STUDY OF FACE TRACKING USING MEANSHIFT AND CAMSHIFT ALGORITHM

Author: Mansa Pabbaraju

Institute: Rochester Institute of Technology

Course: CSCI-631 Spring 2016

Professor: Thomas Kinsman

ABSTRACT:

Face detection and tracking has its uses in various critical applications of computer vision like identity establishment of people, identification of disguised criminals to arrest them, differentiation between the human face and other objects for a robot etc[19]. The main idea behind this project is to study a simple and computationally efficient color based object tracking algorithm - the CAMSHIFT algorithm. CAMSHIFT is a modified version of the Mean Shift algorithm known as Continuously Adaptive Mean Shift. This algorithm was introduced as part of a larger Perceptual user Interface by Gary R. Bradski[2].

In this project, we understand and study the working of Mean Shift and CAMSHIFT algorithm and its application to track a human face in a moving image. We perform various experiments on the CAMSHIFT algorithm to observe the efficiency of CAMSHIFT in tracking a human face in a video.

INTRODUCTION:

The CAMSHIFT algorithm was proposed by Gary R. Bradski in his research paper "Computer Vision Face Tracking For Use in a Perceptual User Interface "[2].This algorithm was proposed as a part of a larger project to build a Perceptual User Interface.

When the computer is given human sensory inputs in analog form - like movement of hands, sound of speech, the electronic sensors detect these analog sensory signals and thereby process these signals[1]. In this case the computer "behaves" and the user "perceives" the computer as a human brain. Thus, this concept is known as **perceptual user interface[1]**.

Face detection and tracking form a major part in the building of a Perceptual User Interface - which revolves around the concept of the computer behaving like a human user when interacting with the user[1]. For example, human gestures like nodding of head, moving back and forth, blinking of the eyelids can be a form of decisive signals to the computer. For this project, the camera is the model which takes in these human gestures as signals and the face detecting algorithm detects these signals to perform computational processes on them[1][2].The main motive of this project was to study how the computer takes the human face movement as input and performs "tracking" of these face movements as an output. The tracking algorithm used is the CAMSHIFT algorithm based on the Mean Shift algorithm.

Study of the Mean Shift algorithm required understanding of color histograms, color probability distribution image (also known as back projection) ,concept of image moments and the intuition behind the mathematical expressions for image moments. While static probability distribution of image pixels is used in Mean Shift, CAMSHIFT required the study of dynamic probability distribution of image pixels in a video. The study for CAMSHIFT also included problems faced in relation to the four degrees of freedom for head movement which CAMSHIFT measures : X position, Y position, Z position and the head roll.

Different input video frames were run on the CAMSHIFT algorithm implemented by MathWorks[4] to understand the working of CAMSHIFT under different inputs. The working of CAMSHIFT under the presence of occlusion was also studied and the results were noted down.

PREVIOUS WORK:

There have been various face trackers developed previously like those involving Eigenspace matching [11], maintaining large sets of statistical hypotheses which are are computationally heavy.[2]

Our need is to develop a computationally efficient face tracker which satisfactorily operates in the presence of noise and occlusion. Thus, CAMSHIFT algorithm was introduced[2] to be a part of a larger perceptual user interface. An object tracking algorithm for a perceptual user interface should consume as few system resources as possible so as be a part of the routinely carried out tasks of the perceptual user interface. It needs to save resources for the main functionality of the perceptual user interface. [2]

MEAN SHIFT ALGORITHM:

Introduction:

CAMSHIFT algorithm is based on the concept of the Mean Shift Algorithm as proposed by Y. Cheng in his research paper "Mean Shift, Mode Seeking, and Clustering"[3].
The Mean Shift Algorithm is stated as a "robust nonparametric technique for climbing density gradients to find the mode (peak) of probability distributions " in [2].

Mean shift algorithm in image processing is based on finding the mean of the neighborhood surrounding pixel's probability density function. The pixel with the highest occurring probability of the considered value is considered to be the mode or peak value. Using the concept of image moments, the centroid location of the target object's probability distribution is found and the search window frame is readjusted.[3][6][7]

Basic Concept [6]:

Given a probability distribution of any parameter of the pixel (can be brightness, hue, or a channel in some other color space) and a particular window (of any given shape),we need to determine the best location of this window where the maximum number of pixels are present.[6]

We use the Mean Shift Algorithm to determine this. Consider Figure 1), where the given window is a circle C1 in blue. Let C1_a be the centre of this circle. Every data point represents the probability of the pixel being a part of our target object. We find the centroid of this blue circle which is C1_r[6]. Thus, the maximum density of probable pixels of the object being tracked is nearer to C1_r. Hence, we place the circular window with centre at C1_r. This shifted value is a vector in the direction of the maximum probability density region and is called **"mean shift".** We repeat the above process until convergence[6]. The problem of Mean Shift was first applied to the problem of mode seeking by Cheng[3].

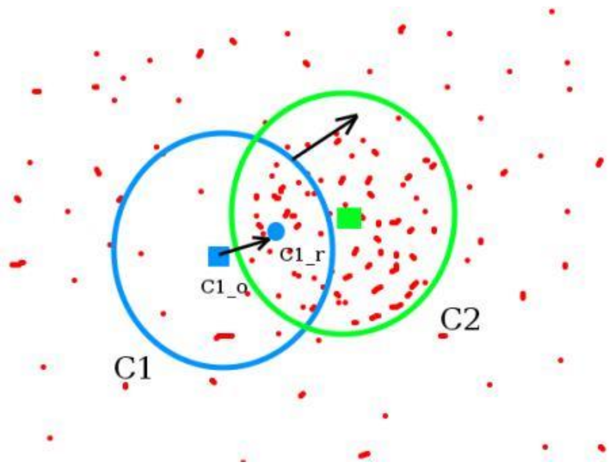


Figure 1) Image taken from http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html#gsc.tab=0 [6]

Face Tracking with Mean Shift Algorithm:

The largest portion of a human face consists of skin. The tones of flesh is the most distinct feature from among the surrounding background objects. Thus it makes it easier to detect the face via the skin feature extraction. We convert the input image frame into HSV color space and consider only the HUE channel. The HUE channel consists of color that particular pixel has, thereby, we can separate the rest of the background and the face based on this HUE difference between the skin of the face and the surrounding objects.

We find the probability distribution of the HUE of the image, and with the help of the Mean Shift Algorithm, we determine the correct position of the search window in the image comprising of the face. We need to keep a continuous track of this mode of probability distribution to keep a track of the movements of the face in a video.

A motion video consists of moving images displayed at a resolution rate of 24 image frames/sec - to give the human eye the perception that it is actually a moving real world image. This phenomenon is known as the persistence of vision[15]. The color probability distribution of face in every image frame in the video changes. Thus, we need to calculate the Mode for every such shifted probability distribution to re calculate the location of the window size in the changed image.

Object Tracking using Mean Shift, search window size remains same in all frames:

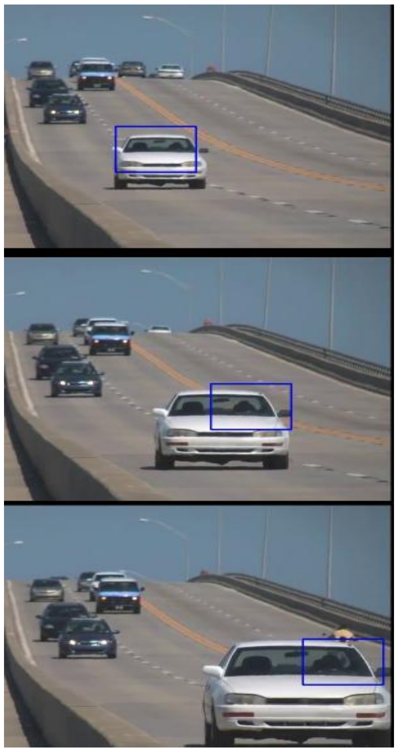


Figure 2) Image taken from http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html#gsc.tab=0 [6]

Issue with Mean Shift Algorithm in Face Tracking:

Figure 2 consists of screenshots where we need to keep a track of the face of the driver. A fixed window size is decided and chosen by the user and manually placed in front of face of the driver. As the car moves ahead, the size of the driver's face and the car appears to become bigger. This is due to the fact that objects in images look bigger when they are closer to the camera sensor than those objects that are further away from the camera sensor[6]. However, the window size in every video frame remains the same regardless of the change in size of object being tracked.

A better way to track the face of the driver would be a re-sizeable window size , so that when the moving car comes near the camera sensor, the search window adapts to the size of the object being tracked.

Object Tracking: Search window size changes as per change in size of object being tracked:

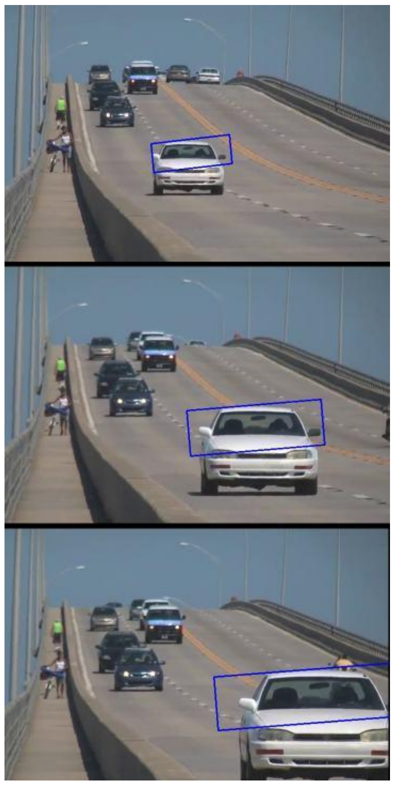


Figure 3) Image taken from http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html#gsc.tab=0[6]

<u>CAMSHIFT</u>

<u>Overview:</u>

As seen in figure 3, when the window size increases as the car moves closer to the camera sensor, the object to be tracked (face) can be seen more clearly. Modifying size of the window according to the changes seen in the object being tracked can be achieved via the CAMSHIFT algorithm. The Mean Shift Algorithm is modified from being a static robust mean-finding algorithm to a dynamic object tracking algorithm which continuously adapts to the change in object being tracked[8]. The size of the search window changes according to the size of the object being tracked. Thus, this algorithm is known as CAMSHIFT - **C**ontinuously **A**daptive **M**ean **S**hift **A**lgorithm[8]. The CAMSHIFT algorithm is essentially an object tracking algorithm based on the color probability distribution of the object being tracked. The CAMSHIFT algorithm climbs the gradient of a back-projected probability distribution to find the nearest peak within a search window[7].

CAMSHIFT Algorithm:

Search window Size:

The CAMSHIFT algorithm detects the change in the probability distribution of the image under the search window size. Thus, to detect a gradient, the search window needs to be greater than 1[2].

Also, for every iteration of the Mean Shift Algorithm, we need to calculate the centroid of the probability distribution. Thus, the size of the search window frame needs to be of odd value after every iteration. In case the centroid is even, it is rounded off to the nearest odd value[2].

Algorithm Steps:

1. Take the color Histogram of object to be tracked - the target object

2. Take the color probability distribution image of the input video frame based on the color histogram from step 1 as look-up table - Back Projection.

3. Compute the centroid using the zeroth moment of the input color probability distribution image under the search window frame and repeat until convergence (Mean Shift)

4. Once the search window location is fixed, store the zeroth moment of the probability distribution found under the search window. Use this to calculate the size of the search window frame for the next input image frame. The first and second moments can also be used to calculate the new size of the search window frame for the next input image frame.

5. Repeat steps 2 to 4 for continuous tracking of the target object in the video frame.

CAMSHIFT Algorithm Step 1:

Store Color Histogram of target object

We convert the input image frame into the HSV color space to separate out the Hue(Color), Saturation (intensity of color) and brightness values for every pixel. As we are using the flesh color for face tracking, we take the Hue channel and create a 1D histogram[2].

The Viola Jones face detector[17] is used to detect the face(our target object) in the MathWorks implementation we are using[4]. This is the initial location of our search window. For the face under the search window, we plot its color histogram. A color histogram is a plot of the frequency of occurrence of each permitted color value in the image and the color itself [16]. The Hue is a 8 bit component, hence the permitted values for the hue is from 0 to 255[2]. We divide these permitted values into groups known as 'bins'. We check which bin each pixel in the input image frame belongs to and count the number of pixels in every bin. The result represents the color distribution of the target object and we store this color histogram as our MODEL, or a LOOKUP table, to convert incoming video pixels to a corresponding probability distribution image of the target object[2].

CAMSHIFT Algorithm Step 2:

Back Projection - Color Probability Distribution

For every incoming set of pixels in the video frame, we check the probability of these pixels belonging to the target object based on the color histogram look-up table obtained from CAMSHIFT Algorithm Step1).

Consider the following colored image:

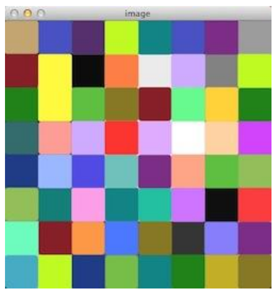


Figure 4) Image taken fromhttp://eric-yuan.me/continuously-adaptive-shift/ [12]

Convert the colored image into HSV color space and take the HUE channel:

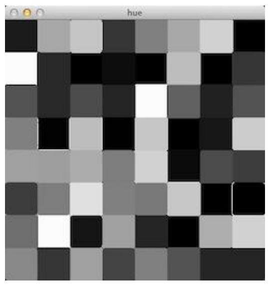


Figure 5) Image taken fromhttp://eric-yuan.me/continuously-adaptive-shift/ [12]

Take the color histogram of the hue channel image:

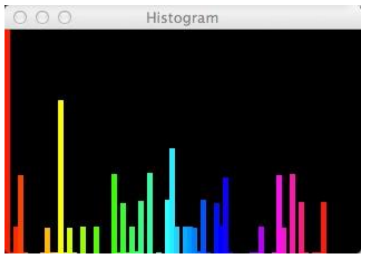


Figure 6) Image taken fromhttp://eric-yuan.me/continuously-adaptive-shift/ [12]

In the above process, we calculate the occurrence of each color. This gives us the weight of each color in its back projection. We change the value of each pixel to the weight of its color in the back projected image[12]. For example, consider the color blue and pink in the histogram. The occurrence of both colors is almost the same. Say blue color is 15% of the entire image and pink is 17% of the color in the entire image. This means the weight of blue is 15% and that of pink is 17%. Thus, we change blue pixel's value 0.15 and pink pixels become 0.17. [12]

Thus, most commonly occurring colors get a higher value in the back projected image and those which occur less frequently get a smaller value[12].

Back projected image:

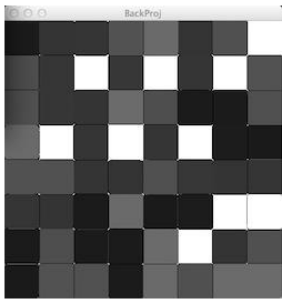


Figure 7) Image taken fromhttp://eric-yuan.me/continuously-adaptive-shift/ [12]

For every incoming video frame, we calculate the back projection that is the probability distribution image based on the target color histogram obtained in CAMSHIFT Algorithm Step 1 as our LOOKUP table.

Note: All images above are taken from http://eric-yuan.me/continuously-adaptive-shift/ [12]

The following figure shows color probability distribution image of a face:

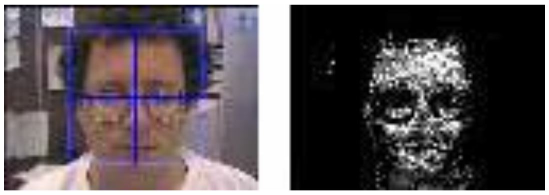


Figure 8) Image taken from G. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface,” Intel Technology Journal, httu://developer.intel.com/technolog.v/iti/1a929 8iarticledart 2.htm, Q2, 1998. [2]

CAMSHIFT Algorithm Step 3:

Calculate the new location of the search window (using Mean Shift Algorithm)

From CAMSHIFT Algorithm Step 2, we have the probability distribution image of the incoming video frame with respect to our target color histogram(obtained from Step 1).

Intuitively, this probability distribution tells us whether the pixel in the search window is part of the target object (face) or not. Those pixels which are a part of the face have higher probability value than those that are not a part of the face.

We use the concept of Mean Shift to find the most dense region in this probability distribution image and centre our search window at the mean of this distribution. Effectively, we are moving the search window towards the target object in every iteration and we converge only when we have centered out search window at the mean of the probability distribution of the target object. This becomes the new location of our search window frame for next input frame of video.

The calculation of this new centre involves the study of image moments. This centre is effectively the centroid of the probability distribution of the pixels found under the search window.

Image Moments:

In pure mathematics, a moment is a quantitative measure of sample points in a distribution[9]. In physics, a moment is used to describe the distribution of mass of a body. 'Moment' in image processing derives its meaning from both the physics and mathematics definition of moments.[13][14]

Just like how moment in physics describes the mass distribution of a body, an 'image moment' describes the distribution of the grey level in the image (Moments are described for either binary of grayscale image). Thus, for a grayscale image, we can say that the image moment is a quantitative measure to give the distribution of the "grayness" in a grayscale image, and the "area" in a binary image.[13]

1.) In pure mathematics, the equation of n^{th}moment is as follows[14]:

$$\mu_n = \int_{-\infty}^{+\infty} (x-c)^n \, f(x)dx$$

2.) This definition is for a function with only one independent variable. However, as we are dealing with 2D images which have x and y coordinates, we need two independent variables[14]. Hence, our equation is as follows:

$$\mu_{m,n} = \iint (x - c_x)^m \left(y - c_y\right)^n f(x,y)\, dy\, dx$$

3.)As images are discrete and we have to deal with 'pixels'. Hence, our function $f(x,y)$ has to be discrete. Thus, the integrals have been replaced by summations[14]. The order of the moment is $m+n$. The $c_x$and c_y are locations around which we calculate the moments. Normally we do this around the origin $(0,0)[14]$. Hence, the above equation becomes:

$$\mu_{m,n} = \sum_{x=0}^{\infty}\sum_{y=0}^{\infty}(x - c_x)^m\left(y - c_y\right)^n\ f(x,y)$$

4.) To calculate the zeroth moment of the image, our equation becomes as follows:

$$\mu_{0,0} = \sum_{x=0}^{w}\sum_{y=0}^{w} x^0 y^0\ f(x,y)$$

As x^0 and y^0 have no effect,

$$\mu_{0,0} = \sum_{x=0}^{w}\sum_{y=0}^{w} f(x,y)$$

Note:

The above equations are referenced from: http://aishack.in/tutorials/image-moments/[14]

Consider a binary image:

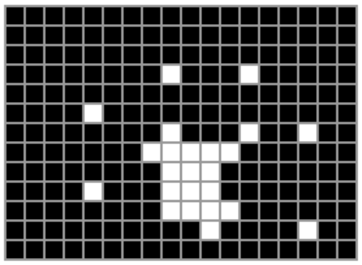


Figure 10) Image taken from http://aishack.in/tutorials/image-moments/ [14]

The pixel values of a binary image are either 1 or 0. Hence, to calculate the zeroth moment of this binary image, a 1 is added for every bright pixel. Thus, we calculate the total area of the foreground object by the zeroth moment equation[14].

A similar concept can be applied for a grayscale probability distribution image in the CAMSHIFT algorithm. Once the final location of the search window is updated, the zeroth moment of the search window frame is calculated. This gives us the total effective area of the target object we are tracking. If the target object's size has increased,(for example, the face moves closer to the camera), we can calculate this increase in size of the target object using the zeroth moment of the area under the search window frame.

For every pixel belonging to the target object, the probability will be very high in the probability distribution image of the incoming input frame. Thus, using the zeroth moment equation in (1),we are effectively calculating the total area of the target object.

Let the $I(x, y)$ be the pixel (probability) value at position (x, y)in the image.

Let and x and y range over the search window.

The zeroth order image moment is taken as follows[2][9]:

$$M_{00} = \sum_x \sum_y I(x, y) \qquad \text{..........} (1)$$

We can think of the zeroth moment as the distribution "area" found under the search window[2]. The window center, window radius (or the height and width) is set to a function of the zeroth moment computed during our search[2].

The first order image moments are taken as follows[2][9]:

$$M_{10} = \sum_x \sum_y xI(x, y)\,; \quad M_{01=} \sum_x \sum_y yI(x, y) \qquad \text{...........} (2)$$

M_{10} : gives sum of all products of the x coordinates and corresponding pixel value for all the probable pixels **belonging** to the target object.

M_{01} : gives sum of all products of the y coordinates and corresponding pixel value for all the probable pixels **belonging** to the target object.

Thus, when we normalize M_{10} and M_{01} with the effective area of the target object (M_{00}), we get the averaged x and y coordinates of the region where the target object is most probable. These resultant x and y coordinates are the centroid of the probability distribution of pixels under the search window.

Let this position be:

$$x_c = \frac{M_{10}}{M_{00}} \quad ; \quad y_c = \frac{M_{01}}{M_{00}}$$

Thus, (x_c, y_c) is the new center of the search window. The above process is repeated until convergence.

Note:

The above equations are referenced from:

1. G. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface,” Intel Technology Journal, httu://developer.intel.com/technolog.v/iti/1a929 8iarticledart 2.htm, Q2, 1998[2].

2."Fast and Robust CAMSHIFT Tracking" by David Exner, Erich Bruns, Daniel Kurz, and Anselm Grundhofer. [9].

CAMSHIFT Algorithm Step 4:

Calculate the new size of the search window as input to next input frame of video

There are two ways we can calculate the size of the new search window:

1) As a function of the zeroth moment.

2) Calculating the second order moments of the distribution area. This also gives us the head roll of the face being tracked.

1) Calculation of size as a function of the zeroth moment:

The zeroth moment is as follows[2][9]:

$$M_{00} = \sum_x \sum_y I(x,y)$$

Now we know that the zeroth moment is the effective area of the probability distribution under the search window frame[2]. Thus, to find the window size, we need to translate the zeroth moment of information into appropriate units[2]. The maximum probability distribution value per discrete cell (histogram bin) is 255 as HUE is a 8 bit component. Thus, we divide the zeroth moment by 256 to convert the calculated area under the search window to units of number of cells[2].

Thus, for 2D probability distributions where maximum pixel value is 255, search window size is:

$$S = \frac{M_{00}}{256}$$

To convert the resulting 2D region to a 1D length, we take the square root[2].

$$S = \sqrt{\frac{M_{00}}{256}}$$

As our goal is to track the whole color target object (entire frame of the face), we further multiply the result by 2 so that the window encompasses the target object's distribution area[2].

Thus, the final search window size becomes:

$$S = 2 * \sqrt{\frac{M_{00}}{256}}$$

Note:

The above equations are referenced from:

1. G. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," Intel Technology Journal, httu://developer.intel.com/technolog.v/iti/1a929 8iarticledart 2.htm, Q2, 1998[2].

2) Calculation of length and width of search window size depending on the second moment:

Using the second order image moments, we can find the first two Eigen values of the probability distribution.

The first Eigen value gives us the 1st Principal Component Axis (Length of the probability distribution) whereas the second Eigen value gives us the 2nd Principal Component Axis (Width of the probability distribution).

Thus, by calculating the Eigen values from the Second order moments of the image, we can calculate the length and width of the probability distribution under the search window frame.

The Second order image moments are as follows:

$$M_{20} = \sum_x \sum_y x^2 I(x,y)\,; \quad M_{02=} \sum_x \sum_y y^2 I(x,y)$$

Calculation of intermediate values,

Let

$$a = \frac{M_{20}}{M_{00}} - x_c^2\,, \quad b = 2 * \left(\frac{M_{11}}{M_{00}} - x_c y_c \right), \quad c = \frac{M_{02}}{M_{00}} - y_c^2$$

Then Length l and width w are given as:

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + x(a-c)^2}}{2}}\,, \quad w = \sqrt{\frac{(a+c) - \sqrt{b^2 + x(a-c)^2}}{2}}$$

Note:

The above equations are referenced from:

1. G. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface,” Intel Technology Journal, httu://developer.intel.com/technolog.v/iti/1a929 8iarticledart 2.htm, Q2, 1998[2].

Calculation of Eigen values from Second order image moments to find the length and width of target object:

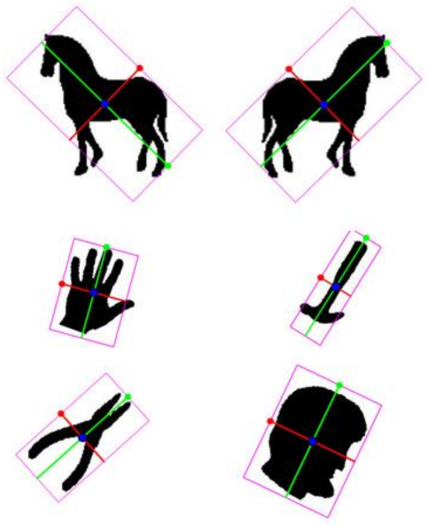


Figure 11: Calculation of Eigen values from Second order image moments to find the length and width of target object. Image is taken from http://www.mukimuki.fr/flashblog/2009/05/20/magic-moments/[13]

Thus, Step 4) gives us the recalculated size of the search window frame depending upon the size of the target object. This size of search window is used as input to the next incoming video frame.

ADVANTAGES OF CAMSHIFT:

Computationally efficient[2]:

When tracking a colored object, CAMSHIFT operates on a color probability distribution of input video image frame based on color histogram of target object. CAMSHIFT calculates the centroid of the color probability distribution within its search window, re-centers the window and then calculates the probability distribution area for the next window size[2]. Thus, we need not calculate the color probability distribution over the whole image, but can instead restrict the calculation of the distribution to a smaller image region surrounding the current CAMSHIFT window[2]. This tends to result in large computational savings when flesh color does not dominate the image. [2]

Robust to Noise[2]:

CAMSHIFT’s windowed distribution gradient climbing causes it to ignore distribution outliers. Therefore, CAMSHIFT produces very little jitter in noise. Thus noise filters do not need to be used[2].

With setting the CAMSHIFT window size to be a value slightly greater than the original probability distribution window size, CAMSHIFT tends to grow to encompass the connected area of the target distribution that is being tracked. Thus CAMSHIFT does not get stuck tracking a particular part, for example, the nose of a face, but instead to track the whole face which is exactly what is desired[2].

SHORTCOMINGS OF CAMSHIFT:

One of the shortcomings of the CAMSHIFT Algorithm is the need for the manual selection of the target object(in our case, the face). This shortcoming can be overcome by creating an "automatic CAMSHIFT algorithm " as proposed in [10].

Also, the CAMSHIFT Algorithm only detects four of the six modes of freedom namely X (the width), Y (the height), Z (the distance between face and the camera) and the head roll. Of the six possible head movements, the head roll is the least common one and is rarely done by the human head. Thus, head roll is the least important of the six possible head movements. CAMSHIFT does not detect pitch and yaw. [2]

In presence of 100% occlusion, CAMSHIFT is not able to track the target object correctly. This is because CAMSHIFT is essentially a color probability distribution object tracking algorithm[2]. Thus, CAMSHIFT tends to follow the occlusion(which has the same HUE as the target object).

EXPERIMENTS AND RESULTS:

Implementation of CAMSHIFT in MATLAB [4]:

The code is taken from MathWorks, (MATLAB R2016a), Computer Vision System toolbox(TM), http://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-camshift.html?refresh=true[4]

In MATLAB, the CAMSHIFT Algorithm is implemented in the following three steps:

Step 1. Detect a Face To Track[4].

We use the MATLAB provided vision.CascadeObjectDetector object to detect the location of a face in a video frame. The cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for detection[17].

Step 2. Identify facial features to track - Skin (HUE Channel)[4].

Once the face is located in the video, the next step is to identify a feature that will help to track the face. For example, we can choose features such as texture, or color. We need to choose a feature that is unique to the object and remains invariant even when the object moves. Hence, we use the HUE of the flesh tones. Here, we use the skin as the selected feature.

Step 3. Track the face[4].

Using the CAMSHIFT algorithm, track the face based on the color histogram probability distribution of any selected feature of the face.

<u>Experiment 1 : Basic working of the CAMSHIFT Algorithm</u>

Sample Input:

The sample input is the video image provided by MathWorks[4] in their vision toolbox called 'visionface.avi'.

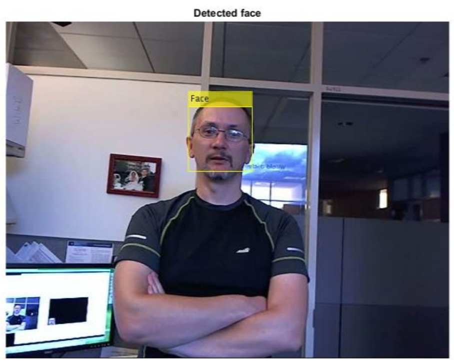
Observations:

In this video sample, CAMSHIFT successfully tracks the human face. The lighting conditions are good and the Viola Jones face detector detects the face correctly using the HUE channel of the face. The human moves left to right and there is also a slight tilt in the head. CAMSHIFT successfully tracks the face of the person continuously.

<u>Results 1:</u>

Step 1: Detect a Face To Track[4]

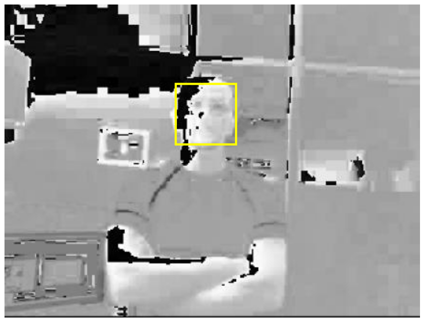
MATLAB Sample Image[4]:



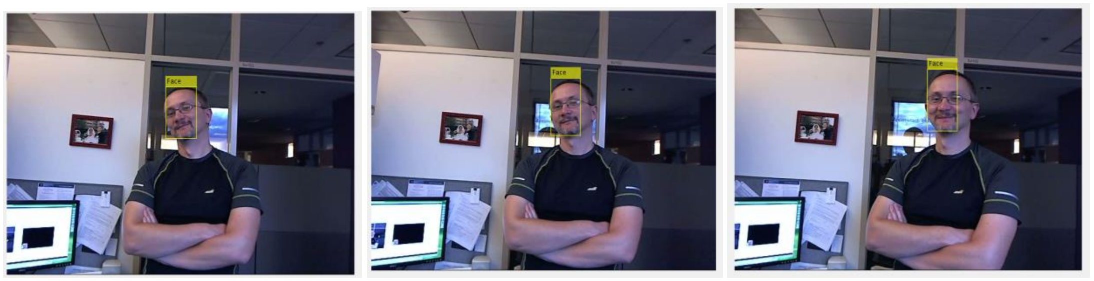
Step 2: Identify facial features to track - Skin (HUE Channel)[4]

MATLAB Sample Image:

Reading the HUE channel of the image and selecting skin as feature



Step 3: Track the face[4]



Note: All images taken from code implemented in MATLAB by MathWorks[4]

<u>Experiment 2 : Check if the CAMSHIFT algorithm correctly tracks a face with different skin tone.</u>

Sample Input:

The sample input is a video of a person (author) having a different skin tone.

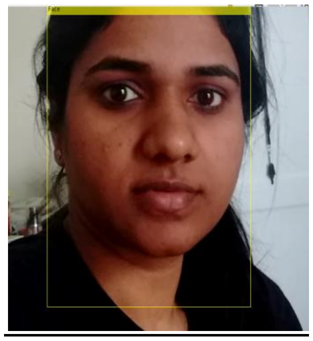
File name: 'Different_SkinTone.mp4'.

Observations:

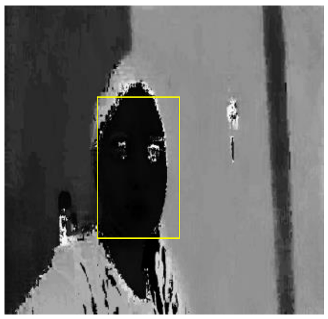
In this video sample, CAMSHIFT successfully tracks the human face. The skin tone of the person does not make a difference in successful tracking. As we are taking the HUE channel, the color or "hue" of skin in both cases is same, the difference only lies in the Saturation channel. The lighting conditions are good and the Viola Jones face detector detects the face correctly using the HUE channel of the face. The human moves left to right and there is also a slight tilt in the head. The search window successfully tracks the face of the person continuously.

Results 2:

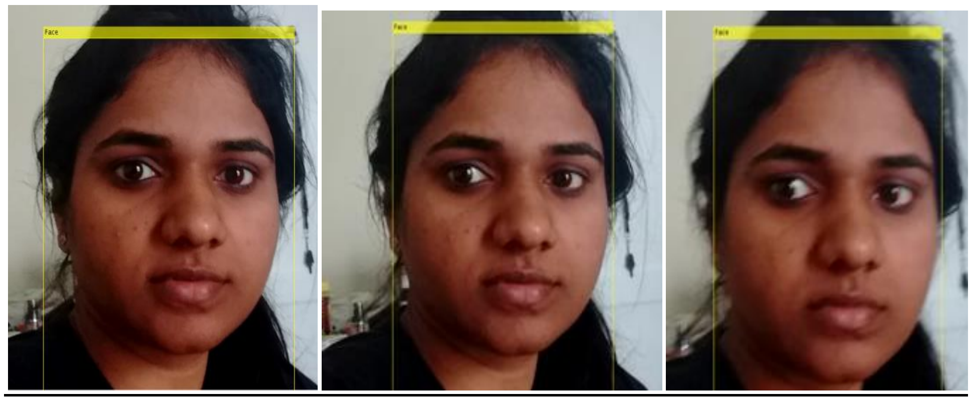
Step 1: Detect a Face To Track



Step 2: Identify facial features to track - Skin (HUE channel)[4]



Step 3: Track the face using CAMSHIFT Algorithm - Different skin tone[4]



Experiment 3 : Check if the CAMSHIFT algorithm correctly tracks a face in the presence of occlusion

Sample Input:

The sample input is a video of a person (author) having a different skin tone. The person brings her hand towards the face to cause an occlusion in the video input sample.

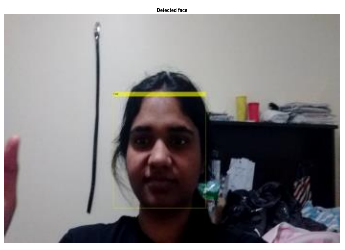
File Name: 'Hand_occlusion.mp4'.

Observations:

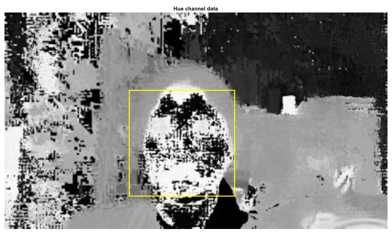
In this video sample, CAMSHIFT successfully tracks the human face in the beginning when there is no presence of hand in the video. Even after the hand comes in the video image frame, CAMSHIFT successfully tracks the face. However, with continuous movement of the hand, after some time, CAMSHIFT tracks the hand of the person in the image instead of tracking the face.

<u>Results 3:</u>

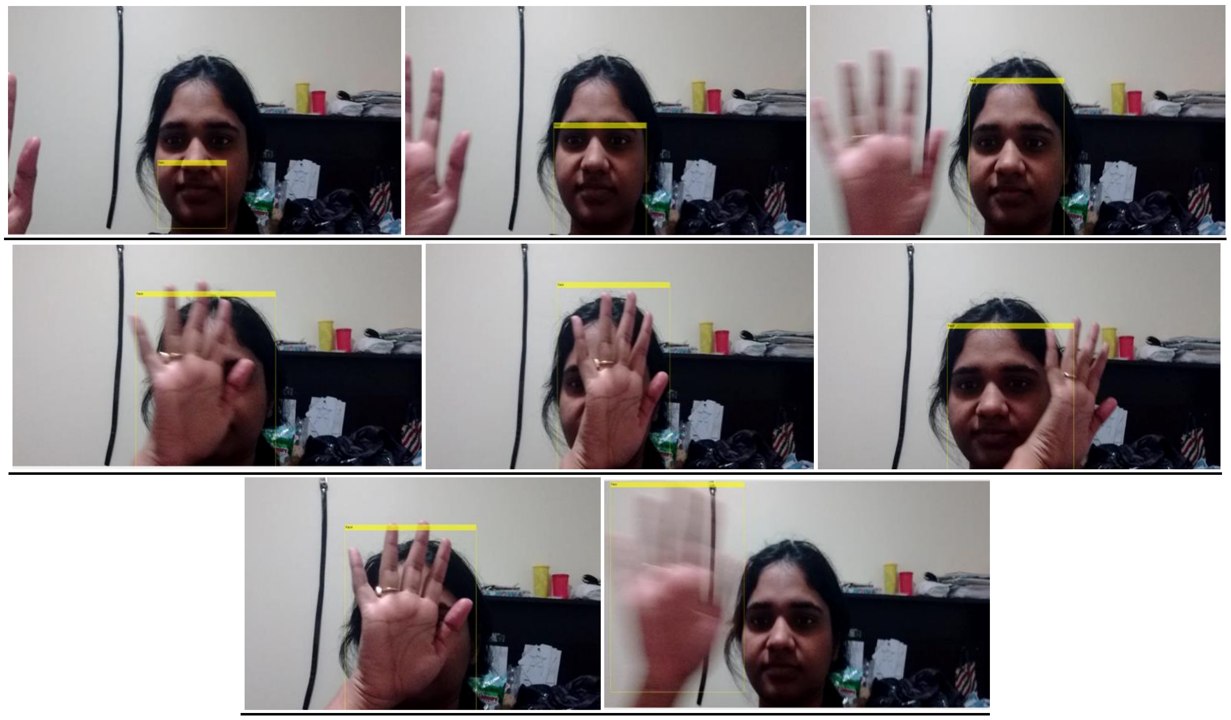
Step 1: Detect a Face To Track[4]



Step 2: Identify facial features to track - Skin (HUE channel)[4]



Step 3: Track the face using CAMSHIFT Algorithm - CAMSHIFT fails under occlusion[4]



Experiment 4 : Check if the CAMSHIFT algorithm correctly tracks a face of varying size as distance of face from camera changes:

Sample Input:

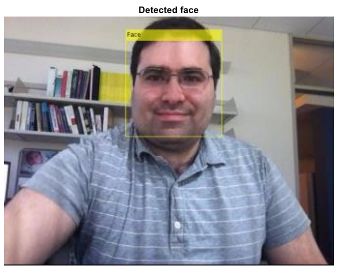
The sample input is the video image provided by MathWorks[4] in their vision toolbox called 'tilted_face.avi'.

Observations:

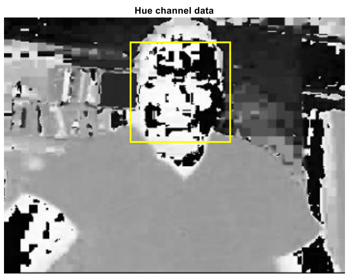
In this video sample, MATLAB successfully tracks the human face. The lighting conditions are good and the Viola Jones face detector detects the face correctly using the HUE channel of the face. The human moves left to right and front to back. Thus the size of the face changes, when the person moves back, face reduces in size and when the person moves towards the camera, the size of the face increases. The CAMSHIFT algorithm successfully tracks the face of the person. It continuously adapts the window size for the face as per the change in size of the face.

Results 4:

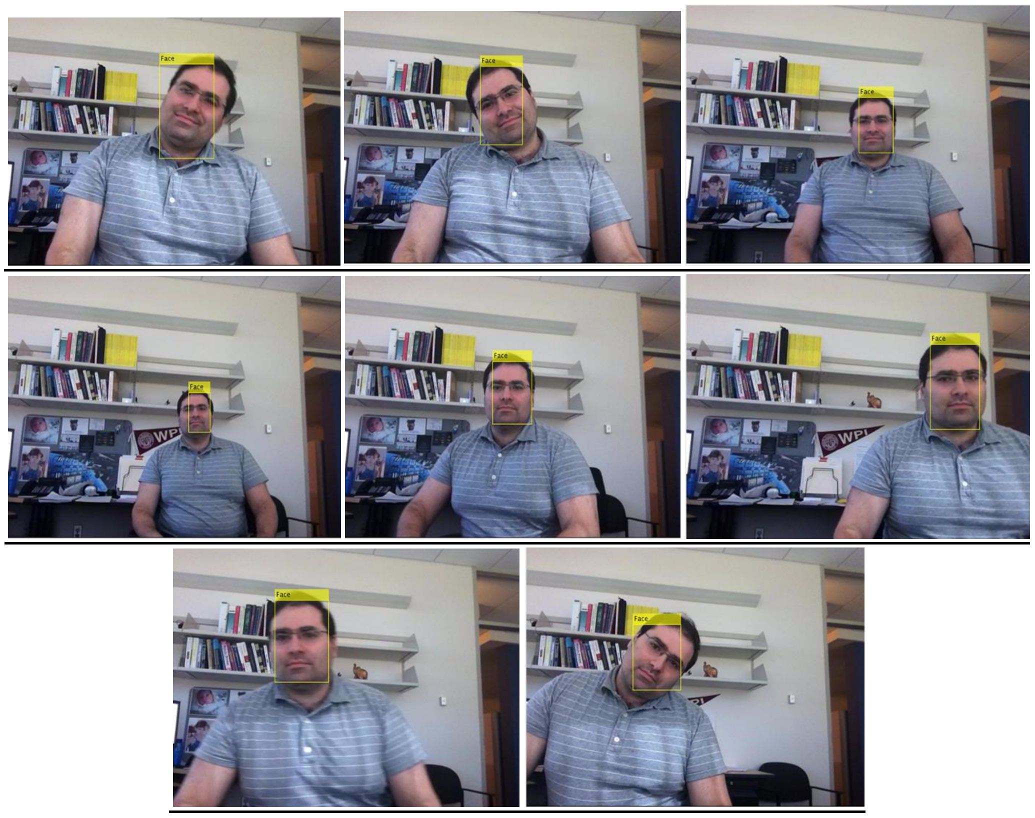
Step 1: Detect a Face To Track[4]



Step 2: Identify facial features to track - Skin (HUE channel)[4]



Step 3: Track the face using CAMSHIFT Algorithm[4]



DISCUSSIONS:

The CAMSHIFT Algorithm implemented by MATLAB tracks the face of the input video successfully in three of the four experiments conducted. Regardless of different skin tones, the CAMSHIFT algorithm successfully tracked the face. Thus, the skin tone did not make a difference to its tracking efficiency. This is because the CAMSHIFT algorithm takes the HUE channel of the HSV image and the selected tracking feature is the color of the flesh tones. The difference in skin tones is due to different levels of saturation in the flesh color. Thus, different skin tones do not affect the efficiency of CAMSHIFT Algorithm[2].

In the presence of occlusion (hand in Experiment 3), the CAMSHIFT algorithm still tracks the color probability distribution of the face. However, when the hand obstructs more than 70%, CAMSHIFT follows the hand and tracks it as the target object rather than the face itself. This is due to the fact that both the hand as well as the face have the same color probability distribution(the HUE level of both hand and face is same as it belongs to the same person). CAMSHIFT searches for the mean of the color probability distribution on the basis of the color histogram model as a look up table. Hence, it detects the same color probability distribution in the hand as well as the face (same HUE) and incorrectly tracks the hand and not the face(target object).[2]

Thus, CAMSHIFT fails in case 100% occlusion.[2]

The size of the face being tracked can change depending on its distance from the camera sensor. In experiment 4), the face size of the person in the video input reduces as he moves back and increases in size as he moves towards the camera sensor. CAMSHIFT algorithm successfully adapts the search window frame size as per the changes in the face size. Also, tilts in the face is also tracked correctly by CAMSHIFT. This is due to the head roll which CAMSHIFT calculates using the second image moments and adjusts the search window frame accordingly[2].

FUTURE WORK:

The main goal of this project was to understand the working of the CAMSHIFT algorithm. As CAMSHIFT is based on Mean Shift, this project also involved studying the Mean Shift algorithm.

In future, we can start with implementation of the preliminary steps of the CAMSHIFT algorithm. Like given a color histogram model of the target object to be tracked, we can create color probability distribution of incoming frame of the input video image. This requires detailed study of back projection to implement it correctly.

Once the back projection is done for one incoming input frame, we can go further ahead with implementing the Mean shift algorithm which involves finding the mean of the color probability distribution in every iteration until convergence and setting the search window frame to the centroid of this color probability distribution.

Once the correct location of the search window frame is fixed, we can use the area under the color probability distribution curve (zeroth moment) to calculate the size of the new search window for the next incoming video frame.

The above steps are done repeatedly to keep a continuous track of the moving target object.

In this way, we can go step by step in implementing the CAMSHIFT algorithm.

CONCLUSION:

The main motive of this project was to study and understand the working of the CAMSHIFT algorithm to track objects based on their color probability distribution.

The CAMSHIFT algorithm is based on various basic principles used in Computer Vision. The CAMSHIFT algorithm was proposed by Gary R. Bradski[2] as a part of a larger perceptual user interface. Thus, the main need of this face tracking algorithm was it to be a simple and computationally efficient algorithm. Computationally heavy face tracking algorithms require a large part of computational resources due to which the main purpose of a perceptual user interface was hindered.

Thus, a tracking algorithm based on the concept of Mean Shift was introduced. The CAMSHIFT is Continuously Adaptive Mean Shift Algorithm. The Mean Shift Algorithm is a "robust nonparametric technique for climbing density gradients to find the mode (peak) of probability distributions " [2]. It is based on the concept of color histograms, back projection and image moments. Thus, this project involved learning color histograms and creation of color probability distribution image (back-projection) based on these color histograms.

In the CAMSHIFT algorithm, the target object in a moving image is tracked using a search window. The location of the search window is recalculated in every iteration of the Mean Shift Algorithm. This new location is the centroid of the probability distribution of every input image frame and is calculated using the zeroth moment[2]. Once the mean shift algorithm converges to a fixed location of the search window, the CAMSHIFT algorithm adjusts the size of the search window based on calculations derived from the zeroth and higher order moments[2]. The output of this process is the input for the next incoming image frame in the video. Thus, this project involved the study of image moments and their understanding conceptually and mathematically.

This project helped in understanding the concepts and working of HSV color space, color histograms, back projection, image moments and derivation of the search window location and its size from these image moments.

Furthermore, the advantages of CAMSHIFT algorithm over other face-tracking algorithm lies in its simplicity and computational efficiency. The main reason for introducing the CAMSHIFT algorithm was its use in a larger Perceptual User Interface. Thus, this project helped in understanding the concept of a Perceptual User Interface and the application of Computer Vision in it.

BIBLIOGRAPHY:

1. http://www.cs.man.ac.uk/~tmorris/IP6.html Downloaded Time 1:33pm, 6th April'16

2. G. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," Intel Technology Journal, httu://developer.intel.com/technolog.v/iti/1a929 8iarticledart 2.htm, Q2, 1998.

3. Y. Cheng, "Mean shift, mode seeking, and clustering," IEEE Trans. Pattern Anal. Machine Intel., 17:790-799, 1995.

4. MathWorks, (MATLAB 2016a), Computer Vision System toolbox(TM), The MathWorks, Inc., Natick, Massachusetts, United States. URL: http://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-camshift.html?refresh=true Downloaded Time 8:50pm, 6th April'16

5. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces John G. Allen, Richard Y. D. Xu, Jesse S. Jin School of Information Technologies University of Sydney Madsen Building F09, University of Sydney, NSW 2006

6. http://docs.opencv.org/3.1.0/db/df8/tutorial_py_meanshift.html#gsc.tab=0 Downloaded Time 1:33pm, 6th April'16

7. https://saravananthirumuruganathan.wordpress.com/2010/04/01/introduction-to-mean-shift-algorithm/ Downloaded Time 1:33pm, 6th April'16

8. Real Time Face and Object Tracking as a Component of a Perceptual User Interface, Gary R. Bradski, Intel Corporation, Microcomputer Research Lab. gary.bradski@intel.com

9. "Fast and Robust CAMShift Tracking" by David Exner, Erich Bruns, Daniel Kurz, and Anselm Grundhofer from Bauhaus-University Weimar, Germany, Oliver Bimber, Johannes Kepler University Linz, Austria

10. "The Research and Implementation of Camshift Algorithm Based on Automatically Target Extraction" by Yang Li, First Author Information Engineering College, Capital Normal University, Hui Ding, Corresponding Author Information Engineering College, Capital Normal University, Shudong Zhang and Yilei Wang from Information Engineering College, Capital Normal University.

11. A. Pentland, B. Moghaddam, T. Starner, "View-based and Modular Eigenspaces for face recognition," CVPR'94, pp. 84-91, 1994.

12. http://eric-yuan.me/continuously-adaptive-shift/ Downloaded 29th April 2016, 7:30pm

13. http://www.mukimuki.fr/flashblog/2009/05/20/magic-moments/ Downloaded 29th April 2016, 6:30pm

14. http://aishack.in/tutorials/image-moments/ Downloaded 29th April 2016, 6:30pm

15. http://www.foundationsmag.com/persistence_of_vision.html Downloaded 2nd May 2016, 6:37pm

16. Fundamentals of Digital Image Processing - A Practical Approach with Examples in Matlab by Chris Solomon,School of Physical Sciences, University of Kent, Canterbury, UK, Toby Breckon School of Engineering, Cranfield University, Bedfordshire, UK

17. Viola, Paul A. and Jones, Michael J. "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.Viola, Paul A. and Jones, Michael J. "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.

18. http://sociograph.blogspot.com/2011/11/accessible-introduction-to-mean-shift.html Downloaded 1st May 2016, 11:09am

19. https://www.kairos.com/blog/inspiring-uses-of-facial-recognition-in-the-real-world

Downloaded 17th March 2016, 5:30 pm