```
videoFile='5289929-preview.mp4';
% Create a video file reader.
i=1;
obj.reader = vision.VideoFileReader(videoFile, 'VideoOutputDataType', 'uint8');
        % Create a video player.
        obj.videoPlayer = vision.VideoPlayer('Position', [29, 597, 643, 386]);
while ~isDone(obj.reader)
        frame = step(obj.reader);
        imwrite(frame,[int2str(i),'.jpg'],'JPEG');
        i=i+1;
end
```

get listing of frames.

```
f_list =  dir('*jpg');
size(f_list)
```

initialize gaussian filter

```
%using fspecial, to make a laplacian of a gaussian (LOG)
hsizeh = 25
sigmah = 4.4
h = fspecial('log', hsizeh, sigmah)
subplot(121); imagesc(h)
subplot(122); mesh(h)
```

iteratively (frame by frame) find flies and save the X Y coordinates!

```
X = cell(1,length(f_list)); %detection X coordinate indice
Y = cell(1,length(f_list));  %detection Y coordinate indice
 img_real = (imread(f_list(1).name)); %for plotting
    figure,image(img_real),truesize;

for i = 1:length(f_list)
%for i = 210:220
    img_real = (imread(f_list(i).name)); %for plotting
   % figure,image(img_real),truesize;
    img_tmp = double(imread(f_list(i).name)); %load in the image and convert to
double
    img = img_tmp(:,:,1); %reduce to just the first dimension

    %do the blob filter
    blob_img = conv2(img,h,'same');

    %threshold the image to blobs only
    idx = find(blob_img < 0.7);
    blob_img(idx) = nan ;


    % 2-d local max/min finder
```

```
    %http://www.mathworks.com/matlabcentral/fileexchange/12275-extrema-m-
extrema2-m

    [zmax,imax,zmin,imin] = extrema2(blob_img);
    [X{i},Y{i}] = ind2sub(size(blob_img),imax);

    %for plotting
    %%{
%     clf
%     subplot(211);
%     imagesc(blob_img)
%        axis off
%     subplot(212)
%     imshow(img_real)
%     hold on
%     for j = 1:length(X{i})
%         plot(Y{i}(j),X{i}(j),'or')
%     end
%     axis off
%     pause(0.1)
    %}

    i
end
%save it
save('raw_fly_detections.mat',  'X','Y')
```

```
load('raw_fly_detections.mat')
%get frame list
f_list =  dir('*jpg');
size(f_list)
```

define main variables for KALMAN FILTER

```
dt = 1;  %our sampling rate
S_frame = 10 % find(cellfun(@length, X)>11,1); %starting frame
u = 0; % define acceleration magnitude to start
HexAccel_noise_mag = 1; %process noise
tkn_x = .1;  %measurement noise in the horizontal direction (x axis).
tkn_y = .1;  %measurement noise in the horizontal direction (y axis).
Ez = [tkn_x 0; 0 tkn_y];
Ex = [dt^4/4 0 dt^3/2 0; ...
    0 dt^4/4 0 dt^3/2; ...
    dt^3/2 0 dt^2 0; ...
    0 dt^3/2 0 dt^2].*HexAccel_noise_mag^2; % Ex convert the process noise (stdv)
into covariance matrix
P = Ex; % estimate of initial position variance (covariance matrix)
```

Define update equations in 2-D (Coefficent matrices): A physics based model [state transition (state + velocity)] + [input control (acceleration)]

```
A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1]; %state update matrice
B = [(dt^2/2); (dt^2/2); dt; dt];
C = [1 0 0 0; 0 1 0 0];  %this is our measurement function C, that we apply to
the state estimate Q to get our expect next/new measurement
```

initialize result variables

```
Q_loc_meas = []; % detecions  extracted by the detection algo
```

initialize estimation variables for two dimensions

```
Q= [X{S_frame} Y{S_frame} zeros(length(X{S_frame}),1)
zeros(length(X{S_frame}),1)]'
Q_estimate = nan(4,2000);
Q_estimate(:,1:size(Q,2)) = Q;  %estimate of initial location estimation
Q_loc_estimateY = nan(2000); %  position estimate
Q_loc_estimateX= nan(2000); %  position estimate
P_estimate = P;  %covariance estimator
strk_trks = zeros(1,2000);  %counter of how many strikes a track has gotten
nD = size(X{S_frame},1); %initize number of detections
nF =  find(isnan(Q_estimate(1,:))==1,1)-1 ; %initize number of track estimates
%for each frame
for t = S_frame:length(f_list)-1

    % load the image
    img_tmp = double(imread(f_list(t).name));
    img = img_tmp(:,:,1);
    % make the given detections matrix
    Q_loc_meas = [X{t} Y{t}];
```

do the kalman filter

Predict next state with the last state and predicted motion.

```
    nD = size(X{t},1); %set new number of detections
    for F = 1:nF
        Q_estimate(:,F) = A * Q_estimate(:,F) + B * u;
    end

    %predict next covariance
    P = A * P* A' + Ex;
    % Kalman Gain
    K = P*C'*inv(C*P*C'+Ez);


    % assign the detections to estimated track positions
    %make the distance (cost) matrice between all pairs rows = tracks, coln =
    %detections
```

```
    est_dist = pdist([Q_estimate(1:2,1:nF)'; Q_loc_meas]);
    est_dist = squareform(est_dist); %make square
    est_dist = est_dist(1:nF,nF+1:end) ; %limit to just the tracks to detection
distances

    [asgn, cost] = ASSIGNMENTOPTIMAL(est_dist); %do the assignment with hungarian
algo
    asgn = asgn';

    %check 1: is the detection far from the observation? if so, reject it.
    rej = [];
    for F = 1:nF
        if asgn(F) > 0
            rej(F) =  est_dist(F,asgn(F)) < 50 ;
        else
            rej(F) = 0;
        end
    end
    asgn = asgn.*rej;


    %apply the assingment to the update
    k = 1;
    for F = 1:length(asgn)
        if asgn(F) > 0
            Q_estimate(:,k) = Q_estimate(:,k) + K * (Q_loc_meas(asgn(F),:)' - C *
Q_estimate(:,k));
        end
        k = k + 1;
    end

    % update covariance estimation.
    P =  (eye(4)-K*C)*P;
```

Store data

```
    Q_loc_estimateX(t,1:nF) = Q_estimate(1,1:nF);
    Q_loc_estimateY(t,1:nF) = Q_estimate(2,1:nF);

    % new detections and lost trackings

    %find the new detections
    new_trk = [];
    new_trk = Q_loc_meas(~ismember(1:size(Q_loc_meas,1),asgn),:)';
    if ~isempty(new_trk)
        Q_estimate(:,nF+1:nF+size(new_trk,2))=  [new_trk;
zeros(2,size(new_trk,2))];
        nF = nF + size(new_trk,2);  % number of track estimates with new ones
included
    end
```

```
    %give a strike to any tracking that didn't get matched up to a
    %detection
    no_trk_list =  find(asgn==0);
    if ~isempty(no_trk_list)
        strk_trks(no_trk_list) = strk_trks(no_trk_list) + 1;
    end

    %if a track has a strike greater than 6, delete the tracking. i.e.
    %make it nan first vid = 3
    bad_trks = find(strk_trks > 6);
    Q_estimate(:,bad_trks) = NaN;

    %%{
    clf
    img = imread(f_list(t).name);
    imshow(img);
    hold on;
    plot(Y{t}(:),X{t}(:),'or'); % the actual tracking
    T = size(Q_loc_estimateX,2);
    Ms = [3 5]; %marker sizes
    c_list = ['r' 'b' 'g' 'c' 'm' 'y']
    for Dc = 1:nF
        if ~isnan(Q_loc_estimateX(t,Dc))
            Sz = mod(Dc,2)+1; %pick marker size
            Cz = mod(Dc,6)+1; %pick color
            if t < 21
                st = t-1;
            else
                st = 19;
            end
            tmX = Q_loc_estimateX(t-st:t,Dc);
            tmY = Q_loc_estimateY(t-st:t,Dc);
            plot(tmY,tmX,'.-
','markersize',Ms(Sz),'color',c_list(Cz),'linewidth',0.1)
            axis off
        end
    end
    pause(0.1)
    %}

    t

end
%reviewing S_frame
for t = 300:length(f_list)-1 %S_frame:length(f_list)
    clf;
    img = imread(f_list(t).name);
    imshow(img);
    hold on;
```

```
    plot(Y{t}(:),X{t}(:),'or'); % the actual tracking
    T = size(Q_loc_estimateX,2);
    Ms = [3 5]; %marker sizes
    c_list = ['r' 'b' 'g' 'c' 'm' 'y'];
    for Dc = 1:nF
        if ~isnan(Q_loc_estimateX(t,Dc))
            Sz = mod(Dc,2)+1; %pick marker size
            Cz = mod(Dc,6)+1; %pick color
            if t < 21
                st = t-1;
            else
                st = 19;
            end
            tmX = Q_loc_estimateX(t-st:t,Dc);
            tmY = Q_loc_estimateY(t-st:t,Dc);
            plot(tmY,tmX,'.-
','markersize',Ms(Sz),'color',c_list(Cz),'linewidth',0.1)
            axis off
        end
    end
    t
    pause(0.1)
end
```