Computer Vision

CSE 5323.03

Final Project Report

Topic: Object Tracking and Segmentation using Feature Points and Cluster Analysis

Due: November 30, 2015

Raghavender Sahdev

214292742

ACKNOWLEDGEMENT

I would like express my gratitude towards Professor Minas E. Spetsakis for providing me not only with the opportunity of carrying out my proposed project under his supervision, but also providing me with the valuable guidance and support throughout the course of my project. I am also grateful to Mr. Mahdi Biparva for his valuable insights during the project proposal and mid way evaluations.

I express my warm thanks to my friends Omar and Chris for their valuable feedback and insights through the course. Additionally I take this opportunity to the whole class for proving a friendly atmosphere which made the course and the project in general way more fun.

My special thanks go to my parents and brother for always supporting me in my studies in a foreign country.

PROJECT ABSTRACT

Project Title: Tracking and Segmentation using Feature Points and Cluster Analysis

Supervisor: Professor Minas E. Spetsakis, York University, Canada

Teaching Assistant: Mr. Mahdi Biparva

Semester: Fall 2015

Name of Student: Raghavender Sahdev Student No. 214292742

In this project we aim to track a moving object taken from an image sequence. The object could be a book, a human, swimmer, skater, a vehicle or anything that can exhibit motion. Object Tracking has numerous applications in the field of Security and Surveillance, traffic monitoring, medical imaging, Human computer Interaction and many more.

In this project we target at tracking objects in videos taken from a fixed camera and a moving camera. We track single objects and multiple objects in both cases. For multiple object tracking using moving camera, we propose a plausible solution to it too. We here propose an algorithm based on harris feature point, flow computation and cluster analysis. We track objects and then compare our results/implementation with a state of the art existing algorithms - Optical flow, Kalman filter, Mean shift or the KLT tracker.

Contents

1. AIM OF THE PROJECT

In this project we aim to track a moving object in a video in real time. The object could be a book, a human, a vehicle or anything that can exhibit motion. Object Tracking has numerous applications in the field of Security and Surveillance, traffic monitoring, medical imaging, Human computer Interaction and many more.

In this project we target at tracking objects in videos taken from a fixed camera. We here propose an algorithm similar to that proposed by Wong and Spetsakis 2003 [1]. We track objects and then compare our results/implementation with a state of the art existing algorithm from one among the following Optical flow, Kalman filter, Mean shift or the KLT tracker.

In this project we aim to solve the following three cases:

1. Tracking a single object from a stationary camera
2. Tracking multiple objects from a stationary camera
3. Tracking a single object when the camera exhibits motion

A rough extension to case 3 is also proposed to track multiple objects from a moving camera.

2. INTRODUCTION

Object Tracking has been a subject of research for a very long time. Object Tracking can be done in various ways:

- Feature Point Based Object Tracking – In such approaches feature points of the objects are extracted and these points are then tracked to track the object motion. The feature points from an object can be Corners as in the Harris Corner Detector [2], SIFT (Scale Invariant Feature Transform) by Lowe [3], SURF (Speeded Up Robust Features) points [4], HOG descriptors as in the famous paper by Dalal and Triggs[5] or any other valid features which can be tracked in consecutive frames from a video.
- Template Matching based tracking – This kind of tracking generally involves tracking a specific object in a scene whose template is known apriori. Examples of such tracking may include tracking a specific object such as a soccer ball in a game of soccer, tracking a specific object (could be a book, a toy, a specific car or anything else that you can think of) in a scene.
- Color Histogram Based tracking
- Probabilistic / Markov Model Based Tracking - This generally involves using the information from the previous frames and predicting the motion based on a probabilistic model. Algorithms like Kalman Filters and Particle Filters fall under this category.
- Contour Based Tracking

We here focus on Feature Point Based Object Tracking. We here aim to track objects in a video taken from a fixed stationary camera. For the simplicity of the project we initially focus on tracking a single object in a video or an image sequence. Later we propose an algorithm to extend this single object based tracking to track multiple objects in a video.

The latter part of the project focuses on tracking a single object from a camera which exhibits motion. We then extend this approach to track multiple objects from a moving camera. It should be noted the initial aim of this project was just to track a single object from a moving camera, however after implementing the former case, it seemed plausible to extend this approach to the multi object tracking too.

3. ASSUMPTIONS

Object Tracking can be done in numerous ways. We here assume the following to track objects:

- The moving objects should not occlude each other – In our proposed algorithm we assume that the objects being tracked do not occlude each other. In case of an occlusion between 2 moving objects, our algorithm is not able to distinguish between the 2 tracked objects.
- The object should have at least 3 corners – this is a very reasonable assumption as most objects have much more than three corners.
- The object should not be moving very fast – this assumption also is very reasonable for the kind of image sequences we focus on. It is assumed that the motion does not suddenly change a lot.
- It is assumed that the object does not undergo any deformation for tracking using a stationary camera.
- The video is taken from a stationary camera – We also assume that the camera does not move while taking the video, we focus on tracking objects when the camera exhibits motion in the latter part of the project.
- In the case of tracking of single object with a camera that exhibits motion, we assume that the object is not very close to the boundary of the image/frame and that the object remains in the seen.
- In our last case we also assume that the foreground object being tracked moves faster than the background. We additionally assume that the object moves at a different speed than the background.
- The foreground and background have corners.

4. OUR PROPOSED ALGORITHM

We here divide the object tracking problem into tracking image sequences captured using a stationary camera and a camera that exhibits motion. For each case we aim to target single object tracking and multiple object tracking. We present the approaches for the 4 cases to track objects below:

4.1 SINGLE OBJECT TRACKING

We here propose an algorithm wherein we follow the steps:

1. Input the Image sequence / video
2. We then do a background subtraction
3. Detect the corners
4. Match the Detected corners in the next frame
5. Draw a convex hull around the detected corners

6. Repeat the above steps for all frames to compute the tracking and segmentation of the object

Input the Image Sequence / Video

The image sequences and the video being used in this project has been taken from the link – cmp.felk.cvut.cz/~vojirtom/dataset and the existing videos in matlab.

We track use the

Background Subtraction

We take 3 frames frame (i+1), frame I, frame (i-1), We then compute the difference of the $(i+1)^{th}$ frame and the i^{th} frame.

$frame\ (i-1)$ $frame\ i$ $frame\ (i+1)$

We compute the difference

$$D_{i,i-1} = frame\ i - frame\ (i-1),$$
$$D_{i+1,i} = frame\ (i+1) - frame\ i$$

After Doing Background subtraction we get rid of the background and only the moving objects are retained.

Corner Detection

We here use the famous Harris Corner Detector to detect the corners in the frames. The detected corner then act as the feature points which we track in the consecutive frames to track the objects. Here we compute the feature points using an approach similar to Shi-Tomasi / Harris Corner Detector. We compute the matrix, M for each pixel and threshold its minimum Eigen value to compute the corner points. We define the matrix M as:

$$M = \begin{bmatrix} I_x^2 * g & I_{xy} * g \\ I_{xy} * g & I_y^2 * g \end{bmatrix}$$

Here the Gaussian filter $g = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ and $I_x^2 = I_x . I_x$, $I_y^2 = I_y . I_y$ and $I_{xy} = I_x . I_y$ and I_x, I_y are the gradients of the image in x and y directions respectively which have been obtained by convolving the images with the following filters:

$$dx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad and \quad dy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

We then apply non maximal suppression to detect the local maximas to reduce the number of feature/corner points. These corner points serve as the candidate points to be chosen as the feature points in our algorithm.

Matching the Detected Corners in the next frame

After background subtraction we do corner detection to get corners in the images $D_{i,i-1} and\ D_{i+1,i}$. We use the one of the following ways to match the corners.

Method1 – Based on SSD and convex hull

1. Detect corners in the image $D_{i,i-1}$ – it should be noted that the corners will only appear in the moving objects because after background subtraction algorithm the background would be eliminated and only the foreground which is the moving objects in this context remains
2. Compute the convex hull using the corner points of the image $D_{i,i-1}$, this has a different approach in case of multiple object tracking.
3. After computing the convex hull, match the points on the convex hull to find their corresponding points using a simple sum of squared differences method.
4. After getting the matched points in the next frame, plot the convex hull again to get the object in the next frame. Here we only match the objects on the hull.
5. Repeat the above steps for all the frames to track the object in the video.

Method 2 – Based on Corner matching based on Euclidean distance

1. Detect Corners in the images $D_{i,i-1} and\ D_{i+1,i}$ – again corners only appear in the moving objects due to background subtraction.
2. Compute the convex hull of the image $D_{i,i-1}$. For each point that lies on this hull, find its corresponding point/corner from the corners detected in the image $D_{i+1,i}$ based on the minimum Euclidean distance. So a point on the hull $p_i$will have a matching point p_i' such that:

   $$p_i' = p_t for\ which\ the\ distance\ (p_i - p_t)^2\ \ is\ least;\ \ p_t\ \epsilon\ corners\ detected\ in\ D_{i+1,i}$$
3. Repeat step 2 to find all feature points / corners in the image $D_{i+1,i}$.
4. After finding the corresponding points compute the convex hull of these points and that will be the next position of the object in the next frame.
5. Repeat the above steps for all the frames to track the object in the image sequence / video

Convex Hull Generation

We compute the convex hull by using the matlab function convhull. A convex hull is a boundary over a set of points which covers each point.

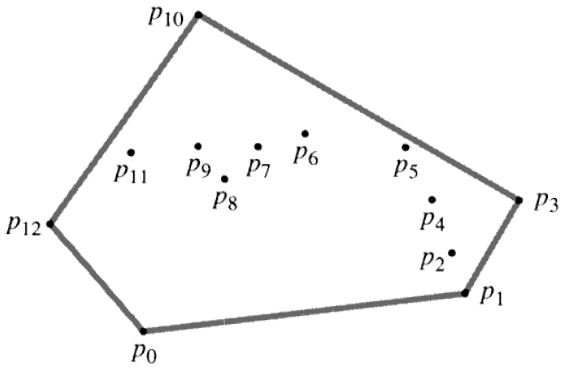

Figure1: A convex hull over a set of points

4.2 MULTIPLE OBJECT TRACKING

We here extend our algorithm to track multiple objects in a scene. Currently the multiple object tracking code is not complete so only a part of it has been implemented. To track multiple objects we use a similar algorithm as we used for tracking a single object. Here it should be noted that our algorithm can not handle occlusion very well. And as of now to track multiple object one must know the number of objects in the scene apriori. We use the following steps to track multiple objects:

1. Input the Image sequence / video
2. Do background subtraction
3. Detect the corners
4. Use K means clustering to cluster the detected corners into k clusters.
5. For each cluster - match the Detected corners in the next frame
6. For each cluster - Draw a convex hull around the detected corners
7. Repeat the above steps for all frames to compute the tracking and segmentation of the object

Input the ImageSequence / Video

Here we use the videos that come with the matlab software – '*atrium.avi'* and '*visiontraffic.avi'.* In each of these videos we have multiple objects moving. We attempt to track them using our proposed algorithm. It is not finished as of now.

Background Subtraction

This step remains the same as that for tracking single object

Detect the corners

This step also remains the same as that for tracking a single object

K means clustering the detected corners into k clusters

After detecting corners in the image we use K means clustering to cluster the detected corners into k clusters. Each of the detected objects will have corners and all corners of a specific object will be close to itself and hence after K means clustering, we will get K clusters corresponding to the K objects in the scene.

Matching /Tracking Feature Points

For each of the clusters we match / track the feature points in the same way as we did for the single object tracking, treating each cluster corresponding to one object.

Convex Hull Generation

We follow the convex hull generation for each cluster in the same way as we did for single object tracking.

4.3 TRACKING A SINGLE OBJECT FROM A MOVING CAMERA

In this section we solve the problem wherein we track a single object from an image sequence taken from a camera which exhibits some motion. We solve this problem by computing the flow between

2 frames and using clustering to cluster the flow to separate out the foreground moving object and the background into 2 clusters.

We propose the following algorithm to track the object in a moving background:

1. Extract the feature points
2. Match / track the feature points
3. Compute the flow using the feature points
4. Generate a feature vector for each point
5. Clustering the features
6. Find out the cluster corresponding to the foreground object
7. Track the foreground cluster
8. Fit a convex hull on the cluster to segment out the object

Extract the Feature Points and compute the Flow

The feature points are extracted in a very similar way as were extracted for the previous 2 cases as in section 4.1 and 4.2. The difference here is that we do not do background subtraction here. We detect the harris corner points in the the ith frame. We call these points as the feature points.

Match / Track the Feature Points, Compute the Flow and Generate the features

We use the feature points computed in the previous section from frame I to match it to the corresponding feature points in frame (i+1). We match the points on a simple Sum of Squared difference (SSD) error measure. We search a window of size $(2w+1) * (2w+1)$ and compute SSDs over the patch of size p. The central pixel of the patch corresponding to the least SSDs is the matched feature point. So say the point P_i in frame i maps to the point P_{i+1}in frame $(i+1)$. We compute the flow of the point P as:

$$\vec{u} = P_i(x, y) - P_{i+1}(x', y')$$

Where $\vec{u}$ is the flow vector and (x', y') is the corresponding point in frame $(i+1)$.

We then use generate the vector

$$V = \begin{bmatrix} \vec{u} \\ P_i \end{bmatrix}$$

So this vector is actually the matrix, S as below:

$$S_i = \begin{bmatrix} u_i \\ v_i \\ P_i\,(x) \\ P_i\,(y) \end{bmatrix}$$

Clustering the Features

From the previous section, we compute the matrix S_i for each of the detected feature points P_i. For each point we call this matrix S_i as the feature vector. We do this for all the detected feature points to come up with features S_1, S_2, S_3 ……S_n where n is the number of feature points. We then use a simple cluster analysis to cluster these features into 2 clusters. We use simple K-means based on the Euclidean distance measure/kernel for clustering the features. The 2 clusters correspond to the background and the foreground.

Find out the cluster corresponding to the foreground

After clustering the features into 2 clusters, each point belongs to either a foreground cluster (object to be tracked) or the background cluster (due to camera motion). We now need to find out the

cluster corresponding to the foreground, we do so by computing the average flow of each cluster. We compute the average flow magnitude over all points in each cluster.

$$Flow1 = \sum_{points\ in\ cluster\ 1} ({u_i}^2 + {v_i}^2)$$

$$Flow2 = \sum_{points\ in\ cluster\ 2} ({u_i}^2 + {v_i}^2)$$

Where $u_i$is the flow in x direction of the feature point i between frame i and $(i+1)$. We label the foreground cluster as the one which has more flow than the other. If $flow\ 1 > flow\ 2$, we say cluster 1 corresponds to the foreground else cluster 2 corresponds to the foreground.

Track the clusters

So assume we have k clusters detected in frame i and k clusters in frame i+1, we map each of the clusters in frame I to the corresponding cluster in frame i+1. It should be noted that this could be made simpler by just considering 2 clusters, however generalizing it for k clusters makes it easy to extend this approach to track not only single objects but also multiple objects from a moving camera. For mapping each cluster to its corresponding cluster, we map the centroids of the clusters in frame I to those in frame i+1. We compute the distance between each of the cluster centroids in frame i and frame i+1.

Fit a convex hull

Here we fit a convex hull to roughly segment out the part in the image corresponding to the object being tracked. We use the convex hull for the points corresponding to the foreground cluster. We use this in exactly the same way as was used in section 4.2

4.4 Tracking multiple objects from a moving camera

Here we extend the approach presented in section 4.4 to track multiple objects from a moving came. We take as input from the user the number of objects in the image sequence as apriori. We then simply set the number of clusters as the number of objects + 1 for the background. It should be noted that this is based on the assumption that the foreground objects do not occlude each other or are in very close proximity with each other and that the flow of the background is different from the foreground objects.

5 OUR RESULTS SO FAR

To show the validation and appropriateness of our proposed algorithm we present our results in this section. We first present results obtained for cases corresponding to the image sequence taken from a stationary camera, later we show the same for the moving camera case.

Single Object Tracking using a fixed camera

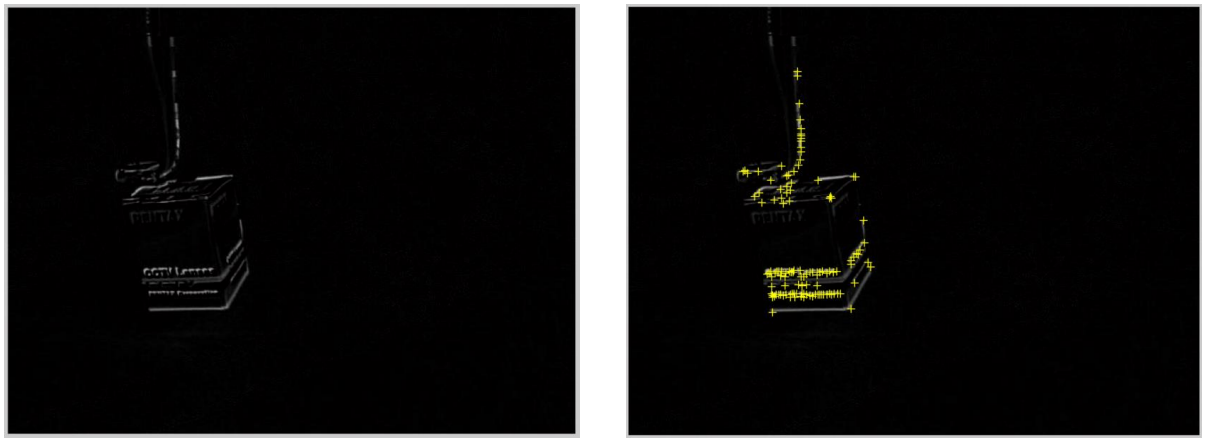

Figure2: Image generated after background subtraction (left); corners/feature points detected in the resulting image

Figure 3: Convex Hull used to segment out the tracked object using method 1 and 2 as descrbed in the previous section

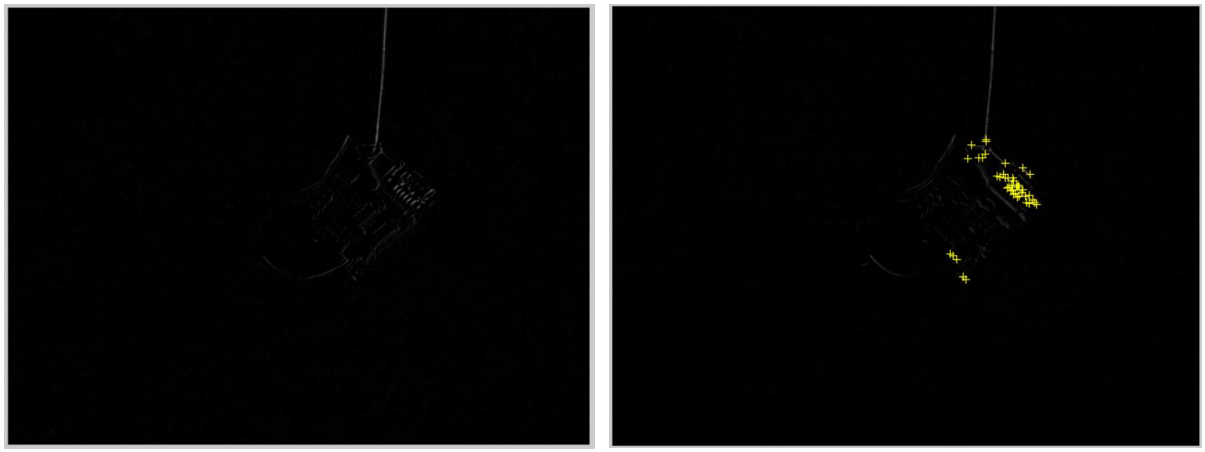

Figure4: Image (electric board) generated after background subtraction (left); corners/feature points detected in the resulting image

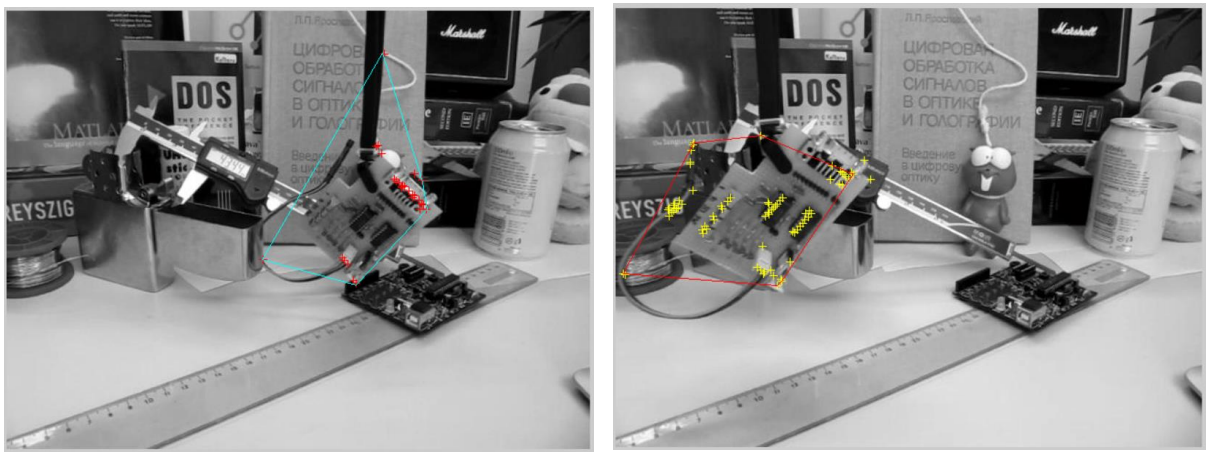

Figure5: Convex Hull used to segment out the tracked object using method 1 and 2 as descrbed in the previous section

Multiple Object Tracking using a fixed camera

Figure 6: Image after background subtraction

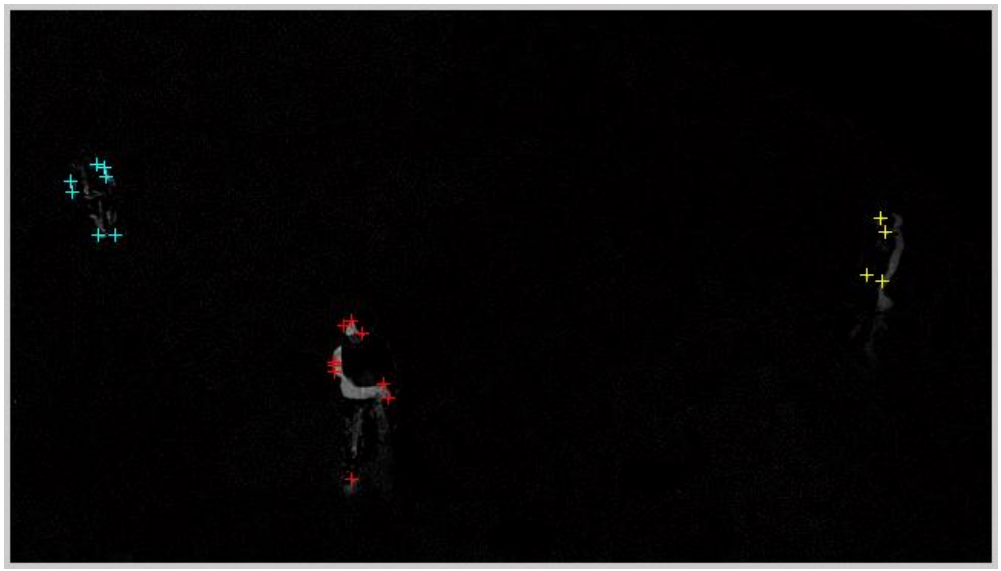

Figure 7: Corner Detector and Clustering of corners in the Background subtracted image

Figure 8: Multiple Object

Single Object tracking using a moving camera

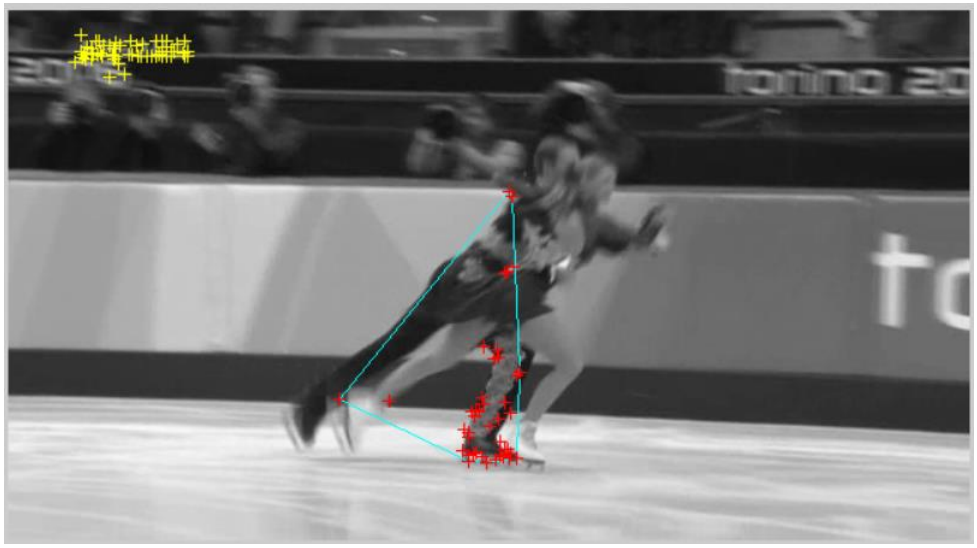

Figure 9: Tracking a couple skiing

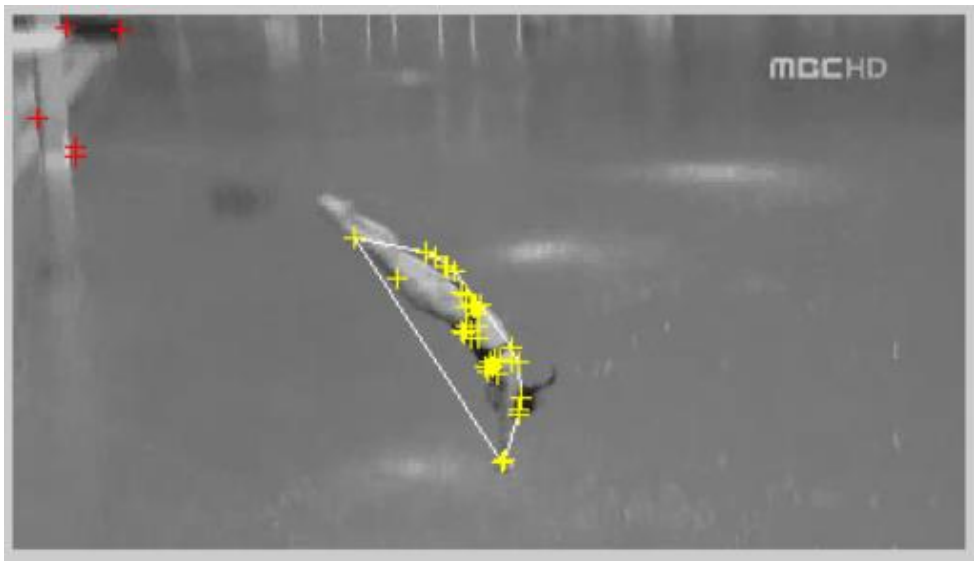

Figure 10: Tracking a swimmer diving

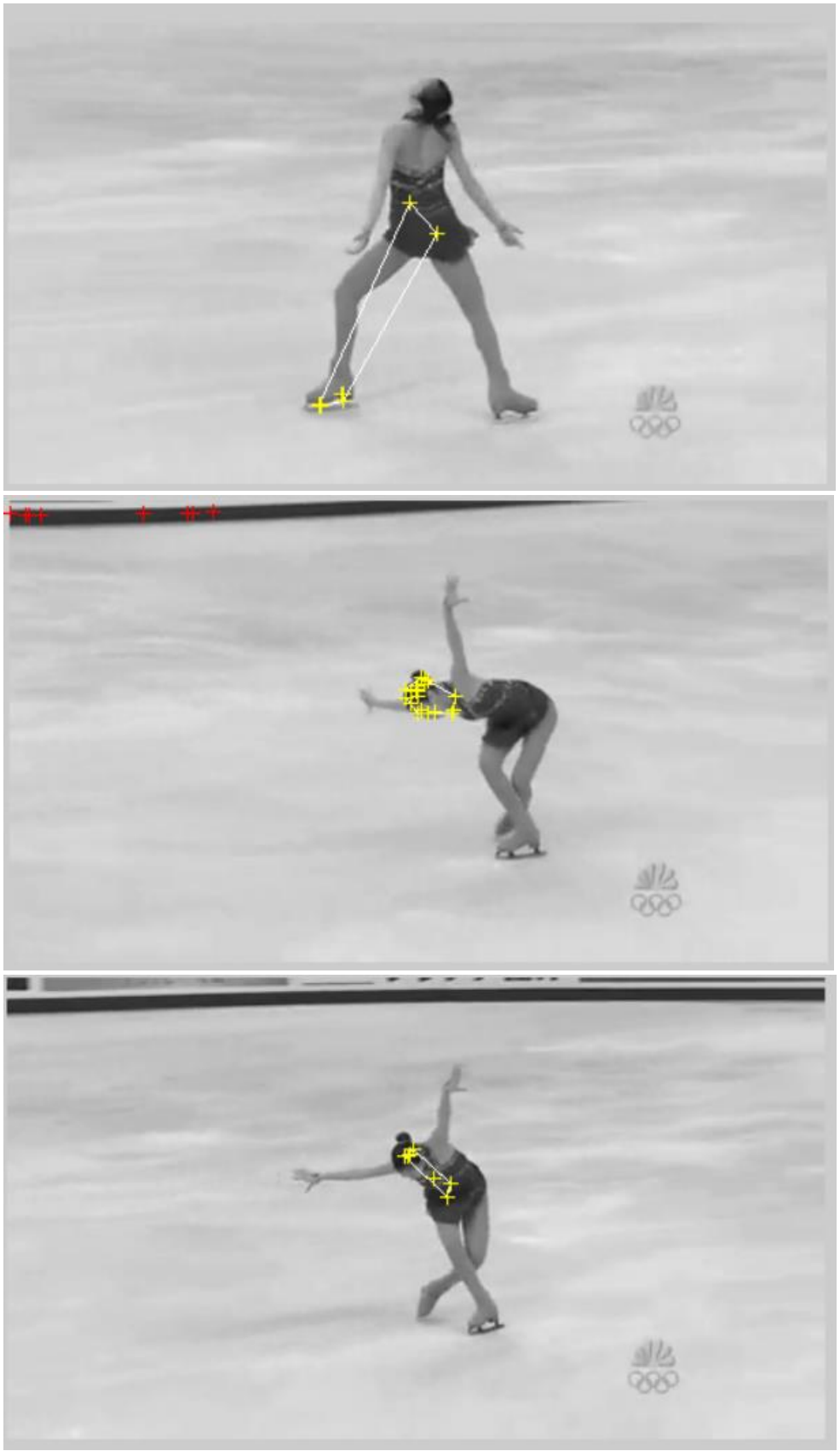

Figure 11: Tracking a skater

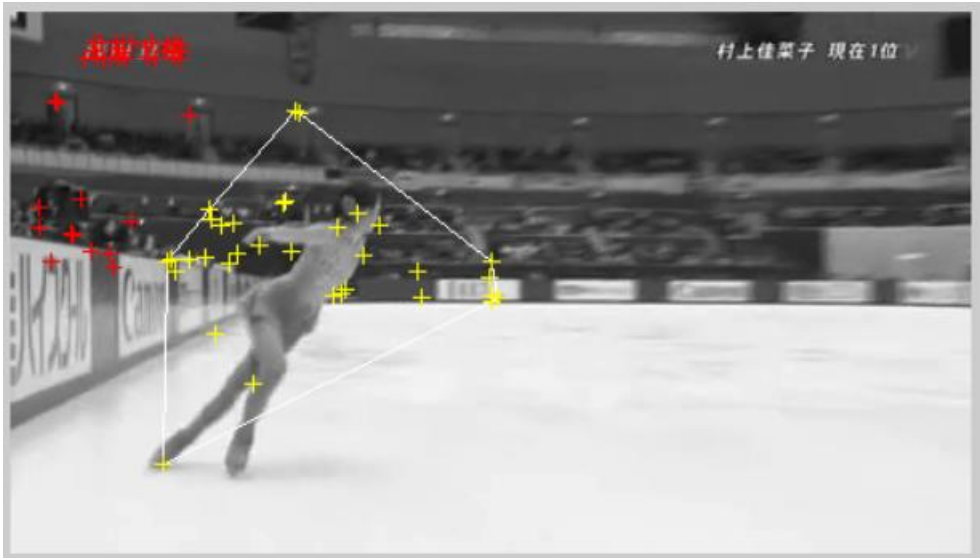

Figure12: Tracking another skater 'Asada'

6. COMPARISON TO A STATE OF THE ART ALGORITHM FOR TRACKING

After implementing the proposed algorithm, our tracking results would be compared with an already existing algorithm (one of these KLT tracker, Optical Flow or Mean shift). We here compare our results with an existing tracking algorithm. Here we present the results by using code taken from the internet and the inbuilt code in matlab which uses a Kalman Filter to track multiple objects in a video. We show below the people being tracked using a Kalman filter and the optical flow of the image. At a later part of the project the KLT Tracker would used to compare the results to get distinct boundaries to segment and track the object.

Using Fixed Camera

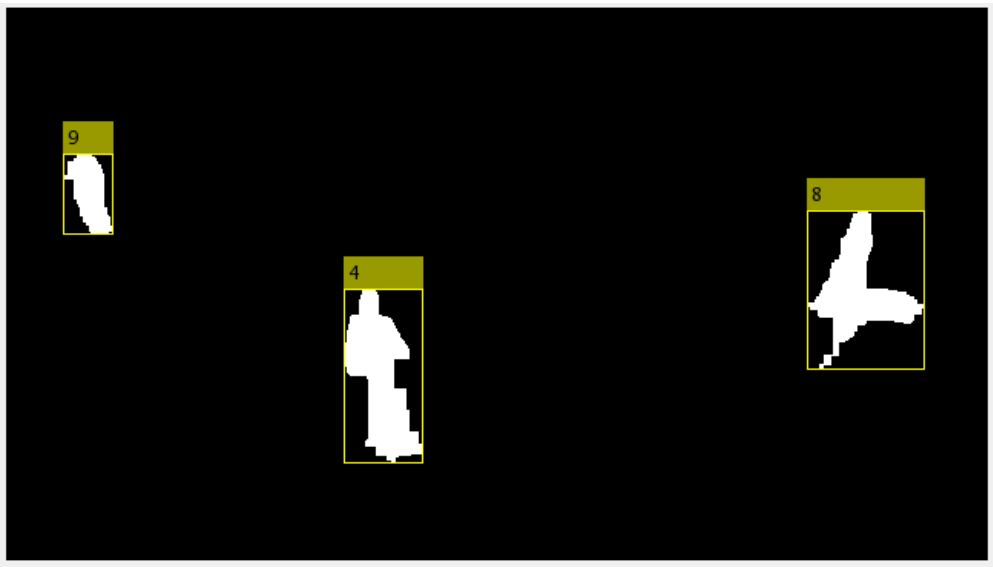

Figure 9: Binary image segmenting out the tracked objects using a Kalman Filter

Figure 10: Kalman Filter being used to track multiple objects in a video

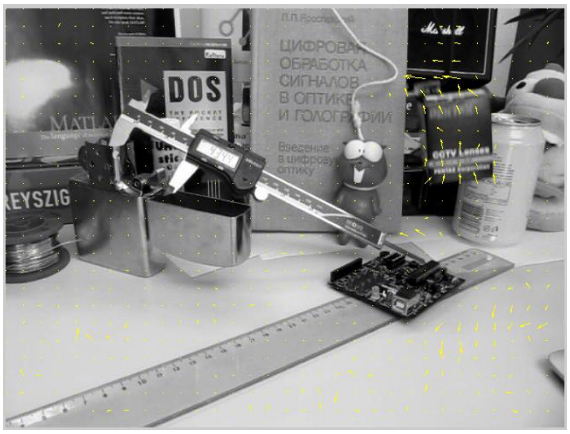

Figure 11: Optical flow computed between 2 frames

Using moving camera

Figure 12: Lucas Kanade Tracker using optical flow to track skaters image sequence from a moving camera

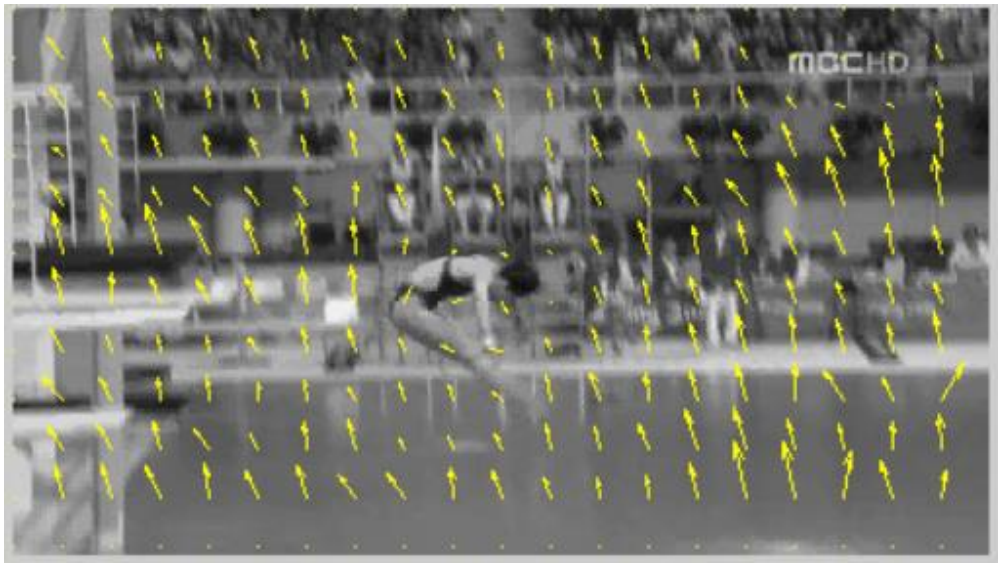

Figure 13: Optical flow of the swimmer clearly segments out the swimmer

7. CONCLUSION SO FAR

We have presented in this report single object tracking and multiple object tracking using a fixed camera and a camera that exhibits motion. We also propose an algorithm to extend the single object tracking to multiple object tracking using moving camera. The code for both the single object and multiple object tracking case has been presented in the Appendix

REFERENCES

1. Tracking, Segmentation and Optical Flow by *King Yuen Wong and Minas E. Spetsakis In proceedings of 16th International Conference on Vision Interface 2003*
2. A Combined Corner and Edge Detector by *Chris Harris & Mike Stephens in proceedings of the Alvey Vision Conference 1988*
3. Distinctive image features from scale-invariant key points by *DG Lowe In proceedings of International journal of Computer Vision 2004*
4. SURF: Speeded Up Robust Features by *Herbert Bay, Tinne Tuytelaars and Luc Van Gool in proceedings of European Conference on Computer Vision 2006*
5. Histogram of Oriented Gradient for Human Detection by *Navneet Dalal and Bill Triggs in Conference on Computer Vision and Pattern Recognition 2005*
6. Motion Segmentation and Tracking by *King Yuen Wong and Minas E. Spetsakis in proceedings of 15th International Conference on Vision Interface 2002*
7. EM Clustering of Incomplete Data Applied to Motion Segmentation by *King Yuen, Lu Ye and Minas E. Spetsakis, In proceedings of British Machine Vision Conference 2004*
8. Determining Optical Flow by *Berthold K.P. Horn and Brian Schunck in proceedings of Artificial Intelligence 1981*

APPENDIX

Tracking objects.m

```
tic
srcFiles =
dir('/home/sahdev/Desktop/Fall2015/ComputerVision5323/project_ideas/Trackin
gDataset/PROST/box/*.jpg');  % the folder in which ur images exists
n = length(srcFiles);
labels = cell(n,1);

parfor i = 1 : n
    filename =
strcat('/home/sahdev/Desktop/Fall2015/ComputerVision5323/project_ideas/Trac
kingDataset/PROST/box/',srcFiles(i).name);
    labels{i} = cellstr(filename);
end
toc
I = imread(char(labels{1}));
sz = size(I);
sz_x = sz(1,1);
sz_y = sz(1,2);

frames = uint8(zeros(sz_x,sz_y,n));
parfor i=1:n
    frame_i = imread(char(labels{i}));
    frame_i = rgb2gray(frame_i);
    frames(:,:,i) = frame_i;

end
toc

%%
strt2 = 2;
end2 = 10
for i=strt2:end2
    temp1 = frames(:,:,i) - frames(:,:,i-1);
    temp1_i = frames(:,:,i);
    temp2_i = frames(:,:,i+1);

%     pts = detectHarrisFeatures(temp1);
    pts = getCorners(temp1);
    y = floor(pts(:,1));  % this is y
    x = floor(pts(:,2));  % this is x
    n2 = numel(x);
    Features1 = zeros(n2,2);
    Features1(:,1) = y;
    Features1(:,2) = x;

    temp2 = frames(:,:,i+1) - frames(:,:,i);
%     pts2 = detectHarrisFeatures(temp2);
    pts2 = getCorners(temp2);
    y2 = floor(pts2(:,1));  % this is y
    x2 = floor(pts2(:,2));  % this is x
    n3 = numel(x2);
```

```
    Features2 = zeros(n3,2);
    Features2(:,1) = y2;
    Features2(:,2) = x2;

%     figure, imshow(frames(:,:,i)), hold on,
plot(y,x,'+','Color','yellow'),hold off
       % sample feature points randomly to track them
    if i == strt2
        P = datasample(Features1,1);

    end
    %% feature matching based on computing minimum euclidean distance
between the detected feaure points
%     distances = pdist2(P,Features2);
%     [dist, matched_index] = min(distances);
%     target_point = Features2(matched_index,:);
%     P = target_point;

    %% build a convex hull from the features detected in the first 2 frames
i and (i-1)
    k = convhull(Features1(:,1),Features1(:,2));
    bounds = zeros(numel(k),2);
    for j=1:numel(k)
        bounds(j,1) = Features1(k(j),1);
        bounds(j,2) = Features1(k(j),2);
    end

    %% feature matching based on computing minimum euclidean distance
between the detected feaure points
    matched_points = zeros(0,2);
    % following for loops matches each of the points on the convex hull
    % generated from frames i and (i-1) and matched it to the feature
    % points generated from frames (i+1) and i, so after execution of this
    % loop we have points matched in from frames i,(i-1) to that of frames
    % (i+1),i;; here we use this approach and not the SSD method to match
    % the corners as the SSD Method would be computationally expensive
    for j=1:numel(k)
        P_temp = [bounds(j,1) bounds(j,2)];

        distances = pdist2(P_temp,Features2);
        [dist, matched_index] = min(distances);
        temp_target_point = Features2(matched_index,:);

        matched_points(j,:) = temp_target_point;
    end


    %% feature matching based on SSD measure
    % here we use another method to match the corners based on SSD we do
    % not detect the corners a second time in this approach, we dont use
    % Features2 here
    P_t1 = getCorners(temp1);
%     P_t2 = getCorners(temp2);
    for j=1:numel(P_t1)/2
        P_Matched(j,:) = matchCorners(temp1_i,temp2_i,P_t1(j,:));
    end

    k2 = convhull(P_Matched(:,1),P_Matched(:,2));
    bounds2 = zeros(numel(k2),2);
```

```
    for j=1:numel(k2)
        bounds2(j,1) = P_Matched(k2(j),1);
        bounds2(j,2) = P_Matched(k2(j),2);
    end
    figure, imshow(frames(:,:,i+1)), hold on, plot(y,x,'+','Color','red'),
plot(bounds2(:,2),bounds2(:,1),'Color','cyan'),hold off

    figure, imshow(frames(:,:,i+1)), hold on,
plot(y2,x2,'+','Color','yellow'),
plot(matched_points(:,1),matched_points(:,2),'Color','red'),hold off


%     figure, imshow(frames(:,:,i+1)), hold on,
plot(target_point(1,1),target_point(1,2),'+','Color','red'),hold off
%     figure, imshow(temp2);% hold on, plot(y,x,'+','Color','yellow'),hold
off
%

end
```

matchCorners.m

```
function P = matchCorners(I,I2,Point_p)
      sz = size(I);
    size_x = sz(1,1);
    size_y = sz(1,2);
    w=2;
    p=3;
    x2 = 0;
    y2 = 0;
    SSD = 0;
    for i=-w:w
        for j=-w:w
            ssd = 0;
            temp1 = Point_p(1,2); % gets the x coordinate
            temp2 = Point_p(1,1); % gets the y coordinate
            px_corr = 15;
            % handling points lying close to the border of the image
            if(temp1 <=px_corr || temp2<=px_corr || temp1>=size_x-px_corr
            || temp2>=size_y-px_corr)
                SSD(i+w+1,j+w+1) = 9999;
                break
            end             mask1 = I(temp1-p:temp1+p,temp2-p:temp2+p);
            mask2 = I2(temp1-p+i:temp1+p+i,temp2-p+j:temp2+p+j);
            dif_mask = mask2-mask1;
            dif_mask = dif_mask.^2;
            SSD(i+w+1,j+w+1) = sum(sum(dif_mask));
        end
    end
    [t1 it1] = min(SSD);
    [t2 it2] = min(t1);
    [t3 it3] = min(SSD,[],2);
    [t4 it4] = min(t3);
    x2 = Point_p(1,2) + it4 -(w+1);
    y2 = Point_p(1,1) + it2 -(w+1);

%     figure, imshow(I);
%     imshow(I2),hold on, plot(y2,x2,'+','Color','red'),hold off
    P = [x2 y2];
end
```

getCorners.m

```
function P = getCorners(I)
    size_xy = size(I);
    size_x = size_xy(1,1);
    size_y = size_xy(1,2);
    % I = imresize(I,[resize_x resize_y]);


    %% create a filter to compute the gradient of the image in x and y
direction
    dx = [-1 0 1 ; -1 0 1 ; -1 0 1];
    dy = [-1 -1 -1 ; 0 0 0 ; 1 1 1];
    Ix = conv2(I,dx,'same');
    Iy = conv2(I,dy,'same');


    %% create the gaussian filter
    hsize = 3;
    sigma = 0.5;
    gauss = fspecial('gaussian', hsize, sigma);


    %% convolving the square of the first derivative with the gaussian
    Ix2 = conv2(Ix.^2, gauss,'same');
    Iy2 = conv2(Iy.^2, gauss,'same');
    Ixy = conv2(Ix.*Iy, gauss,'same');


    %% computing Harris Features

    det_M = Ix2.*Iy2 - Ixy.^2;
    trace_M = (Ix2 + Iy2);
    R = det_M ./trace_M;


    % citation Brown, Szeliski, andWinder (2005) paper use the harmonic
mean: source Richard Szeliski book chapter 4
    %% do non maximal supression on R and store the computed coordinates as
the corners
    hsize = 3;
    threshold = 2000;
    Filtered = ordfilt2(R,hsize^2,ones(hsize)); % dilate to retain the
local maxima
    R = R>threshold & (R==Filtered) ;
    % something really cool happens here we check if after dilation the
values are equal to
    %the original values and only retain the values which are equal, this
retains the local maxima
    [x,y] = find(R);
%     figure,imshow(I), hold on, plot(y,x,'+');
    hold off
    P = [y x];
end
```

MultipleObjectTracking.m

```
tic
video = VideoReader('atrium.avi');
```

```
% video = VideoReader('visiontraffic.avi');
nFrames = video.NumberOfFrames;
vidHeight = video.Height;
vidWidth = video.Width;

mov(1:nFrames) = struct('cdata', zeros(vidHeight,vidWidth, 3,
'uint8'),'colormap',[]);
parfor k = 1 : nFrames
    mov(k).cdata = read(video,k);
    mov(k).cdata = rgb2gray(mov(k).cdata);
end
toc

temp2 =3;
%%
P1=6;
strt2 = 400;
end_it = 402;
occlusion_threshold = 90;
moving_objects = 3;
for i=strt2:end_it
    temp1 = mov(i).cdata - mov(i-1).cdata;


    %% FEATURE 1 using (i) and (i-1)
    pts = detectHarrisFeatures(temp1);
    y = floor(pts.Location(:,1));  % this is y
    x = floor(pts.Location(:,2));  % this is x
    n = numel(x);
    Features1 = zeros(n,2);
    Features1(:,1) = y;
    Features1(:,2) = x;

    %% FEATURE 1 using (i+1) and (i)
    temp2 = mov(i+1).cdata - mov(i).cdata;
    pts2 = detectHarrisFeatures(temp2);
    y2 = floor(pts2.Location(:,1));  % this is y
    x2 = floor(pts2.Location(:,2));  % this is x
    n2 = numel(x2);
    Features2 = zeros(n2,2);
    Features2(:,1) = y2;
    Features2(:,2) = x2;

    %% do clustering to track individual objects
     if i == strt2
        [idx,Centroid] = kmeans(Features1,moving_objects);
        cnt=1;
    %     Group1,Group2,Group3,g1,g2,g3
        g1=1;g2=1;g3=1;
        Group1 =zeros(0,2);
        Group2 =zeros(0,2);
        Group3 =zeros(0,2);

        for j=1:numel(Features1)/2
            if(idx(j,1) == 1)
                Group1(g1,1) = Features1(j,1);
                Group1(g1,2) = Features1(j,2);
                g1 = g1+1;
            elseif(idx(j,1) == 2)
                Group2(g2,1) = Features1(j,1);
```

```
                Group2(g2,2) = Features1(j,2);
                g2 = g2+1;
            elseif(idx(j,1) == 3)
                Group3(g3,1) = Features1(j,1);
                Group3(g3,2) = Features1(j,2);
                g3 = g3+1;
            end
        end
    end
%    figure, imshow(mov(i).cdata), hold on,
plot(y,x,'+','Color','yellow'),hold off


    %% tracking clusters
       % sample feature points randomly to track them
    if i == strt2
        while(1)
            P1 = datasample(Group1(:,:),1);
            if (P1(1,1) ==0 && P1(1,2) ==0)
                continue;
            else
                break;

            end
        end
        while(1)
            P2 = datasample(Group2(:,:),1);
            if (P2(1,1) ==0 && P2(1,2) ==0)
                continue;
            else
                break;

            end
        end
        while(1)
            P3 = datasample(Group3(:,:),1);
            if (P3(1,1) ==0 && P3(1,2) ==0)
                continue;
            else
                break;

            end
        end
    end
    % till here we have 3 sampled points from each of the 3 clusters
    distances = pdist2(P1,Features2);
    [dist, matched_index] = min(distances);
    target_point1 = Features2(matched_index,:);
    P1 = target_point1;

    distances = pdist2(P2,Features2);
    [dist, matched_index] = min(distances);
    target_point2 = Features2(matched_index,:);
    P2 = target_point2;

    distances = pdist2(P3,Features2);
    [dist, matched_index] = min(distances);
    target_point3 = Features2(matched_index,:);
    P3 = target_point3;
```

```
    if(i == strt2)

        k1 = findCluster(target_point1,Centroid);
        k2 = findCluster(target_point2,Centroid);
        k3 = findCluster(target_point3,Centroid);

    end
%     figure, imshow(mov(i+1).cdata), hold on,
plot(target_point1(1,1),target_point1(1,2),'+','Color','red'),hold off
%     figure, imshow(mov(i+1).cdata), hold on,
plot(target_point2(1,1),target_point2(1,2),'+','Color','yellow'),hold off
%     figure, imshow(mov(i+1).cdata), hold on,
plot(target_point3(1,1),target_point3(1,2),'+','Color','cyan'),hold off



%     colors2(k11) =

    %% cluster 1

    kover = convhull(Group1(:,1),Group1(:,2));
    bounds1 = zeros(numel(kover),2);
    for j=1:numel(kover)
        bounds1(j,1) = Group1(kover(j),1);
        bounds1(j,2) = Group1(kover(j),2);
    end
    figure, imshow(mov(i+1).cdata), hold on,
plot(bounds1(:,1),bounds1(:,2),'Color','red'),hold off

    %% cluster 2
    kover = convhull(Group2(:,1),Group2(:,2));
    bounds2 = zeros(numel(kover),2);
    for j=1:numel(kover)
        bounds2(j,1) = Group2(kover(j),1);
        bounds2(j,2) = Group2(kover(j),2);
    end
    figure, imshow(mov(i+1).cdata), hold on,
plot(bounds2(:,1),bounds2(:,2),'Color','cyan'),hold off

    %% cluster 3
    kover = convhull(Group3(:,1),Group3(:,2));
    bounds3 = zeros(numel(kover),2);
    for j=1:numel(kover)
        bounds3(j,1) = Group3(kover(j),1);
        bounds3(j,2) = Group3(kover(j),2);
    end
    figure, imshow(mov(i+1).cdata), hold on,
plot(bounds3(:,1),bounds3(:,2),'Color','yellow'),hold off
end
toc
```

trackingCamera motion.m

```
tic

max_pixel_flow = 4; % this is w, this means actually a movement of
2*max_pixel_flow +1 for the object to be tracked;
```

```
patch_match_size = 3; % this is p, the size of the patch would be
patch_match_size x patch_match_size
boundary_effect = 15;
threshold = 1800; % this is the threshold for the harris corner detector

srcFiles = dir('C:\Users\Raghavender Sahdev\Desktop\York
University\Computer Vision 5323\project\vision\skatings2\skating2\*.jpg');
% the folder in which ur images exists
n = length(srcFiles);
labels = cell(n,1);

parfor i = 1 : n
    filename = strcat('C:\Users\Raghavender Sahdev\Desktop\York
University\Computer Vision
5323\project\vision\skatings2\skating2\',srcFiles(i).name);
    labels{i} = cellstr(filename);
end
toc
I = imread(char(labels{1}));
sz = size(I);
sz_x = sz(1,1);
sz_y = sz(1,2);

boundary_effect = floor(sz_x/30);
frames = uint8(zeros(sz_x,sz_y,n));
parfor i=1:n
    frame_i = imread(char(labels{i}));
    t_s = size(frame_i,3);
    if(t_s == 3)
        frame_i = rgb2gray(frame_i);
    end
    frames(:,:,i) = frame_i;

end
toc

%% set the frame sequence numbers to track
strt2 = 45;
end2 = 60;
% set the number of clusters to track multiple objects set this number to
% the total number of obhects in the scene - 1.
clusters = 2;

Centroid = zeros(2,clusters);
temp_pts = zeros(1,2);
for i=strt2:end2
    temp1 = frames(:,:,i);
    temp2 = frames(:,:,i+1);

    if i==strt2
        pts = getFeatures(temp1,threshold);
%         temp_pts = pts;
        znt=1;
        for j=1:numel(pts)/2
            if(pts(j,1) < boundary_effect || pts(j,1) > (sz_x-
boundary_effect))
                ;
            elseif (pts(j,2) < boundary_effect || pts(j,2) > (sz_y-
boundary_effect))
                ;
```

```
            else
                temp_pts(znt,1) = pts(j,1);
                temp_pts(znt,2) = pts(j,2);
                znt = znt + 1;
            end
        end
    end

    n_corners = numel(temp_pts)/2;
    matched_Points = zeros(n_corners,2);
    for j=1:n_corners
        matched_Points(j,:) =
matchFeatures(temp1,temp2,temp_pts(j,:),max_pixel_flow,patch_match_size,bou
ndary_effect);
    end

    flow_compute = matched_Points - temp_pts;

    %% logic for the actual corner updates
    temp2_pts = matched_Points;
    temp_corners = getFeatures(temp2,threshold);

    actual_index=1;

    actual_corners = zeros(1,2);
    znt2=1;
    for j=1:numel(temp_corners)/2
        if(temp_corners(j,1) < boundary_effect || temp_corners(j,1) >
(sz_x-boundary_effect))
            ;
        elseif (temp_corners(j,2) < boundary_effect || temp_corners(j,2) >
(sz_y-boundary_effect))
            ;
        else
            actual_corners(znt2,1) = temp_corners(j,1);
            actual_corners(znt2,2) = temp_corners(j,2);
            znt2 = znt2 + 1;
        end
    end

    %% this part tracks based on corner matching
    n_corners2 = numel(actual_corners)/2;
%     for j=1:n_corners
%         min=100;
%         for k=1:n_corners2
%             dist = (temp2_pts(j,1)-actual_corners(k,1))^2 +
(temp2_pts(j,1)-actual_corners(k,2))^2;
%             if(dist < min)
%                 actual_index = k;
%             end
%         end
%         matched_Points(j,1) = actual_corners(actual_index,1);
%         matched_Points(j,2) = actual_corners(actual_index,2);
%     end

    %% cluster analysis begins
    temp_pts = matched_Points;
    mag = flow_compute(:,1).^2 + flow_compute(:,2).^2;

    flow2 = [flow_compute, matched_Points];
```

```
      if(i == strt2)
         [idx,Centroid] = kmeans(flow2,clusters);
      end

      [idx2,Centroid2] = kmeans(flow2,clusters);
      temp_Centroid2 = Centroid2;
      temp_idx2 = idx2;

      % following code maps clusters to their respective initial clusters
      % this part basically tracks the clusters

      for j=1:clusters
          min=9999;
          for k=1:clusters
              dist = (Centroid(j,3)-Centroid2(k,3))^2 + (Centroid(j,4)-
Centroid2(k,4))^2 ;
              k;
              if(dist < min)
                  min = dist;
                  C_k = k;
              end
          end

          j;
          C_k;
%            disp('NEXT')
          for k_c=1:n_corners
              if(idx2(k_c) == C_k)
                  temp_idx2(k_c) = j;
              end
          end
      end

     cnt=1;bnt=1;ant=1;ent =1; fnt=1;gnt=1;hnt=1;

     Features1 = zeros(1,2);
     Features2 = zeros(1,2);
     Features3 = zeros(1,2);
     Features4 = zeros(1,2);
     Features5 = zeros(1,2);
     Features6 = zeros(1,2);

     cluster1 = 0;
     cluster2 = 0;
    for j=1:n_corners
         if(temp_idx2(j) == 1)
             Features1(cnt,:) = matched_Points(j,:);
             cnt = cnt+1;
             cluster1 = cluster1 + (matched_Points(j,1)-temp_pts(j,1))^2 +
(matched_Points(j,2)-temp_pts(j,2))^2;
         elseif(temp_idx2(j) == 2)
             Features2(bnt,:) = matched_Points(j,:);
             bnt = bnt+1;
             cluster2= cluster2 + (matched_Points(j,1)-temp_pts(j,1))^2 +
(matched_Points(j,2)-temp_pts(j,2))^2;
         elseif(temp_idx2(j) == 3)
             Features3(ant,:) = matched_Points(j,:);
             ant = ant+1;
         elseif(temp_idx2(j) == 4)
```

```
            Features4(ent,:) = matched_Points(j,:);
            ent = ent+1;
        elseif(temp_idx2(j) == 5)
            Features5(fnt,:) = matched_Points(j,:);
            fnt = fnt+1;
        elseif(temp_idx2(j) == 6)
            Features6(ent,:) = matched_Points(j,:);
            gnt = gnt+1;
        end

    end
    cluster1=cluster1/cnt;
    cluster2=cluster2/bnt;

    %% plotting the convex hull
    bounds=zeros(1,2);
    bounds2=zeros(1,2);
    if (strt2 == i)
        if(cluster1 > cluster2)
            flag=1;
        else
            flag=0;
        end
    end
    if(flag == 1)
        if(size(Features1,1)>2)
            k1 = convhull(Features1(:,1),Features1(:,2));
            bounds = zeros(numel(k1),2);
            for j2=1:numel(k1)
                bounds(j2,1) = Features1(k1(j2),1);
                bounds(j2,2) = Features1(k1(j2),2);
            end
        end
    else
        if(size(Features2,1)>2)
             k2 = convhull(Features2(:,1),Features2(:,2));
            bounds = zeros(numel(k2),2);
            for j2=1:numel(k2)
                bounds2(j2,1) = Features2(k2(j2),1);
                bounds2(j2,2) = Features2(k2(j2),2);
            end
        end
    end
    figure, imshow(temp1), hold on,
    plot(bounds(:,1),bounds(:,2),'Color','cyan')
    plot(bounds2(:,1),bounds2(:,2),'Color','cyan')
    plot(Features1(:,1),Features1(:,2),'+','Color','yellow'),
    plot(Features2(:,1),Features2(:,2),'+','Color','red'),
%     plot(actual_corners(:,1),actual_corners(:,2),'+','Color','cyan'),
    plot(Features3(:,1),Features3(:,2),'+','Color','blue'),
    plot(Features4(:,1),Features4(:,2),'+','Color','green'),
    plot(Features5(:,1),Features5(:,2),'+','Color','cyan'),
    plot(Features6(:,1),Features6(:,2),'+','Color','white'),
    hold off;
end
toc
```

LucasKanadeTracker.m

```
%% input the 2 frames on which you want to compute the flow
fr1 = frames(:,:,9);
fr2 = frames(:,:,10);
figure();
subplot 211
imshow(fr1);
im1t = im2double((fr1));
im1 = imresize(im1t, 0.5); % downsize to half
subplot 212
imshow(fr2);
im2t = im2double((fr2));
im2 = imresize(im2t, 0.5); % downsize to half
%% Implementing Lucas Kanade Method
ww = 11;
w = round(ww/2);
% Lucas Kanade Here
% for each point, calculate I_x, I_y, I_t
Ix_m = conv2(im1,[-1 1; -1 1], 'valid'); % partial on x
Iy_m = conv2(im1, [-1 -1; 1 1], 'valid'); % partial on y
It_m = conv2(im1, ones(2), 'valid') + conv2(im2, -ones(2), 'valid'); %
partial on t
u = zeros(size(im1));
v = zeros(size(im2));
% within window ww * ww
for i = w+1:size(Ix_m,1)-w
   for j = w+1:size(Ix_m,2)-w
      Ix = Ix_m(i-w:i+w, j-w:j+w);
      Iy = Iy_m(i-w:i+w, j-w:j+w);
      It = It_m(i-w:i+w, j-w:j+w);

      Ix = Ix(:);
      Iy = Iy(:);
      b = -It(:); % get b here

      A = [Ix Iy]; % get A here
      nu = pinv(A)*b; % get velocity here

      u(i,j)=nu(1);
      v(i,j)=nu(2);
   end;
end;
% downsize u and v
u_deci = u(1:10:end, 1:10:end);
v_deci = v(1:10:end, 1:10:end);
% get coordinate for u and v in the original frame
[m, n] = size(im1t);
[X,Y] = meshgrid(1:n, 1:m);
X_deci = X(1:20:end, 1:20:end);
Y_deci = Y(1:20:end, 1:20:end);
%% Plot optical flow field
figure();
imshow(fr2);
hold on;
% draw the velocity vectors
 quiver(X_deci, Y_deci, u_deci,v_deci, 'y')
hold off
```