INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(Deemed University)

(A Centre of Excellence in Information Technology Established by Ministry of HRD, Govt. of India)

<u>Project Report</u>

REAL TIME OBJECT SURVEILLANCE IN SPATIAL DOMAIN USING COMPUTER VISION

Project Supervisor:

Mr. Neetish Purohit

Submitted by:

Dushyant Singh (IEC2011019)

Nikhil Pipal (IEC2011020)

Ayush Dobhal (IEC2011081)

Students Declaration

We, hereby declare that the work presented in this project report entitled "**Real Time Object Surveillance in Spatial Domain using Computer Vision**" submitted towards compilation of the 6th semester project of B.Tech (Electronics and Comm. Engg) at the Indian Institute of Information Technology Allahabad, is an authentic record of our original work carried out under the guidance of Dr. Neetish Purohit. The project has been done in full compliance with the requirements and constraints of prescribed curriculum to the best of our knowledge.

Dushyant Singh (IEC2011019)
Nikhil Pipal (IEC2011020)
Ayush Dobhal (IEC2011081)

Certificate

This is to certify that the project work entitled "**Real Time Object Surveillance in Spatial Domain using Computer Vision**" has been carried out for the compilation of 6th semester project by B.Tech (Electronics and Comm. Engg.) students namely Dushyant Singh (IEC2011019), Nikhil Pipal (IEC2011020) and Ayush Dobhal (IEC2011081) at the Indian Institute of Information Technology Allahabad. The above declaration made by the candidates is correct to the best of my knowledge and belief.

Dr.Neetish Purohit
(Associate Professor)

Acknowledgement

Our project titled as “**Real Time Object Surveillance in Spatial Domain using Computer Vision**” would not have been completed if not for the help and encouragement that we got from our mentor Dr. Neetish Purohit. We would like to sincerely thank you for the support and the proper guidelines that we got regarding this project. This genre was completely new for us but with his intense knowledge and constant pushing us to work hard, we could finally complete the project with contention. We thoroughly gained knowledge through the course of this semester. We are thereby submitting this report to you with full honesty and integrity.

Abstract

Real Time Object Surveillance in Spatial Domain using Computer Vision

Object detection and segmentation is the most important and challenging fundamental task of computer vision. It is a critical part in many applications such as image search, image auto-annotation and scene understanding. However it is still an open problem due to the complexity of object classes and images.

The easiest way to detect and segment an object from an image is the color based methods. The colors in the object and the background should have a significant color difference in order to segment objects successfully using color based methods. The objective of this project is to design a hybrid intelligent surveillance system. Due to its programming flexibility, easy availability and user friendly, OpenCV has been used by many researchers to implement object tracking. A tracking system is uttered to be stable if it is robust and accurate. Robust tracking is e.g. when the system is resistant against noise, or partly occlusion, while the accuracy deals with how well the original object is tracked throughout the sequence.

Contents

	Page No.
1. Literature Survey	7
2. Problem Statement and Motivation	8
3. Introduction	9
4. Proposed Framework	10-12
5. Block diagram	13
6. Software Specifications	17
7. Design Specifications	18
8. Activity Time Chart	19-20
9. Schematic	21-24
10. Output	25-27
11. Code	28-35
12. References	36

Literature Survey

Objects are detected in every frame by an object detector. These objects are represented by points, and the association of detected points with the detections in the previous frame is based on the previous object state which can include object position and motion. The problem of association of points is called the point correspondence problem. Deterministic methods for point correspondence define a cost of associating each object in frame $t - 1$ to a single object in frame t using a set of motion constraints. Minimization of the correspondence cost is formulated as a combinatorial optimization problem. Given a cost matrix algorithms like Hungarian algorithm [8] exist to get an optimal correspondence.

Statistical methods model the object properties such as position, velocity etc. as states with some probability. The pdf's of these states are estimated using measurements taken in each frame, which are the object detections obtained by a detection mechanism and by choosing an appropriate motion model to describe the motion of object. Methods like Kalman filtering and particle filtering [4] are used to estimate the state of the object at each time instant, using these measurements.

More recently a different approach for tracking of multiple objects through extended occlusions has emerged. It involves using short-term trackers to track objects until they get occluded or the tracker accuracy drops below a threshold [5]. This short term track is called a tracklet. In the end all these tracklets are linked with each other minimizing a cost function associated with the linking of different tracklets with each other. This is a normal correspondence problem, which can be solved by defining a cost matrix and using Hungarian Algorithm. [6,7] extend the approach used by [5] by allowing a single tracklet to contain multiple objects. They do this by producing a hypothesis of two tracklets which are close in time and space, merging into one tracklet. Also, the cost of linking two tracklets, depends upon the similarity of the appearances of the object at the end of the first tracklet and the start of the second tracklet. If dissimilarity is high then the two tracklets would be perceived as two different objects. In real time videos with unrestricted illumination and pose changes of objects, this possibility is very high.

Kernel is refers to an object region of primitive shape. Tracking is done by computing the motion of this region, which can be translational, affine or projective, from one frame to the next. These algorithms differ by the appearance representation used and the method used to estimate the object motion. Templates and density-based appearance models (histograms) have been widely used because of their simplicity and low computational cost. Comaniciu and Meer [8] use a weighted histogram computed from a circular region to represent the object. Instead of performing a brute force search for locating the object, they use the mean-shift procedure. Extending optical flow methods to compute the translation of a rectangular region inn 1994, Shi and Tomasi [9] proposed the KLT tracker which iteratively computes the translation of a region centered on an interest point. Tao et al. [10] propose an object tracking method for multiple objects based on modelling the whole image as a set of layers. This includes a single background layer and one layer for each object. Each layer consists of shape parameters, motion parameters and appearance parameters, which are continuously, estimated using each other, in an EM algorithm. Mittal and Larry [11] propose a multiple person tracker by matching segmentations of the persons, across multiple views.

Problem Statement and Motivation

Object tracking in real time is one of the most important topics in the field of computer Vision. Detection and tracking of moving objects in the video scenes is the first relevant step in the information extraction in many computer vision applications. This idea can be used for the surveillance purpose, video annotation, traffic monitoring, human-computer interaction, intelligent transportation, and robotics and also in the field of medical. Object tracking can be defined as the process of segmenting an object of interest from a video scene and keeping track of its motion, orientation, occlusion etc. in order to extract useful information. Real-time object detection and tracking is a critical tasks in many computer vision applications such as surveillance, driver assistance, gesture recognition and man machine interface. Here, color object tracking is done using OpenCV library.

Introduction

The goal of the object tracking is to track the position of objects in real-time. It is a critical part in many applications such as image search, image auto-annotation and scene understanding. However it is still an open problem due to the complexity of object classes and images. Unlike a multimedia system, the output of a surveillance system is not always an image
sequence, but could also be the positions of objects image content features and so on. Object tracking can be defined as the process of segmenting an object of interest from a video scene and keeping track of its motion, orientation, occlusion etc. in order to extract useful information. Visual tracking is as the name implies tracking of objects using a camera. Using a camera for object tracking changes the perception from world coordinates to camera coordinates. This means that in the projection from camera, the dimensions is reduced from 3D to 2D. The tracker is made to run in the image, on the 2D projected data of the 3D, and it is thus important have knowledge about the motion models of the object moving in the 3D environment.
The objective of video tracking is to associate target objects in consecutive video frames.
Typically this process involves four stages, namely,

• Object detection: Detecting and localizing a moving object in a video. Typically this is done through background subtraction to identify the moving objects.

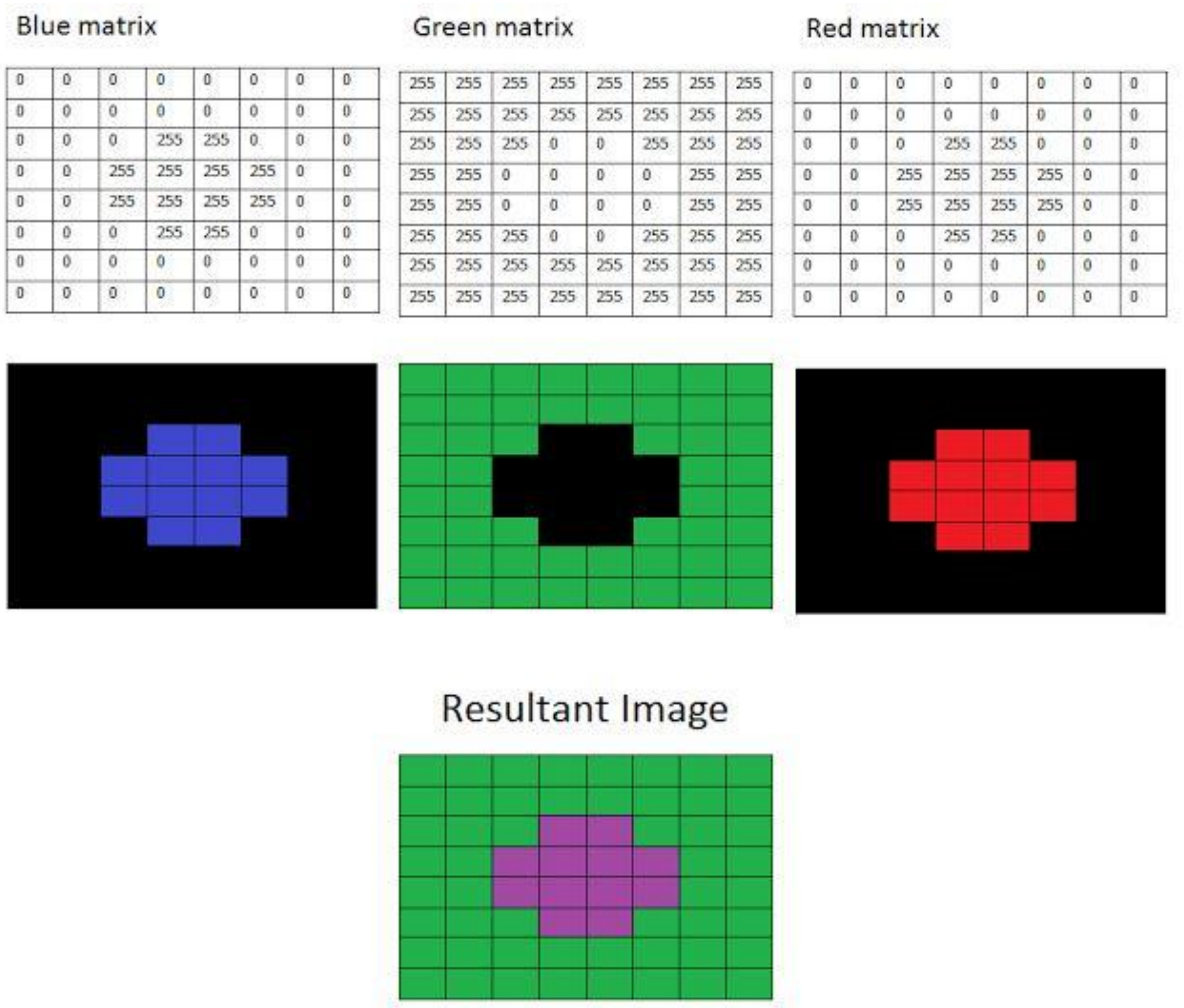
• Object representation: Description of the object in a video frame. For example, an object can be represented by simply its centroid or by simple geometrical shapes like rectangle and ellipse etc. marked around it, or by either its contour or silhouette.

• Feature Selection: After choosing a suitable object representation, a set of features suitable for the application like edges, color histograms, optical flow etc. are extracted over the region of the object. Choice of features has always been governed by the underlying application domain.

• Tracking the features: The chosen features are finally tracked through different methods which can be broadly divided into point tracking, kernel tracking and silhouette tracking. The method of tracking depends on the features chosen.

Proposed Project Framework

The colors in the object and the background should have a significant color difference in order to segment objects successfully using color based methods. OpenCV usually captures images and videos in 8-bit, unsigned integer, BGR format. In other words, captured images can be considered as 3 matrices, BLUE, RED and GREEN with integer values ranges from 0 to 255.



Blue matrix

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Green matrix

255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	0	0	255	255	255
255	255	0	0	0	0	255	255
255	255	0	0	0	0	255	255
255	255	255	0	0	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

Red matrix

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	255	255	0	0	0
0	0	255	255	255	255	0	0
0	0	255	255	255	255	0	0
0	0	0	255	255	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Fig 1 : BGR planes

HSV color space is more often used in machine vision owing to its superior performance compared to RGB color space in varying illumination levels. Often thresholding and masking is done in HSV color space. So it is very important to know the HSV values of the color which we want to filter out. Usually, one can think that BGR color space is more suitable for color based segmentation. But HSV color space is the most suitable color space for color based image segmentation. So, in the above application, we have converted the color space of original image of the video from BGR to HSV image.

HSV color space is consists of 3 matrices, 'hue', 'saturation' and 'value'. In OpenCV, value range for 'hue', 'saturation' and 'value' are respectively 0-179, 0-255 and 0-255. 'Hue' represents the color, 'saturation' represents the amount to which that respective color is mixed with white and 'value' represents the amount to which that respective color is mixed with black.

Hue values of basic colors

- Orange 0-22
- Yellow 22- 38
- Green 38-75
- Blue 75-130
- Violet 130-160
- Red 160-179

Calculating Position of Centre of Object

To calculate the position of the center of the object. We have to calculate 1st order spacial moments around x-axis and y-axis and the 0th order central moments of the binary image.

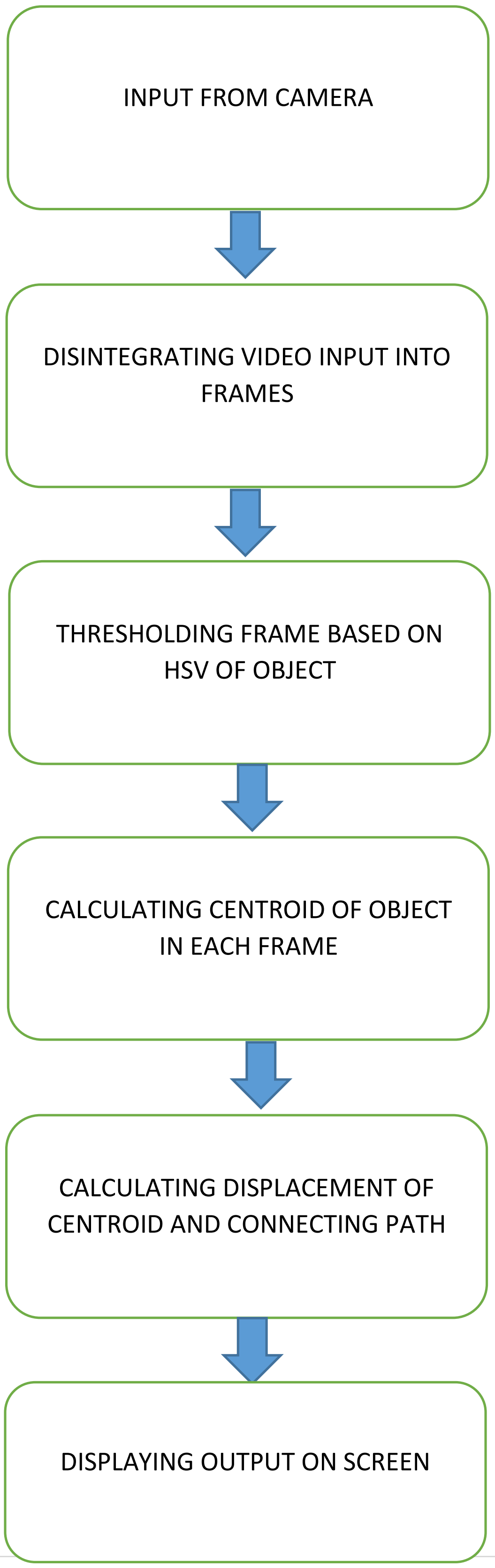
0th order central moments of the binary image is equal to the white area of the image in pixels.

- X coordinate of the position of the center of the object = 1st order spacial moment around x-axis / 0th order central moment
- Y coordinate of the position of the center of the object = 1st order spacial moment around y-axis / 0th order central moment

If there are 2 or more objects in the image, we cannot use this method. And noise of the binary image is also should be at minimum level to get accurate results.

In the above application, we considered that if the white area of the binary image is less than 1000 in pixels, there is no objects in the image because there may be noisy white areas that may have been identified as the object. Usually object always have an area more than 1000 pixels.

Process Flow Chart


INPUT FROM CAMERA
DISINTEGRATING VIDEO INPUT INTO FRAMES
THRESHOLDING FRAME BASED ON HSV OF OBJECT
CALCULATING CENTROID OF OBJECT IN EACH FRAME
CALCULATING DISPLACEMENT OF CENTROID AND CONNECTING PATH
DISPLAYING OUTPUT ON SCREEN

Methods Used

- **cvInRangeS(const CvArr* *src*, CvScalar *lower*, CvScalar *upper*, CvArr* *dst*)**

  Checks that each array element of 'src' lies between 'lower' and 'upper'. If so, array element in the relevant location of 'dst' is assigned '255' , otherwise '0'.

- **cvMoments(const CvArr* arr, CvMoments* moments, int isBinary=0)**

  Calculates all of the spacial and central moments up to the third order

- **cvGetSpatialMoment(CvMoments* *ptr*, int *x_order*, int *y_order*)**

  Retrieve calculated spacial moments from ptr memory block

- **cvGetCentralMoment (CvMoments* *ptr*, int *x_order*, int *y_order*)**

  Retrieve calculated central moments from ptr memory block

- **cvLine (CvArr* *img*, CvPoint *pt1*, CvPoint *pt2*, CvScalar *color*, int *thickness=1*)**

  Draw a line between 2 points, pt1 and pt2

Software Specifications

OpenCV library:
OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. The algorithms available through
OpenCV are mainly written in C and C++ and runs under Windows, Linux and Mac OS X.
Computer vision is a rapidly growing field, partly as a result of both cheaper and more capable cameras, partly because of affordable processing power, and partly because vision algorithms are starting to mature. OpenCV itself has played a role in the growth of computer vision by enabling thousands of people to do more productive work in vision.

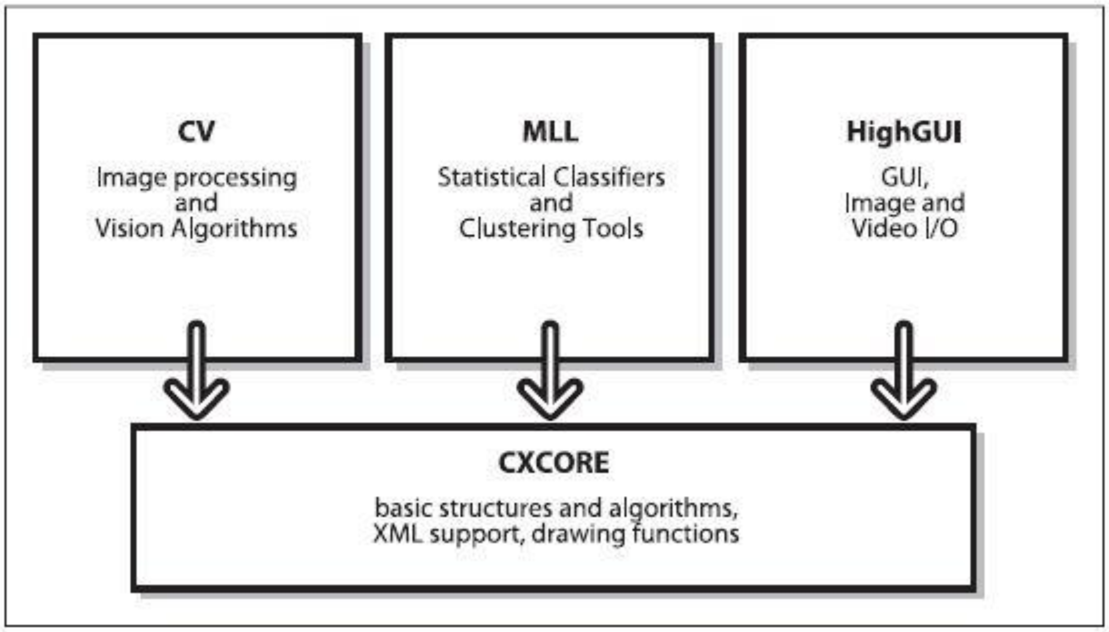

CV
Image processing and Vision Algorithms
MLL
Statistical Classifiers and Clustering Tools
HighGUI
GUI, Image and Video I/O
CXCORE
basic structures and algorithms, XML support, drawing functions

Fig 2: The basic structure of OpenCV

Activity Time Chart

Weeks	**Task**	**Status**
Week 1	Configuring and Installing OpenCV.	Completed
Week 2	Interfacing Camera and learning OpenCV library.	Completed
Week 3	Taking Video input from interfaced camera and processing the video frame by frame.	Completed
Week 4	Detecting a particular using HSV values.	Completed
Week 5	Tracking an object using HSV values and moments.	Completed

Output

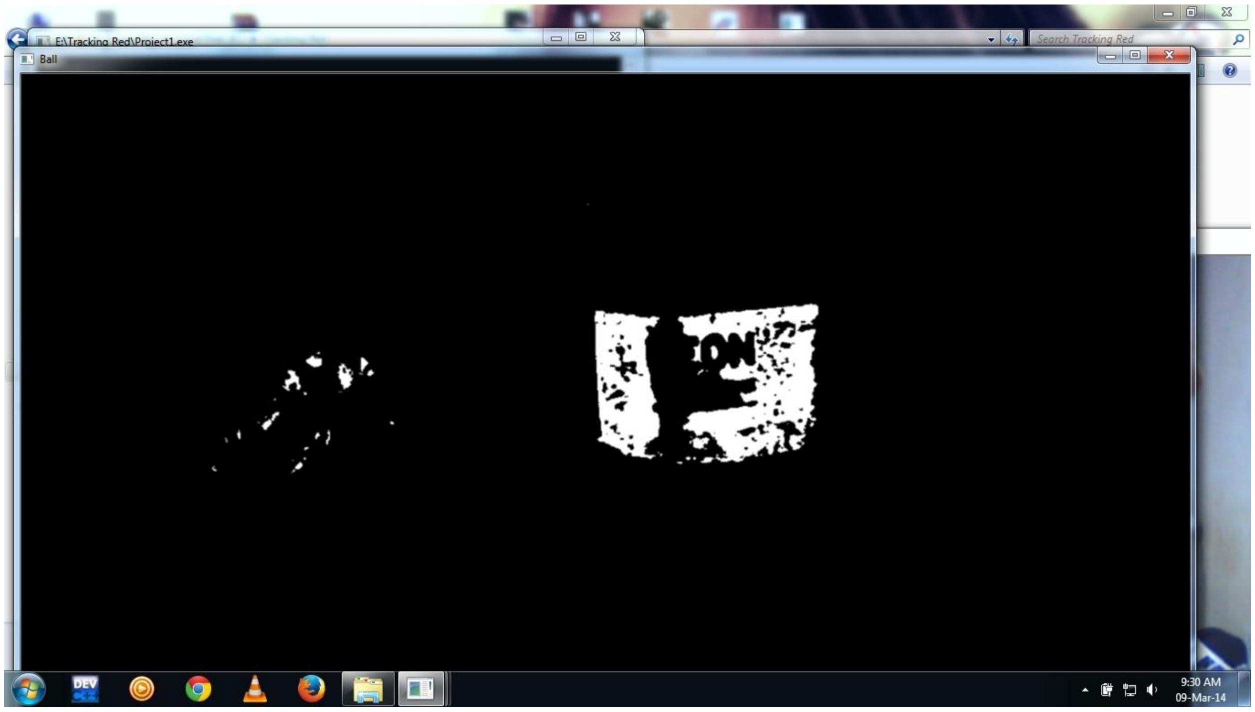


Fig 3: Thresholding red colored object

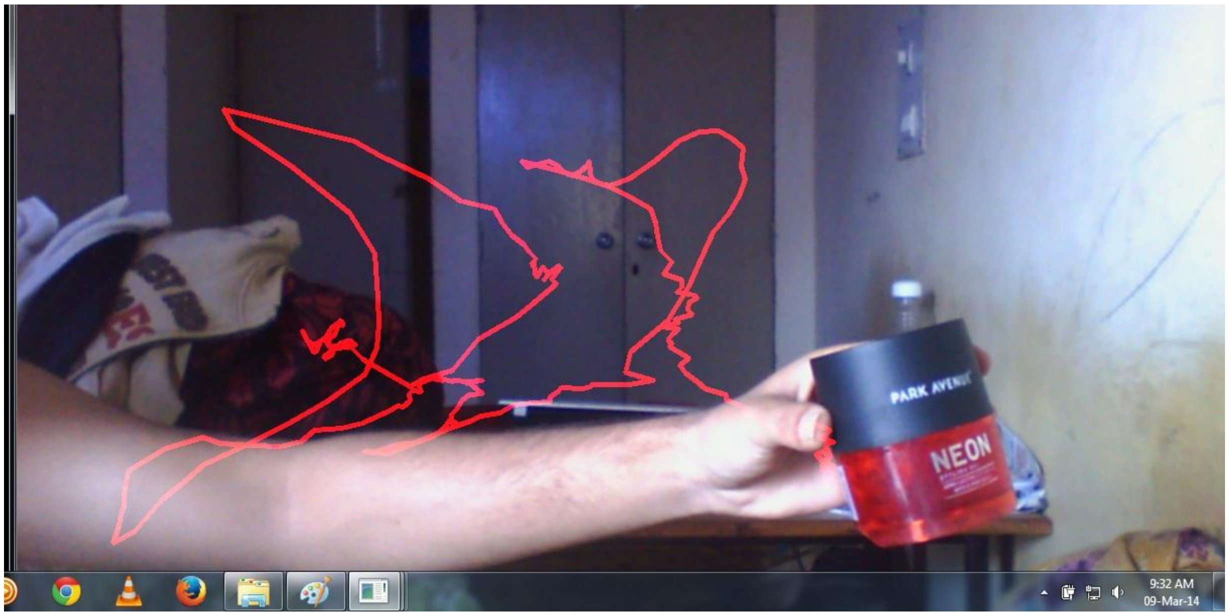


Fig 4: Tracking Red colored object

Future Aspects

More robust techniques can be used which are invariant to angular displacement and scalability and shape invariant. Detection and tracking can be used in various applications like

• Motion-based recognition, i.e. identifying humans based on gait, automatic object detection, etc.

• Automated surveillance, i.e. monitoring a scene to detect suspicious activities or unlikely events automatically, tracking a suspicious person over a long period of time automatically, reducing the amount of manual labor required in surveillance.

• Video indexing, which involves annotating the videos based on the type of the motion of the objects in it and retrieving the videos using a similar analysis on the videos.

• Human-computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc. This is used in systems like SixthSense [22] to provide a more intuitive interface to computing.

• Traffic monitoring, that is, real-time gathering of traffic statistics to direct traffic flow.

• Vehicle navigation, that is, video-based path planning and obstacle avoidance capabilities

References

1. **OpenCV Tutorials** - based on "Learning OpenCV - Computer Vision with the OpenCV Library" by Gary Bradski and Adrian Kaehler http://www.pages.drexel.edu/~nk752/tutorials.html
2. **Oreilly Learning OpenCV -** Gary Bradski and Adrian Kaehler
3. **HighGUI Reference Manual** - http://opencv.jp/opencv1.0.0_org/docs/ref/opencvref_highgui.htm
4. Streit, R. L. and luginbuhl, T. E. Maximum likelihood method for probabilistic multi-hypothesis tracking. In Proceedings of the International Society for Optical Engineering vol. 2235. 394–405 1994.
5. R. Kaucic, A. G. A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. IEEE Conference on Computer Vision and Pattern Recognition 2005, pp. 990-997 .
6. A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-Object tracking through simultaneous long occlusions and Split-Merge conditions. IEEE Conference on Computer Vision and Pattern Recognition 2006, pp. 666-673.
7. João F. Henriques, Rui Caseiro and Jorge Batista. Globally Optimal Solution to Multi-Object Tracking with Merged Measurements. IEEE International Conference on Computer Vision 2011, pp. 2470-2477
8. http://en.wikipedia.org/wiki/Hungarian_algorithm
9. Comaniciu, D. and Meer, P. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Patt. Analy. Mach. Intell. 24, 5, pp. 603–619, 2002.
10. Shi, J. and Tomasi, C. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 593–600, 1994.
11. Tao, H., Sawhney, H., and Kumar, R. 2002. Object tracking with bayesian estimation of dynamic layer representations. IEEE Trans. Patt. Analy. Mach. Intell. 24, 1, pp. 75–89.
12. Mittal, A., Davis, L.S. : “M2 tracker: A multi-view approach to segmenting and tracking people in a cluttered scene”. International Journal of Computer Vision 51 (3) , pp. 189-203

CODE

```
#include <cv.h>
#include <highgui.h>

IplImage* imgTracking;
int lastX = -1;
int lastY = -1;

//This function threshold the HSV image and create a binary image
IplImage* GetThresholdedImage(IplImage* imgHSV){
IplImage* imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
cvInRangeS(imgHSV, cvScalar(170,160,60), cvScalar(180,2556,256), imgThresh);
return imgThresh;
}

void trackObject(IplImage* imgThresh){
// Calculate the moments of 'imgThresh'
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgThresh, moments, 1);
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
double area = cvGetCentralMoment(moments, 0, 0);

// if the area<1000, I consider that the there are no object in the image and it's because of the noise,
the area is not zero
If (area > 1000){
// calculate the position of the ball
int posX = moment10/area;
int posY = moment01/area;

if(lastX>=0 && lastY>=0 && posX>=0 && posY>=0)
{
// Draw a yellow line from the previous point to the current point
cvLine(imgTracking, cvPoint(posX, posY), cvPoint(lastX, lastY), cvScalar(0,0,255), 4);
}

lastX = posX;
lastY = posY;
}

free(moments);
```

```
}


int main(){

CvCapture* capture =0;
capture = cvCaptureFromCAM(0);
if(!capture){
printf("Capture failure\n");
return -1;
}

IplImage* frame=0;
frame = cvQueryFrame(capture);
if(!frame) return -1;

//create a blank image and assigned to 'imgTracking' which has the same size of original video
imgTracking=cvCreateImage(cvGetSize(frame),IPL_DEPTH_8U, 3);
cvZero(imgTracking); //covert the image, 'imgTracking' to black

cvNamedWindow("Video");
cvNamedWindow("Ball");

//iterate through each frames of the video
while(true){

frame = cvQueryFrame(capture);
if(!frame) break;
frame=cvCloneImage(frame);

cvSmooth(frame, frame, CV_GAUSSIAN,3,3); //smooth the original image using Gaussian kernel

IplImage* imgHSV = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 3);
cvCvtColor(frame, imgHSV, CV_BGR2HSV); //Change the color format from BGR to HSV
IplImage* imgThresh = GetThresholdedImage(imgHSV);

cvSmooth(imgThresh, imgThresh, CV_GAUSSIAN,3,3); //smooth the binary image using Gaussian
kernel

//track the possition of the ball
trackObject(imgThresh);
// Add the tracking image and the frame
cvAdd(frame, imgTracking, frame);

cvShowImage("Ball", imgThresh);
cvShowImage("Video", frame);

//Clean up used images
cvReleaseImage(&imgHSV);
cvReleaseImage(&imgThresh);
cvReleaseImage(&frame);

//Wait 10mS
int c = cvWaitKey(10);
//If 'ESC' is pressed, break the loop
if((char)c==27 ) break;
}

cvDestroyAllWindows() ;
cvReleaseImage(&imgTracking);
cvReleaseCapture(&capture);

return 0;
}
```

Suggestion Of Board Members