Kalman Filtering in Robot Soccer

Randal W. Beard

Updated: February 18, 2015

Contents

1 Derivation of the Continuous-Discrete Kalman Filter

1.1 Essentials from Probability Theory

Let $X = (x_1, \ldots, x_n)^\top$ be a random vector whose elements are random variables. The mean, or expected value of X, is denoted by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E\{x_1\} \\ \vdots \\ E\{x_n\} \end{pmatrix} = E\{X\},$$

where

$$E\{x_i\} = \int \xi f_i(\xi)\, d\xi$$

and $f(\cdot)$ is the probability density function for x_i. Given any pair of components x_i and x_j of X, we denote their covariance as

$$\textit{cov}(x_i, x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, that is,

$$\textit{var}(x_i) = cov(x_i, x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)(\xi - \mu_i)\}.$$

The standard deviation of x_i is the square root of the variance:

$$\textit{stdev}(x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

The covariances associated with a random vector X can be grouped into a matrix known as the covariance matrix:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{pmatrix} = E\{(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^\top\} = E\{XX^\top\} - \boldsymbol{\mu}\boldsymbol{\mu}^\top.$$

Note that $\Sigma = \Sigma^\top$ so that Σ is both symmetric and positive semi-definite, which implies that its eigenvalues are real and nonnegative.

The probability density function for a Gaussian random variable is given by

$$f_x(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{(x-\mu_x)^2}{\sigma_x^2}},$$

where μ_x is the mean of x and σ_x is the standard deviation. The vector equivalent is given by

$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left[-\frac{1}{2}(X-\boldsymbol{\mu})^\top \Sigma^{-1}(X-\boldsymbol{\mu})\right],$$

in which case we write

$$X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

and say that X is normally distributed with mean $\boldsymbol{\mu}$ and covariance Σ.

Figure 1 shows the level curves for a 2-D Gaussian random variable with different covariance matrices.

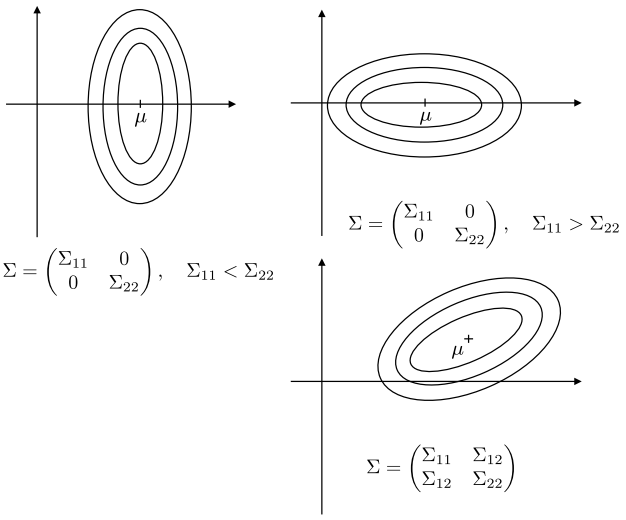

Figure 1: Level curves for the pdf of a 2-D Gaussian random variable.

1.2 Dynamic-observer Theory

The objective of this section is to briefly review observer theory, which serves as a precursor to our discussion on the Kalman filter. Suppose that we have a linear

time-invariant system modeled by the equations

$$\dot{x} = Ax + Bu$$
$$y = Cx.$$

A continuous-time observer for this system is given by the equation

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L\,(y - C\hat{x})}_{\text{correction due to sensor reading}}, \tag{1}$$

where $\hat{x}$ is the estimated value of x. Defining the observation error as $\tilde{x} = x - \hat{x}$ we find that

$$\dot{\tilde{x}} = (A - LC)\tilde{x},$$

which implies that the observation error decays exponentially to zero if L is chosen so that the eigenvalues of $A - LC$ are in the open left half of the complex plane.

In practice, the sensors are usually sampled and processed in digital hardware at sample rate T_s. How do we modify the observer equation shown in Equation (1) to account for sampled sensor readings? One approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu, \tag{2}$$

and then to update the estimate when a measurement is received using

$$\hat{x}^+ = \hat{x}^- + L(y(t_n) - C\hat{x}^-), \tag{3}$$

where t_n is the instant in time that the measurement is received and $\hat{x}^-$ is the state estimate produced by Equation (2) at time t_n. Equation (2) is then re-instantiated with initial conditions given by $\hat{x}^+$. If the system is nonlinear, then the propagation and update equations become

$$\dot{\hat{x}} = f(\hat{x}, u) \tag{4}$$
$$\hat{x}^+ = \hat{x}^- + L(y(t_n) - h(\hat{x}^-)). \tag{5}$$
$$\tag{6}$$

The observation process is shown graphically in Figure 2. Note that it is not necessary to have a fixed sample rate. The continuous-discrete observer can be implemented using Algorithm 1.

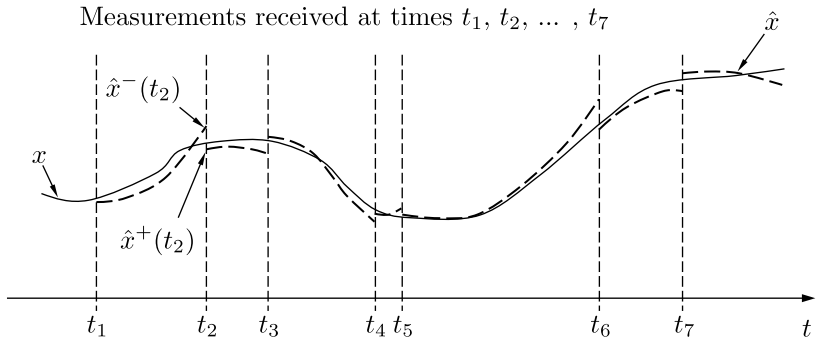

Figure 2: Time line for continuous-discrete dynamic observer. The vertical dashed lines indicate sample times at which measurements are received. In between measurements, the state is propagated using Equation (4). When a measurement is received, the state is updated using Equation (6).

A pseudo-code implementation of the continuous-discrete observer is shown in Algorithm 1. In Line 1 the state estimate is initialized to zero. If additional information is known, then the state can be initialized accordingly. The ODE in Equation (4) is propagated between samples with the for-loop in lines 4-6 using and Euler integration method. When a measurement is received, the state is updated using Equation (6) in line 8.

Algorithm 1 Continuous-Discrete Observer

1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate T_{out} that is less than the sample rates of the sensors.
3: At each sample time T_{out}:
4: **for** $i = 1$ to N **do** {Propagate the state equation.}
5: $\quad \hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
6: **end for**
7: **if** A measurement has been received from sensor i **then** {Measurement Update}
8: $\quad \hat{x} = \hat{x} + L_i \left(y_i - h_i(\hat{x})\right)$
9: **end if**

1.3 Derivation of the Continuous-Discrete Kalman Filter

The key parameter for the dynamic observer discussed in the previous section is the observer gain L. The Kalman filter and extended Kalman filters discussed in the remainder of this chapter are standard techniques for choosing L. If the process and measurement are linear, and the process and measurement noise are zero mean white Gaussian processes with known covariance matrices, then the Kalman filter gives the optimal gain, where the optimality criteria will be defined later in this section. There are several different forms for the Kalman filter, but the form that is particularly useful for MAV applications is the continuous-propagation, discrete-measurement Kalman filter.

We will assume that the (linear) system dynamics are given by

$$\begin{aligned} \dot{x} &= Ax + Bu + \xi \\ y[n] &= Cx[n] + \eta[n], \end{aligned} \tag{7}$$

where $y[n] = y(t_n)$ is the n^{th} sample of y, $x[n] = x(t_n)$ is the n^{th} sample of x, $\eta[n]$ is the measurement noise at time t_n, ξ is a zero-mean Gaussian random process with covariance Q, and $\eta[n]$ is a zero-mean Gaussian random variable with covariance R. The random process ξ is called the process noise and represents modeling error and disturbances on the system. The random variable η is called the measurement noise and represents noise on the sensors. The covariance R can usually be estimated from sensor calibration, but the covariance Q is generally unknown and therefore becomes a system gain that can be tuned to improve the performance of the observer. Note that the sample rate does not need to be fixed.

Similar to Equations (2) and (3) the continuous-discrete Kalman filter has the form

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu \\ \hat{x}^+ &= \hat{x}^- + L(y(t_n) - C\hat{x}^-). \end{aligned}$$

Define the estimation error as $\tilde{x} = x - \hat{x}$. The covariance of the estimation error at time t is given by

$$S(t) \triangleq E\{\tilde{x}(t)\tilde{x}(t)^\top\}. \tag{8}$$

Note that $S(t)$ is symmetric and positive semi-definite, therefore, its eigenvalues are real and non-negative. Also, small eigenvalues of $S(t)$ imply small variance, which implies low average estimation error. Therefore, we would like to choose

$L(t)$ to minimize the eigenvalues of $S(t)$. Recall that

$$tr(S) = \sum_{i=1}^{n} \lambda_i,$$

where $tr(S)$ is the trace of S, and λ_i are the eigenvalues of S. Therefore, minimizing $tr(S)$ minimizes the estimation error covariance. The Kalman filter is derived by finding L to minimize $tr(S)$.

1.3.1 Between Measurements:

Differentiating $\tilde{x}$ we get

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu + \xi - A\hat{x} - Bu \\ &= A\tilde{x} + \xi.\end{aligned}$$

Solving the differential equation with initial conditions $\tilde{x}_0$ we obtain

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}\xi(\tau)\,d\tau.$$

We can compute the evolution of the error covariance S as

$$\begin{aligned}\dot{S} &= \frac{d}{dt}E\{\tilde{x}\tilde{x}^\top\} \\ &= E\{\dot{\tilde{x}}\tilde{x}^\top + \tilde{x}\dot{\tilde{x}}^\top\} \\ &= E\left\{A\tilde{x}\tilde{x}^\top + \xi\tilde{x}^\top + \tilde{x}\tilde{x}^\top A^\top + \tilde{x}\xi^\top\right\} \\ &= AS + SA^\top + E\{\xi\tilde{x}^\top\} + E\{\tilde{x}\xi^\top\}.\end{aligned}$$

We can compute $E\{\tilde{x}\xi^\top\}$ as

$$\begin{aligned}E\{\tilde{x}\xi^\top\} &= E\{e^{(A-LC)t}\tilde{x}_0\xi^\top(t) + \int_0^t e^{(A-LC)(t-\tau)}\xi(\tau)\xi^\top(\tau)\,d\tau \\ &\qquad - \int_0^t e^{(A-LC)(t-\tau)}L\eta(\tau)\xi^\top(\tau)\,d\tau\} \\ &= \int_0^t e^{(A-LC)(t-\tau)}Q\delta(t-\tau)\,d\tau \\ &= \frac{1}{2}Q,\end{aligned}$$

where the $\frac{1}{2}$ is because we only use half of the area inside the delta function. Therefore, since Q is symmetric we have that S evolves between measurements as

$$\dot{S} = AS + SA^\top + Q.$$

1.3.2 At Measurements:

At a measurement, we have that

$$\begin{aligned}\tilde{x}^+ &= x - \hat{x}^+ \\ &= x - \hat{x}^- - L\left(Cx + \eta - C\hat{x}^-\right) \\ &= \tilde{x}^- - LC\tilde{x}^- - L\eta.\end{aligned}$$

We also have that

$$\begin{aligned}S^+ &= E\{\tilde{x}^+\tilde{x}^{+T}\} \\ &= E\left\{\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)^\top\right\} \\ &= E\left\{\tilde{x}^-\tilde{x}^{-\top} - \tilde{x}^-\tilde{x}^{-\top}C^\top L^\top - \tilde{x}^-\eta^\top L^\top\right. \\ &\quad - LC\tilde{x}^-\tilde{x}^{-\top} + LC\tilde{x}^-\tilde{x}^{-\top}C^\top L^\top + LC\tilde{x}^-\eta^\top L^\top \\ &\quad \left. -L\eta\tilde{x}^{-\top} + L\eta\tilde{x}^{-\top}C^\top L^\top + L\eta\eta^\top L^\top\right\} \\ &= S^- - S^-C^\top L^\top - LCS^- + LCS^-C^\top L^\top + LRL^\top, \qquad (9)\end{aligned}$$

where we have used the fact that since η and $\tilde{x}^-$ are independent, $E\left\{\tilde{x}^-\eta^\top L^\top\right\} = E\left\{L\eta\tilde{x}^{-\top}\right\} = 0$.

In the derivation that follows, we will need the following matrix relationships:

$$\frac{\partial}{\partial A}tr(BAD) = B^\top D^\top \qquad (10)$$

$$\frac{\partial}{\partial A}tr(ABA^\top) = 2AB, \text{ if } B = B^\top. \qquad (11)$$

Our objective is to pick L to minimize $\text{tr}(S^+)$. A necessary condition is that

$$\begin{aligned}\frac{\partial}{\partial L}\text{tr}(S^+) &= -S^-C^\top - S^-C^\top + 2LCS^-C^\top + 2LR = 0 \\ &\implies 2L(R + CS^-C^\top) = 2S^-C^\top \\ &\implies L = S^-C^\top(R + CS^-C^\top)^{-1}.\end{aligned}$$

Substituting into Equation (9) gives

$$\begin{aligned}
S^+ &= S^- + S^-C^\top(R + CS^-C^\top)^{-1}CS^- - S^-C^\top(R + CS^-C^\top)^{-1}CS^- \\
&\quad + S^-C^\top(R + CS^-C^\top)^{-1}(CS^-C^\top + R)(R + CS^-C^\top)^{-1}CS^- \\
&= S^- - S^-C^\top(R + CS^-C^\top)^{-1}CS^- \\
&= (I - S^-C^\top(R + CS^-C^\top)^{-1}C)S^- \\
&= (I - LC)S^-.
\end{aligned}$$

We can therefore summarize the Kalman filter as follows. In between measurements, propagate the equations

$$\begin{aligned}
\dot{\hat{x}} &= A\hat{x} + Bu \\
\dot{S} &= AS + SA^\top + Q,
\end{aligned}$$

where $\hat{x}$ is the estimate of the state, and S is the symmetric covariance matrix of the estimation error. When a measurement from the i^{th} sensor is received, update the state estimate and error covariance according to the equations

$$\begin{aligned}
L_i &= S^-C_i^\top(R_i + C_iS^-C_i^\top)^{-1} \\
S^+ &= (I - L_iC_i)S^- \\
\hat{x}^+ &= \hat{x}^- + L_i(y_i(t_n) - C_i\hat{x}^-),
\end{aligned} \tag{12}$$

where L_i is called the Kalman gain for sensor i.

We have assumed that the system propagation models and measurement model are linear. However, for many applications, including the applications discussed later in this chapter, the system propagation model and the measurement model are nonlinear. In other words, the model in Equation (7) becomes

$$\dot{x} = f(x, u) + \xi \tag{13}$$

$$y[n] = h(x[n], u[n]) + \eta[n]. \tag{14}$$

For this case, the state propagation and update laws use the nonlinear model, but the propagation and update of the error covariance use the Jacobian of f for A, and the Jacobian of h for C. The resulting algorithm is called the Extended Kalman Filter (EKF). The Jacobian of a vector function $g : \mathbb{R}^p \to \mathbb{R}^q$, where

$$g(\mathbf{x}) = \begin{pmatrix} g_1(x_1, x_2, \ldots, x_p) \\ g_2(x_1, x_2, \ldots, x_p) \\ \vdots \\ g_q(x_1, x_2, \ldots, x_p) \end{pmatrix},$$

is given by

$$\frac{\partial g}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_p} \\ \vdots & & \vdots \\ \frac{\partial g_q}{\partial x_1} & \cdots & \frac{\partial g_q}{\partial x_p} \end{pmatrix}.$$

The equations for the extended Kalman filter are given by
Between Measurements:

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}, u) \\ A &= \frac{\partial f}{\partial x}(\hat{x}, u) \\ \dot{S} &= AS + SA^\top + Q, \end{aligned} \tag{15}$$

At Measurements of the i^{th} sensor:

$$\begin{aligned} C_i &= \frac{\partial h_i}{\partial x}(\hat{x}) \\ L_i &= S^- C_i^\top (R_i + C_i S^- C_i^\top)^{-1} \\ S^+ &= (I - L_i C_i) S^- \\ \hat{x}^+ &= \hat{x}^- + L_i(y_i(t_n) - C_i \hat{x}^-). \end{aligned} \tag{16}$$

Pseudo-code for the EKF is shown in Algorithm 2. The state is initialized in Line 1. The propagation of the ODEs for $\hat{x}$ and S using an Euler integration scheme are given by the for-loop in Lines 4–8. The update equations for the i^{th} sensor are given in Lines 9–14.

2 Estimation Using an Overhead Camera

2.1 Self Estimate for Three Wheeled Robot

The objective of this section is to design an observer for the three wheeled robot based on Kalman filter theory. The Kalman filter will be used to

1. Filter noise from the vision system,
2. Estimate the current position and angle of the robot,
3. Estimate position and angle between measurements.

Algorithm 2 Continuous-Discrete Extended Kalman Filter

Initialize: $\hat{x} = 0$.
Pick an output sample rate T_{out} which is much less than the sample rates of the sensors.
At each sample time T_{out}:
for $i = 1$ to N **do** {Prediction Step}
  $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
  $A = \frac{\partial f}{\partial x}(\hat{x}, u)$
  $S = S + \left(\frac{T_{out}}{N}\right) \left(AS + SA^\top + Q\right)$
end for
if Measurement has been received from sensor i **then** {Measurement Update}
  $C_i = \frac{\partial h_i}{\partial x}(\hat{x}, u[n])$
  $L_i = SC_i^\top (R_i + C_i S C_i^\top)^{-1}$
  $S = (I - L_i C_i)S$
  $\hat{x} = \hat{x} + L_i \left(y_i[n] - h(\hat{x}, u[n])\right)$.
end if

2.1.1 Mathematical Model

Earlier in the semester we developed a velocity controller for the three wheeled robots. We will assume that the velocity controller have been implemented but that they are not perfect. In other words, the actual velocity of the robot is given by

$$\begin{aligned} V_x &= V_x^c + \delta V_x \\ V_y &= V_y^c + \delta V_y \\ \omega &= \omega^c + \delta\omega, \end{aligned}$$

where V_x^c, V_y^c, and ω^c are commanded velocities and δV_x, δV_y, and $\delta\omega$ are deviations. We will assume that the deviations are zero mean Gaussian random variables with covariance Q. Therefore, a kinematic model of the robot is given by

$$\begin{aligned} \dot{r}_x &= V_x = V_x^c + \delta V_x \\ \dot{r}_y &= V_y = V_y^c + \delta V_y \\ \dot{\theta} &= \omega = \omega^c + \delta\omega. \end{aligned}$$

Using the cameras to provide a measurement of the position and the angle, we get

$$\begin{aligned} y_{r_x} &= r_x + \eta_1 \\ y_{r_y} &= r_y + \eta_2 \\ y_\theta &= \theta + \eta_3, \end{aligned}$$

where $\eta = (\eta_1, \eta_2, \eta_3)^T$ is a zero mean Gaussian random variable with covariance R.

Therefore, the state space model is given by

$$\begin{pmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ \theta \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_x^c \\ V_y^c \\ \omega^c \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta V_x \\ \delta V_y \\ \delta\omega \end{pmatrix}$$

$$\begin{pmatrix} y_{r_x} \\ y_{r_y} \\ y_\omega \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_x \\ r_y \\ \theta \end{pmatrix} + \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix},$$

where $A = 0$, $B = I$, $G = I$, and $C = I$.

2.1.2 Kalman Filter

The continuous-discrete Kalman filter is given by the following algorithm.

System Model.

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + Gw \\ z(kT) &= C\hat{x}(kT) + v_k, \end{aligned}$$

where $w \sim \mathcal{N}(0, Q)$ and $v_k \sim \mathcal{N}(0, R)$.

Initialization. The first step is to initialize $\hat{x}$ and P to $\hat{x}_0$ and P_0.

Time update. In between measurements, the state estimate $\hat{x}$ and the covariance matrix P are propagated according to the differential equations

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu \\ \dot{P} &= AP + PA^T + GQG^T. \end{aligned}$$

In the lab, these differential equations will need to solved numerically. This can be done by approximating the time derivative with an Euler approximation, and propagating the following difference equations

$$\hat{x}[k+1] = \hat{x}[k] + T_p \left(A\hat{x} + Bu\right) \tag{17}$$

$$P[k+1] = P[k] + T_p \left(AP[k] + P[k]A + GQG^T\right), \tag{18}$$

where T_p is the sample rate of propagation, $T_p << T_s$, and T_s is the sample rate of the measurement.

Measurement update. When a measurement is received, we stop the time propagation and let $\hat{x}^-$ and P^- be the values of $\hat{x}$ as currently computed by Equations (23) and (25), respectively. The values of $\hat{x}$ and P are revised, according to the measurement, as follows:

$$L = P^- C^T [CP^- C^T + R]^{-1} \tag{19}$$

$$P = (I - LC)P^- \tag{20}$$

$$\hat{x} = \hat{x}^- + L(z_k - C\hat{x}^-). \tag{21}$$

2.1.3 Application to Three Wheeled Robots

The Kalman filter equations are particularly simple for three wheeled robots. In particular, notice that if Q and R are diagonal matrices, then P and L are always diagonal matrices. Making this assumption we get the following algorithm.

Initialization. Set $\hat{x} = (r_x(0), r_y(0), \theta(0))^T$ as estimated by averaging several camera samples. Set $P = diag([p_1, p_2, p_3]) = diag([1, 1, 1])$, or $p_1 = p_2 = p_3 = 1$.

Time update. In between measurements propagate the following differential equations:

$$\begin{aligned}
\dot{\hat{r}}_x &= V_x^c \\
\dot{\hat{r}}_y &= V_y^c \\
\dot{\hat{\theta}} &= \omega^c \\
\dot{p}_1 &= q_1 \\
\dot{p}_2 &= q_2 \\
\dot{p}_3 &= q_3.
\end{aligned}$$

The Euler approximation of these equations is given by

$$\begin{aligned}
\hat{r}_x[k+1] &= \hat{r}_x[k] + T_p V_x^c \\
\hat{r}_y[k+1] &= \hat{r}_y[k] + T_p V_y^c \\
\hat{\theta}[k+1] &= \hat{\theta}[k] + T_p \omega^c \\
p_1[k+1] &= p_1[k] + T_p q_1 \\
p_2[k+1] &= p_2[k] + T_p q_2 \\
p_3[k+1] &= p_3[k] + T_p q_3.
\end{aligned}$$

Measurement update. At the measurement we have

$$\begin{aligned}
l_1 &= \frac{p_1^-}{p_1^- + r_1} \\
l_2 &= \frac{p_2^-}{p_2^- + r_2} \\
l_3 &= \frac{p_3^-}{p_3^- + r_3} \\
p_1 &= (1 - l_1) p_1^- \\
p_2 &= (1 - l_2) p_2^- \\
p_3 &= (1 - l_3) p_3^- \\
\hat{r}_x &= \hat{r}_x^- + l_1 (y_{p_x} - \hat{r}_x^-) \\
\hat{r}_y &= \hat{r}_y^- + l_2 (y_{p_y} - \hat{r}_y^-) \\
\hat{\theta} &= \hat{\theta}^- + l_3 (y_\theta - \hat{\theta}^-).
\end{aligned}$$

2.2 Estimating other robots

2.3 Ball Prediction with Overhead Camera

2.4 Wall Bounces

The objective of this section is to describe a simple technique to predict future locations of the ball. When the ball is not in contact with the walls or another robot, then we will assume that there is no external force acting upon it (this assumes no air resistance or rolling friction). Under this assumption, by Newton's second law we have

$$m\ddot{\mathbf{b}} = 0,$$

where m is the mass of the ball and $\mathbf{b}$ is its position vector. Dividing by m and integrating once gives

$$\dot{\mathbf{b}}(t) = \dot{\mathbf{b}}(t_1),$$

where $t_1 \leq t$. Integrating again gives

$$\mathbf{b}(t) = \mathbf{b}(t_1) + \dot{\mathbf{b}}(t_1)(t - t_1).$$

Therefore if we measure the ball position at successive frames, then, in the absence of walls and obstacles, we could predict the ball position at for all future times as

$$\mathbf{b}(\tilde{t}) = \mathbf{b}[k] + \frac{\mathbf{b}[k] - \mathbf{b}[k-1]}{T}\tilde{t},$$

where $\tilde{t}$ is interpreted as future time from the current instance.

The ball may, however, collide with walls. We will assume perfectly elastic collisions, i.e., no energy loss. A property of elastic collisions is that the incidence angle is equal to the departure angle as shown in Figure 3.

We will derive the correction factor for the top wall, and leave the walls to the reader. Let YMAX denote the maximum Y coordinate of the soccer field. If $b_y(\tilde{t}) >$ YMAX, then the predicted position of the ball must be corrected to account for the interaction with the ball. Letting $d = b_y(\tilde{t}) - \text{YMAX}$ and assuming that the collision with the ball is elastic, the corrected predicted position of the ball is

$$\begin{aligned} b_y^+(\tilde{t}) &= b_y^-(\tilde{t}) - 2d \\ &= b_y^-(\tilde{t}) - 2b_y^-(\tilde{t}) + 2\ \text{YMAX} \\ &= 2\ \text{YMAX} - b_y^-(\tilde{t}), \end{aligned}$$

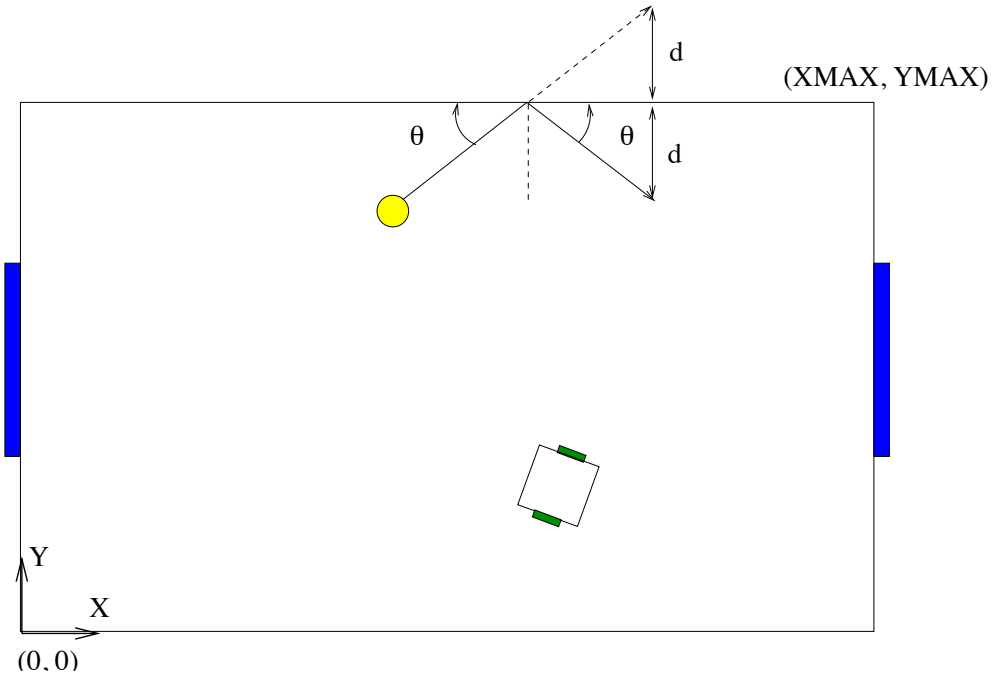

Figure 3: Elastic collision.

where $b_y^-(\tilde{t})$ denotes the predicted y-position of the ball before correcting for the collision, and $b_y^+(\tilde{t})$ denotes the predicted y-position of the ball after correcting for the collision.

You may also want to take into account robot collisions. In particular it would be nice to know how your robot will effect the ball. As a precursor to robot collisions, lets rework the wall collision equations, putting them into a more general form. Let $\dot{\mathbf{b}}^-$ be the velocity of the ball before the collision, and let $\dot{\mathbf{b}}^+$ be the velocity of the ball after the collision. Since the collision is elastic, the magnitude of $\dot{\mathbf{b}}^+$ equals the magnitude of $\dot{\mathbf{b}}^-$, however the direction has changed by an angle of 2θ. Therefore

$$\dot{\mathbf{b}}^+ = R(2\theta)\dot{\mathbf{b}}^-,$$

where $R(\cdot)$ is the rotation matrix given in Equation (**??**). Therefore, the ball prediction equation after one bounce becomes

$$\mathbf{b}(\tilde{t}) = \mathbf{b}[k] + t_{coll}\dot{\mathbf{b}}[k] + (\tilde{t} - t_{coll})R(2\theta)\dot{\mathbf{b}}[k],$$

where t_{coll} is the predicted time of collision. These equations can be extended to multiple bounces in a straightforward manner.

Now suppose that the ball collides with a robot that has velocity vector $\mathbf{v}$ as shown in Figure 4. The collision imparts velocity to the ball. Since the ball dynamics are linear, by superposition the new velocity of the ball will be

$$\dot{\mathbf{b}}^+ = R(2\theta)\dot{\mathbf{b}}^- + \mathbf{v},$$

where θ is the angle of incidence of collision with the face of the robot.

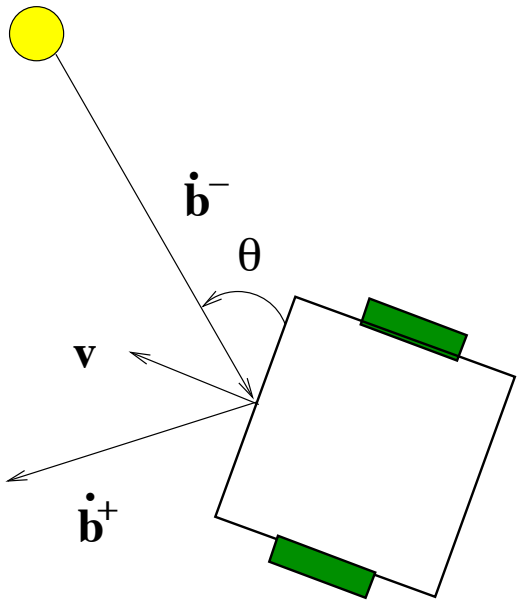

Figure 4: Ball-robot collision

2.4.1 Mathematical Model

When the ball is not in contact with a wall or another robot, its dynamics are governed by Newton's second law:

$$m\ddot{\mathbf{b}} = \mathbf{F},$$

where $\mathbf{b} \triangleq (b_x, b_y)^T$ is the position of the ball in world coordinates, m is the mass of the ball, and $\mathbf{F} \triangleq (F_x, F_y)^T$ is the external force exerted on the ball. The external forces will be composed of a term due to static friction term and a variety of other forces, which we will assume to be small. The force due to static

friction is constant, assuming the ball has a non-zero velocity, and in the direction opposing motion. The static friction term can be expressed as

$$\mathbf{F}_s = -\mu \frac{\dot{\mathbf{b}}}{\left\| \dot{\mathbf{b}} \right\|}.$$

Therefore the total force is

$$\mathbf{F} = -\mu \frac{\dot{\mathbf{b}}}{\left\| \dot{\mathbf{b}} \right\|} + \delta \mathbf{F},$$

where we assume that $\delta \mathbf{F}$ can be modeled as a zero mean, Gaussian random variable with covariance Q. In component form we have

$$\begin{pmatrix} \ddot{b}_x \\ \ddot{b}_y \end{pmatrix} = -\frac{\mu}{m} \begin{pmatrix} \frac{\dot{b}_x}{\sqrt{\dot{b}_x^2 + \dot{b}_y^2}} \\ \frac{\dot{b}_y}{\sqrt{\dot{b}_x^2 + \dot{b}_y^2}} \end{pmatrix} + \frac{1}{m} \begin{pmatrix} \delta F_x \\ \delta F_y \end{pmatrix}.$$

The constant $\frac{\mu}{m}$ is unknown. A common trick is to assume that this constant is also a state of the system and then use the Kalman filter to predict the constant, along with the other states. The dynamics for a constant are

$$\frac{d(\mu/m)}{dt} = 0.$$

Defining the state vector as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \triangleq \begin{pmatrix} b_x \\ b_y \\ \dot{b}_x \\ \dot{b}_y \\ \frac{\mu}{m} \end{pmatrix},$$

and the process disturbance as

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \frac{1}{m} \begin{pmatrix} \delta F_x \\ \delta F_y \end{pmatrix},$$

we can write the nonlinear state space equations as

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ -\frac{x_3 x_5}{\sqrt{x_3^2 + x_4^2}} \\ -\frac{x_4 x_5}{\sqrt{x_3^2 + x_4^2}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{22}$$

Assuming that an overhead camera is used to measure $\mathbf{b}$, the output equation is

$$\mathbf{y} = \begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \mathbf{x} + \begin{pmatrix} r_x \\ r_y \end{pmatrix},$$

where $\mathbf{r} \stackrel{\triangle}{=} (r_x, r_y)^T$ is the measurement noise and is assumed to be a zero mean Gaussian random variable with covariance R.

The application of the extended Kalman filter will require the computation of the Jacobian of f:

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} & \frac{\partial f_1}{\partial x_5} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} & \frac{\partial f_2}{\partial x_5} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} & \frac{\partial f_3}{\partial x_5} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} & \frac{\partial f_4}{\partial x_5} \\ \frac{\partial f_5}{\partial x_1} & \frac{\partial f_5}{\partial x_2} & \frac{\partial f_5}{\partial x_3} & \frac{\partial f_5}{\partial x_4} & \frac{\partial f_5}{\partial x_5} \end{pmatrix}.$$

Given f in Equation (22) we have

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{x_5}{\sqrt{x_3^2+x_4^2}} + \frac{x_3^2 x_5}{(x_3^2+x_4^2)^{3/2}} & \frac{x_3 x_4 x_5}{(x_3^2+x_4^2)^{3/2}} & -\frac{x_3}{\sqrt{x_3^2+x_4^2}} \\ 0 & 0 & \frac{x_3 x_4 x_5}{(x_3^2+x_4^2)^{3/2}} & -\frac{x_5}{\sqrt{x_3^2+x_4^2}} + \frac{x_4^2 x_5}{(x_3^2+x_4^2)^{3/2}} & -\frac{x_4}{\sqrt{x_3^2+x_4^2}} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

2.4.2 Extended Kalman Filter

The continuous-discrete extended Kalman filter is given by the following algorithm.

System Model.

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}, u) + Gq \\ z(kT) &= C\hat{x}(kT) + r_k, \end{aligned}$$

where $q \sim \mathcal{N}(0, Q)$ and $r_k \sim \mathcal{N}(0, R)$.

Initialization. The first step is to initialize $\hat{x}$ and P to $\hat{x}_0$ and P_0.

Time update. In between measurements, the state estimate $\hat{x}$ and the covariance matrix P are propagated according to the differential equations

$$\begin{aligned}\dot{\hat{x}} &= f(\hat{x}, u)\\ A &= \left.\frac{\partial f}{\partial x}\right|_{(x=\hat{x},u)}\\ \dot{P} &= AP + PA^T + GQG^T.\end{aligned}$$

In simulation and in the lab, these differential equations will need to solved numerically. This can be done by approximating the time derivative with an Euler approximation, and propagating the following difference equations

$$\hat{x}[k+1] = \hat{x}[k] + T_p\left(f(\hat{x}, u[k])\right) \tag{23}$$

$$A = \left.\frac{\partial f}{\partial x}\right|_{(x=\hat{x[k+1]},u[k])} \tag{24}$$

$$P[k+1] = P[k] + T_p\left(AP[k] + P[k]A^T + GQG^T\right), \tag{25}$$

where T_p is the sample rate of propagation, $T_p << T_s$, and T_s is the sample rate of the measurement.

Measurement update. When a measurement is received, we stop the time propagation and let $\hat{x}^-$ and P^- be the values of $\hat{x}$ as currently computed by Equations (23) and (25), respectively. The values of $\hat{x}$ and P are revised, according to the measurement, as follows:

$$L = P^-C^T[CP^-C^T + R]^{-1} \tag{26}$$

$$P = (I - LC)P^- \tag{27}$$

$$\hat{x} = \hat{x}^- + L(z_k - C\hat{x}^-). \tag{28}$$

2.4.3 Prediction Equations

If at time t, we would like to predict the state at time $t + T$, where $T > 0$, then we need to add an additional step. In particular we need to integrate the system equations

$$\dot{\hat{x}} = f(\hat{x}, \hat{u})$$

with initial condition $\hat{x}(t)$ for T seconds into the future.

Pseudo code for the complete algorithm is shown below.

Algorithm 3 Ball Predictor

Input: (1) Measurement z, (2) Prediction time T.
In between measurements:
for $i = 1$ to N **do**
    Compute $\hat{x} = \hat{x} + \frac{T_s}{N} f(\hat{x}, u)$
    Correct $\hat{x}$ to take into account wall bounces.
    Compute $A = \frac{\partial f(\hat{x},u)}{\partial x}$
    Compute $P = P + \frac{T_s}{n} \left(AP + PA^T + GQG^T\right)$
end for
At a measurement
        $L = PC^T \left(CPC^T + R\right)^{-1}$
        $P = (I - LC)P$
        $\hat{x} = \hat{x} + L\left(z - C\hat{x}\right)$
        Correct $\hat{x}$ to take into account wall bounces.
At each time t, predict ahead T seconds given the current estimate.
        $\hat{x}_p = \hat{x}$
while $t_p < T$ **do**
            Compute $\hat{x}_p = \hat{x}_p + \frac{T_s}{N} f(\hat{x}_p, u_p)$
            Correct $\hat{x}_p$ to take into account wall bounces.
            Update $t_p = t_p + \frac{T_s}{N}$
end while

3 Estimation Using Range and Bearing Measurements of Known Markers

3.1 Self Configuration

In this section we show how the Kalman filter can be used to estimate the position and orientation of a robot given that the robot has range and bearing estimates. This situation occurs in the robot soccer scenario where each robot has an RGBD sensor that can return range an bearing to markers at known positions.

Let $\mathbf{m}_i \stackrel{\triangle}{=} \begin{pmatrix} m_x, & m_y \end{pmatrix}^\top$ denote the known position of the i^{th} marker, and let $\mathbf{r} \stackrel{\triangle}{=} \begin{pmatrix} r_x, & r_y \end{pmatrix}^\top$ denote the unknown position of the robot, and let ϕ denote the unknown orientation of the robot. Figure 5 shows the geometry. The state of the

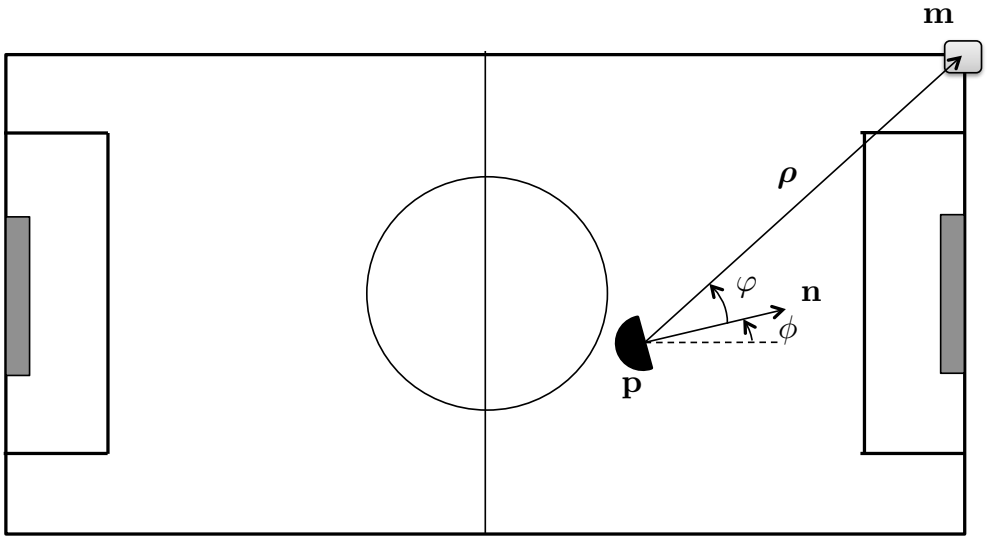

Figure 5: Geometry for the Robot Soccer Estimation Scenario.

robot is its position and orientation and will be denoted by

$$\mathbf{x} = \begin{pmatrix} r_x \\ r_y \\ \phi \end{pmatrix}.$$

The objective is to estimate $\mathbf{x}$ given measurements of the range and the bearing.

The dynamics of the robot are given by

$$\begin{pmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix}, \tag{29}$$

where the input $\mathbf{u} = (v_x, v_y, \omega)^\top$ is assumed to be known. In the notation of Equation (13), $f(x, u) = u$ and $A = \frac{\partial f}{\partial x} = \mathbf{0}_3$, where $\mathbf{0}_3$ is the 3×3 matrix of zeros.

We will assume that the computer vision software on the robot is able to measure the range to each marker we we denote by $\|\boldsymbol{\rho}\|$, and the bearing to each marker which we denote by φ. Let the vector

$$\boldsymbol{\rho} \stackrel{\triangle}{=} \mathbf{m} - \mathbf{r} = \begin{pmatrix} m_x - r_x \\ m_y - r_y \end{pmatrix}$$

denote the vector from the robot to the marker $\mathbf{m}$. The range is therefore given by

$$h_r(\mathbf{x}) \stackrel{\triangle}{=} \|\boldsymbol{\rho}\| = \sqrt{(m_x - r_x)^2 + (m_y - r_y)^2}.$$

The bearing angle from robot $\mathbf{r}$ to marker $\mathbf{m}$ can be found by taking the cross product between the normal vector $\mathbf{n}$ shown in Figure **??** and the line-of-sight vector $\boldsymbol{\rho}$. From the property of the cross product we have that

$$\mathbf{n} \times \boldsymbol{\rho} = \|\mathbf{n}\| \, \|\boldsymbol{\rho}\| \sin\varphi \, \hat{\mathbf{z}},$$

were $\hat{\mathbf{z}}$ is the unit vector perpendicular to both $\mathbf{n}$ and $\boldsymbol{\rho}$. If $\mathbf{n}$ is a unit vector, then the bearing angle ϕ is given by

$$\varphi = \sin^{-1}\left(\frac{n_x \rho_y - n_y \rho_x}{\|\boldsymbol{\rho}\|}\right).$$

For our situation we have that $\mathbf{n} = (\cos\phi, \sin\phi)^\top$. Therefore the bearing angle as a function of robot state is given by

$$h_b(\mathbf{x}) \stackrel{\triangle}{=} \varphi = \sin^{-1}\left(\frac{(m_y - r_y)\cos\phi - (m_x - r_x)\sin\phi}{\sqrt{(m_x - r_x)^2 + (m_y - r_y)^2}}\right).$$

There are therefore two measurements associated with each marker. Using the notation of Equation (14) the measurements are given by

$$\begin{aligned} y_r &= h_r(\mathbf{x}) + \eta_r \\ y_b &= h_b(\mathbf{x}) + \eta_b, \end{aligned}$$

where $\eta_r \sim \mathcal{N}(0, R_r)$ and $\eta_b \sim \mathcal{N}(0, R_b)$. To implement the Kalman filter measurement update equations (12), we need the Jacobians of h_r and h_b. After some algebra, we get

$$\begin{aligned} C_r = \frac{\partial h_r}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_r}{\partial r_x}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial r_y}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \phi}(\mathbf{x}) \right) \\ &= \left(\tfrac{-\rho_x}{\|\boldsymbol{\rho}\|}, \quad \tfrac{-\rho_y}{\|\boldsymbol{\rho}\|}, \quad 0 \right). \end{aligned}$$

Similarly, for h_b we get

$$\begin{aligned} C_b = \frac{\partial h_b}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_b}{\partial r_x}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial r_y}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \phi}(\mathbf{x}) \right) \\ &= \text{sign}\left(\rho_x \cos\phi + \rho_y \sin\phi\right) \left(\tfrac{\rho_y}{\|\boldsymbol{\rho}\|^2}, \quad \tfrac{-\rho_x}{\|\boldsymbol{\rho}\|^2}, \quad -1 \right), \end{aligned}$$

where the sign function is defined as

$$\text{sign}(\zeta) = \begin{cases} 1 & \text{if} \quad \zeta > 0 \\ 0 & \text{if} \quad \zeta = 0 \\ -1 & \text{if} \quad \zeta < 0 \end{cases}.$$

Matlab code implementing a Kalman filter for this scenarios is given below.

```
1     persistent xhat % state estimate = [r_x \\ r_y \\ phi]
2     persistent S % covariance of the estimate
3
4     % initialize the state and the covariance
5     %       <- parameters to be tuned.
6     if t==0,
7         xhat = [-P.field_length/6; 0; -pi/2];
8         S = diag([P.field_length/12; P.field_width/12; pi/20]);
9     end
10
11    % observer gains
12    %       <- parameters to be tuned for performance.
13    P.observer_Q_self = 100*eye(3);
14    P.observer_R_range = .1;
15    P.observer_R_bearing = 1*pi/180;
16
17    % estimate between measurements
18    N = 10;
19    for i=1:N, % solve differential equations using Euler method
```

```
        xhat = xhat + (P.control_sample_rate/N)*(velocity);
        S = S + (P.control_sample_rate/N)*(P.observer_Q_self);
    end

    % measurement updates
    for i=1:P.num_markers,
        % range measurement
        if vision.marker(1,i)~=P.camera_out_of_range,
            rho = P.marker(:,i)-xhat(1:2);
            Rho = norm(rho);
            h = Rho;
            C = [-rho(1), -rho(2), 0]/Rho;
            L = S*C'/(P.observer_R_range+C*S*C');
            S = (eye(3)-L*C)*S;
            xhat = xhat + L*(vision.marker(1,i)-h);
        end
        % bearing measurement
        if vision.marker(2,i)~=P.camera_out_of_range,
            rho = P.marker(:,i)-xhat(1:2);
            Rho = norm(rho);
            phi = xhat(3);
            h = asin((rho(2)*cos(phi)-rho(1)*sin(phi))/Rho);
            C = sign(rho(1)*cos(phi)+rho(2)*sin(phi))...
                *[rho(2)/(Rho^2), -rho(1)/(Rho^2), -1];
            L = S*C'/(P.observer_R_bearing+C*S*C');
            S = (eye(3)-L*C)*S;
            xhat = xhat + L*(vision.marker(2,i)-h);
        end
    end
```

3.2 Ball Estimate Using Range and Bearing Measurements

In this section we show how the Kalman filter can be used to estimate the position, velocity, and drag coefficient of the ball, given the known position of the robot carrying the camera.

As in the privious section, let $\mathbf{r} \stackrel{\triangle}{=} \begin{pmatrix} r_x, & r_y \end{pmatrix}^\top$ be the (assumed) known position of the robot/camera, and again let $\mathbf{n} = (n_x, n_y)^\top = (\cos\phi, \sin\phi)^\top$ be the (known) normal vector along the camera line of sight. The ball position will be denoted by $\mathbf{b} = (b_x, b_y)^\top$, and the ball velocity is $\dot{\mathbf{b}} = (\dot{b}_x, \dot{b}_y)^\top$. We will assume that the dynamic equation of motion for the ball is given by

$$\ddot{\mathbf{b}} = -\mu\dot{\mathbf{b}} + \mathbf{f}_{ext},$$

where μ is the coefficient of rolling friction due to the carpet, and $\mathbf{f}_{ext}$ is any externally applied force due to the walls, or the other robots. Since μ is unknown, the objective is to estimate the ball position $\mathbf{b}$, the ball velocity $\dot{\mathbf{b}}$, and the friction coefficient μ, given the range and bearing measurements to the ball. Accordingly, define the state of the system as

$$x = \begin{pmatrix} b_x \\ b_y \\ \dot{b}_x \\ \dot{b}_y \\ \mu \end{pmatrix}.$$

Since μ is a constant, the differential equation that describes the evolution of x is given by

$$\dot{x} = f(x) + \xi \stackrel{\triangle}{=} \begin{pmatrix} \dot{b}_x \\ \dot{b}_y \\ -\mu\dot{b}_x \\ -\mu\dot{b}_y \\ 0 \end{pmatrix},$$

where the Jacobian of f is given by

$$A = \frac{\partial f}{\partial x}(x) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\mu & 0 & \dot{b}_x \\ 0 & 0 & 0 & -\mu & \dot{b}_y \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

We will assume that the computer vision software on the robot is able to measure the range to the ball which we denote by $\|\boldsymbol{\rho}\|$, and the bearing to the ball which we denote by φ. Let the vector

$$\boldsymbol{\rho} \stackrel{\triangle}{=} \mathbf{b} - \mathbf{r} = \begin{pmatrix} b_x - r_x \\ b_y - r_y \end{pmatrix}$$

denote the vector from the robot to the ball $\mathbf{b}$. The range is therefore given by

$$h_r(\mathbf{x}) \stackrel{\triangle}{=} \|\boldsymbol{\rho}\| = \sqrt{(b_x - r_x)^2 + (b_y - r_y)^2}.$$

The bearing angle from robot $\mathbf{r}$ to ball $\mathbf{b}$ can be found by taking the cross product between the normal vector $\mathbf{n} = (\cos\phi, \sin\phi)^\top$ and the line-of-sight vector $\boldsymbol{\rho}$. From the property of the cross product we have that

$$\mathbf{n} \times \boldsymbol{\rho} = \|\mathbf{n}\| \, \|\boldsymbol{\rho}\| \sin\varphi \; \hat{\mathbf{z}},$$

were $\hat{\mathbf{z}}$ is the unit vector perpendicular to both $\mathbf{n}$ and $\boldsymbol{\rho}$. If $\mathbf{n}$ is a unit vector, then the bearing angle ϕ is given by

$$\varphi = \sin^{-1}\left(\frac{n_x\rho_y - n_y\rho_x}{\|\boldsymbol{\rho}\|}\right).$$

Therefore the bearing angle as a function of robot state is given by

$$h_b(\mathbf{x}) \stackrel{\triangle}{=} \varphi = \sin^{-1}\left(\frac{(b_y - r_y)\cos\phi - (b_x - r_x)\sin\phi}{\sqrt{(b_x - r_x)^2 + (b_y - r_y)^2}}\right).$$

There are therefore two measurements associated with the ball. Using the notation of Equation (14) the measurements are given by

$$\begin{aligned} y_r &= h_r(\mathbf{x}) + \eta_r \\ y_b &= h_b(\mathbf{x}) + \eta_b, \end{aligned}$$

where $\eta_r \sim \mathcal{N}(0, R_r)$ and $\eta_b \sim \mathcal{N}(0, R_b)$. To implement the Kalman filter measurement update equations (12), we need the Jacobians of h_r and h_b. After some algebra, we get

$$\begin{aligned} C_r = \frac{\partial h_r}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_r}{\partial b_x}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial b_y}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \dot{b}_x}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \dot{b}_y}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \mu}(\mathbf{x})\right) \\ &= \left(\tfrac{\rho_x}{\|\boldsymbol{\rho}\|}, \quad \tfrac{\rho_y}{\|\boldsymbol{\rho}\|}, \quad 0, \quad 0, \quad 0\right). \end{aligned}$$

Similarly, for h_b we get

$$\begin{aligned} C_b = \frac{\partial h_b}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_b}{\partial b_x}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial b_y}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \dot{b}_x}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \dot{b}_y}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \mu}(\mathbf{x})\right) \\ &= \text{sign}\left(\rho_x\cos\phi + \rho_y\sin\phi\right)\left(\tfrac{-\rho_y}{\|\boldsymbol{\rho}\|^2}, \quad \tfrac{\rho_x}{\|\boldsymbol{\rho}\|^2}, \quad 0, \quad 0, \quad 0\right). \end{aligned}$$

In summary, the Kalman filter to predict the ball location is given by

Between Measurements:

$$\begin{aligned}\dot{\hat{x}} &= f(\hat{x}) \\ A &= \frac{\partial f}{\partial x}(\hat{x}) \\ \dot{S} &= AS + SA^\top + Q.\end{aligned}$$

When range measurement y_r is available:

$$\begin{aligned}C_r &= \frac{\partial h_r}{\partial x}(\hat{x}) \\ L_r &= S^- C_r^\top (R_r + C_r S^- C_r^\top)^{-1} \\ S^+ &= (I - L_r C_r) S^- \\ \hat{x}^+ &= \hat{x}^- + L_r(y_r - h_r(\hat{x}^-)).\end{aligned}$$

When bearing measurement y_b is available:

$$\begin{aligned}C_b &= \frac{\partial h_b}{\partial x}(\hat{x}) \\ L_b &= S^- C_b^\top (R_b + C_b S^- C_b^\top)^{-1} \\ S^+ &= (I - L_b C_b) S^- \\ \hat{x}^+ &= \hat{x}^- + L_b(y_b - h_b(\hat{x}^-)).\end{aligned}$$

3.3 Estimating the Position of Other Robots

In this section we show how the Kalman filter can be used to estimate the position, and velocity of the other robots, whether they are own team or opponents.

As in the privious section, let $\mathbf{r} \stackrel{\triangle}{=} \begin{pmatrix} r_x, & r_y \end{pmatrix}^\top$ be the (assumed) known position of the robot/camera, and again let $\mathbf{n} = (n_x, n_y)^\top = (\cos\phi, \sin\phi)^\top$ be the (known) normal vector along the camera line of sight. The other robots position will be denoted by $\mathbf{o} = (o_x, o_y)^\top$, and its velocity is $\dot{\mathbf{o}} = (\dot{o}_x, \dot{o}_y)^\top$. We will assume that the dynamic equation of motion for the other robot is given by the constant acceleration model

$$\ddot{\mathbf{o}} = 0,$$

The objective is to estimate the position $\mathbf{o}$, the the velocity $\dot{\mathbf{o}}$, given the range and bearing measurements to the other robot. Accordingly, define the state of the

system as

$$x = \begin{pmatrix} o_x \\ o_y \\ \dot{o}_x \\ \dot{o}_y \end{pmatrix}.$$

The differential equation that describes the evolution of x is given by

$$\dot{x} = f(x) + \xi \stackrel{\triangle}{=} \begin{pmatrix} \dot{o}_x \\ \dot{o}_y \\ 0 \\ 0 \end{pmatrix} + \xi,$$

where the Jacobian of f is given by

$$A = \frac{\partial f}{\partial x}(x) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

We will assume that the computer vision software on the robot is able to measure the range to the other which we denote by $\|\boldsymbol{\rho}\|$, and the bearing to the other which we denote by φ. Let the vector

$$\boldsymbol{\rho} \stackrel{\triangle}{=} \mathbf{b} - \mathbf{r} = \begin{pmatrix} o_x - r_x \\ o_y - r_y \end{pmatrix}$$

denote the vector from the robot to the other robot $\mathbf{o}$. The range is therefore given by

$$h_r(\mathbf{x}) \stackrel{\triangle}{=} \|\boldsymbol{\rho}\| = \sqrt{(o_x - r_x)^2 + (o_y - r_y)^2}.$$

The bearing angle from robot $\mathbf{r}$ to other robot $\mathbf{o}$ can be found by taking the cross product between the normal vector $\mathbf{n} = (\cos\phi, \sin\phi)^\top$ and the line-of-sight vector $\boldsymbol{\rho}$. From the property of the cross product we have that

$$\mathbf{n} \times \boldsymbol{\rho} = \|\mathbf{n}\| \, \|\boldsymbol{\rho}\| \sin\varphi \, \hat{\mathbf{z}},$$

were $\hat{\mathbf{z}}$ is the unit vector perpendicular to both $\mathbf{n}$ and $\boldsymbol{\rho}$. If $\mathbf{n}$ is a unit vector, then the bearing angle ϕ is given by

$$\varphi = \sin^{-1}\left(\frac{n_x\rho_y - n_y\rho_x}{\|\boldsymbol{\rho}\|}\right).$$

Therefore the bearing angle as a function of robot state is given by

$$h_b(\mathbf{x}) \stackrel{\triangle}{=} \varphi = \sin^{-1}\left(\frac{(o_y - r_y)\cos\phi - (o_x - r_x)\sin\phi}{\sqrt{(o_x - r_x)^2 + (o_y - r_y)^2}}\right).$$

There are therefore two measurements associated with the other robot. Using the notation of Equation (14) the measurements are given by

$$\begin{aligned} y_r &= h_r(\mathbf{x}) + \eta_r \\ y_b &= h_b(\mathbf{x}) + \eta_b, \end{aligned}$$

where $\eta_r \sim \mathcal{N}(0, R_r)$ and $\eta_b \sim \mathcal{N}(0, R_b)$. To implement the Kalman filter measurement update equations (12), we need the Jacobians of h_r and h_b. After some algebra, we get

$$\begin{aligned} C_r = \frac{\partial h_r}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_r}{\partial o_x}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial o_y}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \dot{o}_x}(\mathbf{x}), \quad \tfrac{\partial h_r}{\partial \dot{o}_y}(\mathbf{x})\right) \\ &= \left(\tfrac{\rho_x}{\|\boldsymbol{\rho}\|}, \quad \tfrac{\rho_y}{\|\boldsymbol{\rho}\|}, \quad 0, \quad 0\right). \end{aligned}$$

Similarly, for h_b we get

$$\begin{aligned} C_b = \frac{\partial h_b}{\partial \mathbf{x}}(\mathbf{x}) &= \left(\tfrac{\partial h_b}{\partial o_x}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial o_y}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \dot{o}_x}(\mathbf{x}), \quad \tfrac{\partial h_b}{\partial \dot{o}_y}(\mathbf{x})\right) \\ &= \text{sign}\left(\rho_x \cos\phi + \rho_y \sin\phi\right)\left(\tfrac{-\rho_y}{\|\boldsymbol{\rho}\|^2}, \quad \tfrac{\rho_x}{\|\boldsymbol{\rho}\|^2}, \quad 0, \quad 0\right). \end{aligned}$$

In summary, the Kalman filter to predict the location of the other robot is given by

Between Measurements:

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}) \\ A &= \frac{\partial f}{\partial x}(\hat{x}) \\ \dot{S} &= AS + SA^\top + Q. \end{aligned}$$

When range measurement y_r is available:

$$\begin{aligned} C_r &= \frac{\partial h_r}{\partial x}(\hat{x}) \\ L_r &= S^- C_r^\top (R_r + C_r S^- C_r^\top)^{-1} \\ S^+ &= (I - L_r C_r) S^- \\ \hat{x}^+ &= \hat{x}^- + L_r(y_r - h_r(\hat{x}^-)). \end{aligned}$$

When bearing measurement y_b is available:

$$
\begin{aligned}
C_b &= \frac{\partial h_b}{\partial x}(\hat{x}) \\
L_b &= S^- C_b^\top (R_b + C_b S^- C_b^\top)^{-1} \\
S^+ &= (I - L_b C_b) S^- \\
\hat{x}^+ &= \hat{x}^- + L_b(y_b - h_b(\hat{x}^-)).
\end{aligned}
$$