Soccer Ball Tracking using Extended Kalman Filter

Karan Jha

November 8, 2022

Abstract

In this project, we will implement tracking of a soccer ball as well as calculate the spin on the ball using Extended Kalman Filter and use hypothesis testing to determine if the ball has crossed the goal line. In our model, we have a soccer ball that travels based on an initial input (of velocity) and a spin and under influece of gravity in an XYZ-plane. To track the ball movement, we use two pinhole cameras that are kept across each other at the same x-coordinate, each gives its own pan and tilt angle (in degrees) for the ball. We have a process noise in form of wind in all the three dimensions, which provides a Gaussian disturbance. We also check hypothesis testing to test if the ball has crossed the goal line. Since, we have posts and crossbars for checking the y and z coordinates, we only need to check the x coordinate.

1 Theory

1.1 Extended Kalman Filter

In this model we use two pan-tilt pinhole cameras to track state of a soccer ball, each of which gives its separate pan and tilt angles.

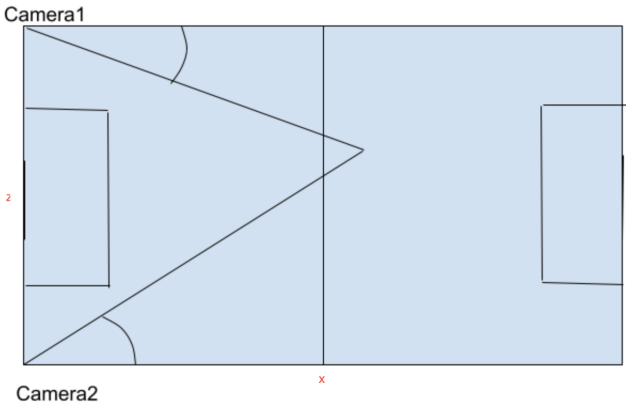

Figure 1: Tracking with Pinhole Camera

In our model, we have a soccer ball that gets an travels after a shot under influence of gravity with given velocities. The process noise is provided by wind in all the three directions. We assume a disturbance for the spin w_s, because we also want to determine the value of the spin provided. Our state space model becomes:-

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & s \end{bmatrix}^\top, \mathbf{w} = \begin{bmatrix} w_x & w_y & w_z & w_s \end{bmatrix}^\top$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + dt \begin{bmatrix} \dot{x} & w_x & \dot{y} & -g + s + w_y & \dot{z} & w_z & w_s \end{bmatrix}_k^\top = \mathbf{f}(\mathbf{x}_k, \mathbf{w}_k)$$

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + G\mathbf{w}_k$$

$$F = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & dt & 0 \\ 0 & 0 & 0 & dt \end{bmatrix}$$

Each camera gives its pan angle (ψ_1 and ψ_2) which is the angle with the corner line and the tilt angle (ϕ_1 and ϕ_2) that is the angle above the ground as follows:-

$$\mathbf{z}_k = \begin{bmatrix} tan^{-1}(\frac{z_k}{x_k}) \\ tan^{-1}(\frac{y_k}{\sqrt{x_k^2+z_k^2}}) \\ tan^{-1}(\frac{(W-z_k)}{x_k}) \\ tan^{-1}(\frac{y_k}{\sqrt{x_k^2+(W-z_k)^2}}) \end{bmatrix} = \mathbf{h}(\mathbf{x}_k)$$

$$H_k = \left[\frac{\delta h}{\delta \mathbf{x}}\right]_{\mathbf{x}_k}$$

$$= \begin{bmatrix} \frac{-z_k}{x_k^2+z_k^2} & 0 & 0 & 0 & \frac{x_k}{x_k^2+z_k^2} & 0 & 0 \\ \frac{-xy}{(x_k^2+y_k^2+z_k^2)\sqrt{x_k^2+z_k^2}} & 0 & \frac{\sqrt{x_k^2+z_k^2}}{(x_k^2+y_k^2+z_k^2)} & 0 & \frac{-zy}{(x_k^2+y_k^2+z_k^2)\sqrt{x_k^2+z_k^2}} & 0 & 0 \\ \frac{-(L-z_k)}{x_k^2+(L-z_k)^2} & 0 & 0 & 0 & \frac{-x_k}{x_k^2+(L-z_k)^2} & 0 & 0 \\ \frac{-xy}{(x_k^2+y_k^2+(L-z_k)^2)\sqrt{x_k^2+(L-z_k)^2}} & 0 & \frac{\sqrt{x_k^2+(L-z_k)^2}}{(x_k^2+y_k^2+(L-z_k)^2)} & 0 & \frac{(L-z_k)y}{(x_k^2+y_k^2+(L-z_k)^2)\sqrt{x_k^2+(L-z_k)^2}} & 0 & 0 \end{bmatrix}$$

Then we use the extended kalman filter with linear model and non linear measurements as:-

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}(\mathbf{x}_k, \mathbf{w_k})$$

$$P_{k+1|k} = FP_{k|k}F^\top + GQG^\top$$

$$K_{k+1} = P_{k+1|k}H_{k+1}^\top(H_{k+1}P_{k+1|k}H_{k+1}^\top + R)^{-1}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + K_{k+1}(\mathbf{z}_{k+1} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}))$$

$$P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^\top K_{k+1}RK_{k+1}^\top$$

1.2 Hypothesis Testing

We use hypothesis testing to test if the ball has crossed the goal line based on our estimations. As we have the cross bar and sideposts for the y and z coordinates, we need to do the hypothesis testing only for the x-direction.

We conclude that it is a goal when $p(goal|\hat{\mathbf{x}}_{k|k}) \geq 95\%$. The all three dimensions are independent and, given we can be sure with the y and z directions, we can write $p(goal|\hat{\mathbf{x}}_{k|k}) = p(goal|\hat{x}_{k|k})p(goal|\hat{y}_{k|k})p(goal|\hat{z}_{k|k})$. With the pitch length L, we can write

$$\begin{aligned} p(goal|\hat{\mathbf{x}}_{k|k}) &= p(goal|\hat{x}_{k|k})p(goal|\hat{y}_{k|k})p(goal|\hat{z}_{k|k}) \\ &= p(goal|\hat{x}_{k|k})\delta(goal|\hat{y}_{k|k})\delta(goal|\hat{z}_{k|k}) \end{aligned}$$

Where $\delta(goal|\hat{y}_{k|k})$ and $\delta(goal|\hat{y}_{k|k})$ are the delta functions that give 0 or 1 based on whether the ball is inside the goal frame or not. Hence we only check that

$$p(goal|\hat{x}_{k|k}) = p(x_k \geq L|\hat{x}_{k|k}) = 1 - p(L|\mathcal{N}(\hat{x}_{k|k}, \sigma_{x_{k|k}})) \geq 95\%$$

2 Results

I have run two separate cases one with initial conditions close to the original values to get prettier graphs, and other with initial conditions quite displaced to test the convergence of the model based on the measurements, where again I have tried lower and higher initial covariances, to understand how the initial covariance affects the convergence over a short time period

2.1 Ball Tracking using EKF

Case 1: Favourable initial conditions
Here we take the initial state close to the actual initial conditions of the ball to get prettier plots to check the EKF performance.

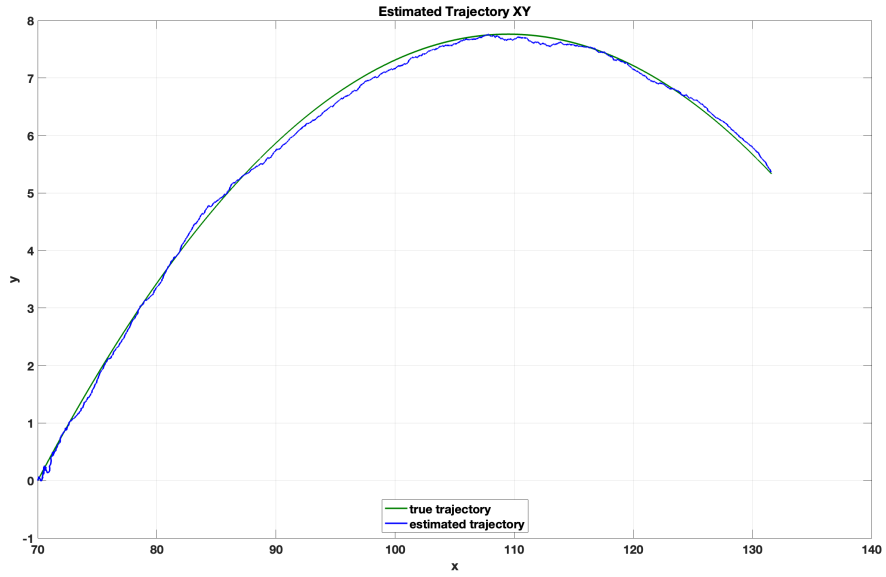

Figure 2: XY estimation plot

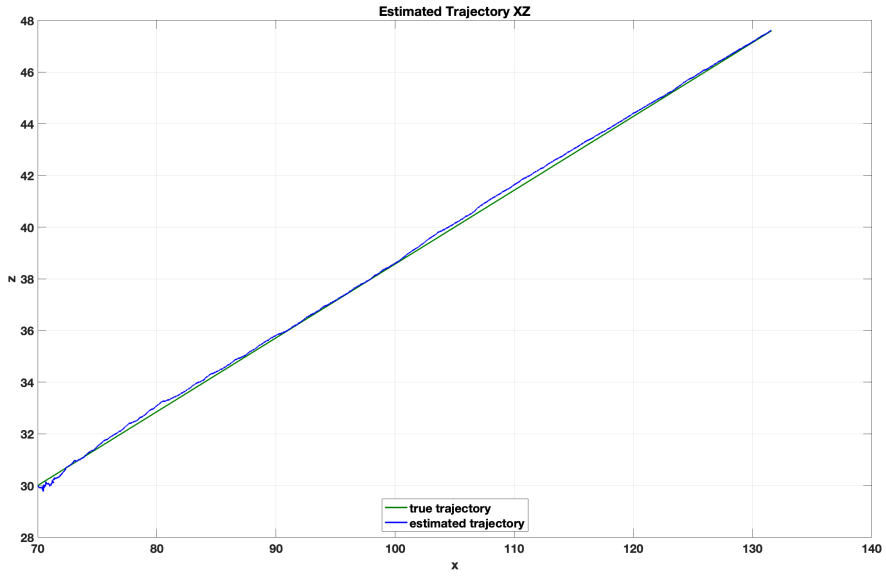

Figure 3: XZ estimation plot

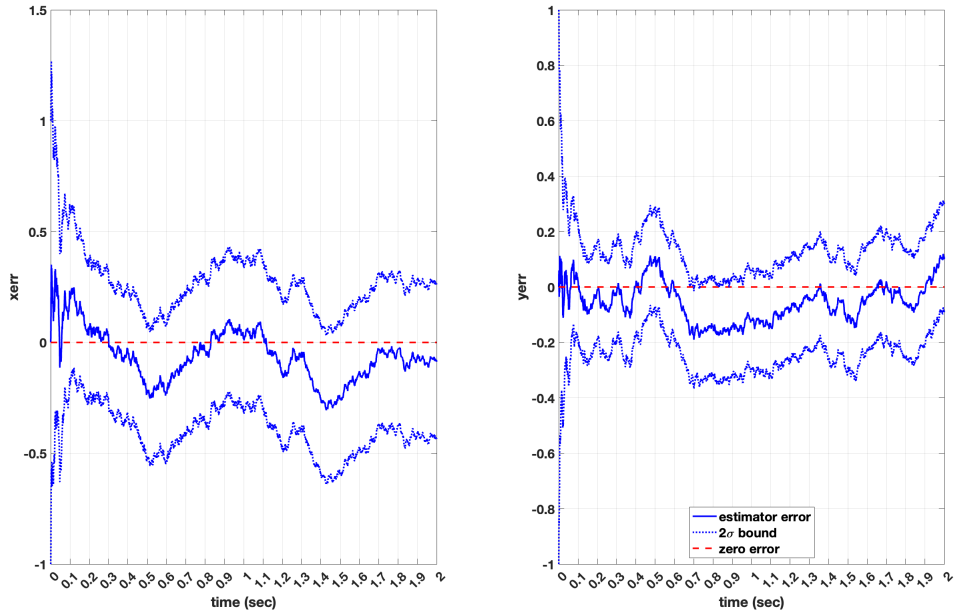

Figure 4: X and Y estimation error plot

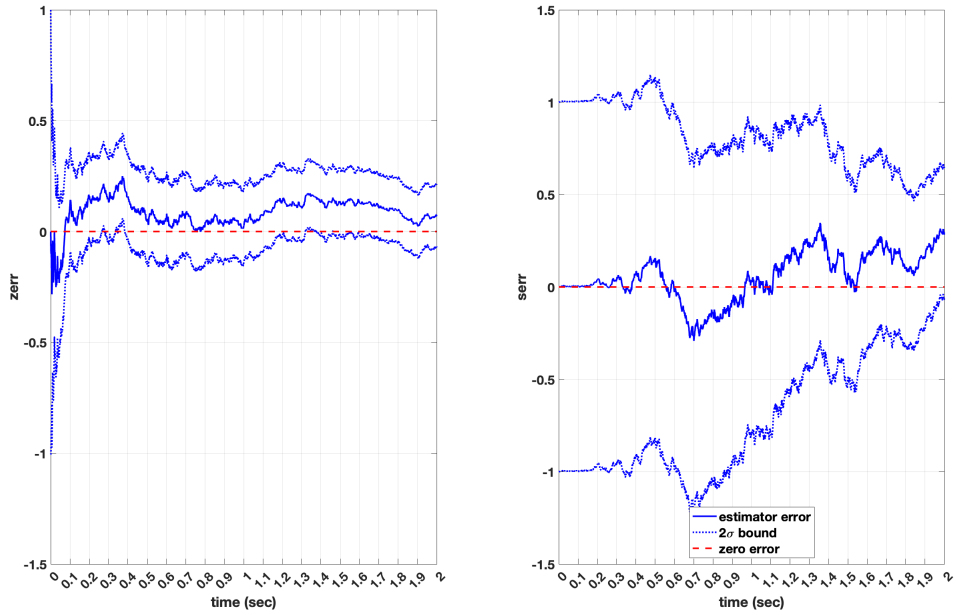

Figure 5: Z and spin estimation plot

Case 2: Displaced initial conditions

Here I have taken the initial state values of the position, velocities and the spin displaced from the original initial values, since I wanted to check how the EKF tackles a biased initial condition problem, and reconciles the predictions with the measurements. I tried this with low and high initial covariances to see how the initial covariances affect the convergence of in this case

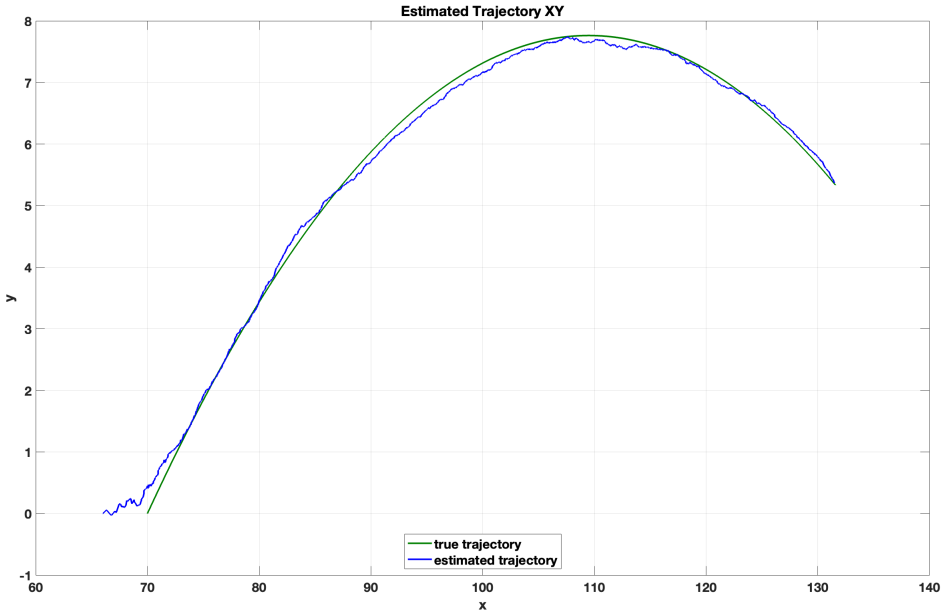

Figure 6: XY estimation plot

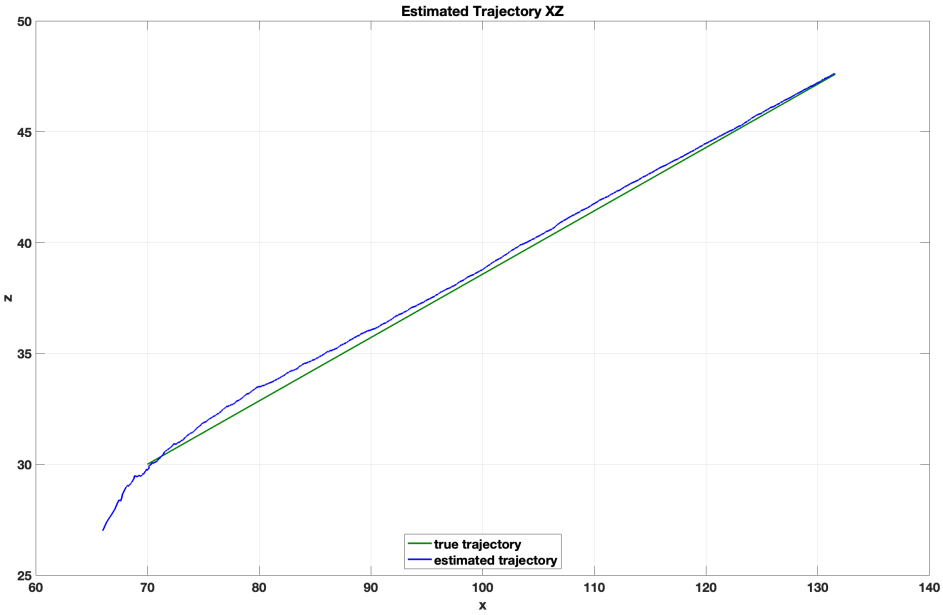

Figure 7: XZ estimation plot

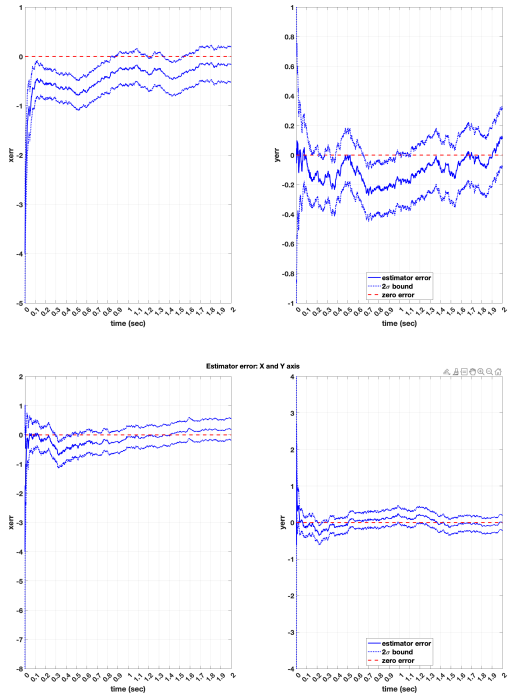

Figure 8: X and Y estimation error plot with
1) low initial covariance, 2) high initial covariance

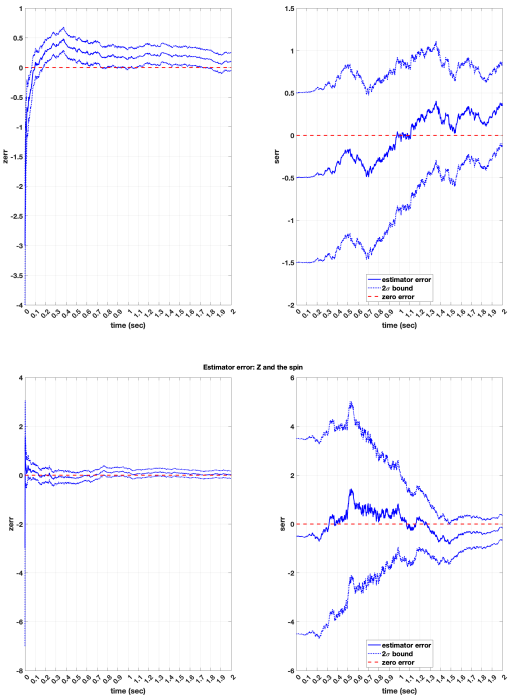

Figure 8: Z and spin estimation error plot with
1) low initial covariance, 2) high initial covariance

2.2 Hypothesis Testing

Here I use hypothesis testing with 95% confidence interval to decide whether the ball has crossed the goal line, based on our estimation of states. I have taken the goal to be higher than reality, because I wanted the ball to have a longer trajectory.

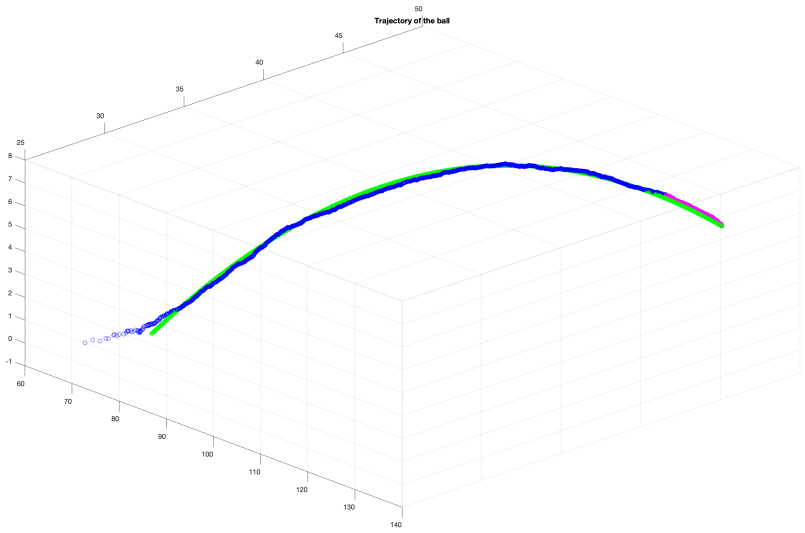

Figure 10: Goal Tracking

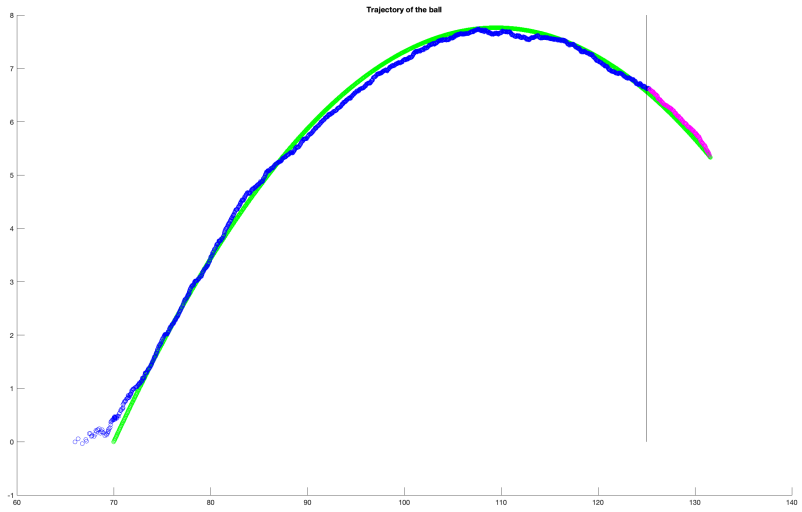

Figure 11: Goal Tracking 2D in XY plane

green	no noise ball trajectory
blue	no goal
magenta	goal
black vertical line	goalpost

3 Discussion and Further Scope

We have used two separate cameras located as in [Figure 1]
to make our system observable. In case of a single camera, we can see that with the given measurements, we won't be able to attain the range measurement for the camera. With two separate camera measurements, we overcome this problem as we get two separate measurements one for z and one for $W - z$ where W is the pitch width along z-axis. This makes our system observable.
Our EKF tracking model seems to work pretty well under the given model and Pan Tilt camera measurements. The EKF captures the non-linearities in our measurements quite well and eventually our estimates converge to the actual no noise case fairly well, in both the cases. I used the two cases to learn how our model works with erroneous initial conditions. I also tried how the initial covariances affect the convergence in this case, and learnt that with higher initial covariance, the convergence is much smoother and faster.
Our hypothesis testing, where we have used 95% confidence intervals on our estimates, gives a fairly accurate goal line tracking as visible in [Figure 11].
This project can be expanded to take into account the curve of the ball in the XZ-plane as well, that is also a variable and forms part of the state vector. Additionally, this can be used to track players' positions as well that can be used for the much controversial offside rule.

4 Matlab Code

```
%% MAE 6760 Final Project

%% Simulation
clc;clear;
n = 7;
pitch_width = 85;
x0 = 70; y0 = 0; z0 = 30;
xdot0 = 28;ydot0 = 11; zdot0 = 8;
s = 2; g = 9.8;
dt = 0.001;
t = 0:dt:2.2;nt = length(t);
Xnonoise = [x0;xdot0;y0;ydot0;z0;zdot0;s];
for k=1:(nt-1)
    xdot=Xnonoise(2,k);
    ydot=Xnonoise(4,k);
    zdot = Xnonoise(6,k);
    s = Xnonoise(7,k);
    Xnonoise(:,k+1) = Xnonoise(:,k) +...
        dt*[xdot;0;ydot;-g+s;zdot;0;0];
end
plot_birdseyeview(Xnonoise,[],[],'True Trajectory XY',[1,3])
PrepFigPresentation(gcf)
plot_birdseyeview(Xnonoise,[],[],'True Trajectory XZ',[1,5])
PrepFigPresentation(gcf)
camup([0 1 0]);
psi1 = atan(Xnonoise(5,:)./Xnonoise(1,:));
phi1 = atan(Xnonoise(3,:)./sqrt(Xnonoise(1,:).^2 + Xnonoise(5,:).^2));
psi2 = atan((pitch_width-Xnonoise(5,:))./Xnonoise(1,:));
phi2 = atan(Xnonoise(3,:)./sqrt(Xnonoise(1,:).^2 + (pitch_width-Xnonoise(5,:)).^2));
nz = 4;
nw = 4;
R=eye(nz)*1^2; v=sqrtm(R)*randn(nz,nt);
Q = diag([1 1 1 1]);w=sqrtm(Q)*randn(nw,nt);
Z = 180*[psi1;phi1;psi2;phi2]/pi+v;
Zacc = w;

%% EKF with favourable initial conditions
x0=[70;28;0;11;30;8;2];
P0=eye(n)*0.5^2;
n=length(x0);
xhatp=x0;Pp(1:n,1:n,1)=P0;
xhatu=x0;Pu(1:n,1:n,1)=P0;
for k=1:(nt-1)
```

```
    %predict state
    xhatp(:,k+1)=predict_state(xhatu(:,k),Zacc(:,k),dt);
    %predict covariance
    [F,G]=getFG(xhatu(:,k),dt);
    Pp(1:n,1:n,k+1) = F*Pu(1:n,1:n,k)*F' + G*Q*G';
    %Kalman Gain
    H = getH(xhatp(:,k+1));
    %Update
    Zp = 180*[atan(xhatp(5,k+1)/xhatp(1,k+1));
        atan(xhatp(3,k+1)/sqrt(xhatp(1,k+1)^2 + xhatp(5,k+1)^2));
        atan((pitch_width-xhatp(5,k+1))/xhatp(1,k+1));
        atan(xhatp(3,k+1)/sqrt(xhatp(1,k+1)^2 + (pitch_width-xhatp(5,k+1))^2))]/pi;
    inn = Z(:,k+1) - Zp;
    S = H*Pp(1:n,1:n,k+1)*H' + R;

    K = Pp(1:n,1:n,k+1)*H'*inv(S);
    xhatu(:,k+1) = xhatp(:,k+1) + K *(Z(:,k+1) - Zp);
    Pu(1:n,1:n,k+1) = (eye(n)-K*H)*Pp(1:n,1:n,k+1)*(eye(n)-K*H)' + K*R*K';
    goal = goal_check(xhatu(:,k+1),Pu(:,:,k+1));
end
plot_birdseyeview(Xnonoise,xhatu,Pu,'Estimated Trajectory XY',[1,3])
PrepFigPresentation(gcf)
plot_birdseyeview(Xnonoise,xhatu,Pu,'Estimated Trajectory XZ',[1,5])
PrepFigPresentation(gcf)
ii_plot=[1 3];
plot_estimator_error(t,Xnonoise,xhatu,Pu,ii_plot,'Estimator error: X and Y axis');
ii_plot=[5 7];
plot_estimator_error(t,Xnonoise,xhatu,Pu,ii_plot,'Estimator error: Z and the spin');

%% EKF with displaced initial conditions
x0=[66;25;0;10;27;10;1.5];
P0=eye(n)*0.5^2;
n=length(x0);
xhatp=x0;Pp(1:n,1:n,1)=P0;
xhatu=x0;Pu(1:n,1:n,1)=P0;
R=eye(nz)*1^2; v=sqrtm(R)*randn(nz,nt);
Q = diag([10 10 10 25]);
for k=1:(nt-1)
    %predict state
    xhatp(:,k+1)=predict_state(xhatu(:,k),Zacc(:,k),dt);
    %predict covariance
    [F,G]=getFG(xhatu(:,k),dt);
    Pp(1:n,1:n,k+1) = F*Pu(1:n,1:n,k)*F' + G*Q*G';
    %Kalman Gain
    H = getH(xhatp(:,k+1));
    %Update
    Zp = 180*[atan(xhatp(5,k+1)/xhatp(1,k+1));
        atan(xhatp(3,k+1)/sqrt(xhatp(1,k+1)^2 + xhatp(5,k+1)^2));
        atan((pitch_width-xhatp(5,k+1))/xhatp(1,k+1));
        atan(xhatp(3,k+1)/sqrt(xhatp(1,k+1)^2 + (pitch_width-xhatp(5,k+1))^2))]/pi;
    inn = Z(:,k+1) - Zp;
    S = H*Pp(1:n,1:n,k+1)*H' + R;

    K = Pp(1:n,1:n,k+1)*H'*inv(S);
    xhatu(:,k+1) = xhatp(:,k+1) + K *(Z(:,k+1) - Zp);
    Pu(1:n,1:n,k+1) = (eye(n)-K*H)*Pp(1:n,1:n,k+1)*(eye(n)-K*H)' + K*R*K';
    goal = goal_check(xhatu(:,k+1),Pu(:,:,k+1));
end
plot_birdseyeview(Xnonoise,xhatu,Pu,'Estimated Trajectory XY',[1,3])
PrepFigPresentation(gcf)
plot_birdseyeview(Xnonoise,xhatu,Pu,'Estimated Trajectory XZ',[1,5])
PrepFigPresentation(gcf)
ii_plot=[1 3];
plot_estimator_error(t,Xnonoise,xhatu,Pu,ii_plot,'Estimator error: X and Y axis');
ii_plot=[5 7];
plot_estimator_error(t,Xnonoise,xhatu,Pu,ii_plot,'Estimator error: Z and the spin');

Plotting Trajectory
figure;
for k=1:(nt)
    goal = goal_check(xhatu(:,k),Pu(:,:,k));
```

```
    scatter3(Xnonoise(1,k),Xnonoise(3,k),Xnonoise(5,k),'green')
    if goal == 1
        scatter3(xhatu(1,k),xhatu(3,k),xhatu(5,k),'magenta')
        hold on;
    else
        scatter3(xhatu(1,k),xhatu(3,k),xhatu(5,k),'blue')
        hold on;
    end
end
title('Trajectory of the ball')
camup([0 1 0])

Hypothesis Testing Functions
function goal = goal_check(Xk,Pk)
prob = 1-cdf('Normal',125,Xk(1),sqrt(Pk(1,1)));
if prob > 0.95
    goal = 1;
else
    goal = 0;
end
end

%% EKF Function Cells
function Xkp1=predict_state(Xk,U,dt)
wx = U(1); wy = U(2); wz = U(3); g = 9.8; ws = U(4);
Xkp1 = Xk +...
    dt*[Xk(2);
        wx;
        Xk(4);
        -g + Xk(7) + wy;
        Xk(6);
        wz;
        ws];
end

function [F,G]=getFG(X,dt)
F = [1 dt 0 0 0 0 0;
    0 1 0 0 0 0 0;
    0 0 1 dt 0 0 0;
    0 0 0 1 0 0 dt;
    0 0 0 0 1 dt 0;
    0 0 0 0 0 1 0;
    0 0 0 0 0 0 1];

G = [0 0 0 0;
    dt 0 0 0;
    0 0 0 0;
    0 dt 0 0;
    0 0 0 0;
    0 0 dt 0;
    0 0 0 dt];
end

function H=getH(X)
L = 85;
x = X(1); y = X(3); z = X(5);
H = 180*[-z/(x^2+z^2) 0 0 0 x/(x^2+z^2) 0 0;
    -x*y/((x^2+y^2+z^2)*sqrt(x^2+z^2)) 0 sqrt(x^2+z^2)/(x^2+y^2+z^2) 0 -z*y/((x^2+y^2+
    z^2)*sqrt(x^2+z^2)) 0 0;
    -(L-z)/(x^2+(L-z)^2) 0 0 0 -x/(x^2+(L-z)^2) 0 0;
    -x*y/((x^2+y^2+(L-z)^2)*sqrt(x^2+(L-z)^2)) 0 sqrt(x^2+(L-z)^2)/(x^2+y^2+(L-z)^2) 0
     (L-z)*y/((x^2+y^2+(L-z)^2)*sqrt(x^2+(L-z)^2)) 0 0]/pi;
end

%% Plotting Functions
function plot_estimator(t,x1,x2,P2,ii_plot,title_name);
% x1 is the true value or reference comparison
% x2,P2 is the estimator state and covariance
% ii_plot: 2x1 vector of which states to plot
%
```

```
axis_names={'North (m)','East (m)','Velocity (m/sec)','Heading (rad)','Accel Bias (m/
    sec^2)','RG Bias (rad/sec)'};
figure;subplot(122);
ii_x1=[];ii_x2=[];ii_P2=[]; %for legend
%
for i=1:length(ii_plot),
    ii=ii_plot(i);
    subplot(1,2,i);
    hold on;
    if ~isempty(x1),
        plot(t,x1(ii,:),'color',[0 0.5 0]);ii_x1=1;
    end
    if ~isempty(x2),
        plot(t,x2(ii,:),'b-');ii_x2=2;
    end
    if ~isempty(P2)
        plot(t,x2(ii,:)'-2*sqrt(squeeze(P2(ii,ii,:))),'b:');
        plot(t,x2(ii,:)'+2*sqrt(squeeze(P2(ii,ii,:))),'b:');ii_P2=3;
    end
    hold off
    xlabel('time (sec)');ylabel(axis_names(ii));grid;
    xlim([0 35]);set(gca,'xtick',[0:5:35]);
end
legend_names={'true state','estimate','2\sigma bound'};
legend(legend_names{ii_x1},legend_names{ii_x2},legend_names{ii_P2},'Location','South')
    ;
%
sgtitle(title_name);
PrepFigPresentation(gcf);
end
%
function plot_estimator_error(t,x1,x2,P2,ii_plot,title_name);
% x1 is the true value or reference comparison
% x2,P2 is the estimator state and covariance
% ii_plot: 2x1 vector of which states to plot
%
axis_names={'xerr','vx','yerr','vy','zerr','vz','serr'};
figure;subplot(122);
%
for i=1:length(ii_plot),
    ii=ii_plot(i);
    subplot(1,2,i);
    err=x2(ii,:)-x1(ii,:);
    plot(t,err,'b-');
    hold on;
    if ~isempty(P2)
        plot(t,err'-2*sqrt(squeeze(P2(ii,ii,:))),'b:');
        plot(t,zeros(length(t),1),'r--');
        plot(t,err'+2*sqrt(squeeze(P2(ii,ii,:))),'b:');
    end
    hold off
    xlabel('time (sec)');ylabel(axis_names(ii));grid;
    xlim([0 2]);set(gca,'xtick',[0:0.1:2]);
end
legend('estimator error','2\sigma bound','zero error','Location','South');
%
sgtitle(title_name);
PrepFigPresentation(gcf);
end
%
function plot_birdseyeview(x1,x2,P2,title_name,ii_plot);
% x1 is the true value or reference comparison
% x2,P2 is the estimator state and covariance
%
label_name={'x','vx','y','vy','z','vz','s'};
ii_x1=[];ii_x2=[];ii_P2=[]; %for legend
figure;
if ~isempty(x1),
    plot(x1(ii_plot(1),:),x1(ii_plot(2),:),'color',[0 0.5 0]);ii_x1=1;
end
hold on;
```

```
if ~isempty(x2),
    plot(x2(ii_plot(1),:),x2(ii_plot(2),:),'b-');ii_x2=2;
end
xlabel(label_name(ii_plot(1)));ylabel(label_name(ii_plot(2)));grid;
hold off;
legend_names={'true trajectory','estimated trajectory','3\sigma bound'};
legend(legend_names{ii_x1},legend_names{ii_x2},legend_names{ii_P2},'Location','South')
%
title(title_name);
PrepFigPresentation(gcf);
end
%
function PrepFigPresentation(fignum);
%
% prepares a figure for presentations
%
% Fontsize: 14
% Fontweight: bold
% LineWidth: 2
%

figure(fignum);
fig_children=get(fignum,'children'); %find all sub-plots

for i=1:length(fig_children),

    set(fig_children(i),'FontSize',16);
    set(fig_children(i),'FontWeight','bold');

    fig_children_children=get(fig_children(i),'Children');
    set(fig_children_children,'LineWidth',2);
end
end
```

Algorithm 1: Matlab Code