0.1 Unicycle Robots Model

The kinematic model each unicycle-type robot can be defined by the following iteration equation:

$$\begin{aligned} x_{k+1} &= x_k + v_k cos(\theta_k) \\ y_{k+1} &= y_k + v_k sin(\theta_k) \\ \theta_{k+1} &= \theta_k + \omega_k \end{aligned} \tag{1}$$

where $X = [x_k, y_k, \theta_k]^T$ is defined as the states vector, it denotes the global position and orientation vector of the robot at time k and $k \in \{1, 2, ..., N\}$. $u_k = [v_k, \omega_k]^T$ is defined as the control input at time k, v_k and ω_k stands for the linear and angular velocities, respectively.

With the state vector and input vector, the system state space model is defined as follow:

$$\begin{aligned} X_{k+1} &= f(X_k, u_k, w_k^1) \\ Z_{k+1} &= h(X_k, w_k^2) \end{aligned} \tag{2}$$

where $f(X_k, u_k, w_k^1)$ is the state transition function, and states can be observed through a measurement function $h(X_k, w_k^2)$. Both function could be nonlinear and w_k^1, w_k^2 are the corresponding perturbation on states and measurement respectively.

0.2 Simulation Setup

The simulation is done with Python and Pygame and the overview of the simulation is shown in the figure 1

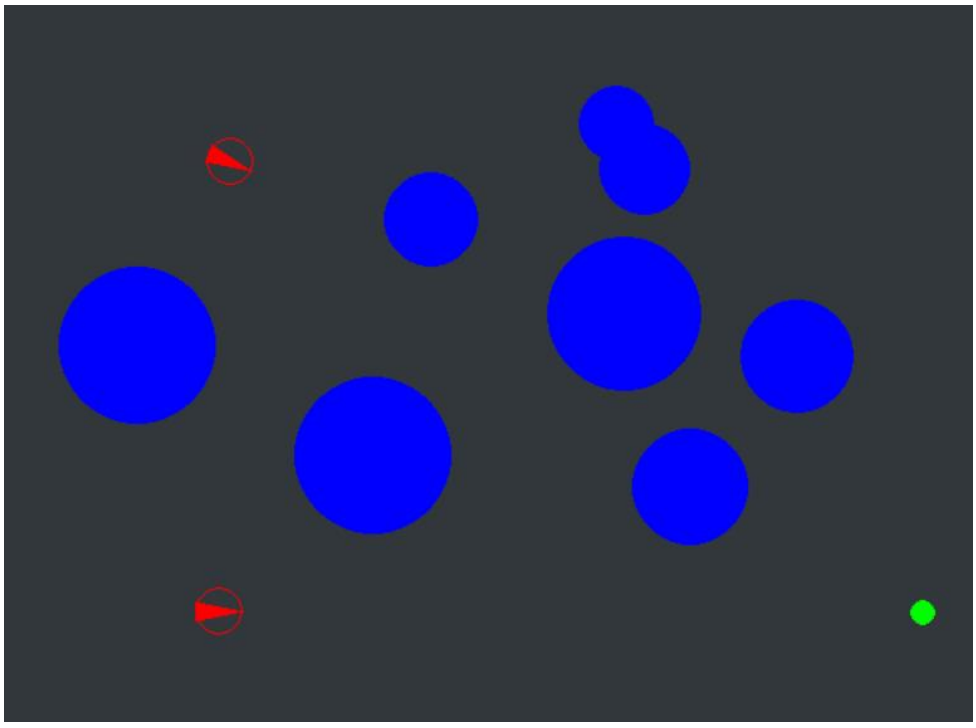

Figure 1: Simulation Overview

The experiment playground size is set as 640x480. For simplicity, only two moving robots(colored as red) are initialized in the simulation, other robots(colored as blue) are assumed remaining still during the experiment. Because of the biased measurement, the blue circles have different radius(would explain more later).

Two robots are initialized in position [100, 100], [100, 400] respectively, the termination point of these two robots are set at [600, 400], the other still robots are placed randomly.

These two robot are performing a simple go-to-goal task with a P controller, they following the equation below:

$$\begin{aligned} e_k &= [x_0 - x_k, y_0 - y_k] \\ v_k &= \|K_1 e_k\| \\ \phi &= \arctan(x_0 - x_k, y_0 - y_k) \\ \omega_k &= K_2 \arctan(sin(\phi - \theta_k), cos(\phi - \theta_k)) \end{aligned} \tag{3}$$

where $K = [K_1, K_2] = [-0.001, -0.001, 0.01]$. Since this control task require robots' position information, if the measurements perturbation present, the trajectory will deviate from the case without perturbation.

0.3 State Perturbation

The kinematic model of unicycle robot is ideal, while in reality the robots could encounter different situations leading the real position differs from the output of the model, such as wheel slipping or stuck.

A noise is add to control input to simulate above situations, the biased control input is defined as follows:

$$\begin{aligned} v_k &= v_k + a, a \in [-0.5v_k, 0.5v_k] \\ \omega_k &= \omega_k + b, b \in [-0.5\omega_k, 0.5\omega_k] \end{aligned} \tag{4}$$

0.4 Measurement Perturbation

Robots can record their historical estimated trajectory, they also receive position measurements from the top camera tracking system. We are assuming robots are receiving measurements from the Optitrack system in every time step.

The measurement perturbation consists of two part, noise and merge condition.

While the robot are standing still, the position readings from the camera have turbulence, which is described as below:

$$\begin{aligned} x_k &= x_k + 0.2a, a \in [-0.1x_k, 0.1x_k] \\ y_k &= y_k + 0.2b, b \in [-0.1y_k, 0.1y_k] \\ \omega_k &= \omega_k + 0.2c, c \in [-0.1\omega_k, 0.5\omega_k] \end{aligned} \tag{5}$$

The other perturbation is merging, this could happen if the distance between two robots are relatively small. When robots are getting too close to each other, it would be difficult for the camera tracking

system to identify each robot, then the tracking system would only give one position(belonging to one of the robot) as the output, since the other position are merged together.

In the simulation, if the distance between moving robot and other agents is small(less than 75 pixels), the tracking system will never give the right measurement(the worst case), it will give other agents' position as a wrong measurement, e.g. in the figure 1, the red triangular getting too close to blue cycles, the position measurement of the red triangular would become the center of the blue cycles.

If there are multiple objects nearby, the simulation would randomly selected one robot(blue one) from all possible options ans assign its center as the wrong measurement at each time step.

0.5 Extended Kalman Filter

In reality, the odometry in one robot could also have drift, therefore an additional information is needed for the correction, in this case, a global camera system is deployed. However, both of sensors are not perfect, their result can not be trusted easily, and adding more additional sensors would cause the system become over-complicated, as a result, the sensor fusion technique is necessary.

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to combine information, which contains noise and other inaccuracies and estimate the state during the transition, [2], [3], [4].

In this case the state transition is a nonlinear function, where the regular Kalman Filter are not suitable to solve. As a result, the extended Kalman Filter known as EKF [1] is used here.

In the simulation, the EKF can be divided into two part, the prediction part and update part, the calculation is listed below.

1. states estimation

   In the prediction part, the state can be estimated using the unicycle kinematic model, the model can be rewritten as follows:

$$X = \begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \gamma_k \end{bmatrix} + \begin{bmatrix} \cos\gamma_k * dk & 0 \\ \sin\gamma_k * dk & 0 \\ 0 & dk \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + \begin{bmatrix} \text{noise}_k \\ \text{noise}_k \\ \text{noise}_k \end{bmatrix} \tag{6}$$

   where dk is time step length and set as 1.

2. Predicted covariance of the state estimate

   Predicted covariance of the state estimate is defined as follows:

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{F}_{k+1}\boldsymbol{P}_{k|k}\boldsymbol{F}_{k+1}^{\top} + \boldsymbol{Q}_{k+1}$$
$$\boldsymbol{P}_{k|k} = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}, \boldsymbol{F}_{k+1} = \boldsymbol{F}_{k+1}^{\top} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \tag{7}$$

   In the predicted covariance, $\boldsymbol{P}_{k|k}$ is initialized as some guess value, and the Q matrix contains small value in the diagonal since in the simulation the camera is less trustful than the odometry.

3. Measurement Residual

$$\tilde{\boldsymbol{y}}_{k+1} = \boldsymbol{z}_{k+1} - h\left(\hat{\boldsymbol{x}}_{k+1|k}\right) \tag{8}$$

   The $\boldsymbol{z}_{k+1}$ is the actual measurements received from the camera system.

4. residual Covariance

$$\boldsymbol{S}_{k+1} = \boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top} + \boldsymbol{R}_{k+1} \tag{9}$$

$\boldsymbol{R}_{k+1}$ is the sensor measurement noise covariance matrix, which is assumed as identity matrix.

5. Near-optimal Kalman Gain

$$\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top}\boldsymbol{S}_{k+1}^{-1} \tag{10}$$

6. Updated State Estimate

$$\hat{\boldsymbol{x}}_{k+1|k+1} = \hat{\boldsymbol{x}}_{k+1|k} + \boldsymbol{K}_{k+1}\tilde{\boldsymbol{y}}_{k+1} \tag{11}$$

$\hat{\boldsymbol{x}}_{k+1|k+1}$ is the final output of the extended Kalman filter algorithm, this value can be used in the P controller to generate new control input u_{k+1}.

0.6 Result

Based on above simulation setup and EKF algorithm, several experiments was conducted, and the results are posted in the Figure 2 and 3, for simplicity, only one trajectory was plotted.

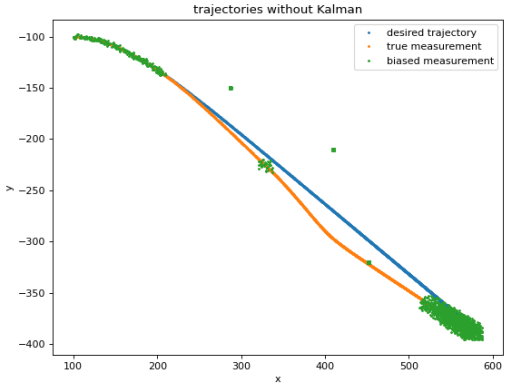

Figure 2: Robot Trajectory without EKF

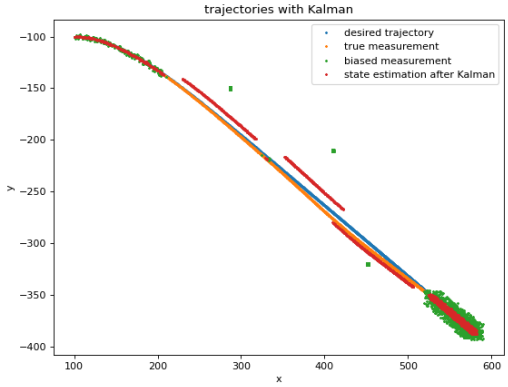

Figure 3: Robot Trajectory with EKF

In the figure 2, the green dots represent the measurements that the camera tracking system send to robots. An obvious discontinuous can be spotted, the reason for that is during this period of time, the robot was close to other object, then camera measurements were wrong.

The orange dots represent the actual trajectory of the robot, and blue dots stand for the desired trajectory where no state or measurement perturbation present.

From the figure 2, there is a large difference between true trajectory stored in the odometry and desired trajectory. That indicating with the wrong camera measurement, the robot will lost its position if they fully trust the information from the camera, since the control input is deviated.

In the figure 3, the green, orange and blue dots have the same meaning as in the figure 2, the red dots represent the output from the EKF. From this plot, the orange and blue dots have little difference, indicating with the help of EKF, the robot can overcome the bias of the camera and have better localization ability.

0.7 Summary

From the simulation result, the localization ability is improved if the robot takes both odometry and camera into account(using EKF to fuse these two sensors' readings).

I think this architecture could work, but I don't know how much we can trust the odometry in the robot, so some of the parameters should be tuned.

A Reference

[1] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*. IEEE, 2015, pp. 74–77.

[2] H. A. Patel and D. G. Thakore, "Moving object tracking using kalman filter," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 4, pp. 326–332, 2013.

[3] T. H. Dinh, M. D. Phung, T. H. Tran, and Q. V. Tran, "Localization of a unicycle-like mobile robot using LRF and omni-directional camera," *CoRR*, vol. abs/1611.09431, 2016. [Online]. Available: http://arxiv.org/abs/1611.09431

[4] J. N. Greenberg and X. Tan, "Dynamic optical localization of a mobile robot using kalman filtering-based position prediction," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2483–2492, 2020.