EXPERIMENT - 2 FINAL REPORT

OBJECT TRACKING ALGORITHM

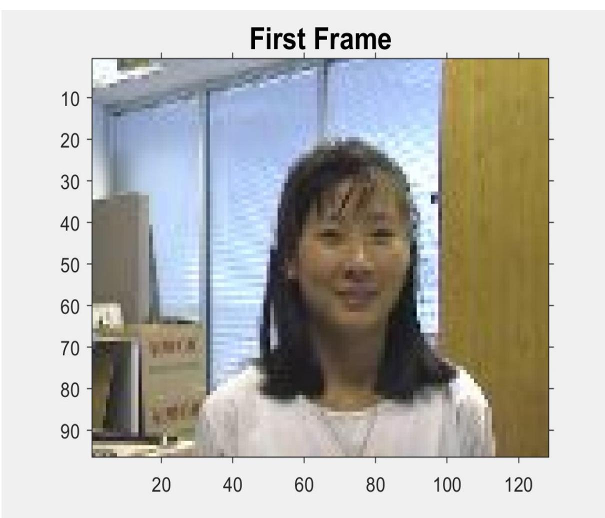
Archana Narayanan

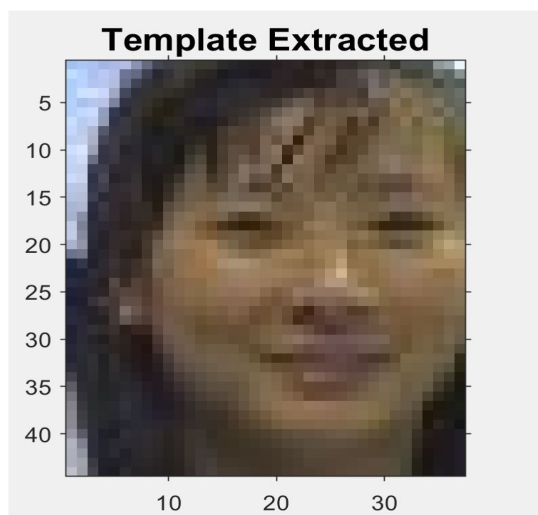
TEMPLATE MATCHING:

For template matching, a portion of the image is extracted from the first frame to form a template. The small portion is extracted using four element position vector values from first frame which will constitute the template for the video. Normalized Cross Correlation is used to calculate the similarity of features between the last frame and template. Normalized Cross Correlation is preferred over the ordinary cross correlation since it is less sensitive to linear changes in the amplitude of illumination in the two compared images. Additionally, the Normalized Cross Correlation is confined in the range between –1 and 1. The setting of detection threshold value is much simpler than the cross correlation[Ref1]. The maximum value of the normalized cross correlation and its location helps identify the location of the template in the last frame.

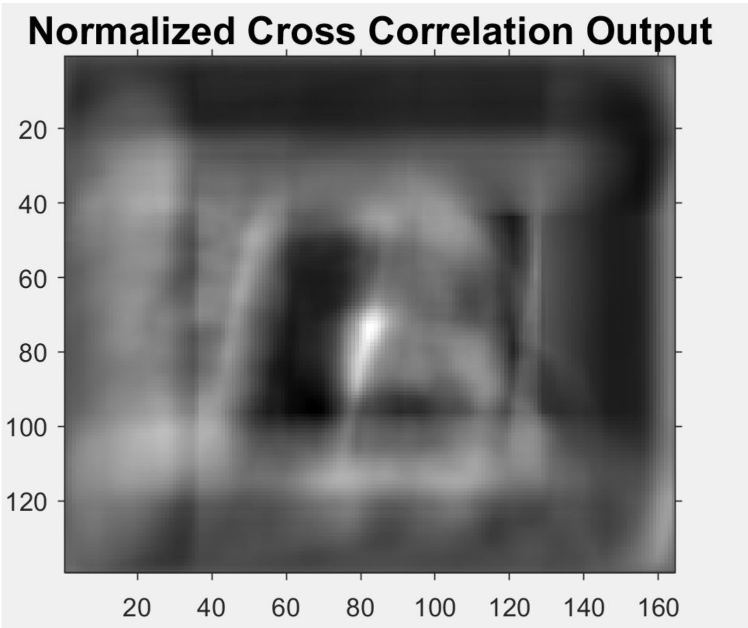
Results of Template Matching on Girl Video :

Template Extracted from First Frame:

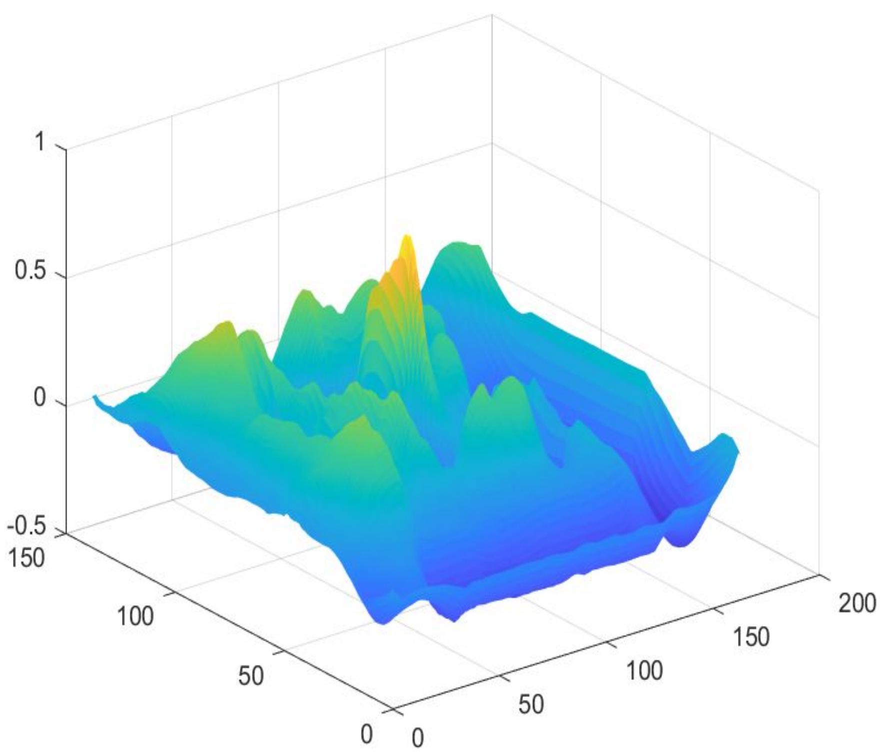




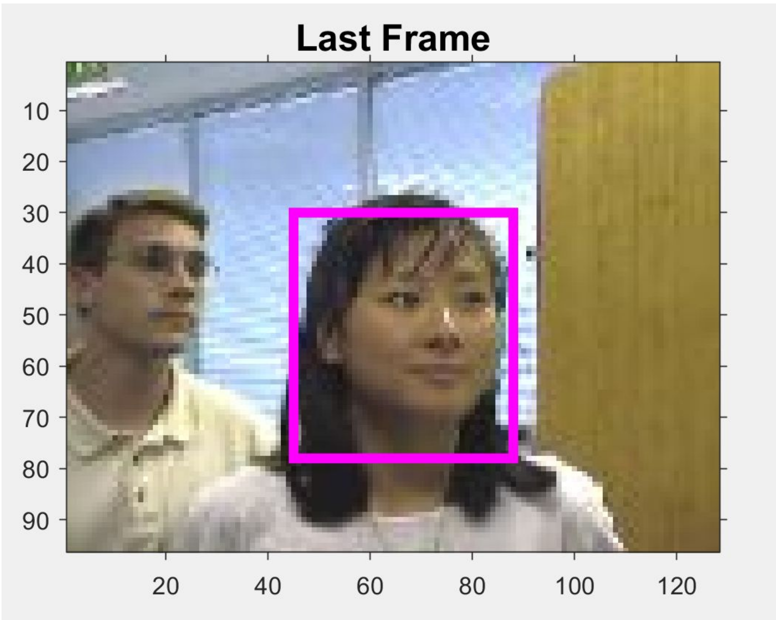
Normalized Cross Correlation Results between the template and last frame:



Peaks of the cross-correlation matrix where the images are best correlated:



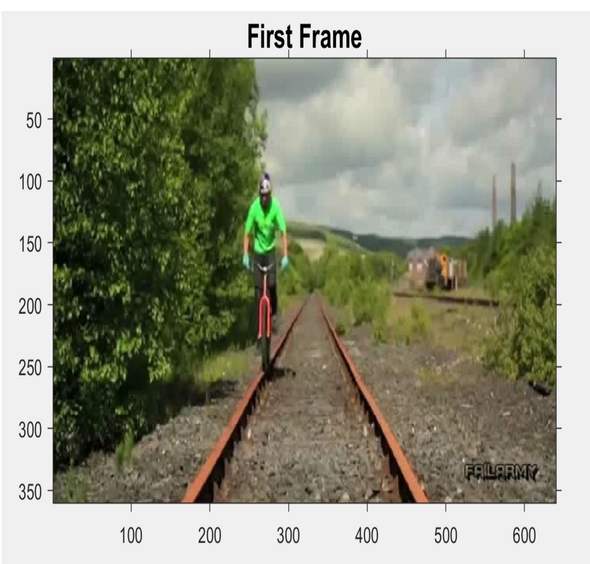
Location of the template in the last frame : (Frame 500)



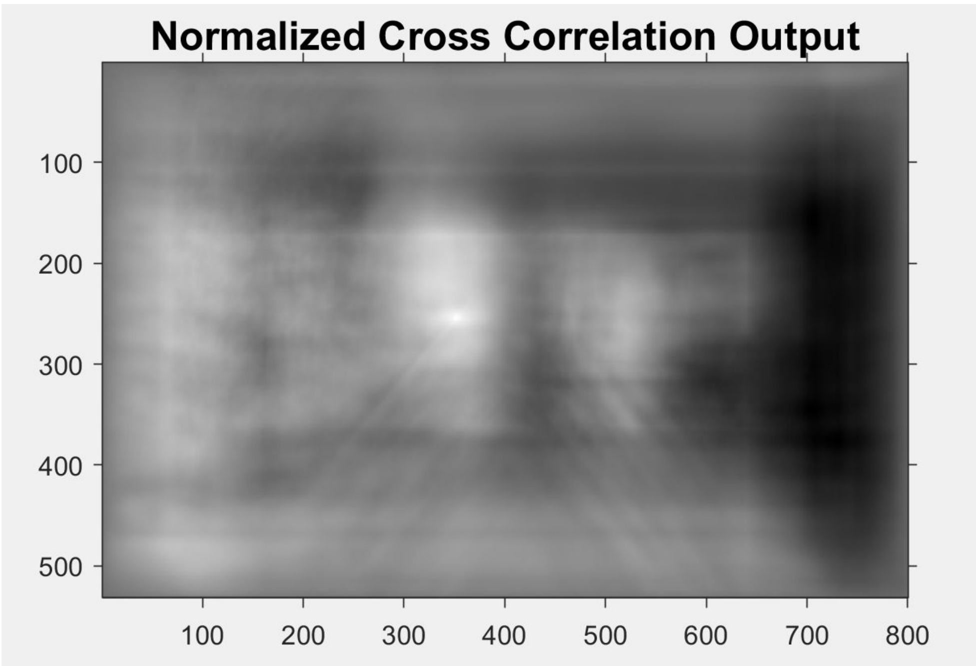
IoU Overlap with Ground Truth of Last Frame : 63.58%

Results of Template Matching on Biker Video :

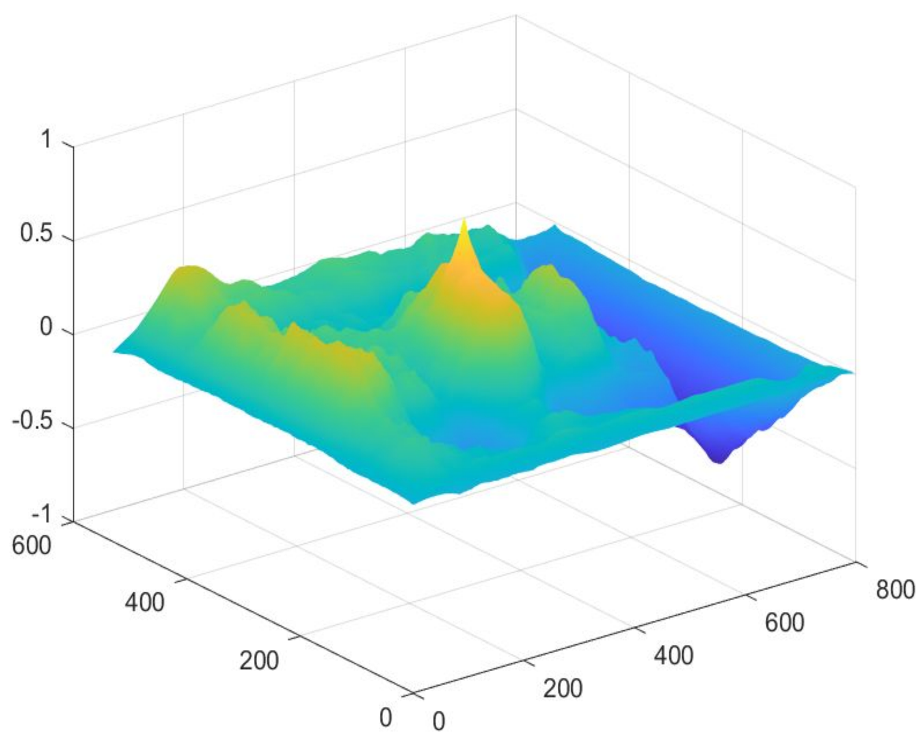
Template Extracted from First Frame:



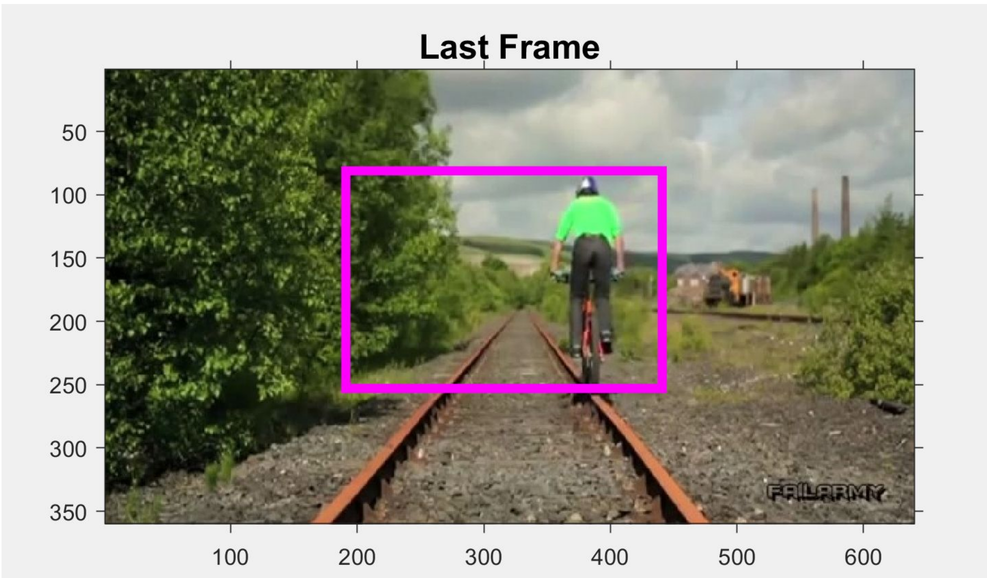
Normalized Cross Correlation Results between the template and last frame:



Peaks of the cross-correlation matrix where the images are best correlated:



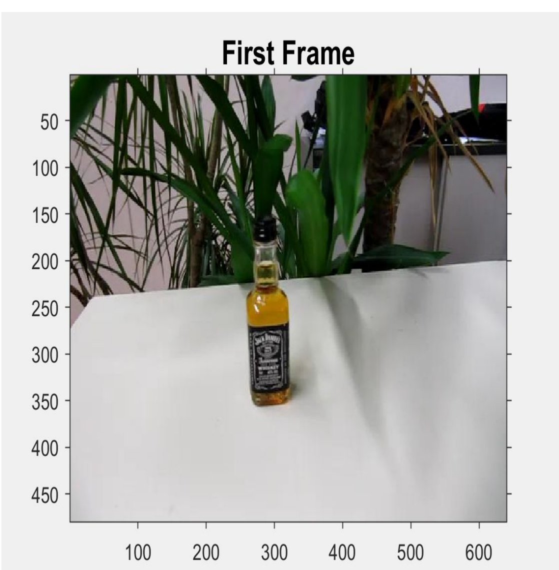
Location of the template in the last frame : (Frame 500)



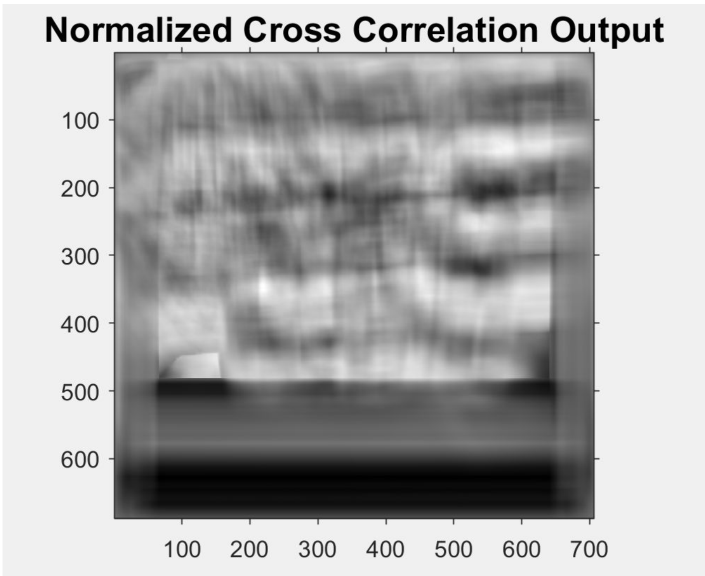
IoU Overlap with ground truth = 76%

Results of Template Matching on Liquor Video(Video of own choice) :

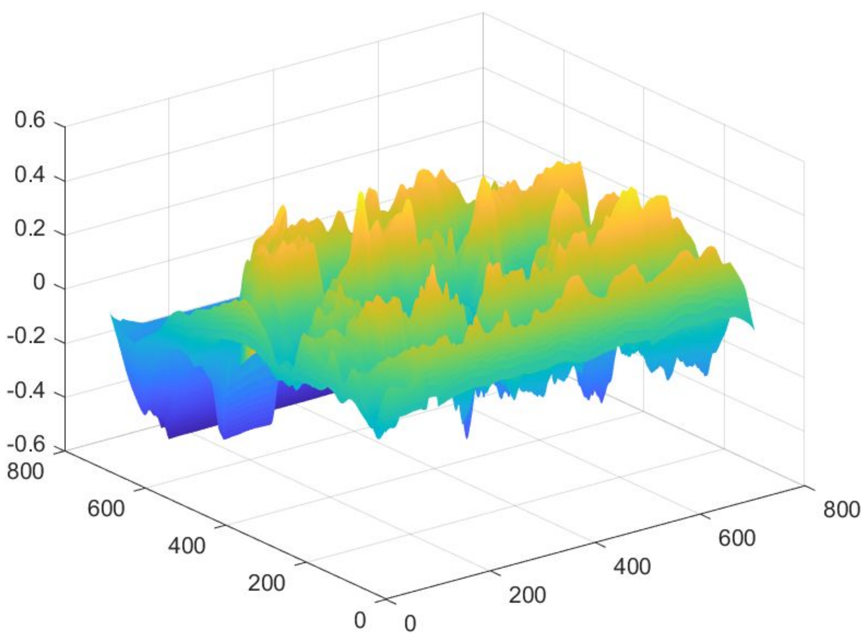
Template Extracted from First Frame:



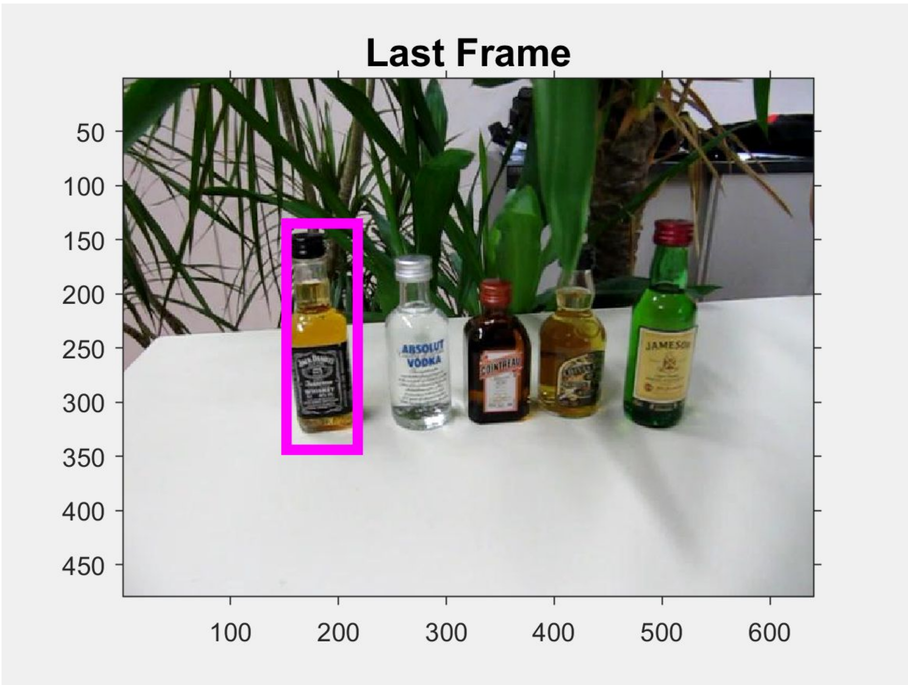
Normalized Cross Correlation Results between the template and last frame:



Peaks of the cross-correlation matrix where the images are best correlated:

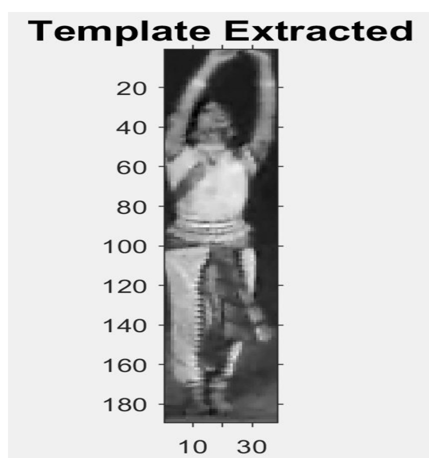


Location of the template in the last frame : (Frame 500)

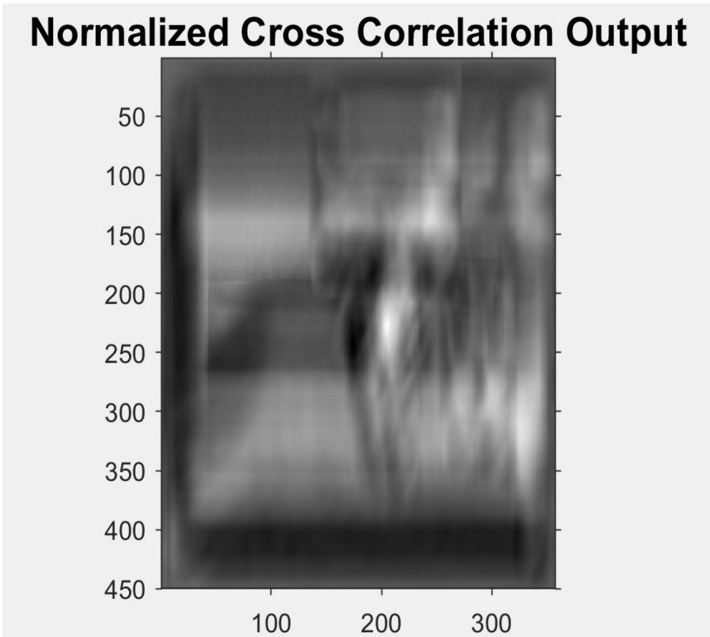


Results of Template Matching on Dance2 Video(Video of own choice) :

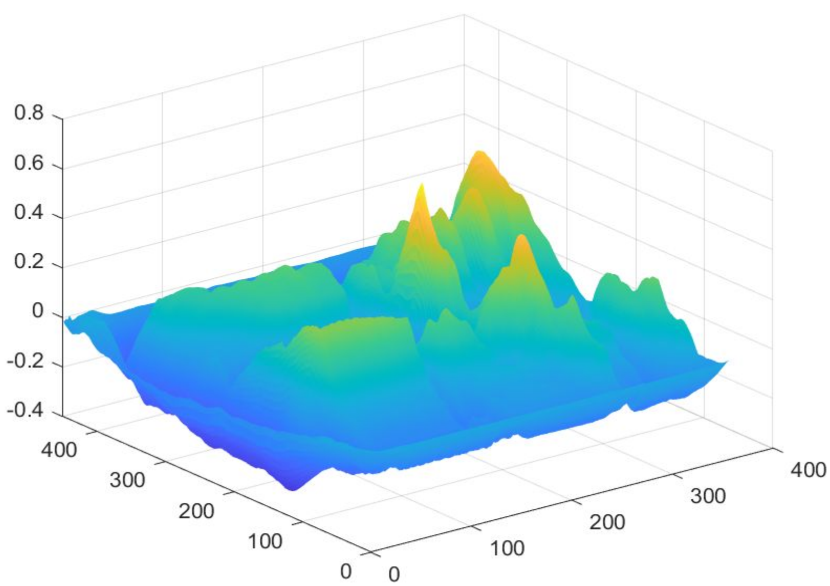
Template Extracted from First Frame:



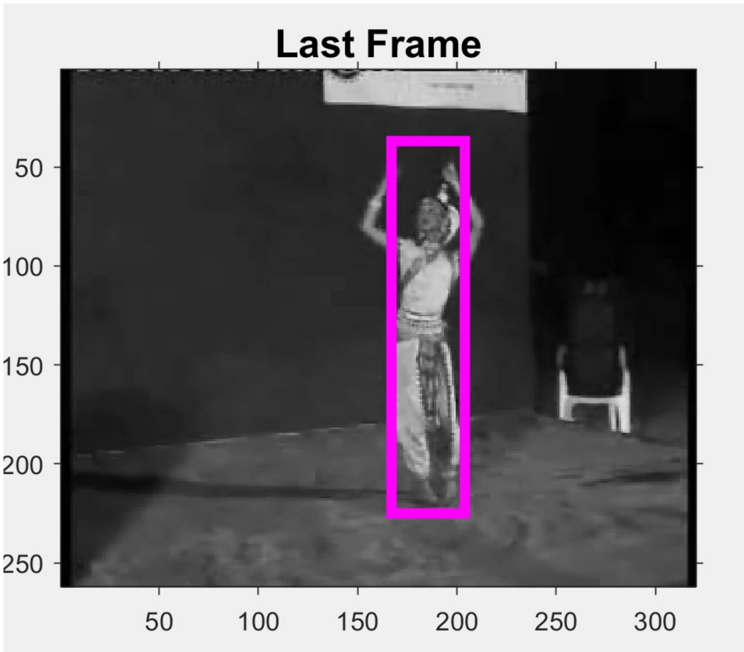
Normalized Cross Correlation Results between the template and last frame:



Peaks of the cross-correlation matrix where the images are best correlated:



Location of the template in the last frame : (Frame 150)



IoU Overlap with ground truth = 71.0630 %

TRACKING ALGORITHM:

For the tracking algorithm, background subtraction is performed to identify the object of interest for tracking since the background remains the same in these videos. By taking the difference between the object of interest and the stationary part of the image, the background is computed across all the frames. From these set of background images the mean/ average value of these frames is considered to form a reference background image.

In addition to this , eigen background is computed using region based spatial information(from literature readings). Eigen model is used to create background using normalized mean and variance of these images. A single frame is represented as a column vector and a sample of N images are considered and their mean is computed. The mean normalized image vectors are gathered to form the singular value decomposition. The Eigen background is computed and image is reduced in this subspace. By computing the difference between the input image and reduced subspace image, the moving object is detected.

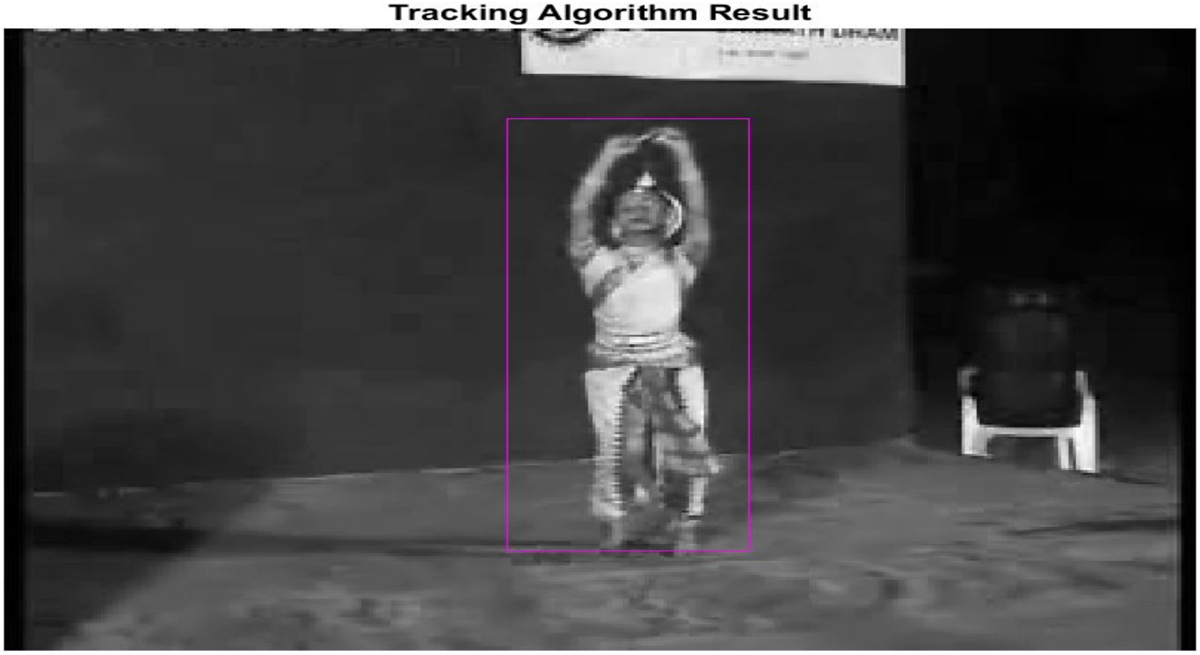
Erode and Dilate operations is performed on the final image for improved results.

I also tried using average gaussian function to create the background image where each pixel's history is modelled as an average gaussian function. The results are attached for biker video. However this did not work well for Girl etc video where there is more than one moving object in frames.

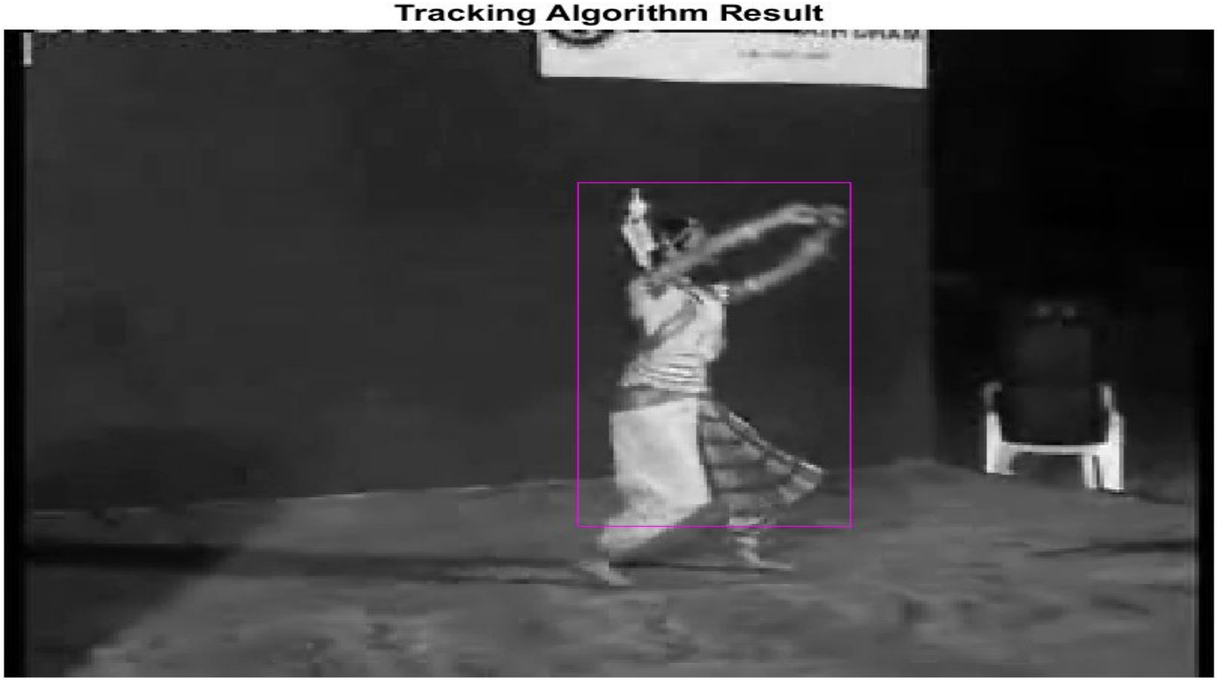
The values of the bounding box are extracted for each frame and compared with their respective ground truth values. For evaluation of the results with the ground truth, the **Intersection Over Union Value** is calculated for each frame and the value is plotted in graph. It can be seen that it produced results >50% for most of the frames. Some of frames have a lesser overlap percentage since the boundaries of the template vary between the ground truth and that produced by the algorithm. This causes a major difference in overlap region and results in lesser IoU value. However, the algorithm tracks the object throughout the video for the most part successfully.

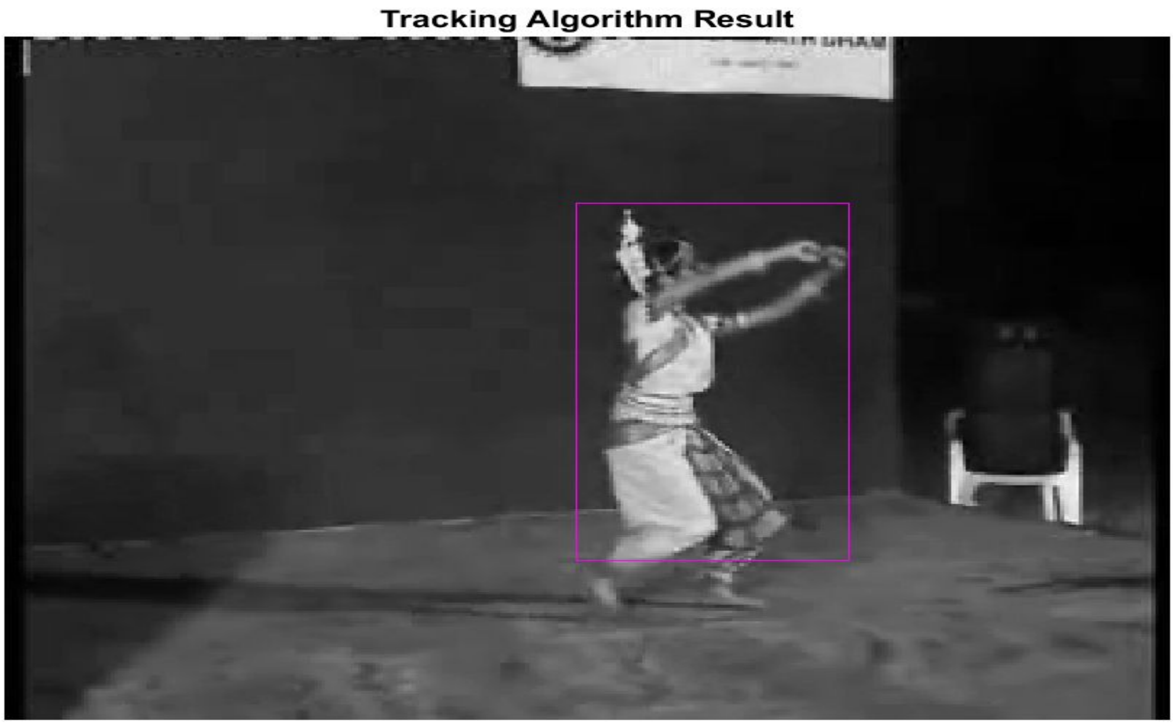
Tracking Algorithm Results for Dancer2 Video :

Tracking Algorithm Result

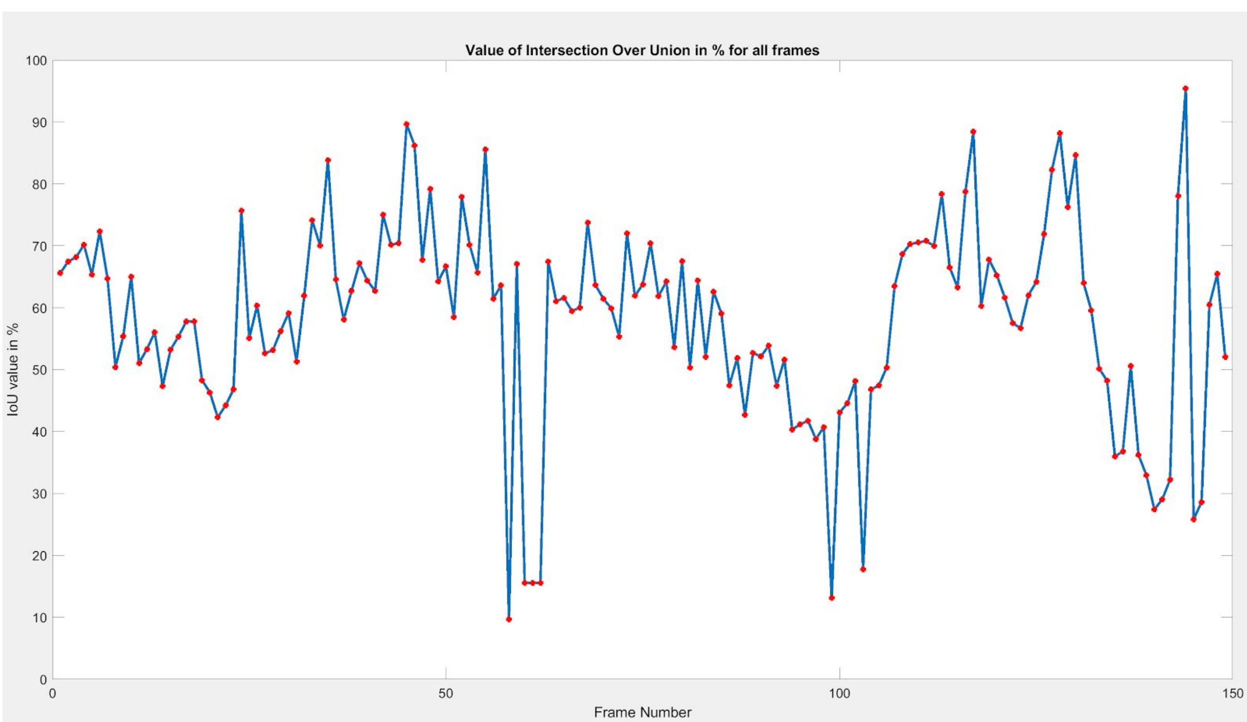


Tracking Algorithm Result



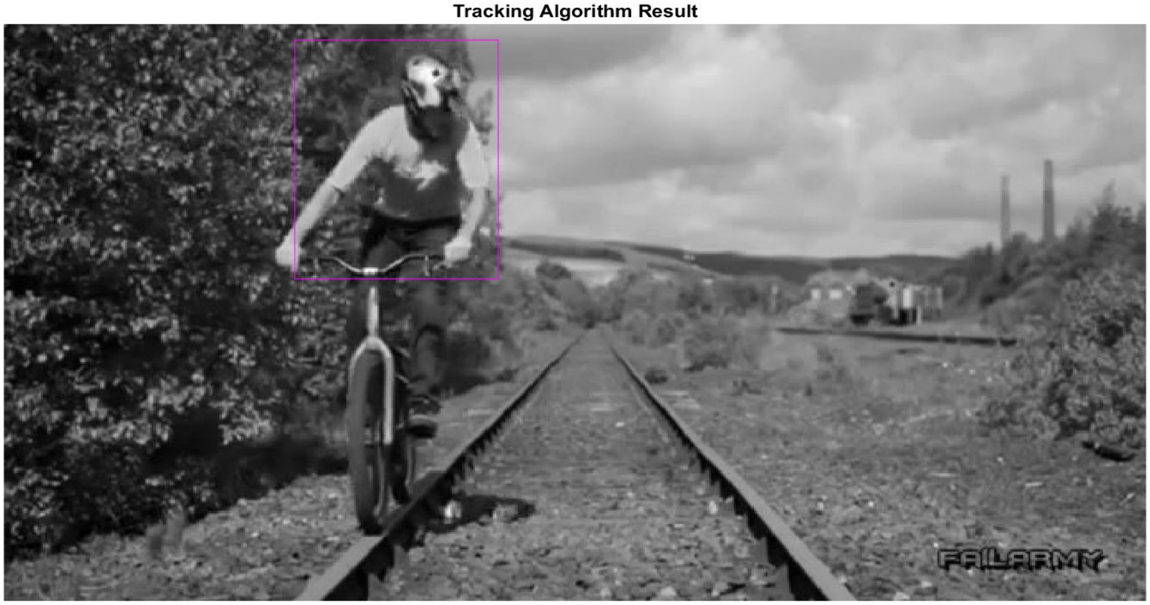


Comparison with Ground Truth : Intersection Over Union Plot for all Frames:



Tracking Algorithm Results for Biker Video : The algorithm remains the same except for converting the frames from RGB to gray and modifying the number of images to be read in the video.

Tracking Algorithm Result

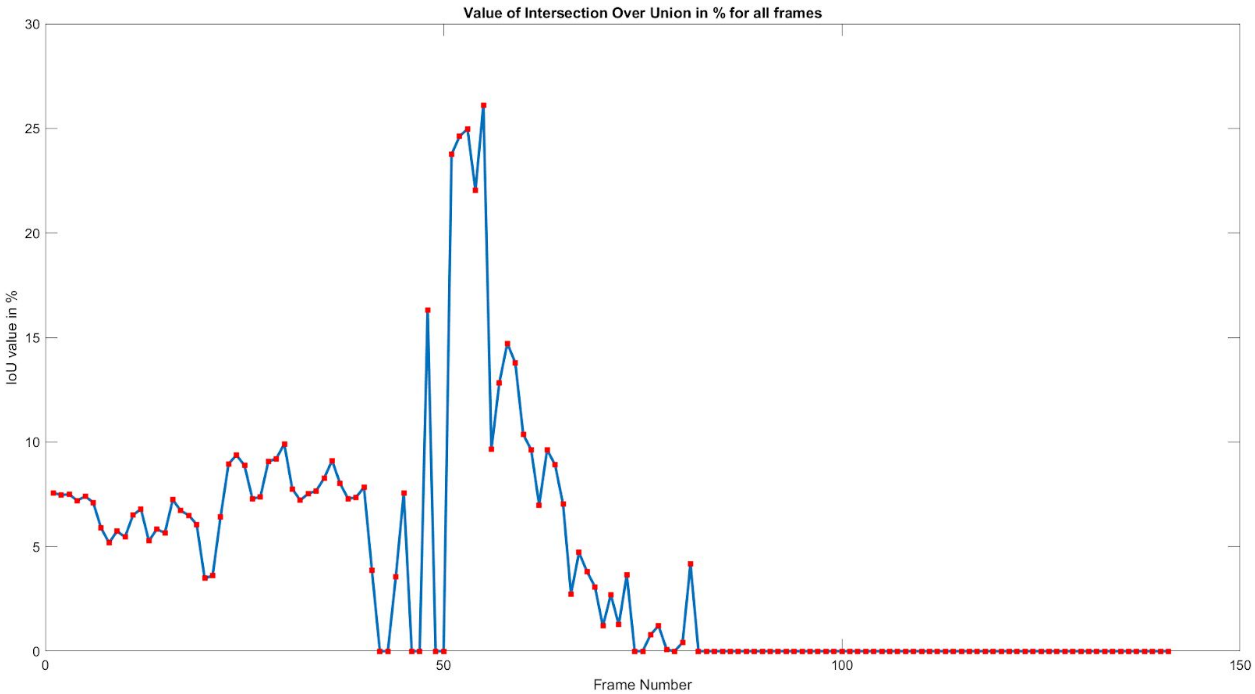


Tracking Algorithm Result

Tracking Algorithm Result

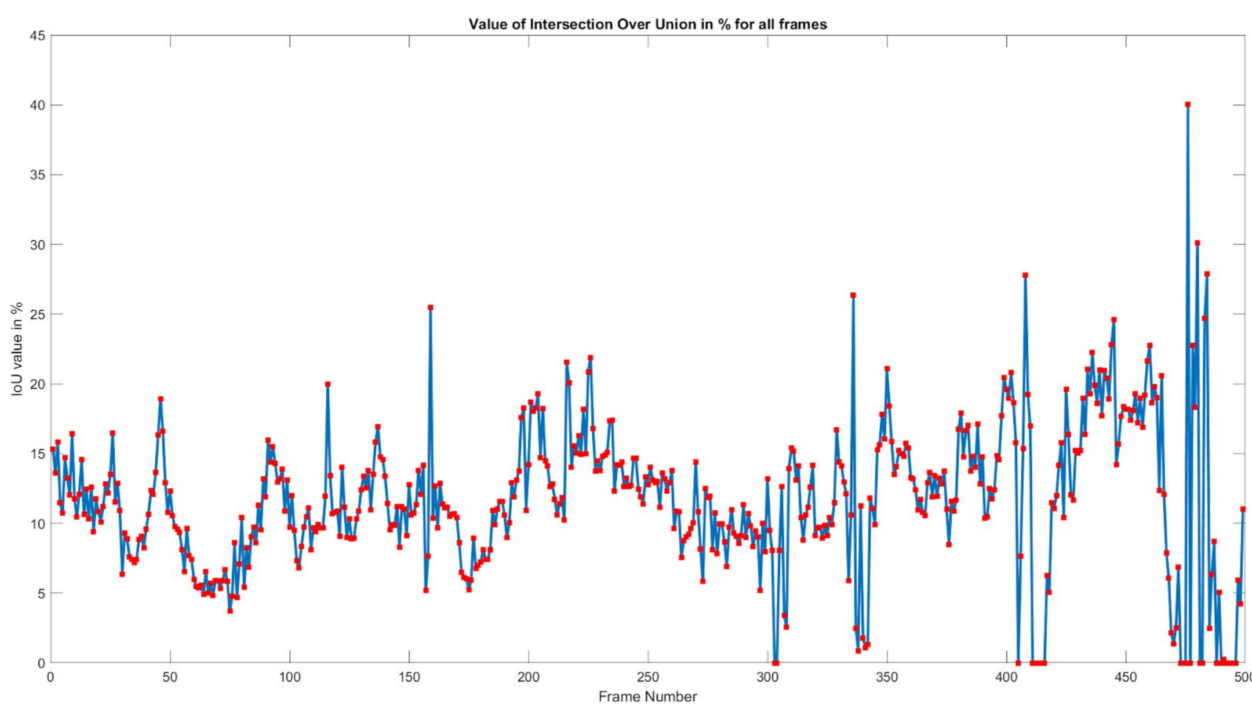
Tracking Algorithm Result

Comparison with Ground Truth : Intersection Over Union Plot for all Frames:



Tracking Algorithm Results for Girl Video :

Comparison with Ground Truth : Intersection Over Union Plot for all Frames:



CITATIONS:

[1]APPLICATION OF NORMALIZED CROSS CORRELATION TO IMAGE REGISTRATION,IJRET: International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308, Y. Raghavender Rao et al.
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.1379&rep=rep1&type=pdf

[2] BACKGROUND SUBTRACTION WITH EIGEN BACKGROUND METHODS, Ilmiyati Sari, Nola Marina
http://ilmiyati.staff.gunadarma.ac.id/Publications/files/2732/EIGENBACKGROUND+SUBSTARACTION+METHOD.pdf

https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/moler/eigs.pdf

APPENDIX A: CODE

Tracking Algorithm for Dancer2 Video :

```
clc
clear all
close all
% Give image directory and extension
imPath = 'C:\Users\Archana Narayanan\Desktop\DIP\Tracking ALgorithm\Template Matching\Dancer2\Dancer2\img'; imExt = 'jpg';
filearray = dir([imPath filesep '*.' imExt]); % get all files in the directory
num_images = size(filearray,1); % get the number of images

% Get image parameters
image_name = [imPath filesep filearray(1).name]; % get image name
I = imread(image_name);
video_w = size(I,2);
video_h = size(I,1);
image_sequence = zeros(video_h, video_w, num_images);

for mean=1:num_images
    image_name = [imPath filesep filearray(mean).name]; % get image name
    image_sequence(:,:,mean) = (imread(image_name)); % load image
end
```

```
%% ---------------- BACKGROUND SUBTRACTION-----------------------------------------------------------------------------
for i = 1:video_h
    for j = 1:video_w
        % Taking median of frame to frame
        I = median(image_sequence(i,j,:));
        background(i,j) = I;     % background Image
    end
end

% Moving object in every frame with threshold
for m = 1:size(image_sequence,3)
    M = image_sequence(:,:,m);                  % Current Frame
    foreground = M- background;
    fore_thresh = foreground>25;
end
%%=================== Eigen background ===========================
% Initialising Variables
N = 30; k = 15; T = 25;
% Reshaping Images
for i = 1:size(image_sequence,3)
   x(:,i) = reshape(image_sequence(:,:,i), [], 1);
end
% The Mean Image
mean = 1/N*sum(x(:,1:N),2);
% Compute mean-normalized image vectors
normalised_mean = x - repmat(mean,1,size(x,2));
% SVD of the matrix X
[U, S, V] = svd( normalised_mean, 'econ');

% Keep the k principal components of U
eigen_background = U(:,1:k);
figure;
```

```
Es = cell(100,1);
for i = 1:size(image_sequence,3)
    img_cons = x(:,i);
    p = eigen_background' * (img_cons - mean);
    Sub_Image = eigen_background * p + mean;

    mask = abs(Sub_Image - img_cons) > T;
    Image_Reshaped = reshape(img_cons .* mask, video_h, video_w);

    %Get the bounding box
    Final_Image = im2bw(Image_Reshaped);
    %% Morphological Operations
    Final_Image = imerode(Final_Image, strel('rectangle', [2 2]));
    Final_Image = imdilate(Final_Image, strel('rectangle', [5 5]));
    bounding_box = regionprops(Final_Image, 'BoundingBox', 'Area');
    area_val = cat(1, bounding_box.Area);
    if(area_val)
        [~,ind] = max(area_val);
        B_box = bounding_box(ind).BoundingBox;
        Es{i} = B_box;
    end

    imshow(image_sequence(:,:,i),[]), title('Tracking Algorithm Result');
    if(area_val)
        hold on;
        rectangle('Position', B_box,'EdgeColor','m');
        hold off;
    end
    drawnow;
end
```

```
fid = fopen('groundtruth_rect.txt', 'rt');
tline = fgetl(fid);
headers = strsplit(tline, ',');     %a cell array of strings
datacell = textscan(fid, '%f%f%f%f', 'Delimiter',',', 'CollectOutput', 1);
fclose(fid);
datavalues = datacell{1};    %as a numeric array

% bounding box values obtained from Es{i}
fid = fopen('dance.txt', 'rt');
tline = fgetl(fid);
headers = strsplit(tline, ',');     %a cell array of strings
datacell_predicted = textscan(fid, '%f%f%f%f', 'Delimiter',',', 'CollectOutput', 1);
fclose(fid);
predicted_values = datacell_predicted{1};    %as a numeric array

%% IoU calculation
Overlap_value = cell(100,1);
for i = 1: 149
    rect1 = datavalues(i,:,:,:);
    rect2 = predicted_values(i,:,:,:);
    intersectionArea = rectint(rect1,rect2);
    unionArea = datavalues(i,3)* predicted_values(i,4) + predicted_values(i,3)* predicted_values(i,4) - intersectionArea;
    overlap = intersectionArea/unionArea * 100;
    Overlap_value{i} = overlap;
    overlapRatio = bboxOverlapRatio(rect1,rect2);
end
plot_value = cell2mat(Overlap_value);
```

```
%% Plot IoU Value
figure();
x_plot = 1:1:149;
plot(x_plot,plot_value,'-s','MarkerSize',3,...
    'LineWidth',2,...
    'MarkerEdgeColor','red',...
    'MarkerFaceColor',[1 .4 .4])
title('Value of Intersection Over Union in % for all frames')
xlabel('Frame Number')
ylabel('IoU value in %')
```

Template Matching:

```
rgbImage = imread('C:\Users\Archana Narayanan\Desktop\DIP\Tracking ALgorithm\Template Matching\Dancer2\Dancer2\img\0001.jpg');
comparison_Image = imread('C:\Users\Archana Narayanan\Desktop\DIP\Tracking ALgorithm\Template Matching\Dancer2\Dancer2\img\0150.jpg');
tMatcher = vision.TemplateMatcher
[rows ,columns, colours] = size(rgbImage);
subplot(2, 2, 1);
imshow(rgbImage, []);
axis on;
title('First Frame', 'FontSize', 15);
set(gcf, 'units','normalized','outerposition',[0, 0, 1, 1]);

%% Template extraction
templateWidth = 40
templateHeight =188
template_I = imcrop(rgbImage, [150.5, 32.5, templateWidth, templateHeight]);
subplot(2, 2, 2);
imshow(template_I, []);
axis on;
title('Template Extracted', 'FontSize', 15);

%% Normalized Cross Correlation
colour_chanel = 1;
correlation_output = normxcorr2(template_I(:,:,1), comparison_Image(:,:, 1));
subplot(2, 2, 3);
imshow(correlation_output, []);
axis on;
title('Normalized Cross Correlation Output', 'FontSize', 15);
[max_corr, index_val] = max(abs(correlation_output(:)));
[y_peak, x_peak] = ind2sub(size(correlation_output),index_val(1))
corr_offset = [(x_peak-size(template_I,2)) (y_peak-size(template_I,1))]

%%   Plot it on the original image.
subplot(2, 2, 4);
imshow(comparison_Image);
```

```
%%   Plot it on the original image.
subplot(2, 2, 4);
imshow(comparison_Image);
ROI = [corr_offset(1) corr_offset(2) templateWidth, templateHeight];
%[location] = tMatcher(rgb2gray(rgbImage),rgb2gray(template_I));
[location] = tMatcher((rgbImage),(template_I));
axis on;
hold on;

%%  Draw the rectangle for the template box
boxRect = [corr_offset(1), corr_offset(2),40,188]
rectangle('position', boxRect, 'edgecolor', 'm', 'linewidth',4);
title('Last Frame', 'FontSize', 15);


figure;
surf(correlation_output), shading flat


rect1 = [corr_offset(1), corr_offset(2),40,188]
rect2 = [166, 63,42,152];
intersectionArea = rectint(rect1,rect2);
unionArea = 40*188 + 42*152 - intersectionArea;
overlap = intersectionArea/unionArea * 100;
```

The above code was modified for different datasets with respect to number of frames and RGB to grayscale conversion of images.

Code for average gaussian function that I tried before Eigenvectors:

```
%------------------------------------------------------------------
% Initializing mean, variance and alpha
mean = image_sequence(:,:,1);
sigma = 1000*ones(video_h,video_w);
alpha = 0.01;
final = zeros(video_h,video_w);
figure;

for m = 1: size(image_sequence,3)
    M = image_sequence(:,:,m);    % Current Frame
    foreground = M - background;

    %Mean
    mean(:,:,m+1) = alpha*image_sequence(:,:,m)+(1-alpha)*mean(:,:,m);
    d = abs(image_sequence(:,:,m)-mean(:,:,m+1));
    %Variance
    sigma(:,:,m+1) = (d.^2)*alpha+(1-alpha)*(sigma(:,:,m));

    final = d>2*sqrt(sigma(:,:,m+1));    % Computed Foreground
    subplot(221),imshow(background,[]), title('Background Image');
    subplot(222),imshow(M,[]), title('Current Image');
    subplot(223),imshow(final,[]), title('Running average gaussian');
    subplot(224),imshow(foreground,[]), title('Object Detection');
    drawnow;
end
```
:

Results :