DESCRIBE THE DATASET

Dataset for object tracking in video

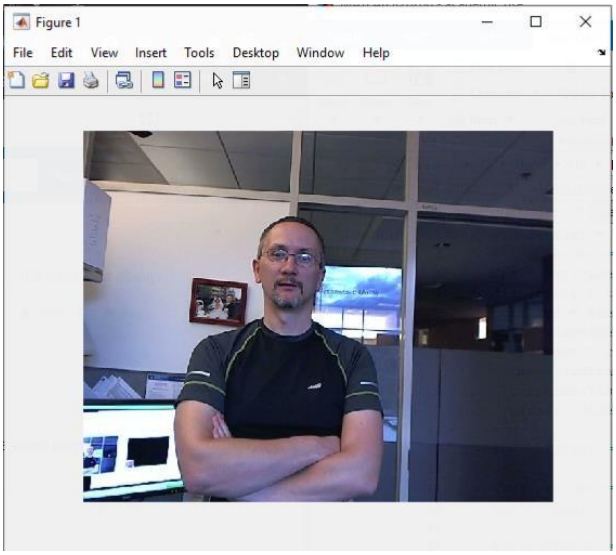
In our experimental study, we used 2 different datasets. These datasets are visionface.avi and tilted_face.avi videos that located in the mathlab library.

In our study, our goals of selecting these two datasets:

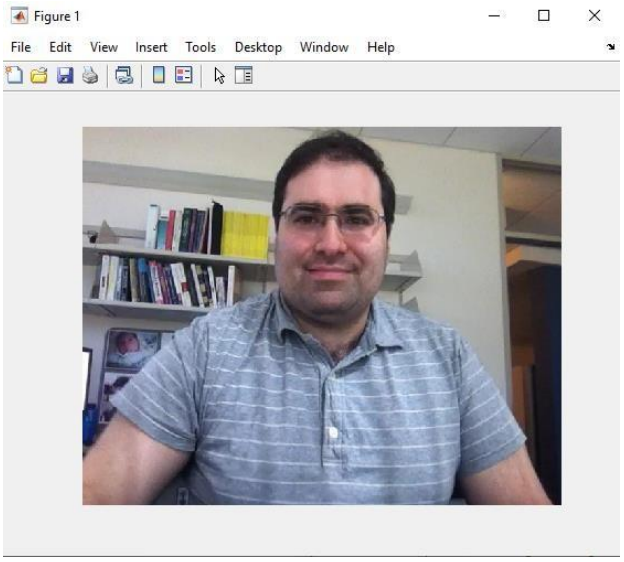
- Videos include the person.
- People's faces are clearly visible in the videos.
- Videos are clear

For this reason, we included videos which contains a human object and the face of the human object is easily detected, in our study

1) visionface.avi



2) tilted_face.avi



SHOW YOUR EXPERIMENTAL RESULTS

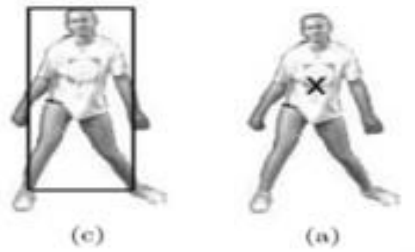
In this experimental study, we included two object tracking methods and three example code. These methods are as follows

1) Kalman Filter
2) Particle Filter

Let's examine these methods and experimental results in order.

1) Kalman Filter

Using this method, we realized the object presentation with centroid and rectangular patchs.



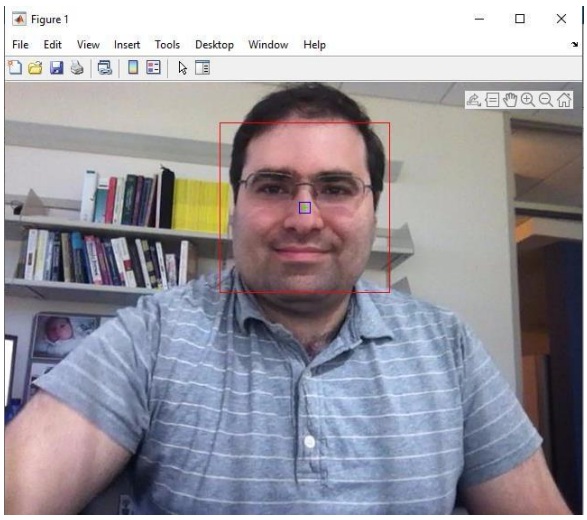
Kalman filters are used to make state predictions of linear systems whose state vector has a Gaussian distribution. It makes a probabilistic prediction about the future motion of the tracked object by looking at its past and present motion. Kalman filter is used to soften the hard and sudden movements of objects and to predict the unmeasurable conditions of the system. In this context, they consist of two steps. The first one is measurement update or correction and the other is forecast / time update. The data to be predicted works in a loop between these two structures and ensures that the forecast converges to the true value in the loop.

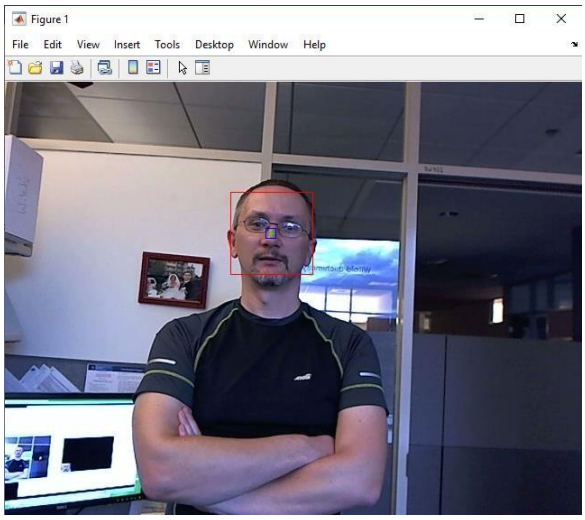
Estimation done by predict the object's previous location in the frame compared with the current frame. The goal is produce detection algorithm by smoothing the result even there are no detect object in the frame. Therefore, even if object does not appear in the scene, we can predict its location by using kalman filter. We start by writing a loop statement which observes the video frame by frame. After getting the frames we initialize kalman filter:

('ConstantVelocity', [240, 180], [1 1]*1e5, [10,10], 15);

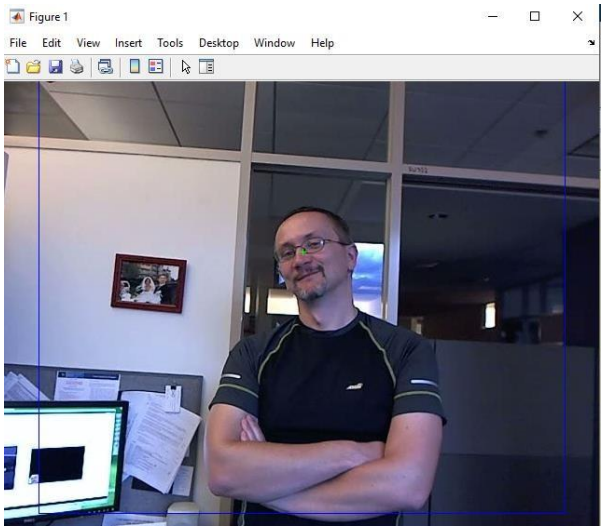
We define the Kalman Filter as the dataset is under a concentrated Gaussian function in the center of the screen.

Therefore, we accomplished results below:





However, due to irreguler behaviors of objects, we can not detect the regions of object but still can track center of object.



We can overcome this incorrect results by particle tracking.

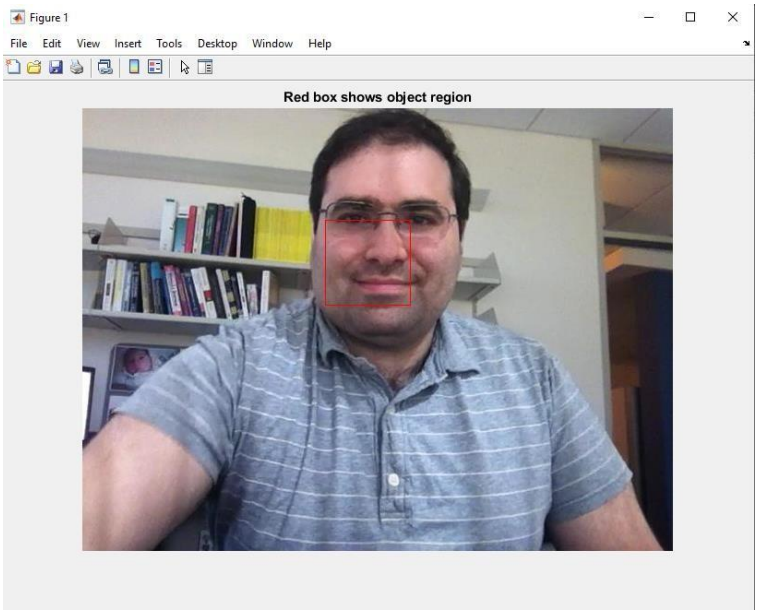
2.a)Particle Filter

Using this method, we realized the object presentation with multiple points.

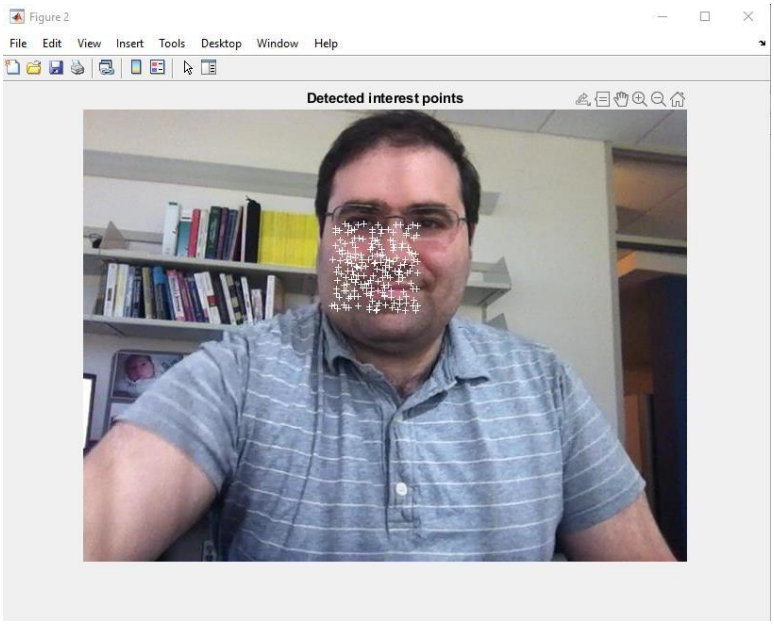
(b)

Thanks to the multiple points, we placed on the face of the object, we could easily follow the object in the video.

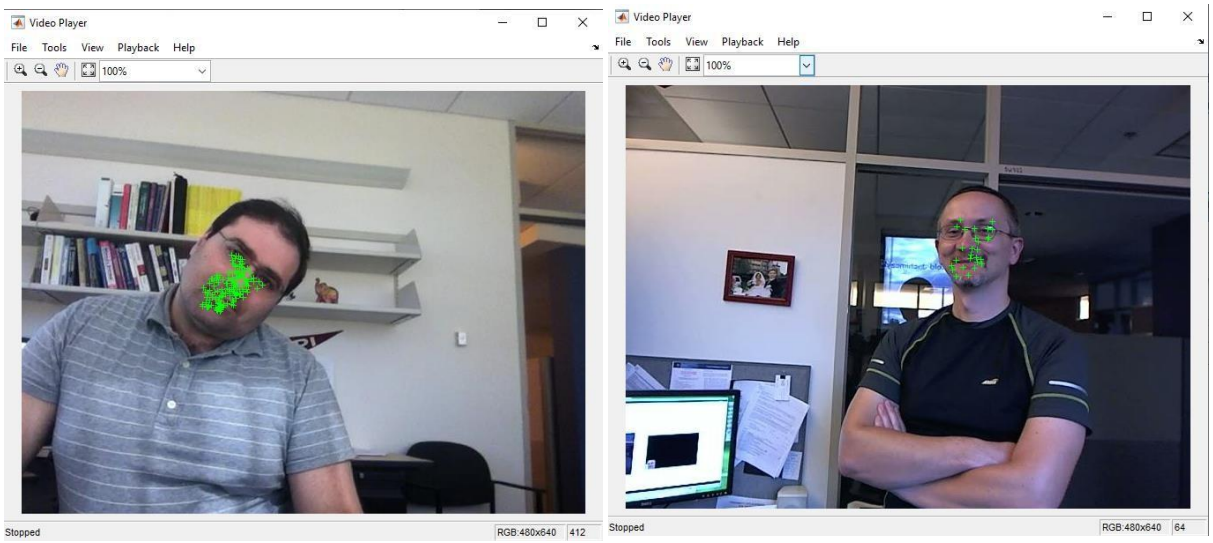
First we initialize regions of object with eigenvalue algorithm.



Then, insert a geometrical shape as border. After that, we fill inside of this shape with points.



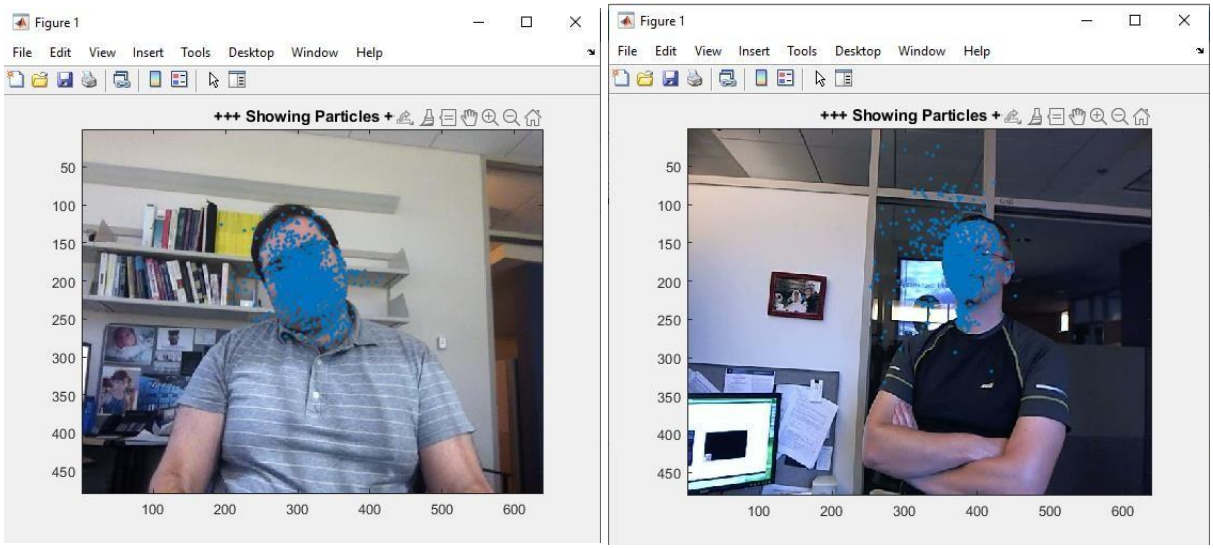
After we filled the shape with points, we initialize tracker variable with point tracking algorithm. Then, until there are no frames left, we update these markers and insert it into frames.



2.b) Particle Filter – 2

We will include our results about the last study whose code we got from the internet.

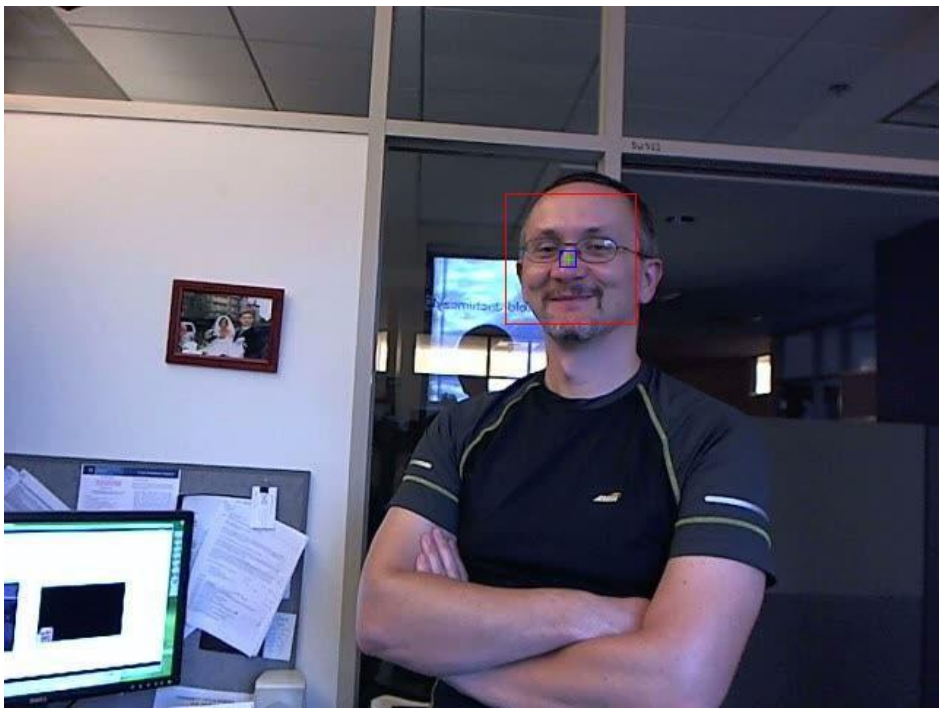
Using this method, the object presentation is realized with multiple points.



1)Kalman Filter

```
bc=ones(1,2);

%%%Creating face detector
faceDetector = vision.CascadeObjectDetector();
faceDetector.MergeThreshold  =25;

%%%Other Datasets that we tried
%vr = vision.VideoFileReader('tilted_face.avi');
vr = vision.VideoFileReader('visionface.avi');
%vr = vision.VideoFileReader('Person.wmv');
% vr = vision.VideoFileReader('atrium.mp4');

%%%Using configureKalmanFilter library to detecte data's Face
%%%We used ConstantVelocity as a motion model
kalmanFilter = configureKalmanFilter('ConstantVelocity', [240, 180], [1 1]*1e5,
[10,10], 15);
while ~isDone(vr)

%%%We returns the current frame with this
    frame = step(vr);
%%%We can predict the object's current position compare with its previous
%%%position(Corrected/smoothed)with the help of motion model
%%% predictedLocation = [x, dot(x), y, dot(y)]
    [~, predictedLocation, P]= predict(kalmanFilter);
    predictedLocation = predictedLocation(1:2:4);
    bbox=step(faceDetector,frame);
%%%This part from internet
    if(~isempty(bbox))
        xpos=bbox(1) + (bbox(3)/2) ;
        ypos=bbox(2) + (bbox(4)/2) ;
        bc=[xpos,ypos];
    end
%%%This ifstatement checks whether there any detection or not
    if numel(bbox(:)) ~= 0
     [~, smooth_bc, P] = correct(kalmanFilter, bc);
     smooth_bc = smooth_bc(1:2:4);
    else
      smooth_bc = predictedLocation;
    end
%%%Draws detection function
    boxInserter  = vision.ShapeInserter('BorderColor','Custom',...
        'CustomBorderColor',[255 0 0]);
%%%Estimated result
```

```
    boxInserterBelief    = vision.ShapeInserter('BorderColor','Custom',...
        'CustomBorderColor',[0 0 255]);
    %Red box
    videoOut = step(boxInserter, frame, bbox);
    belief = [smooth_bc(1)-P(1,1)/2, smooth_bc(2)-P(3,3)/2, P(1,1), P(3,3)];
    %Blue box
    videoOut = step(boxInserterBelief, videoOut, belief);
    %Plus Sign
    videoOut = insertMarker(videoOut, smooth_bc);
    imshow(videoOut,'border','tight');

end
```

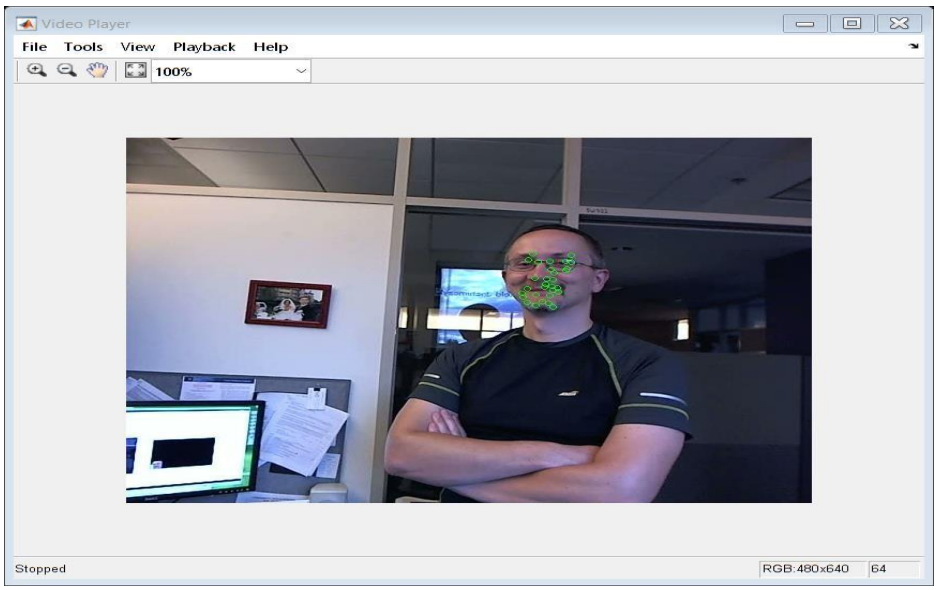


```
release(vr);
```

2.a)Particle Filter

```
%%%Other Datasets that we tried
vr = VideoReader('visionface.avi');
%vr = vision.VideoFileReader('visionface.avi');
% vr = vision.VideoFileReader('atrium.mp4');

%%%For change initial position of video screen
vp = vision.VideoPlayer("Position",[500,100,680,520]);

%%%Read next video frame
```

```
objectFrame = readFrame(vr);
%%%Region of Object
objectRegion = [264,132,93,93];

%%%Detect corners using minimum eigenvalue algorithm and return cornerPoints
object
points = detectMinEigenFeatures(rgb2gray(objectFrame),'ROI',objectRegion);
tracker = vision.PointTracker('MaxBidirectionalError',1);

%%%Executes contents on a model initialize event
initialize(tracker,points.Location,objectFrame);

%%%Tracker updates points until there are no frames
while hasFrame(vr)
        frame = readFrame(vr);
        [points,validity] =  tracker(frame);
        out = insertMarker(frame,points(validity, :),'o');
        vp(out);
end
release(vp);
```



2.b)Particle Filter -2

```
F_update = [1 0 0 0; 0 1 0 1;  0 0 1 0; 0 0  0 1];

Npop_particles = 10000;

color_distribution = 60;
position = 15;
```

```
vec = 5;

target = [255; 0; 0];

vr = VideoReader('visionface.avi');

resolution = [vr.Width vr.Height];
movie = floor(vr.Duration * vr.FrameRate);

%% Object Tracking by Particle Filter
particles = create_particles(resolution, Npop_particles);

for k = 1:movie

    % Getting Image
    frame = read(vr, k);

    %  Forecasting
    particles = update_particles(F_update, position, vec,  particles);

    % Calculating Log Likelihood
    Likelihood = calc_log_likelihood(color_distribution, target,  particles(1:2,
:), frame);

    % Resampling
    particles = resample_particles(particles, Likelihood);

    % Showing Image
    show_particles(particles, frame);
% show_state_estimated(X, Y_k);

end
```

```
function X = create_particles(Npix_resolution, Npop_particles)

X1  =  randi(Npix_resolution(2),  1,    Npop_particles);
X2  =  randi(Npix_resolution(1),  1,    Npop_particles);
X3 = zeros(2, Npop_particles);

X = [X1; X2; X3];
end
function L = calc_log_likelihood(Xstd_rgb, Xrgb_trgt, X, Y)

Npix_h = size(Y, 1);
```

```
Npix_w = size(Y, 2);

N = size(X,2);

L = zeros(1,N);
Y = permute(Y, [3 1 2]);

A =  -log(sqrt(2 * pi) * Xstd_rgb);
B = - 0.5 ./ (Xstd_rgb.^2);

X =  round(X);

for k = 1:N

      m  =  X(1,k);
      n  =  X(2,k);

      I = (m >= 1 & m <= Npix_h);
      J = (n >= 1 & n <= Npix_w);

      if I && J

            C = double(Y(:, m, n));

            D = C - Xrgb_trgt;

            D2 = D' * D;

            L(k)  =    A + B * D2;
      else

            L(k) = -Inf;
      end
end
end
function X = resample_particles(X, L_log)

% Calculating Cumulative Distribution

L = exp(L_log - max(L_log));
Q = L ./ sum(L, 2);
R = cumsum(Q, 2);

% Generating Random Numbers
```

```
N = size(X, 2);
T = rand(1, N);

% Resampling

[~, I] = histc(T, R);
X = X(:, I + 1);
end
function show_particles(X, Y_k)

figure(1)
image(Y_k)
title('+++ Showing Particles +++')

hold on
plot(X(2,:), X(1,:), '.')
hold off

drawnow
end
function X = update_particles(F_update, Xstd_pos, Xstd_vec, X)

N = size(X, 2);

X = F_update * X;

X(1:2,:) = X(1:2,:) + Xstd_pos * randn(2, N);
X(3:4,:) = X(3:4,:) + Xstd_vec * randn(2, N);
end
```