OBJECT TRACKING USING OPENCV AND C#

B.E. Project Report

Submitted in partial fulfilment of the requirements

For the degree of

Bachelor of Engineering
in
Computer Engineering

by

Sachin Biradar 11CE1049

Sushant Alone 11CE1077

Vishal Anand 11CE1103

Dr. Leena Ragha
Supervisor

Prof. Harsha Saxena
Co-supervisor

Department of Computer Engineering

Ramrao Adik Institute Of Technology

Dr. D. Y. Patil Vidyanagar,Sector 7, Nerul, Navi Mumbai 400 706.
(Affiliated to University of Mumbai)

May 2015

Ramrao Adik Institute of Technology

(Affiliated to the University of Mumbai)

Dr. D. Y. Patil Vidyanagar,Sector 7, Nerul, Navi Mumbai 400 706.

CERTIFICATE

This is to certify that, the dissertation titled

"Object Tracking Using OpenCV and C#"

is a bona fide work done by

Sachin Biradar
Sushant Alone
Vishal Anand

and is submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Engineering
in
Computer Engineering
to the
University of Mumbai

Co-Supervisor Prof. Harsha Saxena	Supervisor Dr. Leena Ragha

Project Co-ordinator Prof. Namita Pulgam	Head of Department Dr. Leena Ragha	Principal Dr. Ramesh Vasappanavara

Project Report Approval for B.E.

This is to certify that the dissertation entitled "*Object Tracking Using OpenCV and C#*" is a bona fide work done by *Sachin Biradar, Sushant Alone and Vishal Anand* under the supervision of *Prof. Harsha Saxena*. This dissertation has been approved for the award of *Bachelor's Degree in Computer Engineering, University of Mumbai.*

Examiners :

1. . . . . .

2. . . . . .

Supervisors :

1. . . . . .

2. . . . . .

Principal :

. . . . . .

Date :

Place :

Declaration

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Sachin Biradar 11CE1049
Sushant Alone 11CE1077
Vishal Anand 11CE1103

Date :

ABSTRACT

This report proposes a project using OpenCV and C# to develop a software application that achieves efficient, dynamic and intelligent object tracking via a visual input device. The term “object” in this report loosely means any distinction from a static background, i.e., any place where there seems to be some sort of planar distortion or intersection between the surrounding background and the unexpected change, the object or the foreground.

The project proposed in the report is a software application with the target devices as the laptop and the desktop, with a minimum requirement of no more than a visual sensory device, such as a webcam or an externally connected digital camera. The user interface of the same has been developed using mainly C#, while the code for the internal processes has been written in both OpenCV and C#. The application successfully tracks one or more preset objects for any amount of time, and is extremely space efficient in that it only records the data when untoward or unexpected movement occurs within the frame. In order to do so, it creates a co-ordinate grid of the initial frame and continually compares the current frame to that initial optimal state frame. Hence the memory required is low, and there is found to be a serious improvement in tracking efficiency, since the task of storage is nulled.

INDEX

Index of Figures		
Fig No.	TOPIC	Page No.
3.1	How BGR image is formed	9
3.2	An overview of our proposed method for moving object (hand) detection	13
4.1	The object is tracked by finding the appropriate colour in the image as per the set HSV	16
4.2	Once the object is found, tracking begins, but only for as long as the object is still in the caught frame	17
4.3	The recorded video is stored in the avi format and can then be played back.	17
4.4	Object recognition by way of recognizing the contours and curvature of the object, if present in the preset settings.	18
5.1	Change in pixels in the frame is recorded	20

Chapter 1

Introduction

Real-time object tracking is the critical task in many computer vision applications such as surveillance, perceptual user interfaces, augmented reality, smart rooms, object-based video compression and driver assistance. Because of the advance in technology, there are more affordable digital video acquisition devices in the market. This means more applications for digital video. Having witnessed the success of web camera applications and the appearance of high definition digital video cameras, we believe that digital video will soon become a part of everyday life. Unlike still images, video sequences provide more information about how objects and scenarios change over time, but at the cost of increased space for storage and wider bandwidth for transmission. Therefore, the topic of video compression has drawn more and more attentions during recent years. Many existing algorithms segment each video frame to determine the objects; this action can be computationally expensive and is generally unnecessary if the goal is to track moving objects.

There have been developed several applications on these very algorithms that achieve the function of object tracking, however one using the relatively new computer graphics technology of OpenCV is yet missing. A technology with huge application, OpenCV has been praised by critics and programmers alike, due to its ease of use and powerful inbuilt methods.

The most common approach to track objects is to first detect them using background subtraction and, then, establish correspondence from frame to frame to find the tracks of the objects. This method, although simple, has been considered taboo by several graphics application developers because of its inefficiency in terms of time requires and memory space taken up. However, with the use of OpenCV, and by applying intelligent logic to each camera frame, background subtraction and reference frame comparison actually turn out to be extremely time and space efficient. Therefore, we believe the development of such a project will only lead to better application of existing technologies in a novel way, and will find enormous use in today's IT industry.

Chapter 2

Literature Survey

Visual object tracking is a significant computer vision task which can be applied to many domains such as visual surveillance, human computer interaction, and video compression. Despite extensive research on this topic, it still suffers from difficulties in handling complex object appearance changes caused by factors such as illumination variation, partial occlusion, shape deformation, and camera motion. Therefore, effective modelling of the 2D appearance of tracked objects is a key issue for the success of a visual tracker. In the following literature, researchers have proposed a variety of 2D object detection and tracking models.

A global visual representation reflects the global statistical characteristics of object appearance. Typically, it can be investigated in the following main aspects: (i) raw pixel representation; (ii) optical flow representation; (iii) histogram representation; (iv)covariance representation; (v) wavelet filtering-based representation; and (vi)active contour representation lists several representative tracking methods using global visual representations

—Raw pixel representation. As the most fundamental features in computer vision, raw pixel values are widely used in visual object tracking because of their simplicity and efficiency. Raw pixel representation directly utilizes the raw color or intensity values of the image pixels to represent the object regions. Such a representation is simple and efficient for fast object

tracking. In the literature, raw pixel representations are usually constructed in the following two forms: vector-based and matrix-based. The vector-based representation directly flattens an image region into a high-dimensional vector, and often suffers from a small-sample-size problem.

Motivated by attempting to alleviate the small-sample-size problem, the matrix-based representation directly utilizes 2D matrices or higher-order tensors as the basic data units for object description due to its relatively low-dimensional property. However, raw pixel information alone is not enough for robust visual object tracking. Researchers attempt to embed other visual cues (e.g., shape or texture) into the raw pixel representation. Typically, the color features are enriched by fusing other visual information such as edge.

—Optical flow representation: In principle, optical flow represents a dense field of misplacement vectors of each pixel inside an image region, and is commonly used to capture the spatio-temporal motion information of an object. Typically, optical flow has two branches: constant-brightness-constraint (CBC) optical flow and non-brightness-constraint (NBC) optical flow. The CBC optical flow has a constraint on brightness constancy while the NBC optical flow deals with the situations with varying lighting conditions.

—Histogram representation: Histogram representations are popular in visual object tracking because of their effectiveness and efficiency in capturing the distribution characteristics of visual features inside the object regions. In general, they have two branches: single-cue and multi-cue.

(i) A single-cue histogram representation often constructs a histogram to capture the distribution information inside an object region. For example, Bradski [1998] uses a color histogram in the Hue Saturation Value (HSV) color space for object representation, and then embeds the color histogram into a continuously adaptive mean shift (CAMSHIFT) framework for object tracking. However, the direct use of color histogram may result in the loss of spatial information.

(ii) A multi-cue histogram representation aims to encode more information to enhance the robustness of visual representation. Typically, it contains three main components: a) spatial-color; b) spatial-texture; c) shape-texture;

For a), two strategies are adopted, including joint spatial-color modeling and patch division. The goal of joint spatial-color modeling is to describe the distribution properties of object appearance in a joint spatial-color space (e.g., (x, y, R, G, B). The patch-division strategy is to encode the spatial information into the appearance models by splitting the tracking region into a set of patches. By considering the geometric relationship between patches, it is capable of capturing the spatial layout information.

For b), an estimate of the joint spatial-texture probability is made to capture the distribution information on object appearance.

For c), the shape or texture information on object appearance is incorporated into the histogram representation for robust visual object tracking.

Chapter 3

System Architecture

3.1 Problem Statement

The easiest way to detect and segment an object from an image is the colour-based methods. The object and the background should have a significant colour difference in order to successfully segment objects using colour based methods. Moving object detection is the basic step for further video analysis tasks. The performance of this step is particularly significant because subsequent processing of the video data is greatly dependent on this step. Moving object detection aims at extracting moving objects that are of interest in video sequences. Commonly used techniques for moving object detection are background subtraction, temporal frame differencing, and optical flow. Background subtraction, in particular, is a commonly used technique for foreground segmentation in static scenes because it has a very low computational cost.

The goal of background subtraction is to “remove” the background in a scene by describing an adequate model of the background, the result is that only “interesting objects” are left in the scene for tracking and further analysis. The problems with dynamic environmental conditions make moving object detection very challenging. Some examples of these are shadows, sudden or gradual illumination changes, and repetitive motion from scene clutter such as waving tree branches and leaves. The next step in the video analysis is object augmented reality, traffic control, medical imaging, gesture recognition, and video editing. Another way to look at object tracking is the creation of temporal correspondence

among detected object from frame to frame. The next step in the video analysis is object tracking, which can be considered as a hidden state estimation problem given available observations. Object tracking is an important component of many vision systems. It is used not only for visual surveillance, but also for augmented reality, traffic control, medical imaging, gesture recognition, and video editing. Another way to look at object tracking is the creation of temporal correspondence among detected object from frame to frame.

3.2 Technologies Used:

1. OpenCV:

OpenCV (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center. 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. OpenCV's application areas include: 2D and 3D feature toolkits, Egomotion estimation, Facial recognition system, Gesture recognition, Human–computer interaction (HCI), Mobile robotics, Motion understanding, Object identification, Segmentation and Recognition, Stereopsis (Stereo vision: depth perception from 2 cameras), Structure from motion (SFM), Motion tracking, Augmented reality.

To support some of the above areas, OpenCV includes a statistical machine learning library that contains: Boosting (meta-algorithm), Decision tree learning, Gradient boosting trees, Expectation-maximization algorithm, k-nearest neighbor algorithm, Naive Bayes classifier, Artificial neural networks, Random forest and Support vector machine (SVM).

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. OpenCV runs on Windows, Android, Maemo, FreeBSD, OpenBSD, iOS, BlackBerry 10, Linux and OS X. The user can get official releases from SourceForge, or take the current snapshot under SVN from there. OpenCV uses CMake.

2] C#:

C# is intended to be a simple, modern, general-purpose, object-oriented programming language. The most recent version is C# 5.0, which was released on August 15, 2012. Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or Fortran. Some notable features of C# that distinguish it from C and C++ (and Java, where noted) are:

- C# supports strongly typed implicit variable declarations with the keyword var, and implicitly typed arrays with the keyword new[] followed by a collection initializer.
- Meta programming via C# attributes is part of the language. Many of these attributes duplicate the functionality of GCC's and VisualC++'s platform-dependent preprocessor directives.

. 3.3 Moving object detection

– Foreground-background pixel classification using adaptive thresholding

– Adaptive pixel-wise background model update techniques

– A technique that is robust to sudden illumination changes using a illumination model and a statistical test

– A fast implementation of the methods using an extension of integral images

3.4 Object tracking

– Robust tracking with dynamic parameter setting for the likelihood model for particle filtering

– Investigation of the trade-off between object tracking performance and computational cost by subsampling the pixels for color density estimation

– A fast method for construction of the appearance model for particle filtering for object tracking

3.5 System Design:

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, GREEN and RED (hence the name BGR) with integer values ranges from 0 to 255. The following image shows how a colour image is represented using 3 matrices.

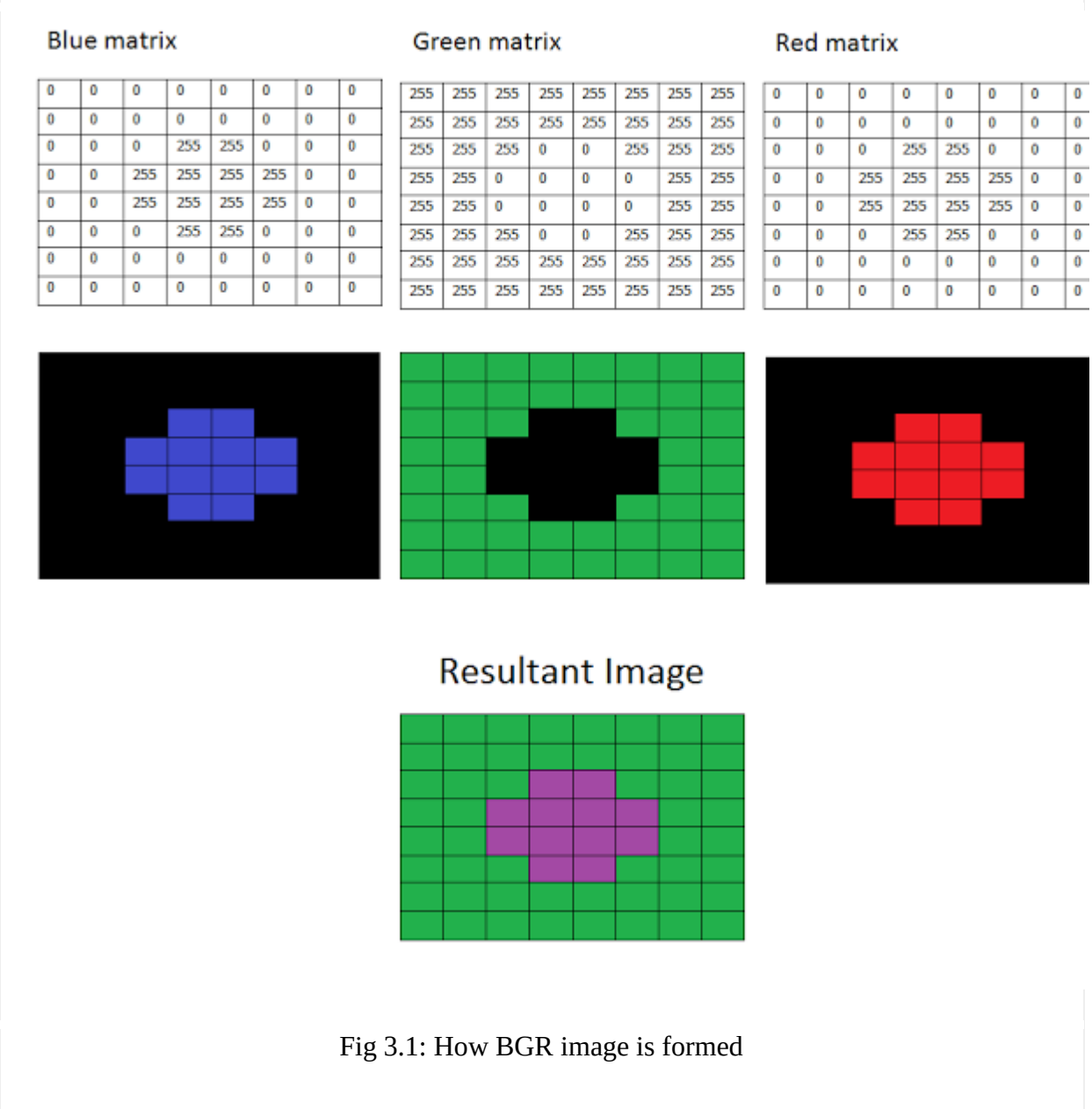

Fig 3.1: How BGR image is formed

In the above image, each small box represents a pixel of the image. In real images, these pixels are so small that human eye cannot differentiate. Usually, one can think that BGR

colour space is more suitable for colour based segmentation. But HSV colour space is the most suitable colour space for colour based image segmentation. So, in the above application, I have converted the colour space of original image of the video from BGR to HSV image.

HSV colour space is also 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 colour, SATURATION represents the amount to which that respective colour is mixed with white and VALUE represents the amount to which that respective colour is mixed with black.

In the developed application, we have considered that the red object as HUE, SATURATION and VALUE in between 170-180, 160-255, 60-255 respectively. Here the HUE is unique for that specific colour distribution of that object. But SATURATION and VALIE may vary according to the lighting condition of that environment.

Hue values of basic colours:

Orange 0-22

Yellow 22- 38

Green 38-75

Blue 75-130

Violet 130-160

Red 160-179

These are approximate values. You have to find the exact range of HUE values according to the colour of the object. I found that the range of 170-179 is perfect for the range of hue values of my object. The SATURATION and VALUE is dependent on the lighting condition of the environment as well as the surface of the object.

How to find the exact range of HUE, SATURATION and VALUE for an object is discussed in this post.

After thresholding the image, you'll see small white isolated objects here and there. It may be because of noises in the image or the actual small objects which have the same colour as our

main object. These unnecessary small white patches can be eliminated by applying morphological opening. Morphological opening can be achieved by a erosion, followed by the dilation with the same structuring element.

Thresholded image may also have small white holes in the main objects here and there. It may be because of noises in the image. These unnecessary small holes in the main object can be eliminated by applying morphological closing. Morphological closing can be achieved by a dilation, followed by the erosion with the same structuring element.

3.5.1 OpenCV Libraries:

- The function inRange() checks that each element of 'src' lies between 'lowerb' and 'upperb'. If so, that respective location of 'dst' is assigned '255', otherwise '0'. (Pixels with value 255 is shown as white whereas pixels with value 0 is shown as black).

  Arguments -

  InputArraysrc - Source image

  InputArraylowerb - Inclusive lower boundary (If lowerb=Scalar(x, y, z), pixels which have values lower than x, y and z for HUE, SATURATION and VALUE respectively is considered as black pixels in dst image)

  InputArrayupperb - Exclusive upper boundary (If it is upperb=Scalar(x, y, z), pixels which have values greater or equal than x, y and z for HUE, SATURATION and VALUE respectively is considered as black pixels in dst image)

  OutputArraydst - Destination image (should have the same size as the src image and should be 8-bit unsigned integer, CV_8U)

- The function erode the source image and stores the result in the destination image. In-place processing is supported (which means you can use the same variable for the source and destination image). If the source image is multi-channel, all channels are processed independently and the result is stored in the destination image as separate channels.

  Arguments -

InputArraysrc - Source image

OutputArraydst - Destination image (should have the same size and type as the source image)

InputArray kernel - Structuring element which is used to erode the source image

Point anchor - Position of the anchor within the kernel. If it is Point(-1, -1), the center of the kernel is taken as the position of anchor

int iterations - Number of times erosion is applied

intborderType - Pixel extrapolation method in a boundary condition

const Scalar&borderValue - Value of the pixels in a boundary condition if borderType = BORDER_CONSTANT

- The dilate() function dilates the source image and stores the result in the destination image. In-place processing is supported (which means you can use the same variable for the source and destination image). If the source image is multi-channel, all channels are processed independently and the result is stored in the destination image as separate channels.

  Arguments-

  InputArraysrc - Source image

  OutputArraydst - Destination image (should have the same size and the type as the source image)

  InputArray kernel - Structuring element which is used to dilate the source image

  Point anchor - Position of the anchor within the kernel. If it is Point(-1, -1), the center of the kernel is taken as the position of anchor

  int iterations - Number of times dilation is applied

  intborderType - Pixel extrapolation method in a boundary condition

  const Scalar&borderValue - Value of the pixels in a boundary condition if borderType = BORDER_CONSTANT

- The convert() function converts a source image from one colour space to another. In-place processing is supported. (which means you can use the same variable for the source and destination image)

  Arguments-

  InputArraysrc - Source image

  OutputArraydst - Destination image (should have the same size and the depth as the source image)

  int code - Colour space conversion code (e.g– COLOR_BGR2HSV, COLOR_RGB2HSV, COLOR_BGR2GRAY, COLOR_BGR2YCrCb, COLOR_BGR2BGRA, etc.)

  intdstCn - Number of channels in the destination image. If it is 0, number of channels is derived automatically from the source image and the colour conversion code.

3.5.2 System Diagram:

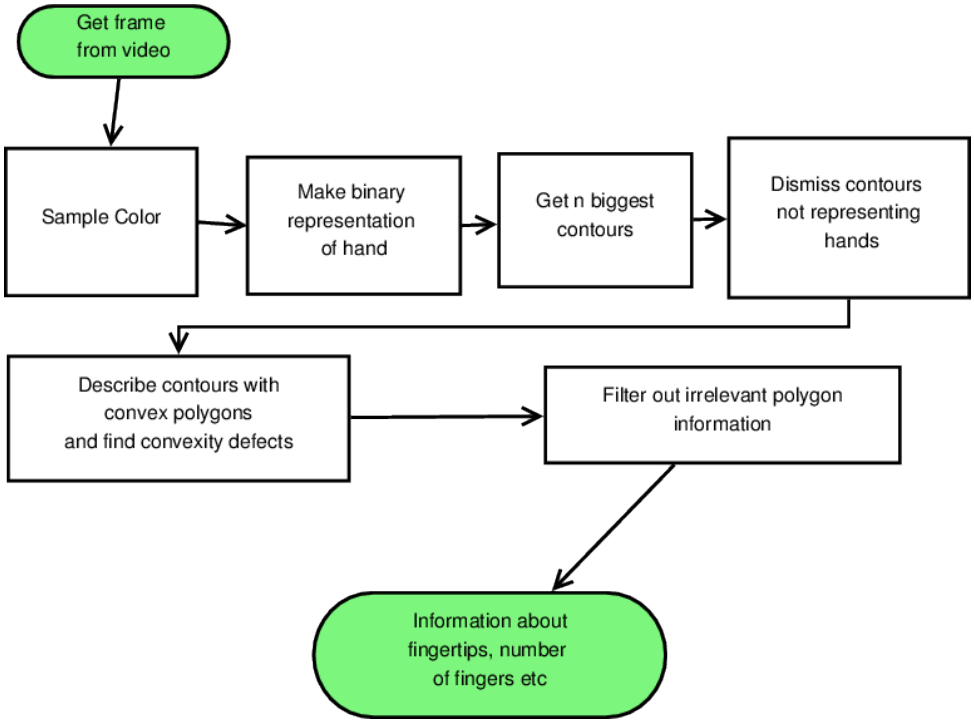

Fig. 3.2: An overview of our proposed method for moving object (hand) detection

Chapter 4

Result & Analysis

4.1 Moving Object Detection

In this application, we use moments to calculate the position of the centre of the object. We have to calculate 1st order spatial 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 spatial moment around x-axis / 0th order central moment

Y coordinate of the position of the center of the object = 1st order spatial 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 application developed above, I considered that if the white area of the binary image is less than or equal to 10000 pixels, there are no objects in the image because my object is expected to have an area more than 10000 pixels.

Now, let's discuss new OpenCV methods that can be found in the above application.

- Moments moments(InputArray array, bool binaryImage=false):
  This OpenCV function calculates all of the spatial moments up to the third order and returns a Moments object with the results.

Arguments-

InputArray array - Single channel image

boolbinaryImage - If this is true, all non zero pixels are considered as ones when calculating moments.

- void line(Mat&img, Point pt1, Point pt2, const Scalar&color, int thickness=1, intlineType=8, int shift=0):

  This function draws a line between two points on a given image.

  Arguments-

  Mat&img - image which you want to draw the line

  Point pt1 - First point of the line segment

  Point pt2 - Other point of the line segment

  const Scalar&color - Colour of the line (values of Blue, Green and Red colours respectively)

  int thickness - Thickness of the line in pixels

- static MatExpr zeros(Size size, int type)

  This function returns a black image (with pixels with zero values) with a given size and type.

  Size size - Size of the required image (Size(No of columns, No of rows))

  int type - Type of the image (e.g - CV_8UC1, CV_32FC4, CV_8UC3, etc)

How to Find Exact Range for 'Hue', 'Saturation' and 'Value' for a Given Object:

Finding the optimum HUE, SATURATION and VALUE ranges for an object is a 4 step process.

1. Track bars should be placed in a separate window so that ranges for HUE, SATURATION and VALUE can be adjusted. And set the initial ranges for HUE, SATURATION and VALUE as 0-179, 0-255 and 0-255 respectively. So, we will see a complete white image in the 'Control' window.

2. First, adjust 'LowH' and 'HighH' track bars so that the gap between 'LowH' and 'HighH' is minimized. Here you have to be careful that white area in 'Ball' window that represents the object should not be affected, while you are trying to minimize the gap.

3. Repeat the step 2 for 'LowS' and 'HighS' trackbars.

4. Repeat the step2 for 'LowV' and 'HighV' trackbars. Now you can find the optimum HUE, SATURATION and VALUE ranges for the object.

4.2 Object Tracking:

Simple Example of Tracking Red objects:

There are 3 steps involving to achieve this task:

1. Detect the object

2. Find the exact position (x, y coordinates) of the object

3. Draw a line (visible or invisible) along the trajectory of the object. Process this trajectory according to the application as desired.

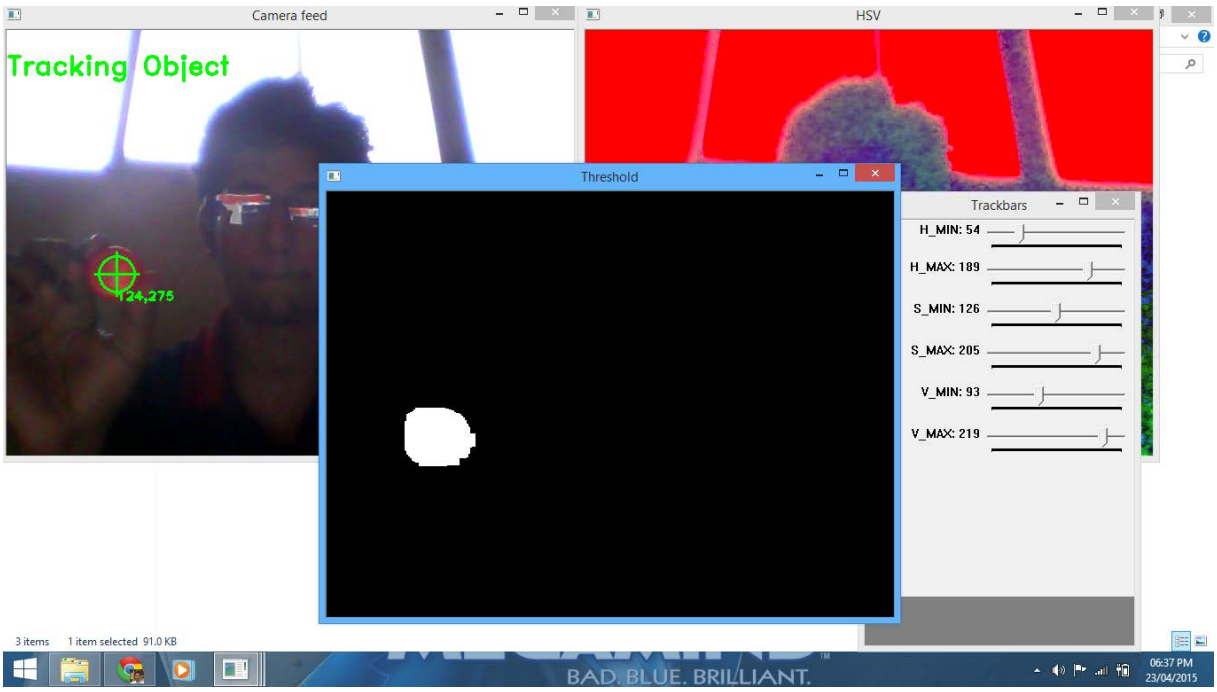

Fig. 4.1: The object is tracked by finding the appropriate colour in the image as per the set HSV

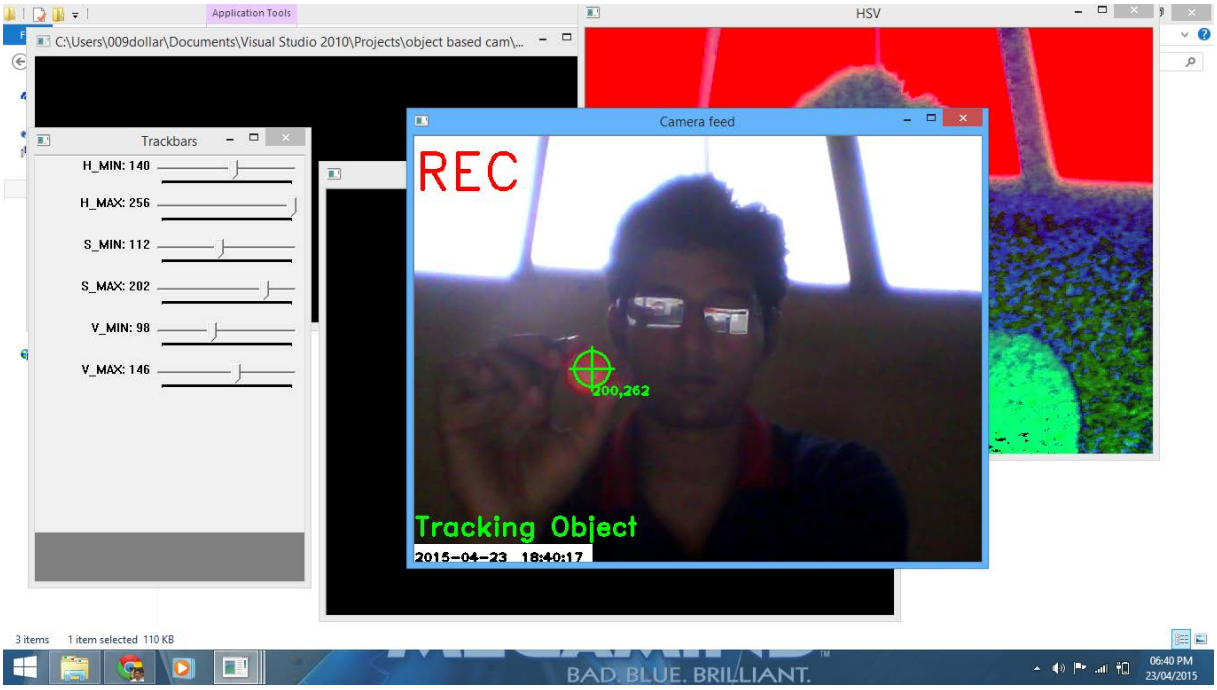

Fig. 4.2: Once the object is found, tracking begins, but only for as long as the object is still in the caught frame.

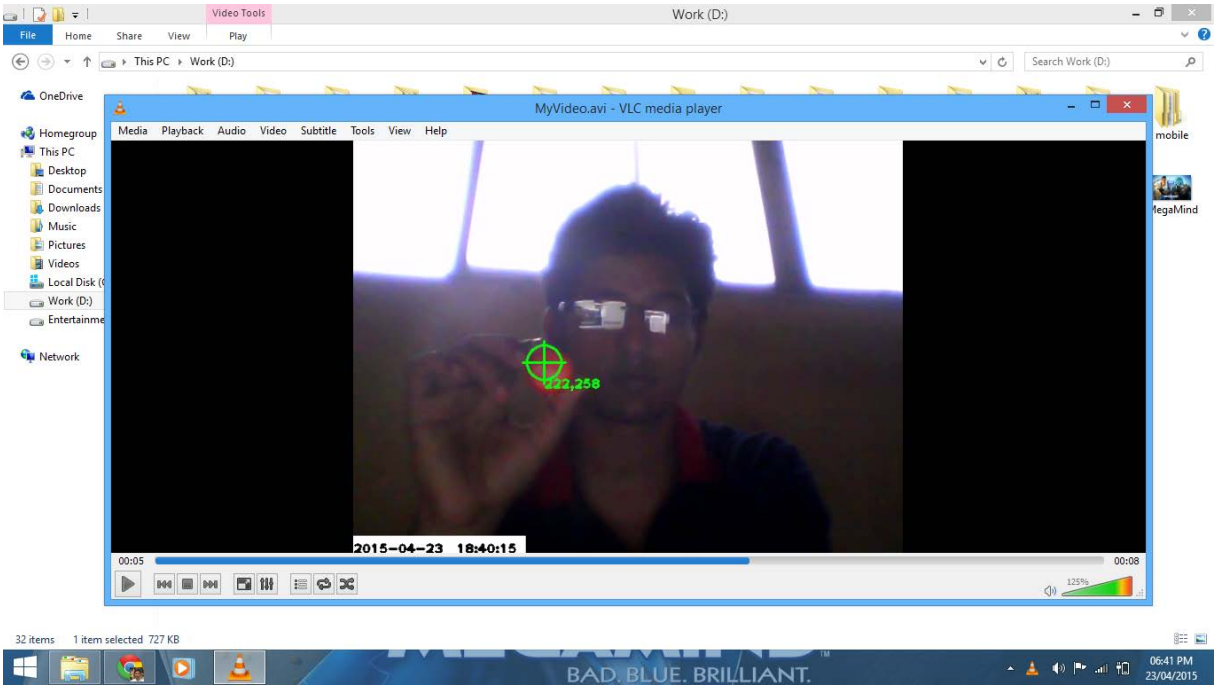

Fig. 4.3: The recorded video is stored in the avi format and can then be played back.

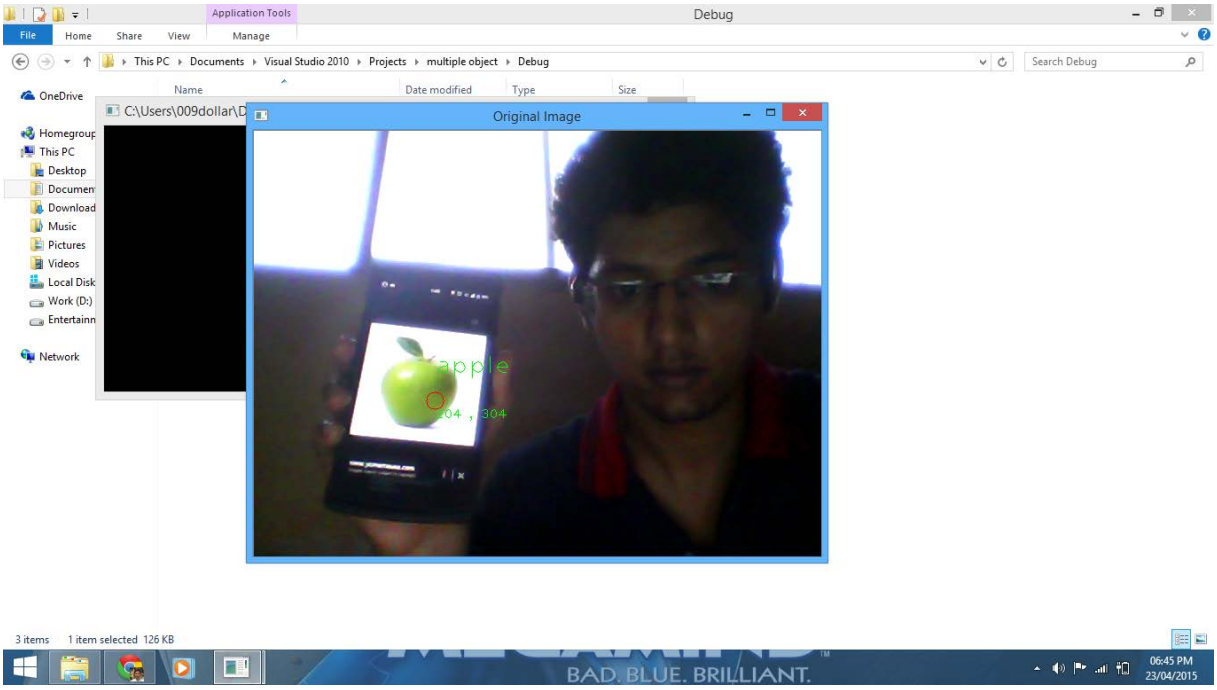

Fig.4.4: Object recognition by way of recognizing the contours and curvature of the object, if present in the preset settings.

4.3 Analysis

The proposed system has been decided upon through extensive research, and the components used as well as the techniques recommended are the optimal ones currently available. Though we have tried to minimize them, the system is not without a few flaws that, although only marginally, cause the system to produce unexpected outputs in some cases.

Below are mentioned the advantages this system has over others, as well as a few disadvantages that it encounters that are currently insoluble but may be minimized by future research.

4.3.1 Advantages:

1] The technologies used into developing the application are all open-source, as will be the code of the application itself. Therefore, as a result of constant developments in the underlying technologies, the application too will find itself developing at a very high rate, therefore leading to greater chances of adoption of the application by external users.

2] The application uses techniques that have been proven to be simple yet effective, such as background subtraction, particle filtering and foreground intensity comparisons. Therefore the application should be extremely low-cost computationally, taking up the least amount of time and memory space on the user's computer/server.

3] The application has been so devised as to cache every new incoming frame for as much time as is required for comparison with the background model, update the model if required and then wait for the next frame. This space is freed as soon as the operations on that frame are completed. If and only if some untoward motion occurs that affects a certain preset target object does the application record it and store it in its permanent storage.

4] Since the application does not exist on the camera but only on the server or the computer, unless and until an intruder has physical access to the server, the application cannot be disabled or tampered with. Being an open-source application, optimistically any bugs present too will find immediate resolution. Therefore in terms of security, the application is secure and unalterable except in the worst of cases.

4.3.2 Disadvantages:

The proposed application is unsuitable for purposes where constant monitoring is required, for instance in hospital Intensive Care Units or in high-sensitivity government zones. These areas have access to high storage servers for continuous recording of the video, and hence might not find use of the application. It is proposed that the application contain two modes – one that allows for continuous monitoring and another that works as above.

Chapter 5

Future Scope

Some of the important applications of object tracking are:

1. Automated video surveillance:

Video Surveillance is one of the primary places where his application finds use. Computer vision system is designed to monitor the movements in an area, identify the moving objects and report any doubtful situation. The system needs to discriminate between natural entities and humans, which require a good object tracking system. The proposed application is extremely effective in distinctive recognition, and can also perform crowd analysis very well.

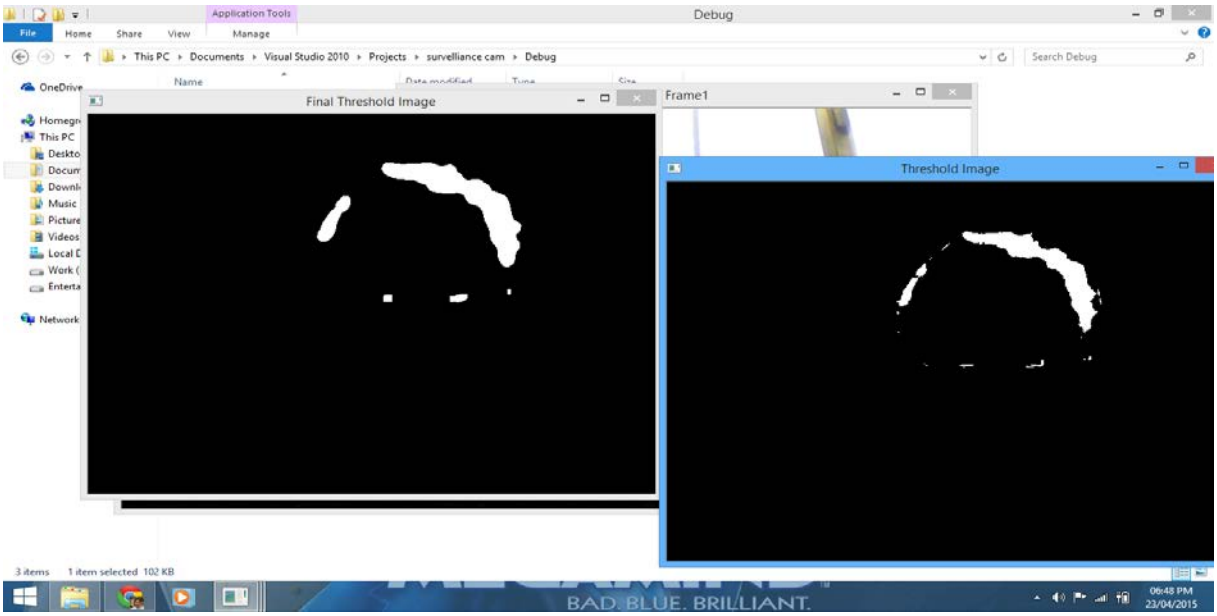

Fig. 5.1: Change in pixels in the frame is recorded.

2. Robot vision:

In robot navigation, the steering system needs to identify different obstacles in the path to avoid collision. If the obstacles themselves are other moving objects then it calls for a real-time object tracking system. The application developed can help the robot to distinguish surroundings as objects and background. With a few developments, the application can be easily configured to process all inputs obtained from the visual sensory devices of robots, and enable him to recognize various objects allowing for safe, unrestricted motion.

3. Traffic monitoring:

In some countries highway traffic is continuously monitored using cameras. Any vehicle that breaks the traffic rules or is involved in other illegal act can be tracked down easily if the surveillance system is supported by an object tracking system. Thus even as huge a sector as traffic monitoring can also employ the concepts explained in the report, if not the application itself.

4. Animation:

Object tracking algorithm can also be extended for animation. Tracking normal motion of humans, animals and other objects will enable better framework development of the objects to be animated making the job of the animator easy.

CHAPTER 7

Conclusion

In this report, we developed several methods for moving object detection (specifically background model initialization, background maintenance, and foreground detection), particle filtering based object tracking (the appearance model and methods for parameter settings for the likelihood model), a fast implementation for appearance model construction, and crowd analysis (crowd flow estimation). We consider robustness and computational cost as the major design goals of our work. Our methods are designed to be proactive and capable of processing the video stream for event analysis in real-time. Ideally they can assist human operators with identification of important events in videos and responding to them in a timely manner.

The application is one that should easily find a number of uses in the mainstream, that too in various distinct areas such as security, traffic monitoring, industrial and financial systems etc. If extended properly, even areas as extensive as robotics can find application of the proposed project. It is composed of the best and most effective algorithms proposed in literature, and is therefore extremely efficient in terms of time and space complexities. We even propose the installation of this application in the college's surveillance system once it comes to fruition.

References

[1] M. Valera and S. Velastin, “Intelligent distributed surveillance systems: a review,” IEE Proceedings - Vision, Image and Signal Processing, vol. 152, no. 2, pp. 192–204, April 2005.

[2] M. Shah, O. Javed, and K. Shafique, “Automated visual surveillance in realistic scenarios,” IEEE Transactions on Multimedia, vol. 14, no. 1, pp. 30–39, January 2007.

[3] R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, “Image change detection algorithms: a systematic survey,” IEEE Transactions on Image Processing, vol. 14, no. 3, pp. 294–307, March 2005.

[4] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” ACM Com

[5] N. Haering, P. L. Venetianer, and A. Lipton, “The evolution of video surveillance: an overview,” Machine Vision and Applications, vol. 19, no. 5-6, pp. 279–290, September 2008.

[6] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol. 34, no. 3, pp. 334–352, August 2004.

[7] M. F. Abdelkader, R. Chellappa, Q. Zheng, and A. L. Chan, “Integrated motion detection and tracking for visual surveillance,” Proceedings of the IEEE International Conference on Computer Vision Systems, Washington, DC, USA, January 2006, p. 28.

[8] www.engineering.purdue.edu

[9] www.wikipedia.com

Acknowledgement

It is a matter of great honour to bring out the synopsis report on the project: “Online Programming Portal Integrated With Compilers/Interpreters And Text Analysis For College-Level Coding”.

We experience immense pleasure in stating that we have secured the excellent guidance of Prof. Harsha Saxena. We have received her whole hearted assistance, inspiration, encouragement and valuable guidance in all phases of our project.

We also take the opportunity to thank our HOD, **Dr. Leena Ragha** and Principal **Dr. Ramesh Vasappanavara** for their valuable help.

Lastly we cannot forget to thank our group members and our class friends who directly or indirectly gave us ideas and help for this project. Also we are thankful to the non teaching staff for providing the various facilities for this project.