Target Tracking with Kalman Filtering
Project 05 - Report

Usama Munir Sheikh
ECSE 6650: Computer Vision with Dr. Qiang Ji, RPI
Troy, New York
sheiku@rpi.edu

***Abstract/Goals*— The goal of this project was to implement the Kalman Filter (KF) to track a face as it moves, over multiple frames. Multiple images from a video sequence of the face were given. Its initial position and bounding box from the first couple of frames were used to track the position of the face in the image sequence. The results obtained using the Kalman Filter are quite robust, presented as a red bounding box that is used to show the tracked position of the face in the resulting video.**

Keywords- Kalman Filter; Sum of Squared Differences; Correspondance; Feature Tracking;

I. Introduction

"Tracking is a process that matches a sparse set of feature points from frame to frame."[1] It has a lot of applications, ranging from navigation to robot manipulation.

For this project we have implemented the Kalman Filter for feature tracking. Kalman Filter is a very popular recursive technique to estimate the state of a dynamic system. The way a Kalman Filter works is that it fuses information from observations or measurements from the sensor with the dynamical information or predictions made through the system model.

We are tracking the face of a person in a video sequence composed of one hundred images. More specifically, we track the x and y position of the face as it moves. We will also keep a track on the evolution of the x and y velocities, v_x and v_y, as the other two states in the system.

The first two frames in the video image sequence will be used to initialize the states for the Kalman Filter. The x and y coordinates of the position of the center of the face and a bounding box around it have been provided to us as shown in figure 1.

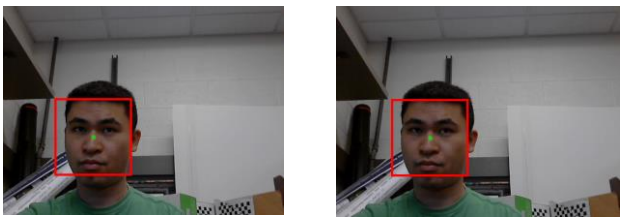

Figure 1. First two frames of the video

II. Theory and Methods

A. *Kalman Filter & Tracking*

The linear Kalman Filter is a "classic tool of optimal estimation theory" that "produces optimal estimates (at each time instant) of the state of a dynamic system, on the basis of noisy measurements and an uncertain model of the system dynamics."[2] Therefore, we have a system or process model and a measurement model that are needed for the optimal estimation of the states of a dynamic system.

The system model represents the evolution of the system states over time. The states can be represented as a time-dependent column vector, s and the evolution of s can be modeled through a state transition matrix, φ. For feature tracking, the states will be the position of the feature point, $p = (x_t, y_t)$ and $v_t = \left(v_{x,t}, v_{y,t}\right)^t$, where t represents the time instant which in our case also happens to be the index of the video frames. Therefore, s becomes,

$$s_t = \left[x_t, y_t, v_{x,t}, v_{y,t}\right]^t \tag{1}$$

According to the system, the next state, s_{t+1}, can be computed as follows,

$$s_{t+1} = \phi s_t + w_t \tag{2}$$

Where, ϕ is the state transition matrix and w_t represents the process noise/disturbance or the system perturbation. It can be normally distributed as $w_t \sim N(0, Q)$. In this case, the state/system model describes the temporal parts of the system.

The other part of the Kalman Filter is the measurement model. A noisy measurement of the state vector or part of it is assumed to be taken and it can be related to the state vector through the following linear relationship,

$$z_t = H s_t + \varepsilon_t \tag{3}$$

Where z_t represents the measurement of the states and the matrix H, called the measurement matrix, relates the current state to the current measurement. ε_t represents the measurement uncertainty, normally distributed as $\varepsilon_t \sim N(0, R)$. In our case, the measurement model describes the spatial features of the system since we will only measure the x and y position coordinates through feature detection by establishing

correspondences. This will make our measurement vector as $z_t = (z_{x,t}, z_{y,t})$.

For tracking, we can assume that "the features movement between two consecutive frames is small enough to consider the motion of the feature positions from frame to frame to be uniform." Therefore, the state transition matrix can be given as,

$$\phi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Also, for simplicity and since z_t only involves position, we can assume that $z_t$is the same as $s_t$plus some noise. Therefore, H can be represented as,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (5)$$

Now that we have knowledge about the state/system model and the measurement model, we can use it Kalman Filtering. Kalman Filtering basically consists of two key steps: 1) State Prediction, 2) State Updating. State Prediction is performed using the state model while the state updating is performed using the measurement model. A detailed step-by-step procedure description of the algorithm will follow in the experimental procedure section (Section III).

B. *Limitations with Kalman Filtering*

There are two main limitations of Kalman Filtering found in [1].

1) We assume that the state dynamics (state transition) can be modeled as linearly.

2) We assume that the state vector has a uni-modal and it has a Gaussian distribution and therefore, it cannot track multiple feature points without the use of multiple Kalman Filters. Also, it cannot track non-Gaussian distributed features.

We can however, overcome these limitations through the use of the Extended Kalman Filter that overcomes the linearity assumption or the Unscented Kalman Filter which overcomes the Gaussian assumption.

C. *Establishing Correspondance – Sum of Squared Differences (SSD)*

To get the measurement value, z_t, we need to use a correlation technique. We must establish a correspondence between the current and the next image frame to get the postions of the center of the face which is the point/position-state that we are tracking.

The underlying idea for correlation methods is to establish correspondence between points based on intensity distribution of their neighborhood. These correlation methods have few assumptions, the assumptions are:

1) The points are visible in both image frames,
2) There's a single light source,
3) Corresponding image regions look similar.

One of the correlation methods in computer vision is the sum of squared differences (SSD). This method creates two windows. The first window in the current image frame and the second window in the next image frame. Then, we compute the squared difference between the elements of the windows. And finally, we sum all these differences to get the SSD value. The formula for SSD is

$$SSD = (W_1 - W_2)^T (W_1 - W_2) \quad (6)$$

Where W_1 Is a window about a point in the current image frame, and W_2 is a window about a point in the next image frame. For correspondence search, our window in the current image frame will be fixed around the current estimate of the state. The size of this window will be the size of the bounding box that has a height and width of 172. The size of the window in the next image will also be the same. However, this window will slide from the start of the search region to its end.

The search region will be centered at the predicted state position, defined by a square of $3\sigma_x \times 3\sigma_y$, where σ_x and σ_y are the two Eigen-Values of the first 2×2 submatrix of the covariance estimation matrix, Σ_{t+1}^-. The process correspondence search is described by the following steps:

1) Pick the point defined by the current state in the current image frame.
2) Create a window W_1 of 172×172 around it.
3) Pick the point defined by the predicted state in the next image frame.
4) Create the search region around it and make a window, W_2 of 172×172, at the start of the search region
5) Compute the SSD using (6) for that window in the next image.
6) Now slide the window W_2 along the search region and compute new SSD values.
7) Find the minimum SSD value. This will correspond to the matching window in the next image and the center of this window will be our matched point z_{t+1}.

III. Experimental Procedure, Results & Discussion

A. *The Kalman Filter Algorithm*

The Kalman filter, as discussed before, has two main steps. The step by step procedure of these key steps is given below:

1) **Prediction:** Given the current state, s_t., and its covariance matrix, Σ_t ., which is its confidence measure, we can predict the new state, s_{t+1}^-., using the state model and the correspoinding covariance estimate, Σ_{t+1}^-, using the following equations:

$$s_{t+1}^- = \phi\, s_t \quad (7)$$

$$\Sigma_{t+1}^- = \phi \sum_t \phi^t + Q \quad (8)$$

2) **Updating:** Updating consists of the following three steps:

- *First, we get z_{t+1} using the correspondence measurement described in section II C.*
- *Second, we combine s_{t+1}^{-} with z_{t+1} to obtain the final state estimate s_{t+1}*
  To fuse them together, we first need to compute the Kalman Gain using the following equation:

$$K_{t+1} = \frac{\Sigma_{t+1}^{-} H^T}{H\Sigma_{t+1}^{-} H^T + R} \quad (9)$$

This gain matrix, K, is a weighting factor to determine the contribution of the measurement and the prediction on the posterior state estimate, s_{t+1}
After the gain is computed we can use it to get s_{t+1} using the following equation

$$s_{t+1} = s_{t+1}^{-} + K_{t+1}(z_{t+1} - Hs_{t+1}^{-}) \quad (10)$$

- *Finally, we obtain the posteriori error covariance matrix estimate as follows:*

$$\Sigma_{t+1} = (I - K_{t+1}H)\Sigma_{t+1}^{-} \quad (11)$$

The trace of the state covariance matrix is a measure of uncertainty of the estimated position and it can be plotted w.r.t time to see if the uncertainty in state estimation decreases over time or not.

B. *Initializations*

The start Kalman filtering we need initial estimates of the state, the Q and R matrices and the covariance matrix, $\sum$.

The initial estimates of the state vector, s_0 can be specified as,

$$\begin{aligned} x_0 &= x_{i+1} \\ y_0 &= y_{i+1} \\ v_{x_0} &= x_{i+1} - x_i \\ v_{y_0} &= y_{i+1} - y_i \end{aligned} \quad (12)$$

For this project, the initial states were given in the first two frames of the video along with the corresponding bounding boxes.

The Initial covariance matrix was taken as,

$$\Sigma_0 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \quad (13)$$

The Q and the R matrix which are the system model uncertainty and measurement noise covariance respectively.These matrices are constant throught out the process of Kalman Filtering. The error in the state prediction is taken to be 4 pixels, here, in the x and y positions as seen in the Q matrix and the measurement error is taken to be 2 pixels as seen in the R matrix. The Q and the R matrices have been taken as the following:

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \quad (14)$$

$$R = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \quad (15)$$

Q and R can be tuned to make the tracking better. For this project I found out that by increasing the diagonal elements of Q had a great impact on the tracking of the face and I got better results with the following Q matrix, called Q_{bar} here to differentiate it from the other one. Q_{bar} assumes greater uncertainty in the state model.

$$Q_{bar} = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \quad (16)$$

C. *Results*

The final results were pretty good and we see that the Kalman Filter tracks the face with both Q and Q_{bar}. Better tracking was acheieved with Q_{bar} which assumes a greater uncertainty in the state model.

The final results (achieved using Q_{bar}) have been provided as a video. It can be found on the link below along with the resulting frames. (The mp4 version is clearer but an .avi file has also been provided as stated in the project handout).

https://drive.google.com/file/d/0B59A4eWmXoQ-dkx0T3l2YXFiSUE/view?usp=sharing

https://drive.google.com/file/d/0B59A4eWmXoQ-WC15cjFEMG91eDA/view?usp=sharing

Some of the frames from the video have been shown in figures 2 and 3 for $Q$and Q_{bar} respectively. We see that there is a greater tracking error by using Q. This is because the state model is based on an assumed linear model that predicts the new position state based on the x and y velocities of the current and the next image. These velocities are just the difference of the positions between the two images. Therefore, we see some tracking error with Q. If, however, we lower our confidence in the state prediction by increasing the diagonal elements of Q, we see that the tracking error has reduced and the tracking is more robust. However, we still get some drift in the system. This is expected as the simple linear Kalman Filter is not perfect. We can make the results better and more robust by using the extended version of the Kalman Filter or the

Unscented version which takes care of the inherent limitations of the Kalman Filter.

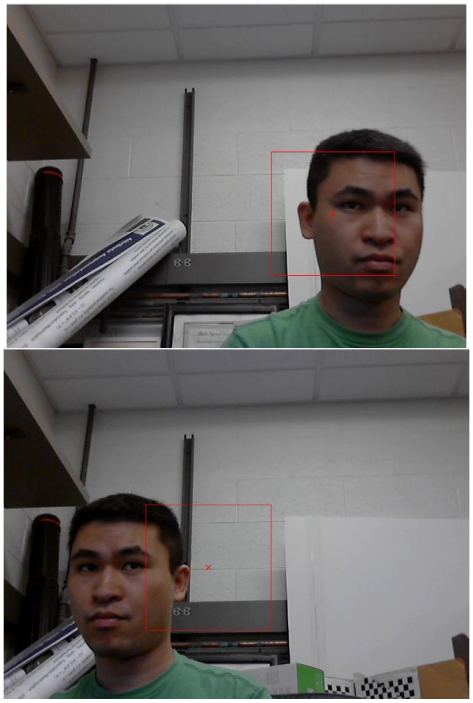

Figure 2. Results for the 17th and the 100th frame using Q – We can see the tracking error is quite a lot as compared to figure 3.

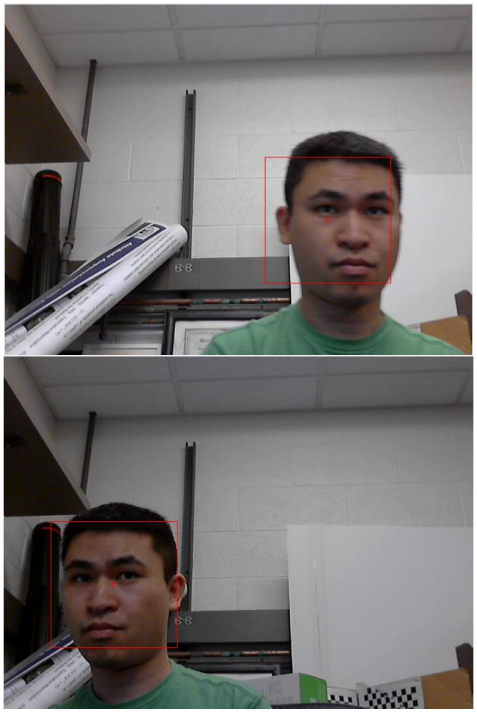

Figure 3. Final Results for the 17th and 100th image frame using Q_{bar}- We can see that there is a significantly lower tracking error and especially for the final frame where the whole face is encompassed in the binding box

Finally, we can also plot the trace of the estimated state covariance matrix to see how it changes over the frames. The trace of the Σ_{t+1} matrix is given in figure 4 (figure given after the References Section of the report for a clearer view). We can clearly see how the trace decreases. The slope is really steep at the beginning and then it starts to fall down until it stabilizes (around the fifth frame) to a constant value which means that it has found a good, consistent region of confidence around the estimate state that's closer to the actual state.

IV. Summary & Conclusions

I achieved all the goals as laid out by the project guidelines. I have achieved good results for feature tracking using the Kalman Filter.

Further work can be done to improve said algorithm by using the Extended or the Unscented version of the Kalman Filter. Also, we can still play around with different Q and R matrices to make the tracking better and see how fast the trace of the estimated state covariance matrix decreases.

Another thing I wanted to discuss in this section are the factors that could have affected the accuracy of the Kalman Filtering. It's that the raw version of the first two frames weren't provided. Remember, these frames are used to provide our initialization values. Therefore, when we use the second from to compute z_{t+1} for the third frame using SSD, we are actually also including the intensity value of the green dot at the center of the face which disrupts the results. This was eradicated by predicting the state of the third image by just using the prediction value with no dependence on the measurements. Alternatively, as some other students may have done, we could skip one or two image frames at the start. Even though my effort tries to add a little drift, I think it adheres better to the specifications of the project. Also, we see that due to the robustness of the Kalman Filter, the tracking results are very accurate.

One of the main problems encountered in this project was the choosing the right $Q$and R matrix. Other than that, I had to debug a very naïve error in my SSD computation. It turned out that the error didn't lie in my SSD code but rather in the way I was loading the image frames into *MATLAB*.

Finally the code and the final state estimate results are given at the end of the report.

References

[1] Ji, Qiang. "ECSE 6650 Lecture Notes." N.p., n.d. Web. 10 May. 2016. https://www.ecse.rpi.edu/Homepages/qji/CV/ecse6650lecture notes.html

[2] Trucco, Emanuele, and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 1998. Print.

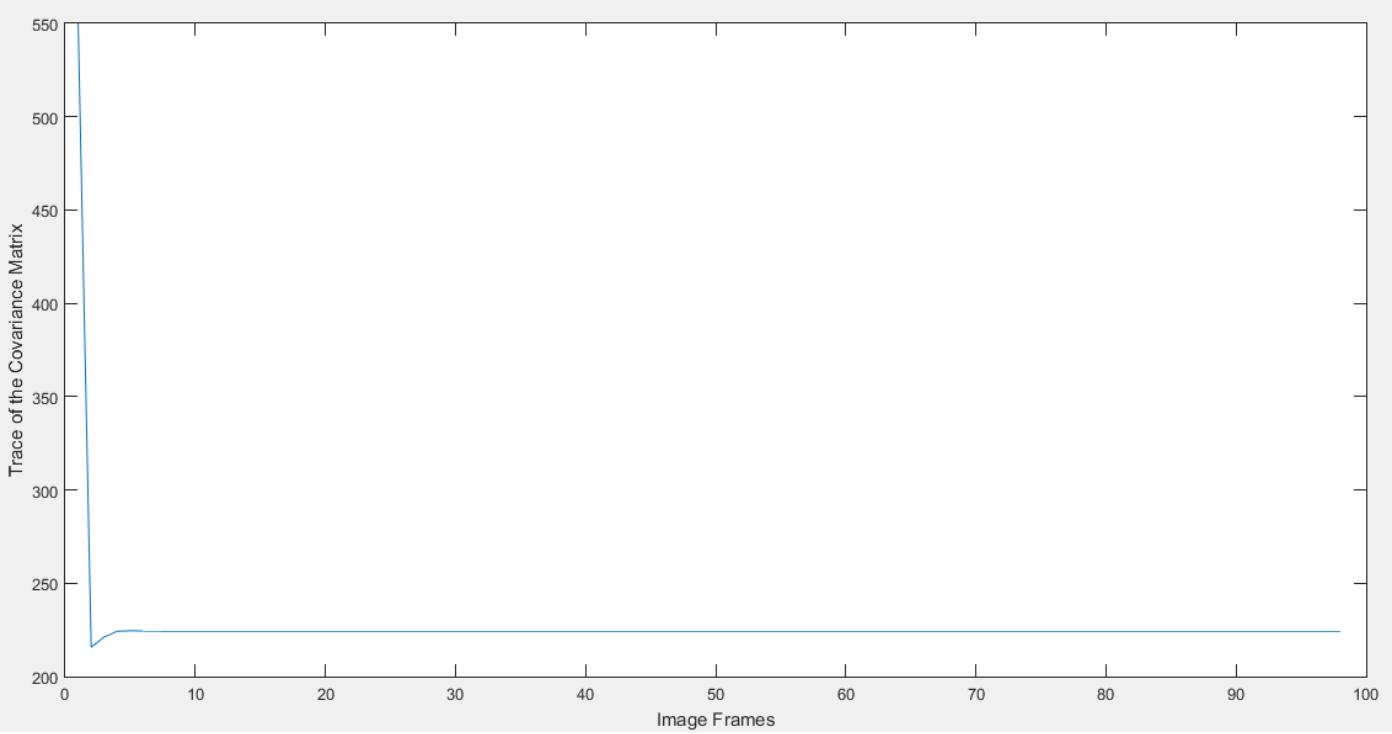

Figure 4. Trace of the Covariance Matrix of the Estimated State

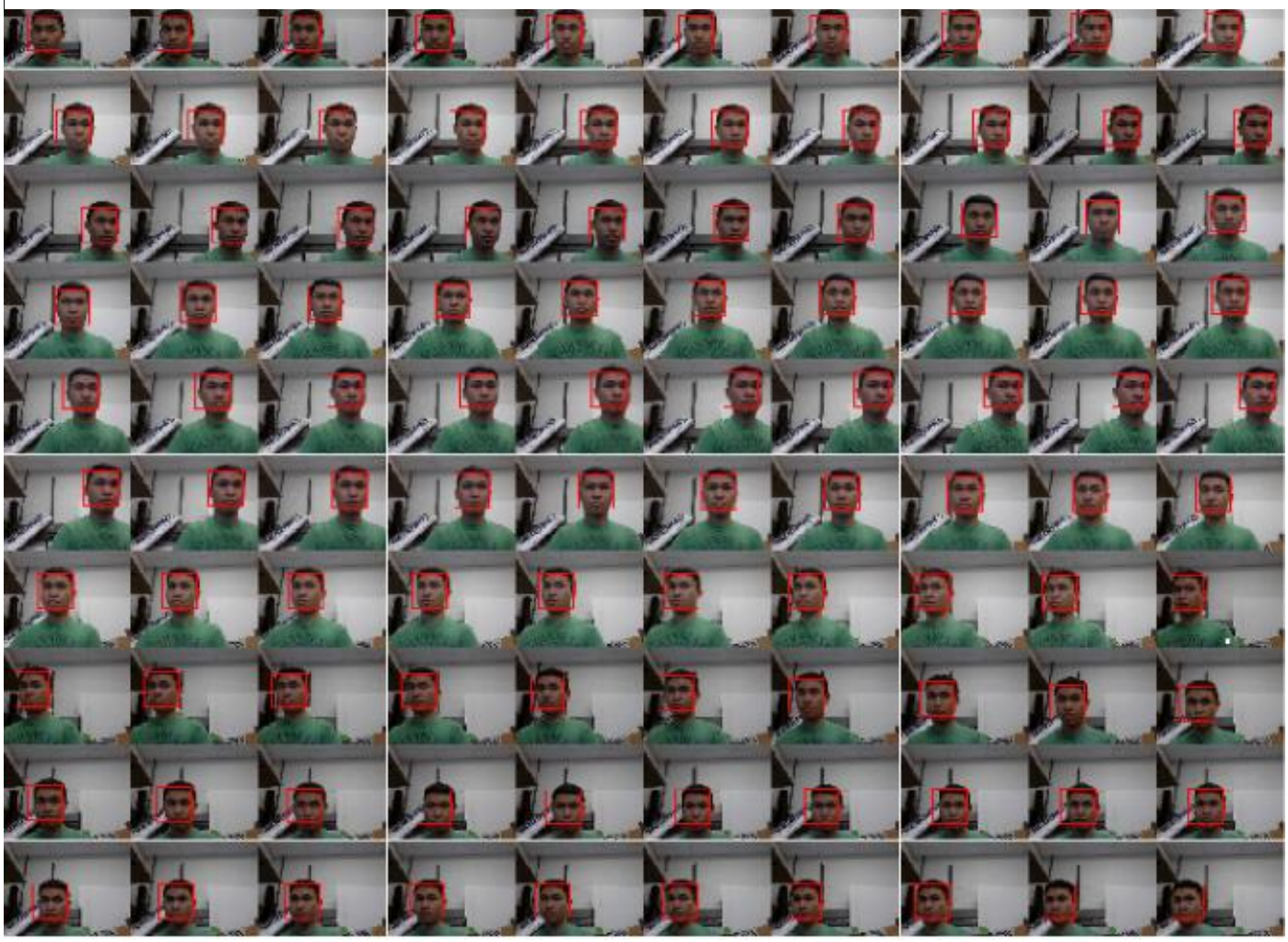

Figure 5. All the Resulting Image Frames

%% CODE

%% MAIN:

```
%% House Keeping
clc;clear;close all;
%%  Read Me.txt
% face_info = [face_center_x, face_center_y, face_width, face_height]
% face_tracking_1.png = [206, 303, 172, 172]
% face_tracking_2.png = [213, 303, 172, 172]
%
% Starting tracking from the face_tracking_3.png until face_tracking_100.png.
% Only need to track the face_center, the face size can be used to plot the
% tracking results(bounding box).
%% Read Images
N_Images = 100;%Number of Images
I = cell(1,N_Images);
%Set the 3D data equal to the chosen set
for i = 1 : N_Images
    filename = strcat('face_tracking_',num2str(i));
    filename = strcat(filename,'.png');
    I{i} = rgb2gray(imread(filename));
    %figure, imshow(I{i});
end
%% Initializations
global k;
Q = eye(4,4);
Q(1,1)=100;Q(2,2)=100;Q(3,3)=25;Q(4,4)=25;
R = diag((50)*ones(1,2));
P = eye(4,4);
P(1,1)=100;P(2,2)=100;P(3,3)=25;P(4,4)=25;
%% Initialize the state vector
face_width = 172;
face_height = 172;
x0 = 213;%centre column of k+1
y0 = 303;%centre row of k+1
vx0 = 213-206;%centre column vel of k+1
vy0 = 303-303;%centre row vel of k+1
s = [x0;y0;vx0;vy0];
%% Kalman Filter
N = 100;
state_x(1,1) = 206;
state_y(1,1) = 303;
state_x(2,1) = x0;
state_y(2,1) = y0;
Ptrace(1) = trace(P);
Ptrace(2) = trace(P);
for k = 3:N
    %img1 = double(I{k-1});
    img1 = double(I{k-1});
    img2 = double(I{k});
    [s_est,P_est] = EKF(s,P,img1,img2,Q,R);
    s = s_est; P = P_est;
    state_x(k) = s(1,1);
    state_y(k) = s(2,1);
    Ptrace(k) = trace(P_est);
```

```
end
close all;
IM = cell(1,N_Images);
for i = 1 : N_Images
    filename = strcat('face_tracking_',num2str(i));
    filename = strcat(filename,'.png');
    IM{i} = imread(filename);
    image = uint8(IM{i});
    %RGB = insertMarker(image,[state_x(i),state_y(i)]);
    imshow(image);
    hold on;
    scatter(state_x(i),state_y(i),'rx');
    rectangle('position',[(state_x(i)-172/2),(state_y(i)-172/2),172, 172],
...
    'EdgeColor','r')

    f=getframe(gca);
    [RGB, map] = frame2im(f);
%     filename = strcat('E:\Documents\Usama RPI\Computer Vision
Ji\Projects\Project 05\Result\result_',num2str(i));
%     filename = strcat(filename,'.png');
%     imwrite(RGB, filename,'png');
    hold off;
end
```

%% EKF

```
function [xest, P] = EKF(lastEst,lastP,im1,im2,Q,R)
global k
x_1 = lastEst(1,1);
x_2 = lastEst(2,1);
x_3 = lastEst(3,1);
x_4 = lastEst(4,1);
if k ==3;
    phi = [1 0 1 0;0 1 0 1;0 0 1 0;0 0 0 1];
    x_predict = phi*lastEst;
    P_predict = phi*lastP*phi' + Q;
    xest = x_predict;
    P = P_predict;
else
    %% Phi Matrix
    phi = [1 0 1 0;0 1 0 1;0 0 1 0;0 0 0 1];
    %% Compute x_predict
    x_predict = phi*lastEst;
    %% Compute P_predict
    P_predict = phi*lastP*phi' + Q;
    %% Compute z (SSD)
    col = round(lastEst(1,1));
    row = round(lastEst(2,1));
    col_p = round(x_predict(1,1));
    row_p = round(x_predict(2,1));
    submatrixP = P_predict(1:2,1:2);
    eigen=eig(submatrixP);
    width = 3*sqrt(eigen(1,1));
    height = 3*sqrt(eigen(2,1));
```

```
    win = (173-1)/2;
    smallest_sum = 10000000000000000;

    for i = (col_p-round(width/2)):(col_p+round(width/2)) %column %search
region
        for j =(row_p-round(height/2)):(row_p+round(height/2)) %row
            difference = 0;
            sum = 0;
            for ii = -win:win
                for jj = -win:win
                    difference = im1(row+jj,col+ii)-im2(j+jj,i+ii);
                    difference = difference*difference;
                    sum = sum+difference;
                end
            end
            if sum < smallest_sum
                smallest_sum = sum;
                z = [i;j];
            end
        end
    end
    %% H Matrix
    H = [1 0 0 0;0 1 0 0];
    %% Compute S and K
    S = H*P_predict*H' + R;
    K = P_predict*H'*pinv(S);
    %% Update State and P
    xest = x_predict + K*(z - H*x_predict);
    %P = (eye(4,4) - K*H)*P_predict;
    P = (eye(4,4) - K*H)*P_predict*(eye(4,4) - K*H)' + K*R*K';
end
end
```