VIRTUAL ADVERTISING

A Camera Calibration and Perspective Projection Technique

MAH Abdul Cader Hasanain, m.a.h.abdulcaderhasanain@student.utwente.nl, 1996134, EMSYS
Darshan Srirangachar Ramesh, d.srirangacharramesh@student.utwente.nl, s1998978, EMSYS

***Abstract*— In today's digital era, the most preferred ways for advertisement is to virtually project the images in a live telecast or in a pre-recorded event. Virtual advertisement is an innovative digital technology which enables the advertisers to promote their brand dynamically in an ongoing event in an impactful way. This way, the advertisements creates an impression on the viewer that the he/she is seeing the ad on the screen is real. This technique is majorly employed during sports telecasts like football, volley ball, tennis etc. and in news broadcasts. Virtual advertising needs real time video processing as it requires the mixing of two signal, the actual video and the advertisement. To do this, one such approach is discussed in this paper using MATLAB image processing and computer vision toolbox. Techniques and algorithms used to project the ad virtually on an ongoing tennis match are discussed.**

***Keywords*— Computer Vision, Image Processing, MATLAB, Virtual advertisement.**

I. INTRODUCTION

In the recent years, the number of devices getting connected to the broadband is rising rapidly. Thus, a lot of data, either in the form of video or images, termed as multimedia content, are continuously being pushed to the web. As a result, advancements in the field of image and video processing technology are rapidly growing. This rapid development of computer video technology has given birth to a new kind of instrument, virtual advertisement insertion. Conventional advertisements involve in interrupting a program for a short while to play out a video recording of one or more advertisements. This often adds to the length of the program and causes viewers to lose out on some precious minutes of live action. Consequently, embedding virtual ads in videos during such live events provides an effective & innovative solution. [1]

Virtual advertisement system, the main application of this technology, are providing opportunities to improve the commercial value of television programming while enhancing the contents and the entertainment aspect of these programs [2]. The system can create enormous economic benefit and be widely used in sports broadcasting, particularly live sporting events. It inserts ads seamlessly into the video without interrupting programs or disrupting the viewer's attention. Consequently, the true cost of advertising is not only reduced, but also the quality and the value of the program are increased-the television program becomes the central focus and the sponsors' ads are delivered to audience effectively. [3]

Advancements in image processing, computer vision, and computer graphics have made real time application of virtual or augmented reality (VR/AR) in sports programming feasible. A typical virtual advertisement insertion system consists of three subsystems, i.e., camera tracking subsystem, real-time virtual ad generation subsystem and video composition subsystem. [4]

This paper focuses on one such way of projecting the virtual ad on a pre-recorded tennis match using camera calibration and perspective projection techniques.

The work is organized as follows. In Section II we discuss materials used and methods implemented. Section III details the proposed algorithm. Section IV discusses about the results. Section V concludes the paper.

II. METHODS AND MATERIALS

A. Materials

A short Tennis clip between Garcia-Lopez and A. Seppi in ATP World Tour is used to project the University of Twente advertisement. MATLAB 2017b with Image processing and Computer Vision toolbox is used in the assignment.

B. Methods

1) Analysis:

The aim of the assignment is to project an advertisement on a sport clip and transform it, to make it appear, as if it is on the field. The projection of the advertisement on the original video requires analyzing thoroughly all the frames in the video. Different aspects of the video, if not carefully analyzed might result in undesirable projection of the advertisement. The foremost technique to find the area of projection, is to establish a relation between the image plane and the point of projection. This can be handled effectively, if the camera calibration is done. For camera calibration it is necessary to find six or more feature points on every image, to calibrate the camera across all the frames.

To find the feature points, landmarks which are visible across all the frames is found. Since, there is no specific calibration object available, we choose the court lines and intersections as the feature points. The court lines can be detected by various line detection and edge detection techniques. The Pseudo Code of the assignment is shown in figure 1.

a). Edge Detection

Two different methods were applied to find the Edges of the court lines in the frames, and the best method which provided the better result was implemented in the assignment. The MATLAB inbuilt function 'edge' supports several edge detection filters, and the most popular filter 'Canny' was used. The Canny edge detector is widely used in computer vision to locate sharp intensity changes and to find object boundaries in an image. The MATLAB function finds edges by looking for local maxima of the gradient of the input image. Then it

calculates the gradient using the derivative of a Gaussian filter. The function detects the strong and weak edges, and so it is susceptible to noise [5]. The resultant frame after the implementation is shown in figure 2.

Algorithm ***advertisment_placement*** is
Input: RGB image frames from vidReader
Canny Edge detection
Hough Transform
Feature points extraction
World Coordinate construction
Camera Parameters estimation
Projection of Advertisement
Output: Original RGB image frames from vidReader with projected advertisement

While for each RGB image frames in vidReader
RGB image frame ← edge ('canny')
Return RGB image frames in vidReader1

For all frames in vidReader1
RGB image frame ← Hough Transform ()
Line segments ← RGB image frame
Feature points ← Line-Line intersection formula (Line segments)
Image points for all frames ← Feature points
Return Image points for all frames

For all points in Image points
Camera parameters← World coordinates + Image points
Intrinsic parameter (k) ← Camera Parameters
Rotation matrices (R) ← Camera Parameters
Translation matrices (t) ← Camera Parameters
Return k, R, t

Input: Points for advertisement in World Coordinates as P
While for each RGB image frames in vidReader2
Image points for advertisement ← P * K * [R t]
Return RGB frames with advertisement on the Image points for advertisement

Figure 1 Pseudo Code

The other method that can be applied for detecting the edges is segmentation. Image segmentation is the process of partitioning a digital image into multiple sets of pixels. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze [6].

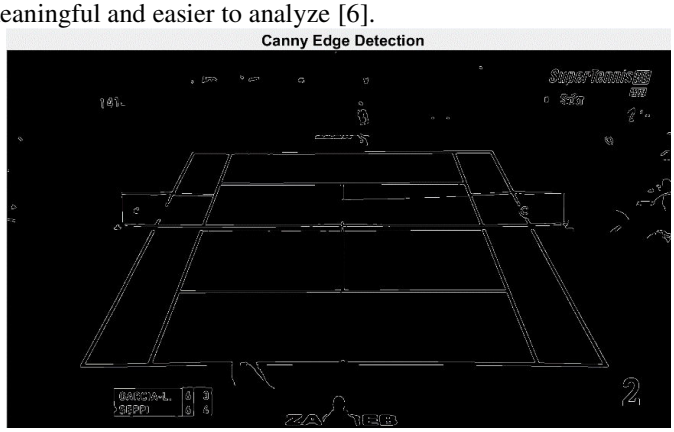

Figure 2 Edges by Canny Filter

By creating a marker (lower threshold) and mask (higher threshold) of the input image, it is possible to detect the edges of the courts. It is also helpful, as the court lines are white and have higher intensities. Thus, when a specific range of threshold is picked, it is possible to eliminate all extraneous detail in the image. By applying a marker of 0.8 and mask of 0.9, it was possible to eliminate unwanted details and only get the lines of the image as shown in the figure 3.

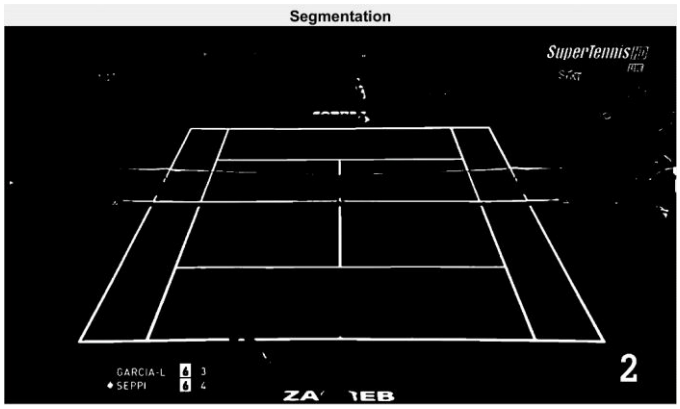

Figure 3 Edges by Segmentation

b). Line Detection

The feature points in the image were found to be the intersection point of court lines. These points dont change shape or heed to lens distortion, so they were the perfect candidates as the camera calibration parameters. To find the lines along the edges, we use Hough transform. The purpose of the Hough transform is to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects. Thus, even when some pixels in the court lines are lost, performing Hough transform on the edges detects the possible lines that can be formed. MATLAB has an inbuilt function for Hough transform, which returns the angle of the detected lines with respect to the x axis and distance from origin to the line along vector perpendicular to the line and two points from the detected line called rho. Multiple spurious lines will be detected, but gap filling and maximum length for line segments are implemented to restrict the selection. This information is used to get the lines of interest. The Lines detected are shown in the figure 4.

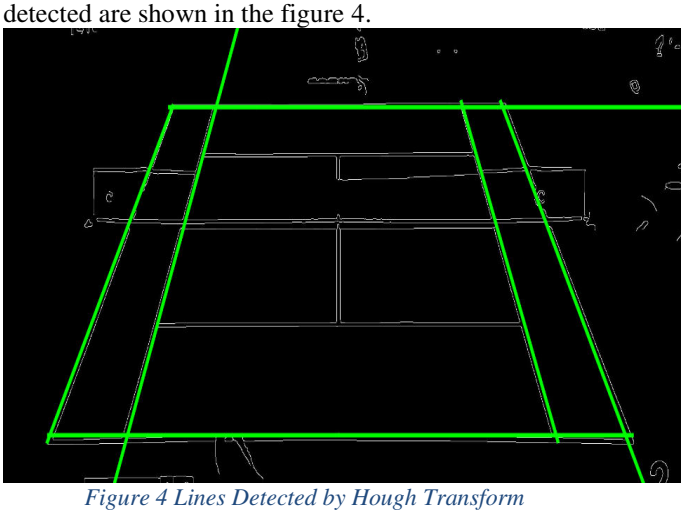

Figure 4 Lines Detected by Hough Transform

c). Feature Point Detection

The Hough lines return the lines and endpoints of the court lines. This information is enough to find the point of intersections and hence, the feature points. A necessary condition for two lines to intersect is that they are in the same plane. Since, this condition is satisfied, the two point formula implemented.

If two lines have end points (x1, y1)(x2, y2) and (x3, y3)(x4, y3), the resultant point of intersection can be found by:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

By this formula, we find all the intersection points possible by the line segments. It is important to note that, the court lines are indeed parallel in world coordinate plane but are not parallel in the image plane. These lines intersect at a point called the Vanishing point. These points appear outside the image plane and has a negative x or y value, depending on the orientation. We however neglect these values in this method. Now, we find all the intersection points only in the image plane and pass them as the camera parameters. The found intersection points can be found in the figure 5.

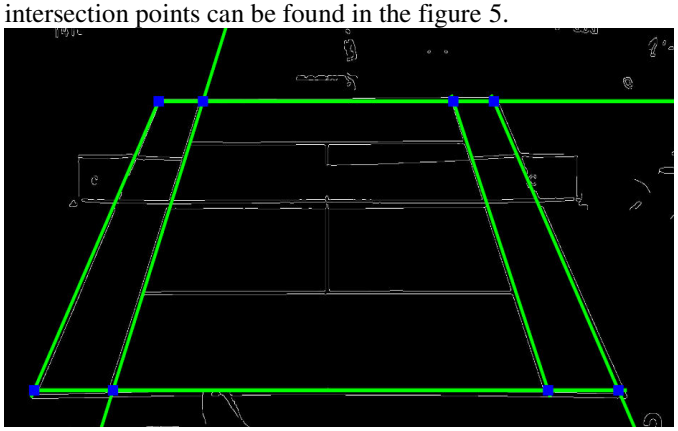

Figure 5 Points of Intersection

d). Camera Calibration

Camera calibration estimates parameters of a lens and image sensor of an image or video camera. These parameters can be used to correct lens distortion, measure the size of an object in world units, or determine the location of the camera in the scene. For Calibration, it is necessary to find at least six feature points. We however, found eight points of intersections as the camera parameters from the Hough transform. To find the camera parameters, we use the MATLAB function estimateCameraParameters, from which we obtain the intrinsic and extrinsic parameters. The Tsai algorithm and Zhang algorithm are very popular algorithm which can establish a relation between the world coordinates to image coordinates. The MATLAB function encompasses the Tsai algorithm which consists of two steps: I) From World coordinate system to Camera coordinate system. II). From Camera coordinate system to image coordinates. The estimateCameraParameters returns the intrinsic parameter 'K' and extrinsic parameter 'R' and 'T', rotation matrix and translation matrix which can be used to implement the two steps of the Algorithm.

I). World to Camera Coordinate system

We first establish a 3D world coordinate system of the feature points. To implement the coordinate system, we find the dimensions of the court lines. The dimension of the court lines can be found in the figure 6.

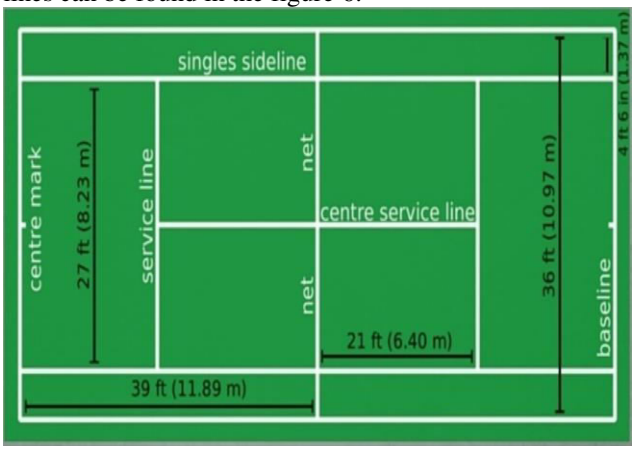

Figure 6 Court Line Dimensions

The dimensions are considered in meters, and the top left corner is found as the origin. So now according to the dimensions, we find the (x,y,z) points of all the feature points.

Point1 – (0,0,0)
Point2 - (1.37,0,0)
Point3 - (9.6,0,0)
Point4 - (10.97,0,0)
Point5 - (0,23.78,0)
Point6 - (1.37,23.78,0)
Point7 - (9.6,23.78,0)
Point8 - (10.97,23.78,0)

The points and the corresponding feature points are numbered in the figure 7.

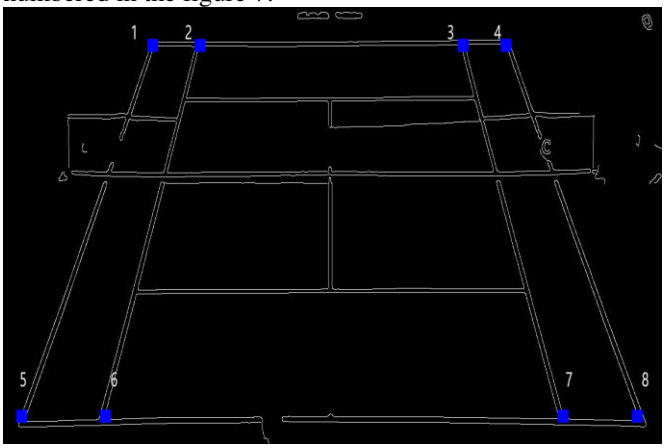

Figure 7 The numbered feature points

The points in 3D world Coordinate system (Xw, Yw, Zw) when combined with the Rotation 'R' and Translation Vector, gives the Points in Camera Coordinate System (Xc, Yc, Zc), which also happens to be in 3D coordinates.

$$\begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix} = R \begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix} + T$$

2). Camera Coordinate System to Image Coordinates

The 3D camera coordinates are transformed to Image coordinates by finding the intrinsic parameters. The intrinsic matrix consists of the focal length, image sensor format and

principal point. This matrix 'K' is estimated and returned as a parameter of the estimatecameraparameter function.

$$K = \begin{bmatrix} f & s & a \\ 0 & f & b \\ 0 & 0 & 1 \end{bmatrix}$$

The 'f', 's', 'a &b' represents the focal length, skew and principal points. These points can be used to find the relation between the camera coordinates and the image coordinates.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix}$$

The resulting matrix are the homogeneous coordinates in the image coordinates. To find the homogeneous coordinates, we divide all the elements by the third element.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix}$$

The resulting (u/w, v/w) are the corresponding image coordinates of the world coordinates (Xw, Yw, Zw).

e). Projection of the advertisement

By finding the camera parameters and all the feature points, we can find the desired position of our advertisement in terms of world coordinates and the corresponding image coordinates. To project the banner in such a way that, there is no geometrical difference between the virtual banner and a real banner of same size, we implement the perspective projection technique.

The intrinsic and extrinsic parameter returns the image coordinates of the corresponding world coordinates, which was discussed in camera calibration section

$$W[x\ y\ 1] = [X\ Y\ Z\ 1]\begin{bmatrix} R \\ t \end{bmatrix} K$$

X, Y, Z are the World Coordinates, "R, t" are the rotation and translation matrix, K is the intrinsic parameters. The resultant value is non-homogeneous image coordinates.

The results and performance of the above-mentioned aspects are discussed briefly with accordance to the assignment in the performance evaluation section.

2) Performance evaluation:

Both the edge detection methods were implemented, and the lines along the edges were found by the Hough transform. On using the Segmentation technique, a lot of spurious lines were detected. Even though, they can be eliminated by filtering out lines with similar rho and theta values, canny operation provided excellent lines along the edges, and hence, we chose to implement Canny edge operator in the final assignment.

To obtain the feature points across all frames, the Hough lines should be maintained in the entire video. We used a low threshold, less minimum length and greater gap filling to find as many lines possible. The lines that doesn't bode well were filtered with the rho and theta values.

When finding the feature points and storing the values for correlating with the world coordinates, it is essential to match the same points across all frames. We have obtained the intersection points by a "for loop", which finds the intersections across all lines formed in the Hough transform. In some frames, the points are interchanged, the MATLAB produces an error stating a huge amount of Lens distortion has been produced, and thus it was unable to find the camera parameters. This was rectified by perusing the points obtained across all frames, finding the specific frames and reconstruction the points.

The camera parameters are obtained conveniently by the estimateCameraParameters function. The important condition to note is to change the indexing of the points. The MATLAB index for points is generally (y,x), but in the estimateCameraParameters, the coordinates should be (x,y) for both the world coordinates and the feature points. It was fairly easy to find the parameters, with a reprojection error of under 3 pixels which was sufficient to provide a good projection result.

The perspective projection of the advertisement includes creating masks of the image and advertisement. The masked advertisement is then subjected to geometric projection by the estimateGeometricTransform function, which returns a transformed image. This can be warped into the original image. Sometimes, a player might superimpose the advertisement, upon which the structure of the player should be visible. i.e. the player should overlap the advertisement. This is not considered in our project, as the player doesn't superimpose our advertisement.

TABLE I
CORNER POINTS FROM CAMERA CALIBRATION

World Coordinates (Xw, Yw, Zw, 1)	Image Coordinates (u, v)
(13,14,0,1)	[1746,537]
(11,14,0,1)	[1538,537]
(11,18,0,1)	[1600,652]
(13,18,0,1)	[1830,653]

III. RESULTS

Figure 8 shows the advertisement overlaid on the tennis court. The corner points of the advertisement and their corresponding image coordinates are shown in the table I.

Figure 8 Final Result with UT advertisement

IV. DISCUSSION

The framework mentioned on sections 2, gave us desirable results. We also found some other methods, by which the similar results might be obtained.

In the line detection and feature points extraction section, as we have mentioned, we found points that are intersection of lines which appear to be parallel, called the vanishing point. By [7], it is also possible to find the camera calibration with one or more vanishing points. But, by applying this method, we cannot estimate the intrinsic parameters, as they are unknown. And so, we might be forced to specify the image points of the advertisement explicitly with respect to the vanishing point. Hence, this method although promising was not used.

We also found in our video, the camera doesn't pan much and so the pixel points don't change much across the frames. This can be found by the Extrinsic parameters calculated as shown in the figure 9. Since, all the frames are in the same plane, we can explicitly mention the pixel points of the advertisement, and not find any huge discrepancy.

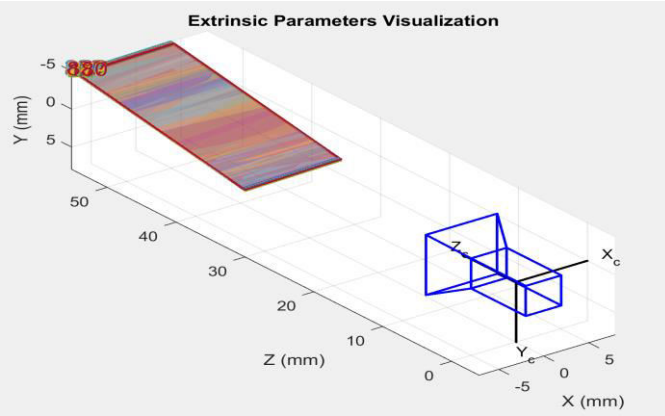

Figure 9 Extrinsic Parameters Visualization

Although, the Edge detection methods provide very good results, some pixels are lost which cause discontinuity along the edges. This causes serious problems when the Hough transform for line detection is applied. We tried the same techniques mentioned in the sections above on a volleyball game. Even by using lower threshold values, it was almost impossible to detect the same lines across all frames. We tried finding the feature points in the first frame alone and then tracking the points across all other frames. For this method, we applied vision.pointtracker, a MATLAB computer vision toolbox command to track the specific points across all frames in vain. It was particularly problematic as we could not find the same points in all frames.

V. Conclusion

The Camera Calibration and Perspective projection gives us the desirable results. Several improvements on the algorithm can be done. Finding the points in just one frame and updating it across all frames can be a much easier and effective method, as maintaining the points across all frames is a challenge.

Appendix

MATLAB script

```
%% Tennis Game
close all
clear all
%% Reading the video
fname = 'C:\UT Files\Image Processing and Computer
Vision\Final Project\tennis.mp4';
vidReader = VideoReader(fname);

%% Canny Edge Detection & Writing the video
      edgeplay=VideoWriter('cannyedge_tennis.avi');
      open(edgeplay);
while hasFrame(vidReader)
   frameRGB = readFrame(vidReader);
   frameGray = rgb2gray(frameRGB);
    edge1=edge(frameGray,'Canny',[0.25 0.55]);
        bw = colormap(gray(2));
        frame = im2frame(uint8(edge1), bw);
        writeVideo(edgeplay,frame);
end
       close(edgeplay);
%% Feature Points Estimation across all present frames
gname='C:\UT Files\Image Processing and Computer
Vision\Final Project\cannyedge_tennis.avi';
vidReader1=VideoReader(gname);
count =0;
nFrames = vidReader1.NumberOfFrames;
for iFrame=1:1:nFrames
   frame1=read(vidReader1,iFrame);
   frameGray1 = rgb2gray(frame1);
   count = count+1;
   [accum, theta, rho] =hough(frameGray1);
peaks=houghpeaks(accum,1000,'Threshold',ceil(0.35*max(acc
um(:))));
line_seg=houghlines(frameGray1,theta,rho,peaks,'FillGap',600
,'MinLength',600);
   seg= [];
    imshow(frame1); hold on
   for k=1: length(line_seg) % Filtering the spurious line
segments
      if ((((line_seg(k).rho <-800 && line_seg(k).rho >-950)||
line_seg(k).rho >-300)&& line_seg(k).theta==-
90)||(line_seg(k).rho>0 && (line_seg(k).theta<-90 ||
line_seg(k).theta<80)))
       seg=cat(1,seg,line_seg(k));
       endpoint=[line_seg(k).point1; line_seg(k).point2];
       end
   end
   f= [];
```

```
    g= [];
    for i=1: length (seg) % Finding the Point of Intersection of
each lines
        for j=i:length(seg)
        [a]=seg(i).point1;
        [b]=seg(i).point2;
        [c]=seg(j).point1;
        [d]=seg(j).point2;
        res=((a(2)-b(2))*(c(1)-d(1)))-((a(1)-b(1))*(c(2)-d(2)));
        if (res~=0)
        [e]=[(((a(2)*b(1)-a(1)*b(2))*(c(2)-d(2))-(a(2)-
b(2))*(c(2)*d(1)-c(1)*d(2)))/res),(((a(2)*b(1)-
a(1)*b(2))*(c(1)-d(1))-(a(1)-b(1))*(c(2)*d(1)-
c(1)*d(2)))/res)];
       if (e(1)>0)
        f=cat(1,f,e);
       end
        end
        end
    end
    f=fliplr(f); % Maintaining the same feature point order
across all frames in (x,y) indexs
     fi=f(1:end/2,:);
        fi=sort(fi);
        fi1=f(end/2 +1:end,:);
        fi1=sort(fi1);
        g=cat(1,fi,fi1);
        g=g';
        g=sortrows(g,2);
        g=g';
        % Changing the index and point order of certain frames
found by
        % peruse
        if (iFrame==123 || iFrame==124 || iFrame==125 ||
iFrame==128 || iFrame==131 || iFrame==133 || iFrame==153 ||
iFrame==154 || iFrame==224 || iFrame==225 || iFrame==226 ||
iFrame==227 || iFrame==228)
            g1=g(1:end/2,:);
            g1=fliplr(g1);
            g2=g(end/2 +1:end,:);
            g2=fliplr(g2);
            g=cat(1,g2,g1);
        end
     % Drawing a rectangle on the point of Intersection
      for l =1:1:length(g)
        width = 10;
        height = 5;
        xCenter = g(l,1);
        yCenter = g(l,2);
        xLeft = xCenter - width/2;
        yBottom = yCenter - height/2;
        rectangle('Position', [xLeft, yBottom, width, height],
'EdgeColor', 'b', 'FaceColor', 'r', 'LineWidth', 4);
        grid on;
      end
   hold off;
   imagePoints(:,:,count)=g; % Feature points across all frames
end
%% Camera Parameters
worldPoints=[0,23.78;1.37,23.78;9.6,23.78;10.97,23.78;0,0;1.
37,0;9.6,0;10.97,0];
  params =
estimateCameraParameters(imagePoints,worldPoints,'ImageSi
ze', size(frameGray1));
   showExtrinsics(params);
%% Advertisement projection
    k=params.IntrinsicMatrix; % Intrinsic parameter from the
estimateCameraParameter function
    hname='C:\UT Files\Image Processing and Computer
Vision\Final Project\tennis.avi';
    vidReader2=VideoReader(hname);
    nFrames = vidReader2.NumberOfFrames;
     edgeplay2=VideoWriter('final_tennis.avi');
       open(edgeplay2);
    for iFrame1=1:1:nFrames
        frame1=read(vidReader2,iFrame1);
        r=params.RotationMatrices(:,:,iFrame1);
        t=params.TranslationVectors(iFrame1,:,:);
        point1=[13;14;0;1]; % world coordinates of the corners
of advertisement
        point2=[11;14;0;1];
        point3=[11;18;0;1];
        point4=[13;18;0;1];
        mat1 =
[r(1,1),r(1,2),r(1,3);r(2,1),r(2,2),r(2,3);r(3,1),r(3,2),r(3,3);t(1,1)
,t(1,2),t(1,3)];
        pixel1 = point1' * mat1 *k; % Calculating with Intrinsic
and Extrinsic Parameters
        pixel1 = [pixel1(1)/pixel1(3);pixel1(2)/pixel1(3)];
        pixel1 = round(pixel1');
        pixel2 = point2' * mat1 *k;
        pixel2 = [pixel2(1)/pixel2(3);pixel2(2)/pixel2(3)];
        pixel2 = round(pixel2');
        pixel3 = point3' * mat1 *k;
        pixel3 = [pixel3(1)/pixel3(3);pixel3(2)/pixel3(3)];
        pixel3 = round(pixel3');
        pixel4 = point4' * mat1 *k;
        pixel4 = [pixel4(1)/pixel4(3);pixel4(2)/pixel4(3)];
        pixel4 = round(pixel4');
        p_bound =[pixel1;pixel2;pixel3;pixel4];
        ad = imread('ut.png'); % Reading the Image

rpoints=[1,1;1,size(ad,1);size(ad,2),size(ad,1);size(ad,2),1]; %
Dimensions of the ad
        h = estimateGeometricTransform(rpoints,
p_bound,'projective'); % Geometric Transform
        adv_h = imwarp(ad,h); % Warping the ad to transform
        mask = imwarp(ones(size(ad)),h); % Create a mask with
size of ad and warp with transform
        mask = mask ~= 0;
        xposition = round(min(p_bound(:,1))); % Position of x
corner
        yposition = round(min(p_bound(:,2))); % Position of y
corner
        t = zeros(size(frame1),'logical'); % mask of the original
frame
        mask_org = t(yposition:(yposition+size(adv_h,1)-
1),xposition:(xposition+size(adv_h,2)-1),:);
```

```
        mask_org(mask) = 1;
        t(yposition:(yposition+size(adv_h,1)-
1),xposition:(xposition+size(adv_h,2)-1),:) = mask_org;
        frame2 = frame1;
        frame2(t)=adv_h(mask);
        filt = fspecial('motion');
        frame2 = imfilter(frame2,filt,'replicate');
        imshow(frame2);
        pause(0.1);
        writeVideo(edgeplay2,frame2);
    end
close (edgeplay2);
```

REFERENCES

[1] A. Hegde, D. A. Kumar and G. Srinivasa, "Contour-traversal based algorithm for insertion of virtual advertisements in videos," 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Mumbai, 2012, pp. 1-5.
doi: 10.1109/ICCICT.2012.6398213

[2] Yi Tan, "Virtual imaging in sports broadcasting: an overview," Proc. SPIE 4756, Third International Conference on Virtual Reality and Its Application in Industry, (1 April 2003); doi: 10.1117/12.498030, Hangzhou, China

[3] C. Linqiang, H. Jianping and Z. Linfeng, "Video Segmentation Algorithm in Virtual Advertisement System," 2008 International Conference on Cyberworlds, Hangzhou, 2008, pp. 735-740. doi: 10.1109/CW.2008.89

[4] S. Li and B. Lu, "Automatic Camera Calibration Technique and its Application in Virtual Advertisement Insertion System," 2007 2nd IEEE Conference on Industrial Electronics and Applications, Harbin, 2007, pp. 288-292. doi: 10.1109/ICIEA.2007.4318417

[5] Nl.mathworks.com. (2018). Find edges in intensity image - MATLAB edge-MathWorks Benelux. [online] Available at: https://nl.mathworks.com/help/images/ref/edge.html#d119e43753 [Accessed 29 Apr. 2018].

[6] En.wikipedia.org. (2018). Image segmentation. [online] Available at: https://en.wikipedia.org/wiki/Image_segmentation [Accessed 29 Apr. 2018].

[7] R. Orghidan, J. Salvi, M. Gordan and B. Orza, "Camera calibration using two or three vanishing points," *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Wroclaw, 2012, pp. 123-130.

[8] M. Sharifi, M. Fathy and M. T. Mahmoudi, "A classified and comparative study of edge detection algorithms," *Proceedings. International Conference on Information Technology: Coding and Computing*, 2002, pp. 117-120. doi: 10.1109/ITCC.2002.1000371

[9] F.V.D. Heijden, "Camera Models", University of Twente, October 2016

[10] F.V.D. Heijden, "Key point matching and optical flow", University of Twente, December 2016