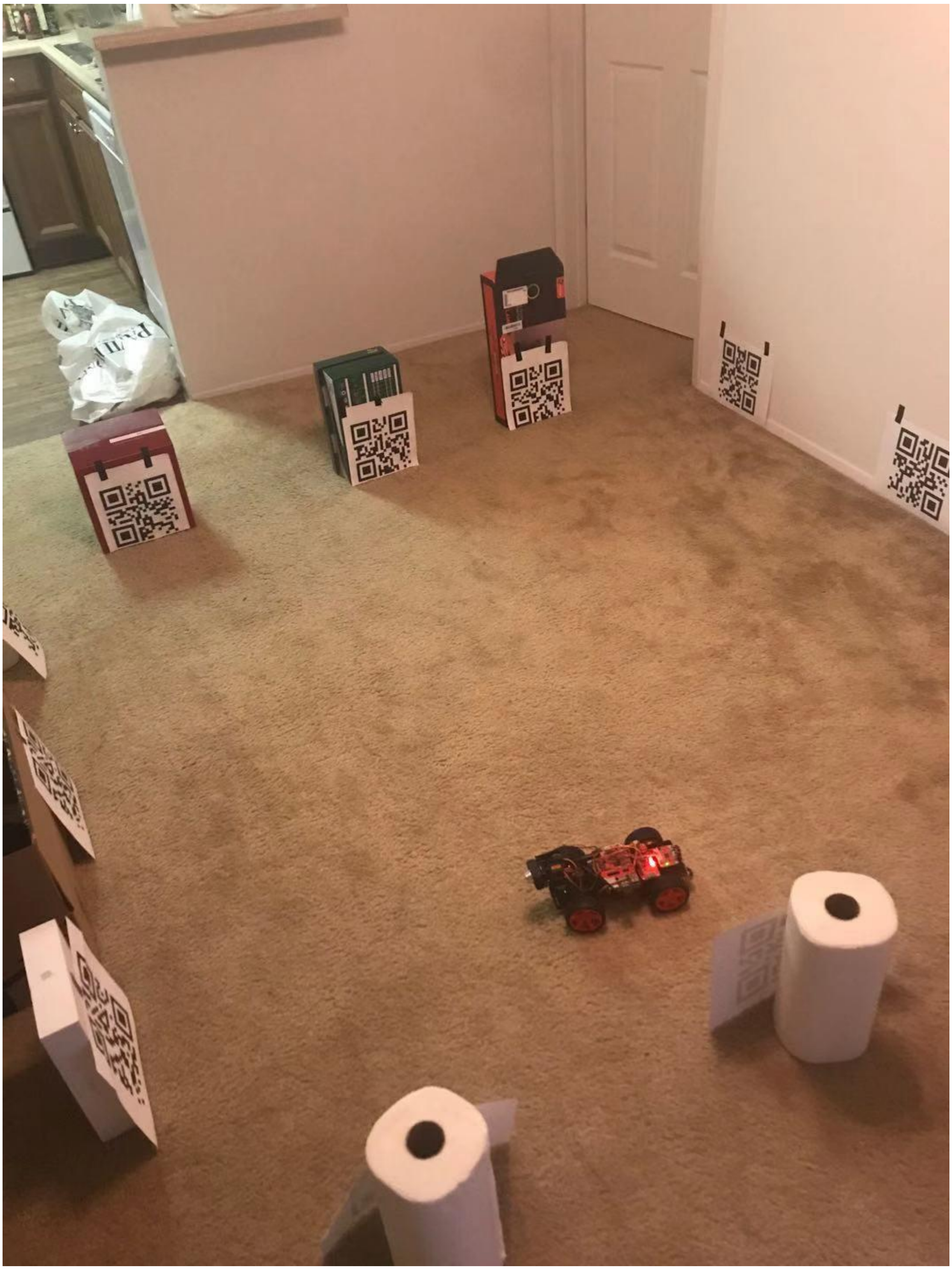
Suppose ~~a~~ robot first detects landmark 1. and it will update μ_t, Σ_t based on the measurement. This time, $\begin{bmatrix} \alpha \sin\theta & -\alpha\cos\theta \\ \beta\cos\theta & \beta\sin\theta \end{bmatrix}$ are the first two colums.

Next, it would detect landmark 2, and the 2x2 matrix are the second two columns.

After our robot drives in a round, it should have detected 12 landmarks and the 13th landmarka is indeed the 1st landmark. For this, I assume that our robot could not miss any landmarks.

All of the above descripts the algorithms.

These six pictures mainly describes the algorithms I use for this homework. As you can see, I write down all of the necessary derivatives for the final results.



This picture shows how I place the 12 QR landmarks.

Then, I will show how I calibrate the measurement model to get the alpha and belta which help me build the linear relationship between the measurements I directly read from decode function in zbar and the 2D positions of landmarks(x,y) in the robot coordinate.

First, I put our robot one meter in front of one of all the QR landmarks:

From the screenshot, you can see that the width of the landmark is 86.
Then I put the robot half a meter in front of the same landmark:

From this picture, you can see that the width of the landmark is 168.
Based on these two relations(1,86) and (0.5,168), I could tell the linear relationship between measurements and the depth as follow:

$$\frac{1}{u} = \frac{1}{86} * y$$

In the above equation, 1/u is what I could obtain from decode function and y is the distance between the landmark and our robot along y axis.
Hence, I successfully build a linear model which helps simplify our model.
You can see alpha=1/86 in my python codes.

Second, I put the robot between two landmarks. The two landmarks are sticked on a wall and the robot heads towards the wall.

From the picture, you can see that there are two landmarks, one is on the left and the other one is on the right.
The x-difference between the center of the left landmark with the image center is -138 pixels and the x-difference for the right one is 217.
Besides, the real distances are -0.2m and 0.3m for for the left and right landmarks, respectively.
Hence, I could tell the linear relationship:

$$v = 700 * x$$

In the above equation, v is what I could obtain from decode function and x is the distance between the landmark and our robot along x axis.
Hence, I successfully build another linear model which also helps simplify our model.
You can see belta=700 in my python codes.

II).The results obtained with a circular motion

The final values in state vector and convariance matrix are saved in two txt files, you could open them and see each value.
It is obvious that in a round, our robot at least miss three landmarks, since there are six values equal to 0, which means they haven't been detected. However, it is mainly due to the deficiency of the hardwares(like the cheap camera). And if our robot could run in more rounds, **this problem may be solved** but the data association could become another difficult problem since we have totally 12 landmarks with only four different tags. It is really hard to tell the difference between the landmarks with the same tag. If we could not match the measurements with the landmarks in the state vector succefully, it is a diaster ro run EKF and the results could be weired.

III).The results obtained with figure 8 motion

The final values in state vector and convariance matrix are saved in two txt files, you could open them and see each value.
For figure 8 motion, the good news is that there is only one landmark missed in a round, so maybe I could say it is easier for our robot to find landmarks when it runs in a figure 8. And if our robot run in more rounds, the data association problem become much more difficult to handle with. If we could deal with data association problem, the accuracy could improved a lot.

IV).Future work

If we want to improve the accuracy, we need to find a method for data association problem.