Bachelor Thesis Project

CRICKET UMPIRE ASSISTANCE AND BALL TRACKING SYSTEM USING A SINGLE CAMERA

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF B.E (COMPUTER ENGINEERING)

SUBMITTED BY:

Sarthak Sahni (356/CO/12)

Sohit Verma (365/CO/12)

Udit Arora (381/CO/12)

GUIDED BY:

Dr. Pinaki Chakraborty

(Assistant Professor, Division of Computer Engineering)

DIVISION OF COMPUTER ENGINEERING
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to our mentor Dr. Pinaki Chakraborty for giving us the opportunity to work under his able supervision. He has been a pillar of strength throughout this research, showing us the way from the very beginning. The frequent brainstorming sessions have helped us bring out the best and have made us feel excited about the work we are doing. His encouragement in times of distress and anxiety; patience in hearing our problems; rigorous reviewing to make sure that everything falls in place; and the immense knowledge that he possesses have helped us carry this project forward in the right spirits. Without his invaluable and continuous contributions, this project could not have been completed.

We are also indebted towards the faculty of NSIT, whose guidance and teaching for the past four years has helped us in understanding the theory and practices of Computer Engineering. The progress in our Bachelor Thesis Project is a reflection of the progression of our education in the 4 years that we have spent at NSIT.

We also thank our parents and families who always supported us unconditionally in our thick and thin. Their belief in our capabilities always pushed us to give our best in this project.

नेताजी सुभाष प्रौद्योगिकी संस्थान
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi-110 045
Telephone : 2509 9036-42, 2509 9050 Fax : 2509 9022 Website : http://www.nsit.ac.in

DECLARATION

This is to certify that the project entitled, "**CRICKET UMPIRE ASSISTANCE AND BALL TRACKING SYSTEM USING A SINGLE CAMERA**" by Sarthak Sahni (356/CO/12), Sohit Verma (365/CO/12) and Udit Arora (381/CO/12) is a record of bonafide work carried out by us, in the Department of Computer Engineering, Netaji Subhas Institute of Technology, University of Delhi, New Delhi. The thesis is being submitted by Sarthak Sahni (356/CO/12), Sohit Verma (365/CO/12) and Udit Arora (381/CO/12) in partial fulfilment of requirements for the Degree of Bachelor of Engineering in Computer Engineering, University of Delhi in the academic year 2015-2016.

The results presented in this thesis have not been submitted in part or in full, to any other university or institute for the award of any degree or diploma.

Sarthak Sahni (356/CO/12)

Sohit Verma (365/CO/12)

Udit Arora (381/CO/12)

Date: 31.05.2016

CERTIFICATE

This is to certify that the final year project entitled "**CRICKET UMPIRE ASSISTANCE AND BALL TRACKING SYSTEM USING A SINGLE CAMERA**" being submitted by Sarthak Sahni (356/CO/12), Sohit Verma (365/CO/12) and Udit Arora (381/CO/12) to Netaji Subhas Institute of Technology, Delhi University for the partial fulfilment of requirements for the degree of Bachelor of Engineering in Computer Engineering is a record of the bonafide research work carried out by them in the Department of Computer Engineering, under my guidance and supervision.

The results contained in this report have not been submitted in part or in full, to any other university or institute for the award of any degree or diploma.

Dr. Pinaki Chakraborty

Assistant Professor

Division of Computer Engineering

Netaji Subhas Institute of Technology

University of Delhi

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This thesis aims to develop a product for assisting the umpire in the sport of Cricket in making decisions like detection of no-balls, wide-balls, leg before wicket etc with the help of technology. It involves the implementation of Computer Vision algorithms for object detection and motion tracking, as well as the integration of machine learning algorithms to optimize the results.

Several ball tracking algorithms have been reported in literature. However, most of them use high quality video and multiple cameras and the emphasis has been on coordinating the cameras, or visualising the tracking results. In this project, we propose to develop a ball tracking mechanism for low quality off-air video recorded with a single camera.

Techniques like Histogram of Gradients (HOG) and Support Vector Machine (SVM) are used for object classification and recognition.

Frame subtraction, minimum enclosing circle and contour detection algorithms are optimised and used for the detection of a cricket ball. These algorithms are assisted by the use of Open Source Python Library - OpenCV.

Machine Learning techniques - Linear and Quadratic Regression are used to track and predict the motion of the ball. The project also involves the use of open source Python library VPython for the visual representation of the results.

The thesis describes the design and structure for the approach undertaken in the project for analysing and visualizing off air low quality cricket videos.

INTRODUCTION

In cricket, an umpire is a person who has the authority to make judgements on the cricket field, according to the laws of cricket. Besides making decisions about legality of delivery, appeals for wickets and general conduct of the game in a legal manner.

Umpire may call, and signal, No Ball, for a ball which is illegally delivered (bowled). A Wide Ball is an illegal delivery in cricket, which is illegal due to it being “wide of the striker where he is standing and would also have passed wide of him standing in a normal guard position or the ball passing above a batsman’s head”. The umpire may rule a batsman out Leg before wicket (lbw) if the ball would have struck the wicket, but was instead intercepted by any part of the batsman's body (except the hand holding the bat). The umpire's decision will depend on a number of criteria, including where the ball pitched, whether the ball hit in line with the wickets, and whether the batsman was attempting to hit the ball.

Over the years, the game of cricket has evolved greatly and has incorporated the use of technology in various forms. Whether it is the use of high quality cameras and advanced computer systems to enhance the viewer experience or the use of sophisticated technologies for increasing the accuracy and correctness of umpiring decisions, the impact of technology has been significant in making the game better.

The Umpire Decision Review System (abbreviated as UDRS or DRS) is a technology-based system used in cricket. The system was introduced in cricket, for the sole purpose of reviewing controversial decisions made by the on-field umpires as to whether or not a batsman had been dismissed.

Research in the field of technology in cricket has resulted in some very useful hardware and software technologies. The most successful products to date include Hawk Eye, Hot Spot, Snickometer.

Hawk-Eye, Eagle Eye, or Virtual Eye is a ball-tracking technology that plots the trajectory of a bowling delivery that has been interrupted by the batsman, often by the pad, and can determine whether it would have hit the wicket or not. It uses high resolution cameras to calculate the trajectory using an advanced position triangulation method. Hot Spot is an Infra-red imaging system that illuminates where the ball has been in contact with bat or pad. Real time Snickometer, is a tool that relies on directional microphones to detect small sounds made as the ball hits the bat or pad.

In the past few decades researchers and engineers have developed quite a few technologies for assisting the umpire in making decisions. These solutions have mostly involved the use of high quality hardware (expensive high resolution cameras, microphones etc).

The projects have mostly been developed keeping in mind the professional and international level of cricket due to the high costs of the hardware involved.

The game of cricket is very widely played and followed in India and many other countries in the world. There are millions of amateur and aspiring cricketers involved. But the high cost and heavy technological requirements of the above mentioned technologies restricts their use in any matches, competitions and training academies other than the ones operating at an international level.

Computer vision seems like a natural choice for these applications. In spite of the success of computer vision technology in several other fields (such as robot navigation, surveillance, and

user interface), very few computer vision systems and algorithms which enhance the experience of cricket are there which operate at a low cost

A promising research direction is the use of computer vision to detect, identify and track the cricket ball (and other relevant objects in the context of cricket), and machine learning techniques to optimize and further predict various results and decisions.

The use of just one camera, which may be of a quality equivalent to modern day smartphone cameras, along with the various algorithms and techniques of Computer Vision and Machine Learning can help us achieve a system that reliably assists the umpire and operates at a cheap cost.

This thesis aims to develop a low-cost computer system which assists the umpire in cricket, operates at a low cost, has lesser technological (software and hardware) requirements, and can be used at sub-international levels in the sport of cricket.

PROJECT REQUIREMENTS

Operating System Requirements:

The application was developed on Ubuntu 14.10 (LTS) and Mac OS X 10.11 (El Capitan)

Software Requirements:

1. Python 2.7: Python is a widely used general-purpose, multi-paradigm, dynamically-typed high-level programming language. Python was chosen for this project for its ability to allow rapid prototyping of applications and for its wide support-base. Open Source OpenCV library also supports an interface for Python.

2. Python Libraries used:

   A. OpenCV2: OpenCV (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision. OpenCV is written in C++ and its primary interface is in C++, but it provides a binding in Python.

   B. Imutils: It contains a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

C. NumPy: NumPy is a general-purpose array-processing package designed to efficiently manipulate large multi-dimensional arrays of arbitrary records without sacrificing too much speed for small multi-dimensional arrays.

D. VPython: VPython is the Python programming language plus a 3D graphics module called Visual. VPython allows users to create objects such as spheres and cones in 3D space and displays these objects in a window. This makes it easy to create simple visualizations.

E. Scikit: Scikit-learn is a free software machine learning library for the Python programming language.

F. Kivy: It is an open source (pronounced as Kiwi) Python library for developing mobile apps and other multitouch application software with a natural user interface (NUI).

Hardware Requirements:

1. Smartphone Camera: A digital camera or digicam is a camera that encodes digital images and videos digitally and stores them for later reproduction. Smartphones (advanced mobile phone devices) typically come equipped with a digital camera feature. The camera needs to have capabilities for recording videos at a high frame rate.
2. Tripod: A tripod is used to capture a stable video using a digital camera.

PROJECT APPROACH

- **Topic Research and Selection:** We read various research papers, University websites to identify the topic where there was scope for research and implementation and hence, decided to develop a prototype of the cricket umpiring assistance system.
- **Research:** We read about 15 – 20 research papers, eliminated irrelevant ones and selected about 24 related papers to do an exhaustive study, from our end. We studied about the theoretical foundations of the field of Computer Vision and understood the working of existing image processing algorithms.
- **Dataset Collection and Preparation:** A smartphone camera was used to capture raw videos of cricket in amicable surroundings which were simulated in a manner that was representative of a real cricket match or nets training session.
- **Object Detection and Image Processing:** Computer Vision algorithms were used to detect various relevant objects (ball, batsman, bowler etc.). Histogram of Gradients (HOG) and Support Vector Machine (SVM) techniques were used in the classification of these objects and in improving the accuracy of detection. Image processing techniques were used to improve the results.
- **Ball Tracking:** Computer Vision techniques and Machine learning features of linear and quadratic regression (unweighted and weighted) are used to improve the ball tracking results.
- **Visual Representation of Results:** The results and various information were represented visually using VPython library.

OBJECTIVES

The objective of this thesis is to develop a product for assisting the umpire in the sport of cricket in making decisions using a single camera. The thesis involves the development of algorithms using computer vision and machine learning techniques for ball detection and tracking, along with various cricket decision making rules.

This thesis focuses on three areas:

1. Implementing computer vision algorithms such as contour detection, frame subtraction, minimum enclosing circle, thresholding for processing real-time images.
2. Using machine learning to optimise the results and improve their accuracy and correctness by implementing classification and regression techniques.
3. Visually representing the tracking results along with decisions based on the results, in accordance with the rules of cricket.

LITERATURE REVIEW

Computer Vision has a dual goal. From the biological science point of view, computer vision aims to come up with computational models of the human visual system. From engineering point of view, computer vision aims to build autonomous systems which could perform some of the tasks which the human visual system can perform (and even surpass it in many cases). Many vision tasks are related to the extraction of 3D and temporal information from time-varying 2D data such as obtained by one or more television cameras, and more generally the understanding of such dynamic scenes. The properties and characteristics of the human visual system often give inspiration to engineers who are designing computer vision systems. Conversely, computer vision algorithms can offer insights into how the human visual system works.

It is commonly accepted that the father of Computer Vision is Larry Roberts, who in his Ph.D. thesis (cir. 1960) at MIT, discussed the possibilities of extracting 3D geometrical information from 2D perspective views of blocks (polyhedra). Many researchers, at MIT and elsewhere, in Artificial Intelligence, followed this work and studied computer vision in the context of the blocks world.

Later, researchers realized that it was necessary to tackle images from the real world. Thus, much research was needed in the so called “low-level” vision tasks such as edge detection and segmentation. A major milestone was the framework proposed by David Marr (cir. 1978) at MIT, who took a bottom-up approach to scene understanding.

Low-level image processing algorithms are applied to 2D images to obtain the “primal sketch” (directed edge segments, etc.), from which a 2.5 D sketch of the scene is obtained using binocular stereo. Finally, high-level (structural analysis, *a priori* knowledge) techniques are used to get 3D model representations of the objects in the scene. This is probably the single most

influential work in computer vision ever. Many researchers cried: “From the paradigm created for us by Marr, no one can drive us out.”

Nonetheless, more recently a number of computer vision researchers realized some of the limitation of Marr’s paradigm, and advocated a more top-down and heterogeneous approach. Basically, the program of Marr is extremely difficult to carry out, but more important, for many if not most computer vision applications, it is not necessary to get complete 3D object models. For example, in autonomous vehicle navigation using computer vision, it may be necessary to find out only whether an object is moving away from or toward your vehicle, but not the exact 3D motion of the object. This new paradigm is sometimes called “Purposive Vision” implying that the algorithms should be goal driven and in many cases could be qualitative. One of the main advocates of this new paradigm is Yiannis Aloimonos, University of Maryland.

Looking over the history of computer vision, it is important to note that, because of the broad spectrum of potential applications, the trend has been the merge of computer vision with other closely related fields. These include: Image processing (the raw images have to be processed before further analysis). Photogrammetry (cameras used for imaging have to be calibrated. Determining object poses in 3D is important in both computer vision and photogrammetry). Computer graphics (3D modeling is central to both computer vision and computer graphics. Many exciting applications need both computer vision and computer graphics.

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast

to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

In scientific imaging where spatial correlation is more important than intensity of signal (such as separating DNA fragments of quantized length), the small signal to noise ratio usually hampers visual detection.

Histogram equalization often produces unrealistic effects in photographs; however it is very useful for scientific images like thermal, satellite or x-ray images, often the same class of images to which one would apply false-color. Also histogram equalization can produce undesirable effects (like visible image gradient) when applied to images with low color depth. For example, if applied to 8-bit image displayed with 8-bit gray-scale palette it will further reduce color depth (number of unique shades of gray) of the image. Histogram equalization will work the best when applied to images with much higher color depth than palette size, like continuous data or 16-bit gray-scale images.

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as superpixels). The goal of segmentation is to

simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. Various ball tracking have been proposed and implemented for the sport of cricket and tennis in particular. The problem of tracking a ball moving at high speeds has been dealt with different approaches. Most approaches use some kind of background subtraction to identify the moving objects and then track the required ball.

Further, various approaches for a smart and automated cricket field have been proposed for automating the crucial umpiring decisions in a match. However most of these approaches propose the use of multiple cameras, or provide little to no implementation details.

Reliable ball tracking often requires the use of multiple high quality cameras, as used by high-quality commercial systems like Hawk-Eye which is widely used in sports.

THEORETICAL BACKGROUND

Computer Vision Techniques

Throughout this thesis, a number of Image Processing and Computer vision techniques are mentioned. They serve different purposes such as image smoothing, object detector, regression and contour detection. In the following section these techniques are briefly described.

Data Classification

Classification algorithm is the core of detection system. In brief, the role of classification is to determine whether the input sub-window contain an object or not. Broadly speaking, given a set of training examples (image, human, data, etc.), each marked as belonging to one of M categories (generally, M = 2. ball or non-ball), a classification algorithm builds a system or model that can assigns new examples which do not belong to the training examples into one category or others. The same process happened in image target detection.

Classification algorithm in machine vision and image processing is that of determining whether or not the input image contains some specific object (human-face, pedestrian, animal, traffic sign, etc.). After we have the features to describe the image's details, an algorithm is needed to justify whether the image is our target. The algorithm has two main effects, the first one is to process the image feature in training step to generate the classifier based on the image database, the second one is to analyse the input image and detect the target.

In this chapter, popular classification algorithm is introduced, Support Vector Machine(SVM).

Support Vector Machine (SVM)

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples.

In which sense is the hyperplane obtained optimal? Let's consider the following simple problem:

For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line.

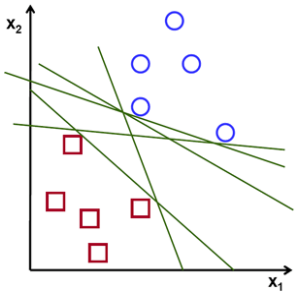


Fig. 11.1: Linear Separation

In the above picture you can see that there exists multiple lines that offer a solution to the problem. Is any of them better than the others? We can intuitively define a criterion to estimate the worth of the lines:

A line is bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, our goal should be to find the line passing as far as possible from all points.

Then, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of **margin** within SVM's theory. Therefore, the optimal separating hyperplane *maximizes* the margin of the training data.

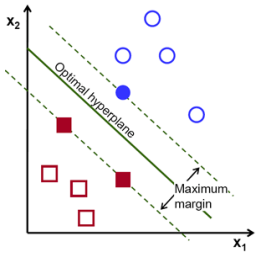


Fig. 11.2: Hyperplane separation

Computation of optimal hyperplane

Let's introduce the notation used to define formally a hyperplane:

$$f(x) = \beta_0 + \beta^T x,$$

where β is known as the *weight vector* and β_0 as the *bias*.

The optimal hyperplane can be represented in an infinite number of different ways by scaling of β and β_0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

$$|\beta_0 + \beta^T x| = 1$$

where x symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called **support vectors**. This representation is known as the **canonical hyperplane**.

Now, we use the result of geometry that gives the distance between a point x and a hyperplane (β, β_0):

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is

$$\text{distance}_{\text{ support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}.$$

Recall that the margin introduced in the previous section, here denoted as M, is twice the distance to the closest examples:

$$M = \frac{2}{\|\beta\|}$$

Finally, the problem of maximizing M is equivalent to the problem of minimizing a function $L(\beta)$ subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples x_i. Formally,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i,$$

where y_i represents each of the labels of the training examples.

This is a problem of Lagrangian optimization that can be solved using Lagrange multipliers to obtain the weight vector β and the bias β_0 of the optimal hyperplane.

Linear SVM

We are given a training dataset of n points of the form

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$$

where the y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0,$$

Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

The essential thought behind the histogram of oriented gradients descriptor is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is the concatenation of these histograms. For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

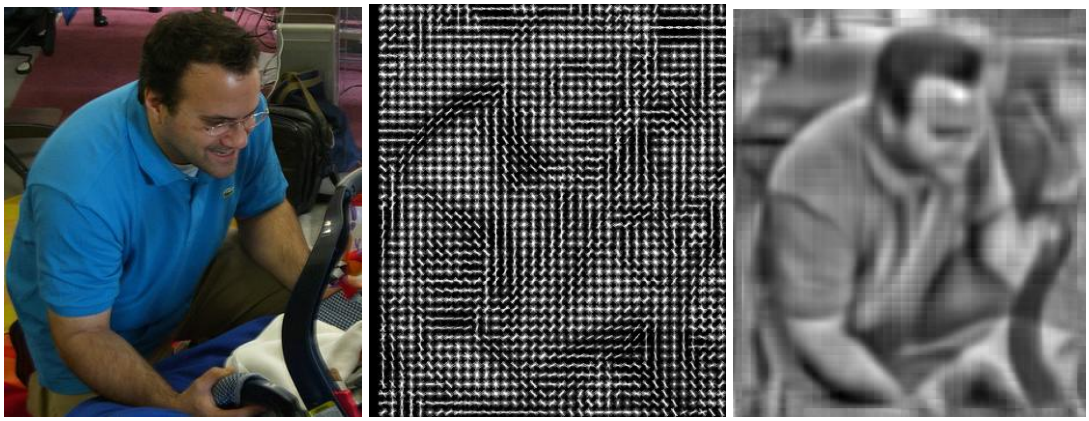


Fig. 11.3: Orginal Image (left), HOG features (center), Inverse HOG (Right)

The above example shows an actual image, which is then converted into a compilation of cells according to intensity pattern. The third image is how HOG perceives the real world image after applying the intensity gradient normalization on it.

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation. Such changes would only appear in larger spatial regions. Moreover, as Dalal and Triggs discovered, coarse spatial sampling, fine orientation sampling, and strong local photometric normalization permits the individual body movement of pedestrians to be ignored so long as they maintain a roughly upright position. The HOG descriptor is thus particularly suited for human detection in images.

Gradient Computation

The first step of calculation in many feature detectors in image pre-processing is to ensure normalized color and gamma values. As Dalal and Triggs point out, however, this step can be omitted in HOG descriptor computation, as the ensuing descriptor normalization essentially achieves the same result. Image pre-processing thus provides little impact on performance.

Instead, the first step of calculation is the computation of the gradient values. The most common method is to apply the 1-D centered, point discrete derivative mask in one or both of the horizontal and vertical directions. Specifically, this method requires filtering the color or intensity data of the image with the following filter kernels:

$$[-1, 0, 1] \text{ and } [-1, 0, 1]^\top.$$

Dalal and Triggs tested other, more complex masks, such as the 3x3 Sobel mask or diagonal masks, but these masks generally performed more poorly in detecting humans in images. They also experimented with Gaussian smoothing before applying the derivative mask, but similarly found that omission of any smoothing performed better in practice.[2]

Orientation Binning

The second step of calculation is creating the cell histograms. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel based on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is “unsigned” or “signed”. Dalal and Triggs found that unsigned gradients used in conjunction with 9 histogram channels performed best in their human detection experiments. As for the vote weight, pixel contribution can either be the gradient magnitude itself, or some function of the magnitude. In tests, the gradient magnitude itself generally produces the best results. Other options for the vote weight could include the square root or square of the gradient magnitude, or some clipped version of the magnitude.

Descriptor Blocks

To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the concatenated vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor. Two main block geometries exist: rectangular R-HOG blocks and circular C-HOG blocks. R-HOG blocks are generally square grids, represented by three parameters: the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram. In the Dalal and Triggs human detection experiment, the optimal parameters were found to be four 8x8 pixels cells per block (16x16 pixels per block) with 9 histogram channels. Moreover, they found that some minor improvement in performance could be gained by applying a Gaussian spatial window within each block before tabulating histogram votes in order to weight pixels around the edge of the blocks less. The R-HOG blocks appear quite similar to the scale-invariant feature transform (SIFT) descriptors; however, despite their similar formation, R-HOG blocks are computed in dense grids at some single scale without orientation alignment, whereas SIFT descriptors are usually computed at sparse, scale-invariant key image points and are rotated to align orientation. In addition, the R-HOG blocks are used in conjunction to encode spatial form information, while SIFT descriptors are used singly.

Circular HOG blocks (C-HOG) can be found in two variants: those with a single, central cell and those with an angularly divided central cell. In addition, these C-HOG blocks can be described with four parameters: the number of angular and radial bins, the radius of the center bin, and the expansion factor for the radius of additional radial bins. Dalal and Triggs found that the two main variants provided equal performance, and that two radial bins with four angular bins, a

center radius of 4 pixels, and an expansion factor of 2 provided the best performance in their experimentation. Also, Gaussian weighting provided no benefit when used in conjunction with the C-HOG blocks. C-HOG blocks appear similar to shape context descriptors, but differ strongly in that C-HOG blocks contain cells with several orientation channels, while shape contexts only make use of a single edge presence count in their formulation.[4]

Block Normalization

Dalal and Triggs explored four different methods for block normalization. Let v be the non-normalized vector containing all histograms in a given block, $\|v\|_k$ be its *k*-norm for $k = 1, 2$ and e be some small constant (the exact value, hopefully, is unimportant). Then the normalization factor can be one of the following:

L2-norm: $$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing, as in [5]

L1-norm: $$f = \frac{v}{(\|v\|_1 + e)}$$

L1-sqrt: $$f = \sqrt{\frac{v}{(\|v\|_1 + e)}}$$

In addition, the scheme L2-hys can be computed by first taking the L2-norm, clipping the result, and then renormalizing. In their experiments, Dalal and Triggs found the L2-hys, L2-norm, and L1-sqrt schemes provide similar performance, while the L1-norm provides slightly less reliable

performance; however, all four methods showed very significant improvement over the non-normalized data.[6]

SVM Classifier

The final step in object recognition using histogram of oriented Gradient descriptors is to feed the descriptors into some recognition system based on supervised learning. The support vector machine (SVM) classifier is a binary classifier which looks for an optimal hyperplane as a decision function. Once trained on images containing some particular object, the SVM classifier can make decisions regarding the presence of an object, such as a human, in additional test images.

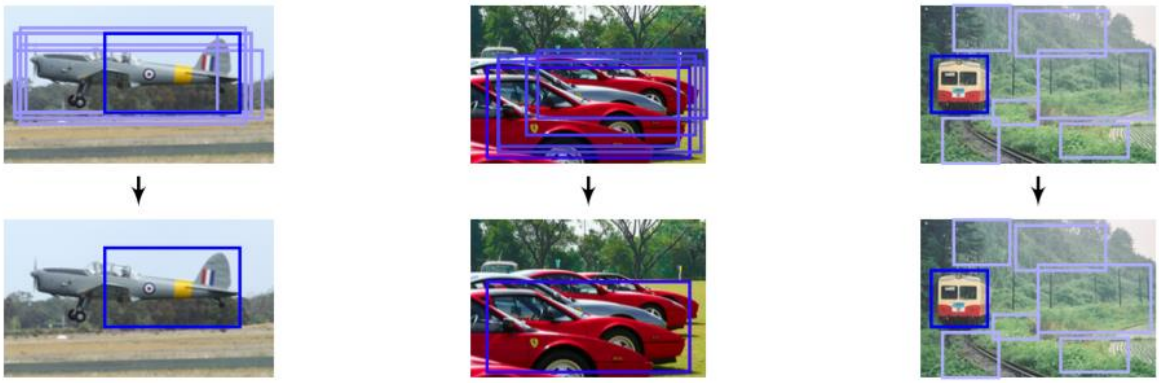
Neural Network Classifier

The feature of the gradient descriptors are also fed into the neural network classifiers which provides more accuracy in the classification comparing other classifiers (SVM). The neural classifiers can accept the descriptor feature as the binary function or the optimal function.

Non Maxima Suppression

Non-maximum suppression (NMS) has been widely used in several key aspects of computer vision and is an integral part of many proposed approaches in detection, might it be edge, corner or object detection . Its necessity stems from the imperfect ability of detection algorithms to localize the concept of interest, resulting in groups of several detections near the real location.

In the context of object detection, approaches based on sliding windows typically produce multiple windows with high scores close to the correct location of objects. This is a consequence of the generalization ability of object detectors, the smoothness of the response function and visual correlation of close-by windows. This relatively dense output is generally not satisfying for understanding the content of an image. As a matter of fact, the number of window hypotheses at this step is simply uncorrelated with the real number of objects in the image. The goal of NMS is therefore to retain only one window per group, corresponding to the precise local maximum of the response function, ideally obtaining only one detection per object. Consequently, NMS also has a large positive impact on performance measures that penalize double detections.



(a) The top-scoring box may not be the best fit. (b) It may suppress nearby objects. (c) It does not suppress false positives.

Fig. 11.4: Examples of possible failures when using a greedy procedure for NMS.

The most common approach for NMS consists of a greedy iterative procedure, which we refer to as Greedy NMS. The procedure starts by selecting the best scoring window and assuming that it indeed covers an object. Then, the windows that are too close to the selected window are suppressed. Out of the remaining windows, the next top-scoring one is selected, and the procedure is repeated until no more windows remain. This procedure involves defining a measure of similarity between windows and setting a threshold for suppression. These definitions vary substantially from one work to another, but typically they are manually designed. Greedy NMS, although relatively fast, has a number of downsides, as illustrated in Fig. 1. First, by suppressing everything within the neighborhood with a lower confidence, if two or more objects are close to each other, all but one of them will be suppressed. Second, Greedy NMS always keeps the detection with the highest confidence even though in some cases another detection in the surrounding might provide a better fit for the true object. Third, it returns all the

bounding-boxes which are not suppressed, even though many could be ignored due to a relatively low confidence or the fact that they are sparse in a subregion within the image.
As these problems are due to greediness and hard-thresholding, in this paper we propose to consider NMS as a clustering problem that is solved globally, where the hard decisions taken by Greedy NMS are replaced with soft penalties in the objective function.

Regression

In statistical modelling, regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables – that is, the average value of the dependent variable when the independent variables are fixed. Less commonly, the focus is on a quantile, or other location parameter of the conditional distribution of the dependent variable given the independent variables. In all cases, the estimation target is a function of the independent variables called the regression function. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function which can be described by a probability distribution.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable, for example, correlation does not imply causation.

Many techniques for carrying out regression analysis have been developed. Familiar methods such as linear regression and ordinary least squares regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data. Nonparametric regression refers to techniques that allow the regression function to lie in a specified set of functions, which may be infinite-dimensional.

The performance of regression analysis methods in practice depends on the form of the data generating process, and how it relates to the regression approach being used. Since the true form of the data-generating process is generally not known, regression analysis often depends to some extent on making assumptions about this process. These assumptions are sometimes testable if a sufficient quantity of data is available. Regression models for prediction are often useful even when the assumptions are moderately violated, although they may not perform optimally. However, in many applications, especially with small effects or questions of causality based on observational data, regression methods can give misleading results.

In a narrower sense, regression may refer specifically to the estimation of continuous response variables, as opposed to the discrete response variables used in classification. The case of a

continuous output variable may be more specifically referred to as metric regression to distinguish it from related problems.

Linear regression

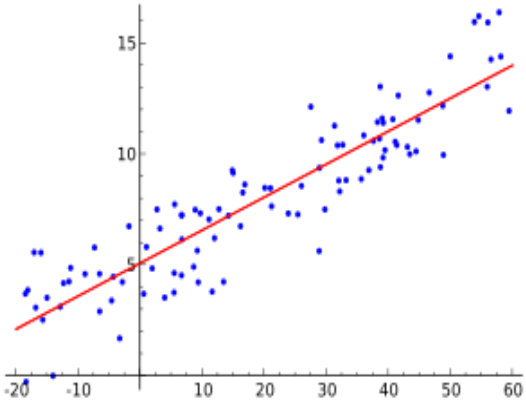


Fig. 11.5 Linear Regression

In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable *y* and one or more explanatory variables (or independent variables) denoted *X*. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*. (This term should be distinguished from *multivariate linear regression*, where multiple correlated dependent variables are predicted, rather than a single scalar variable.)

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.[3] Most commonly, the conditional mean of *y* given the value of *X* is assumed to be an affine function of *X*; less commonly, the median or some other quantile of the conditional distribution of *y* given *X* is expressed as a linear function of *X*. Like all forms of regression analysis, linear

regression focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

- If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.
- Given a variable y and a number of variables X_1, ..., X_p that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the X_j, to assess which X_j may have no relationship with y at all, and to identify which subsets of the X_j contain redundant information about y.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the

least squares approach can be used to fit models that are not linear models. Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.

Polynomial Regression

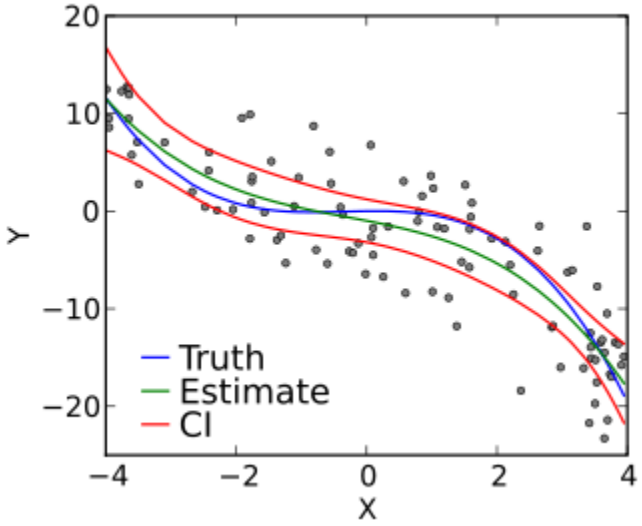


Fig.11.6: Quadratic Regression

n statistics, polynomial regression is a form of linear regression in which the relationship between the independent variable *x* and the dependent variable *y* is modelled as an *n*th degree polynomial. Polynomial regression fits a nonlinear relationship between the value of *x* and the corresponding conditional mean of *y*, denoted E(*y* | *x*), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics. Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(*y* | *x*) is linear in the unknown parameters that are estimated from the data. For this reason, polynomial regression is considered to be a special case of multiple linear regression.

The predictors resulting from the polynomial expansion of the "baseline" predictors are known as interaction features. Such predictors/features are also used in classification settings.[4]

Weighted Regression

The method of ordinary least squares assumes that there is constant variance in the errors (which is called homoscedasticity). The method of weighted least squares can be used when the ordinary least squares assumption of constant variance in the errors is violated (which is called heteroscedasticity). The model under consideration is

$$Y=X\beta+\epsilon,Y=X\beta+\epsilon,$$

where now $\epsilon\epsilon$ is assumed to be (multivariate) normally distributed with mean vector 0 and non constant variance-covariance matrix

$$\begin{pmatrix} \sigma 21 & 0 & \cdots & 0 \\ 0 & \sigma 22 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma 2n \end{pmatrix}.(\sigma 120\ldots 00\sigma 22\ldots 0\vdots\vdots\ddots\vdots 00\ldots \sigma n2).$$

If we define the reciprocal of each variance, $\sigma 2i\sigma i2$, as the weight, $wi=1/\sigma 2iwi=1/\sigma i2$, then let matrix W be a diagonal matrix containing these weights:

$$W=\begin{pmatrix} w1 & 0 & \cdots & 0 \\ 0 & w2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & wn \end{pmatrix}.W=(w10\ldots 00w2\ldots 0\vdots\vdots\ddots\vdots 00\ldots wn).$$

The weighted least squares estimate is then

$$\beta\hat{}WLS=\arg\min\beta\sum i=1n\epsilon*2i=(XTWX)-1XTWY.\beta\hat{}WLS=\arg\min\beta\sum i=1n\epsilon i*2=(XTWX)-1XTWY.$$

With this setting, we can make a few observations:

- Since each weight is inversely proportional to the error variance, it reflects the information in that observation. So, an observation with small error variance has a large weight since it contains relatively more information than an observation with large error variance (small weight).
- The weights have to be known (or more usually estimated) up to a proportionality constant.

Background Subtraction

Background subtraction, also known as Foreground Detection, is a technique in the fields of image processing and computer vision wherein an image's foreground is extracted for further processing (object recognition etc.). Generally an image's regions of interest are objects (humans, cars, text etc.) in its foreground. After the stage of image preprocessing (which may include image denoising, post processing like morphology etc.) object localisation is required which may make use of this technique. Background subtraction is a widely used approach for detecting moving objects in videos from static cameras. The rationale in the approach is that of detecting the moving objects from the difference between the current frame and a reference frame, often called “background image”, or “background model”. Background subtraction is mostly done if the image in question is a part of a video stream. Background subtraction provides important cues for numerous applications in computer vision, for example surveillance tracking or human poses estimation. However, background subtraction is generally based on a static background hypothesis which is often not applicable in real environments. With indoor scenes, reflections or

animated images on screens lead to background changes. In a same way, due to wind, rain or illumination changes brought by weather, static backgrounds methods have difficulties with outdoor scenes.

Frame Differencing

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Here using simple arithmetic calculations, we can segment out the objects simply by using image subtraction technique of computer vision meaning for each pixels in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B].

In mathematical equation, it is written as:

$$P[F(t)] = P[I(t)] - P[B]$$

The background is assumed to be the frame at time *t*. This difference image would only show some intensity for the pixel locations which have changed in the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static. A threshold "Threshold" is put on this difference image to improve the subtraction (see Image thresholding).

$$|P[F(t)] - P[F(t+1)]| > \text{Threshold}$$

This means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Threshold. [4] The accuracy of this approach is dependent on speed of movement in the scene. Faster movements may require higher thresholds.

Color conversion

Threshold is an image segmentation to convert grayscale to binary image. During the threshold process, individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label.

Contrast Limited Adaptive Histogram Equalization (CLAHE)

While performing AHE if the region being processed has a relatively small intensity range then the noise in that region gets more enhanced. It can also cause some kind of artifacts to appear on those regions. To limit the appearance of such artifacts and noise, a modification of AHE called Contrast Limited AHE can be used. The amount of contrast enhancement for some intensity is directly proportional to the slope of the CDF function at that intensity level. Hence contrast enhancement can be limited by limiting the slope of the CDF. The slope of CDF at a bin location is determined by the height of the histogram for that bin. Therefore if we limit the height of the

histogram to a certain level we can limit the slope of the CDF and hence the amount of contrast enhancement.

The only difference between regular AHE and CLAHE is that there is one extra step to clip the histogram before the computation of its CDF as the mapping function is performed

Following is the overview of the algorithm for this function:

1. Calculate a grid size based on the maximum dimension of the image. The minimum grid size is 32 pixels square.
2. If a window size is not specified chose the grid size as the default window size.
3. Identify grid points on the image, starting from top-left corner. Each grid point is separated by grid size pixels.
4. For each grid point calculate the histogram of the region around it, having area equal to window size and centered at the grid point.
5. If a clipping level is specified clip the histogram computed above to that level and then use the new histogram to calculate the CDF.
6. After calculating the mappings for each grid point, repeat steps 6 to 8 for each pixel in the input image.
7. For each pixel find the four closest neighboring grid points that surround that pixel.
8. Using the intensity value of the pixel as an index, find its mapping at the four grid points based on their cdfs.
9. Interpolate among these values to get the mapping at the current pixel location. Map this intensity to the range [min:max) and put it in the output image.

Clipping the histogram itself is not quite straight-forward because the excess after clipping has to be redistributed among the other bins, which might increase the level of the clipped histogram. Hence the clipping should be performed at a level lower than the specified clip level so that after redistribution the maximum histogram level is equal to the clip level.

Contour Tracing

Contour tracing is one of many preprocessing techniques performed on digital images in order to extract information about their general shape. It is a technique that is applied to digital images in order to extract their boundary. Once the contour of a given pattern is extracted, it's different characteristics will be examined and used as features which will later on be used in pattern classification. Therefore, correct extraction of the contour will produce more accurate features which will increase the chances of correctly classifying a given pattern.

the contour pixels are generally a small subset of the total number of pixels representing a pattern. Therefore, the amount of computation is greatly reduced when we run feature extracting algorithms on the contour instead of on the whole pattern. Since the contour shares a lot of features with the original pattern, the feature extraction process becomes much more efficient when performed on the contour rather on the original pattern.

In conclusion, contour tracing is often a major contributor to the efficiency of the feature extraction process -an essential process in the field of pattern recognition.

Minimum Enclosing Circle Algorithm

The smallest-circle problem or minimum covering circle problem is a mathematical problem of computing the smallest circle that contains all of a given set of points in the Euclidean plane. The

corresponding problem in *n*-dimensional space, the smallest bounding-sphere problem, is to compute the smallest *n*-sphere that contains all of a given set of points.

Algorithm:

1. Draw any circle around all the points. This circle can clearly be made smaller.
2. Make your circle smaller by finding the point A farthest from your circle's center, and drawing a new circle with the same center and passing through the point A. This produces a smaller enclosing circle, since it still contains all the points, but now passes through A rather than enclosing it.
3. If the circle passes through 2 or more points, proceed to step 4. Otherwise, make the circle smaller by moving its center towards point A, until the circle makes contact with another point B from the set.
4. At this stage, you will have a circle, C, passing through two or more points from the set. If the circle contains an interval of arc greater than half the circle's circumference on which no points lie, the circle can be made smaller. We will call such an interval a *point-free interval*. Let D and E be the points on the ends of this point-free interval. While keeping D and E on the circle's boundary, reduce the diameter of the circle until
    1. the diameter is the distance DE, or
    2. the circle C touches another point from the set, F.
5. In case a), we are finished. In case b), we must check whether there are now no point-free arc intervals of length more than half the circumference of C. If there are none, we are done. If there does exist such a point-free interval, we need to repeat step 4. In this case, three points must lie on an arc less than half the circumference in length. We repeat step 4 on the outer two of the three points on the arc.

IMPLEMENTATION

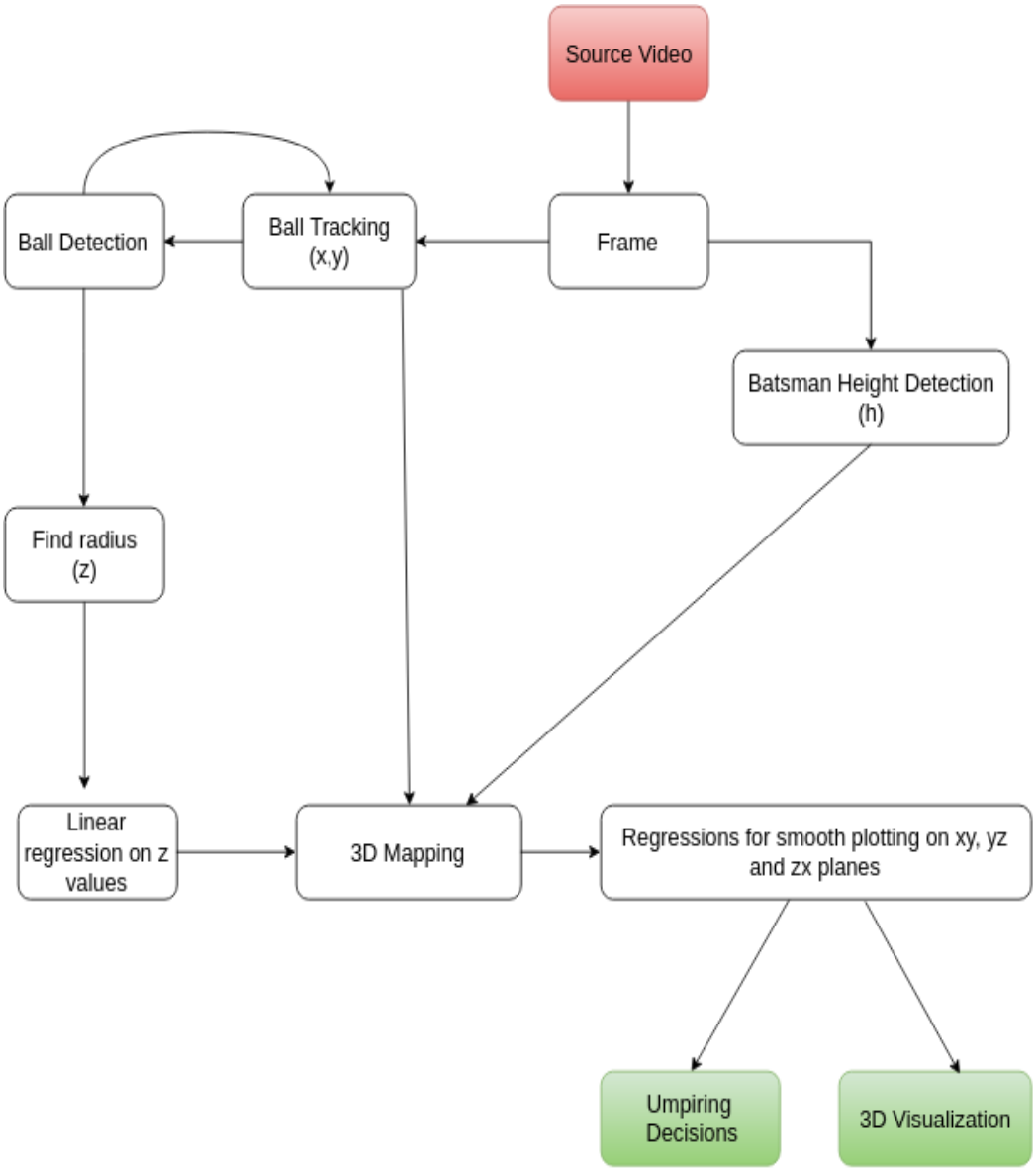


Fig. 12.1: Flowchart of the Process

Python

Python is a widely used high-level, general-purpose,interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles.

Python is used for the implementation of this project as it provides various utilities and modules and integrates seamlessly with third party libraries like OpenCV, NumPy , VPython for computer vision related functionalities.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, etc.

OpenCV and its features are used for detection and tracking of the cricket ball and batsman, which is the core feature of the project.

Ball Detection

Proposed Algorithm

The cricket ball is detected using Support Vector Machine (SVM) and Histogram of Oriented Gradient (HOG). Positive and negative data samples are collected and used for building the SVM models for HOG objects (ball, batsman).

The video is sliced into multiple image frames separated by a fixed time duration. Frame difference technique is used to find areas which may have a high possibility of having a ball. Each window frame, after converting to grayscale, is used to extract HOG features and fed to the SVM model to detect the ball in the areas found to be different in subsequent frames by the frame subtraction method.

Training Data Preparation

Data samples are collected by manually scraping through the captured videos and taking screenshots of relevant objects. There are two types of samples: negative and positive. Negative samples correspond to non-object images. Positive samples correspond to images with detected objects.

Negative Samples

Negative samples are taken from arbitrary images. These images must not contain detected objects. Note that negative samples and sample images are also called background samples or background images, and are used interchangeably in this document.

After taking a few negative samples, we apply hard negative mining i.e manually including all the false positive detections made by the SVM into the negative data sample set. A user interface was developed for the same which made the task easier and more efficient.

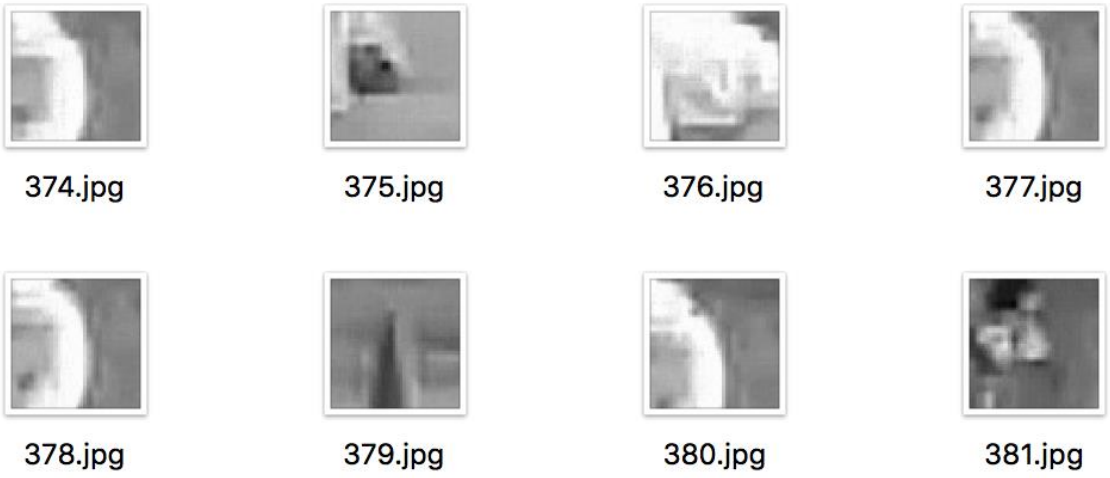


Fig. 12.2: Negative Dataset

Positive Samples

Positive samples are collected by taking screenshots of a standardized size of the cricket ball from the captured video data. The positive sample set basically consists of large number of images of a cricket ball differing in various parameters like orientation, color intensity, brightness, background etc.

Please note that you need a large dataset of positive samples. Hundreds and even thousands of positive samples for ball are needed. The object instances are taken as screenshot images from

the videos. Then they are resized to target samples size and stored in the positive sample set. No distortion is applied.

Fig. 12.3: Positive Dataset

Histogram of Oriented Gradients (HOG)

OpenCV provides the functionality of extracting the HOG features of images using the various parameters mentioned below.

- Minimum Window Size (min_wdw_sz) signifies the size of the window of which the HOG features are to be extracted.
- Step Size (step_size) indicates the amount by which the window moves in one iteration.
- Pixels Per Cell (pixels_per_cell) is the number of pixels that are there in a cell.
- Cells per block (cells_per_block) is the number of cells in one block of the window.

Building the SVM

HOG features are extracted for the collected positive and negative data samples. The SVM model is built using the OpenCV SVM library using the extracted HOG features of the positive and negative data sample sets. As a result, we get a SVM model which is used to classify objects into those that are a ball and those that are not.

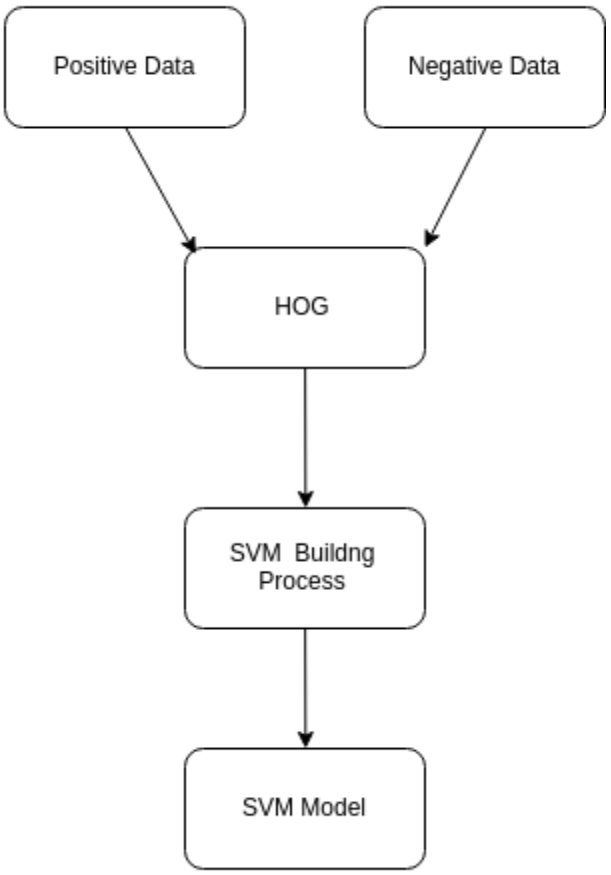


Fig. 12.4 Building the SVM Model

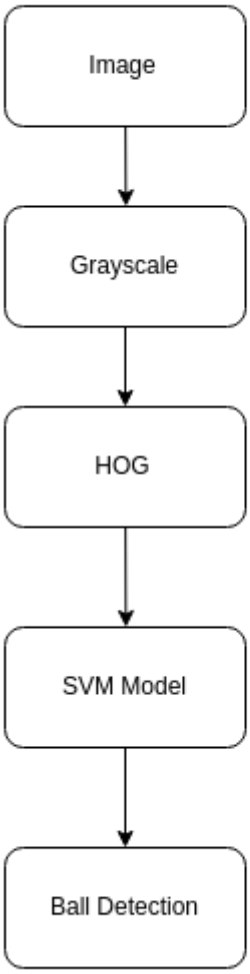


Fig. 12.5 Ball Detection Process

Ball Tracking

Sliding Window Approach

The window where the ball is detected in a particular frame is used in the subsequent frame as a reference and the SVM model is used to detect the presence of the ball in an inflated window in the neighbourhood of the previous frame's window. This ensures higher accuracy in ball detection.

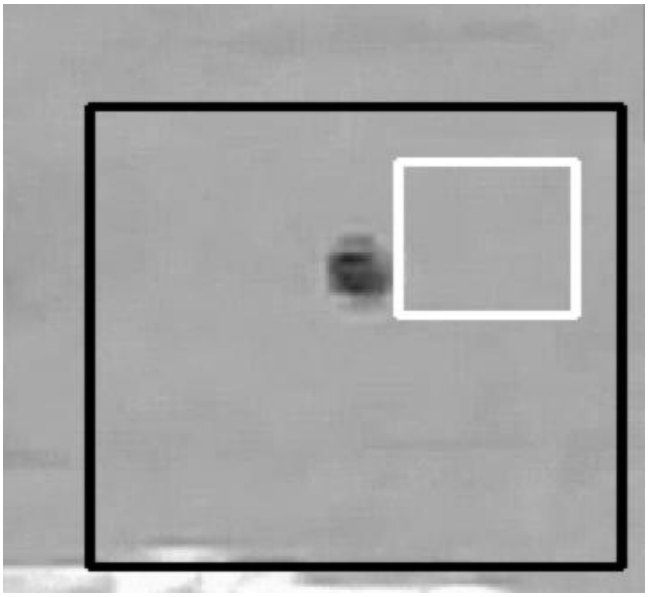


Fig. 12.6: Sliding window in progress

The obtained position of the ball is stored as a set of ‘x’ and ‘y’ coordinates in a separate file.

Bounce Point Detection

The point where the ball bounces is the point where the ‘y’ coordinate is the least.

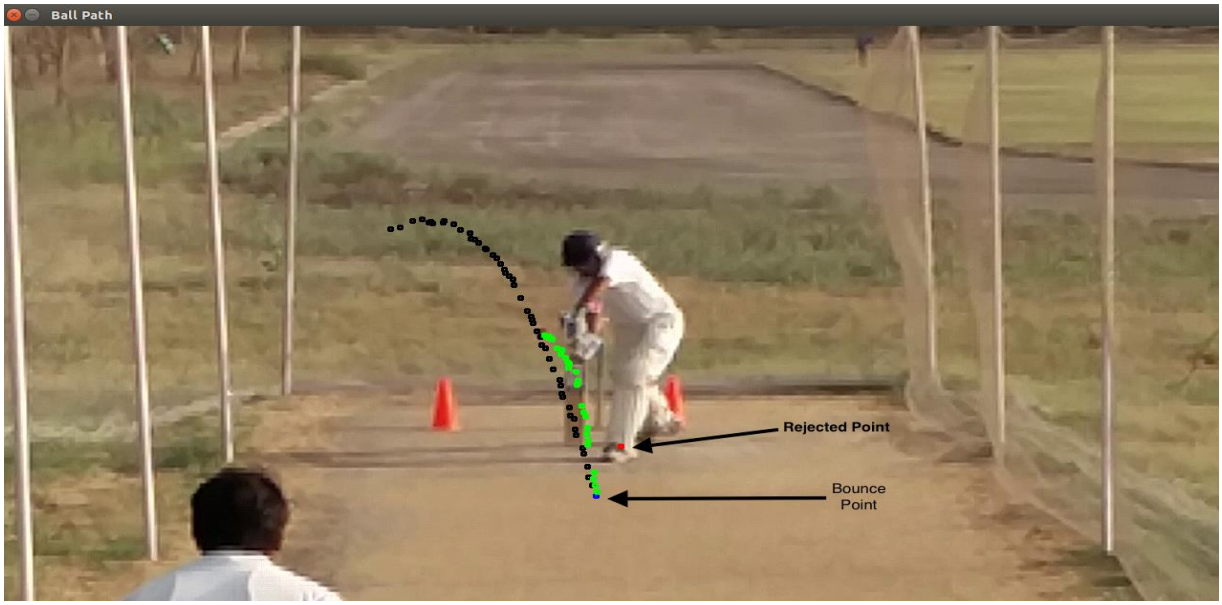


Fig. 12.7: Bounce point detection

Rejection of False Detections

Consecutive detections separated by a value less than a maximum distance are included in our correct detections.

If the distance is greater than a particular value, the difference of angle between subsequent points on the trajectory path followed by the ball is used, and false detections are rejected using maximum permissible value of this angle.

Further, some points are rejected while mapping to 3D coordinates based on the THRESHOLD values and the (x,y,radius) of the detected ball.

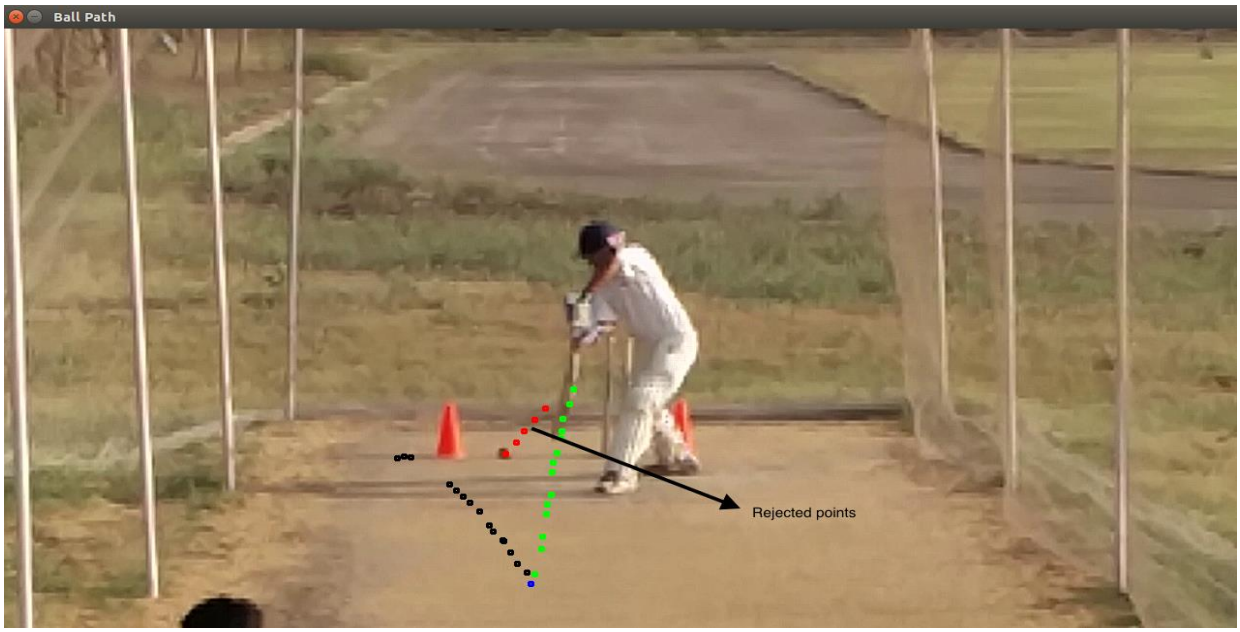


Fig. 12.8 : Red dots indicating rejected points

Detection of Ball Hitting the Bat or Batsman

The above mentioned maximum distance and angle difference methods are used to detect drastic changes in path due to impact of the bat or batsman's body on the ball. All detections made after that would be false and hence are rejected.

Batsman Detection and Tracking

The batsman is detected using the same approach used for ball detection. OpenCV's built in People Detector SVM models are used.

Further, NMS is applied on the detected objects to obtain the final result.

The movement of the batsman is tracked using the same approach used for ball tracking, ie, limiting the search window to the neighbourhood of first detection, and the detected batsman window is displayed on the frame.

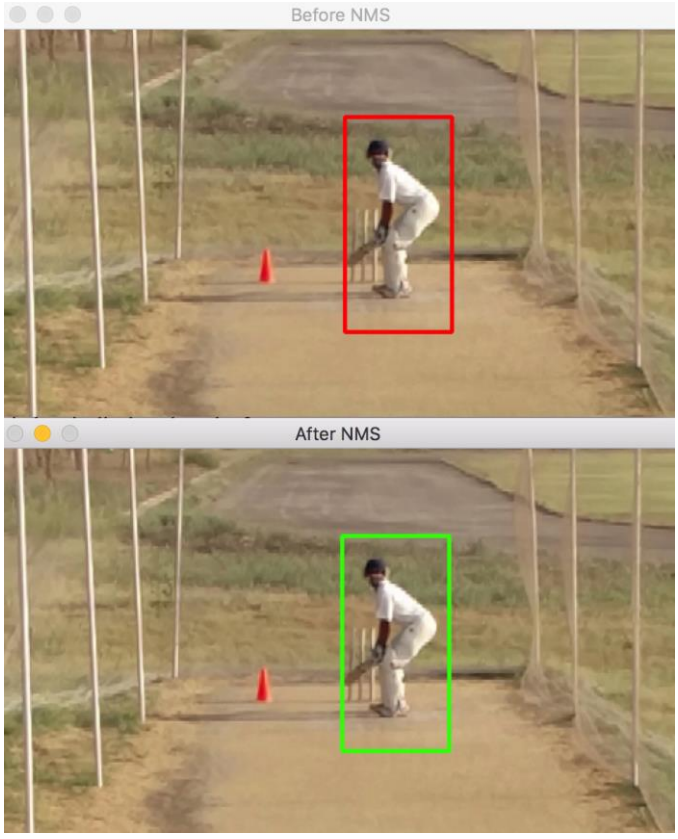


Fig. 12.9: The tracked batsman before and after NMS

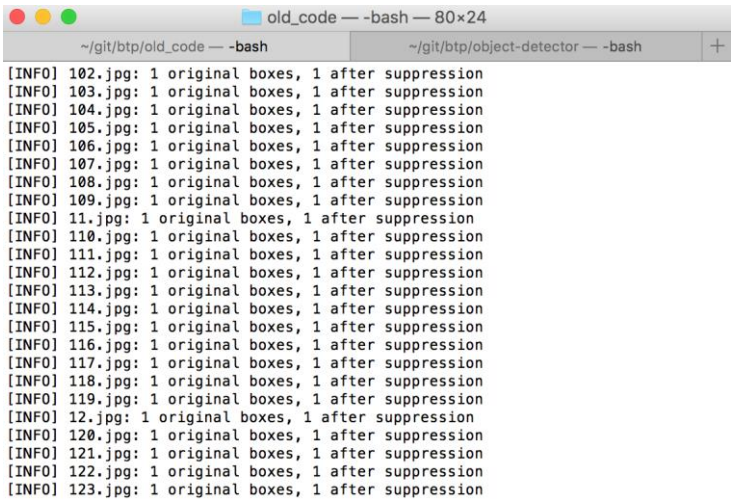


Fig. 12.10: Tracked batsman log across frames

Mapping 2D Image coordinates to 3D world coordinates

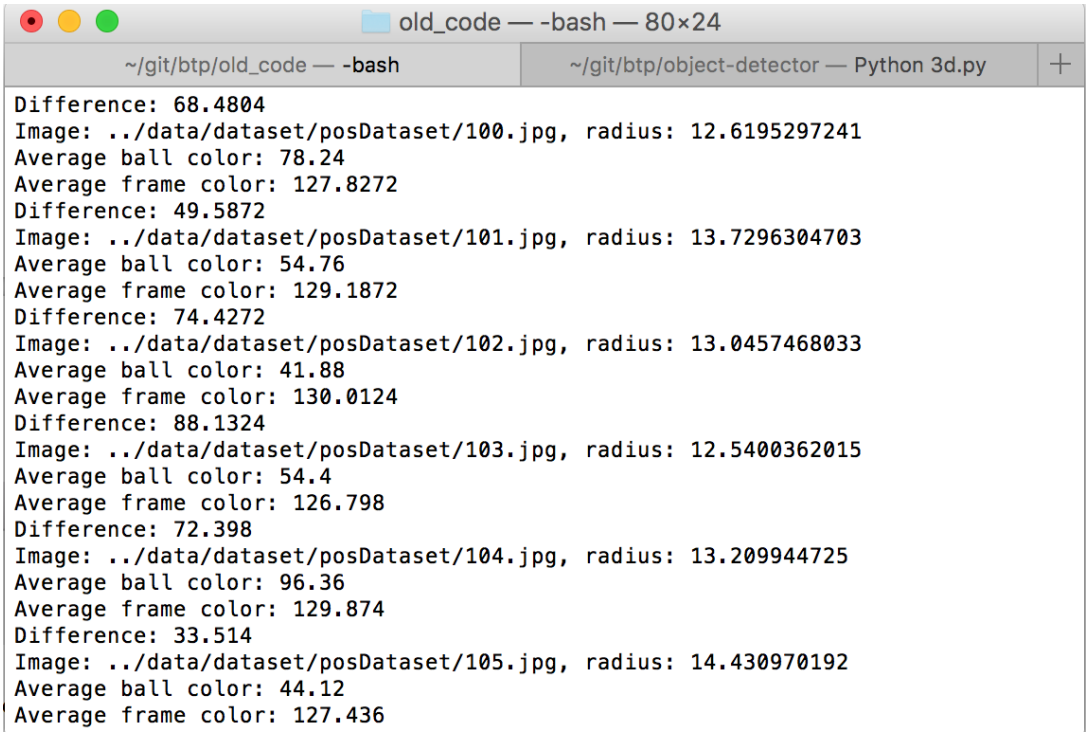


Fig. 12.11: Finding x,y and radius of the tracked ball

1. The x,y coordinates are obtained from the tracked ball using the **Minimum Enclosing Circle** algorithm after a series of image transformations.

    a. The tracked ball window is cropped and converted to Grayscale.

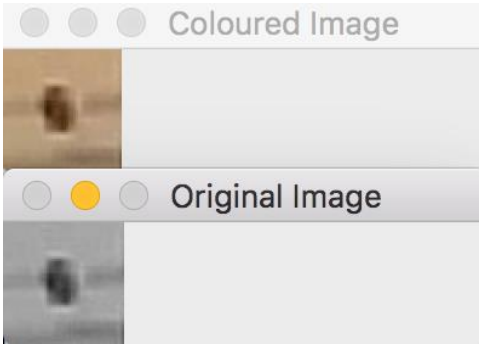


Fig. 12.12: The tracked ball in color and converted to grayscale

    b. CLAHE is used to improve the contrast of the image and the image is blurred using Gaussian Blur technique.

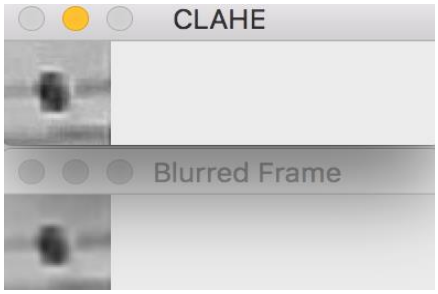


Fig. 12.13: Result of CLAHE algorithm and Gaussian blurring

    c. The average colour around the center of the frame is found and used to find a suitable THRESHOLD brightness value to extract the ball from the frame. The frame is then thresholded by setting intensity of the pixels with value below THRESHOLD to 1, and those with value above THRESHOLD to 0.

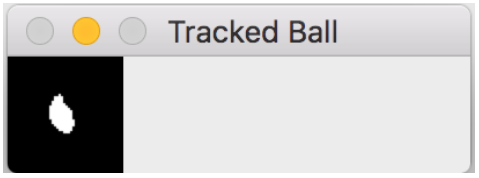


Fig. 12.14: Result of thresholding

d. Contour detection is now used the find the boundary of the tracked ball and the maximum sized contour closest to the center of the frame is selected.

Fig. 12.15: Detected Contours

e. Minimum enclosing circle algorithm is now applied on the detected contour to find the centre and radius of the tracked ball.

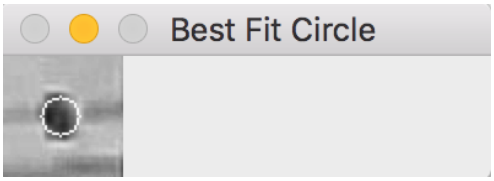


Fig. 12.16: Minimum enclosing circle shown on blurred image

2. The obtained (x,y) image coordinates are scaled to world coordinates according to the cricket pitch dimensions.
3. The z-coordinate (depth) is obtained the comparing the radius of the tracked ball with the radius obtained at the beginning of the pitch vs the radius at the end of the pitch, and scaling this ratio to the pitch length.
4. The scaled (x,y) coordinates are transformed using perspective projection to obtain the real world (x,y) coordinates.
5. Weighted linear and quadratic regression is applied across various dimensions to smoothen out the obtained coordinates for better visualization.
6. The trajectory of the ball is predicted further using the polynomial equations obtained via regression and the predicted ball coordinates are found.

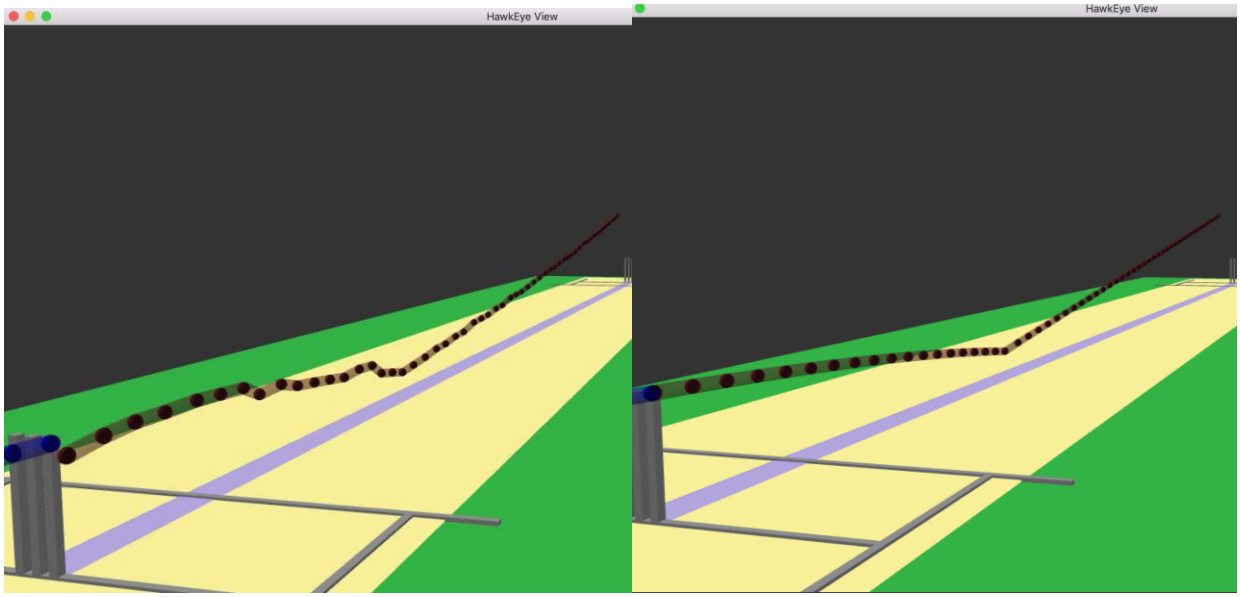


Fig 12.17 : Visualization without (L) and with (R) regression

Umpiring Decisions

Information related to the ball and batsman is calculated using the obtained ball coordinates and batsman coordinates.

- The speed of the ball is found by taking an average of the speed of the ball before the bouncing point (if available) or after.
- The height of the batsman is found by finding the midpoint of the detected batsman window, mapping the midpoint to real world coordinates, and taking a double of the real world height.

The obtained coordinates and other information is then used for umpiring decisions.

Leg Before Wicket (LBW) Decision

1. The impact location of the ball with the batsman is checked to see if it is inside the impact zone.
2. The pitching location of the ball is found.
3. The location of the ball when it is nearest to the wickets is found from the predicted ball trajectory and this location is then checked to see if the ball would hit the wickets.

These three conditions are checked to arrive at the final LBW decision.

Wide Decision

1. The location of the ball when it is nearest to the batsman's wicket crease is found.
2. If the location is towards the leg side of the batsman, it is marked as wide.
3. If the location is towards the off side of the batsman, the location is compared with the location of the wide crease line.
4. If the height of the ball near the crease is greater than the height of the batsman, it is marked as wide.

No Ball Decision

1. The location of the ball when it is nearest to the batsman's wicket crease is found.
2. If the ball hasn't bounced off the ground, and the height of the ball is greater than half of the batsman's height, it is marked as a no-ball.

Bouncer Decision

1. The location of the ball when it is nearest to the batsman's wicket crease is found.

2. If the ball has a valid bounce point, and the height of the ball is greater than shoulder height of the batsman, it is marked as a bouncer.

3D Visualization of Tracked Objects and Cricket Pitch

Using the object (x,y,z) coordinates obtained by mapping algorithm, the objects are visualized in a 3D space using VPython visualization library.

VPython

VPython is the Python programming language plus a 3D graphics module called Visual. VPython allows users to create objects such as spheres and cones in 3D space and displays these objects in a window. This makes it easy to create simple visualizations, allowing programmers to focus more on the computational aspect of their programs. The simplicity of VPython has made it a tool for the illustration of simple physics, especially in the educational environment.

Visualization

1. The pitch and the wickets are drawn by using boxes in VPython, taking ideal pitch measurements.
2. The tracked ball is displayed on the screen using spheres with a cylindrical trajectory between them.
3. The trajectory of the ball is predicted further using the polynomial equations obtained via regression and the predicted ball positions are displayed on the screen using spheres of different colour than the detected balls.
4. The various decision parameters are shown on the screen.

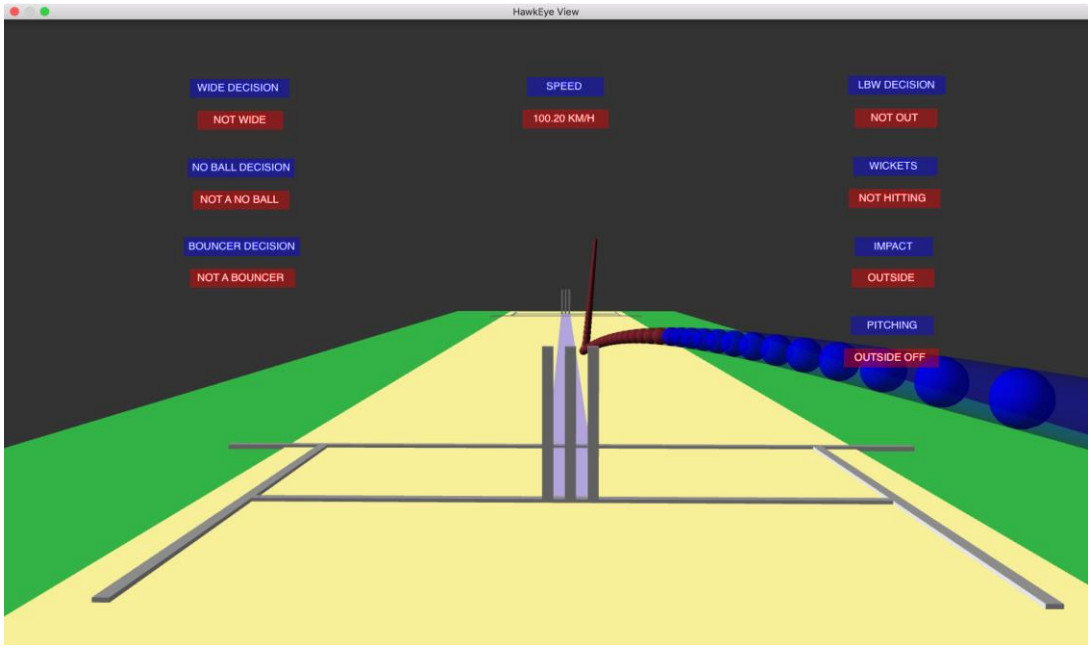


Fig. 12.18: The visualization result

Visualization in Stereoscopic 3D

VPython allows visualization of the results in passive stereoscopic 3D as shown below.

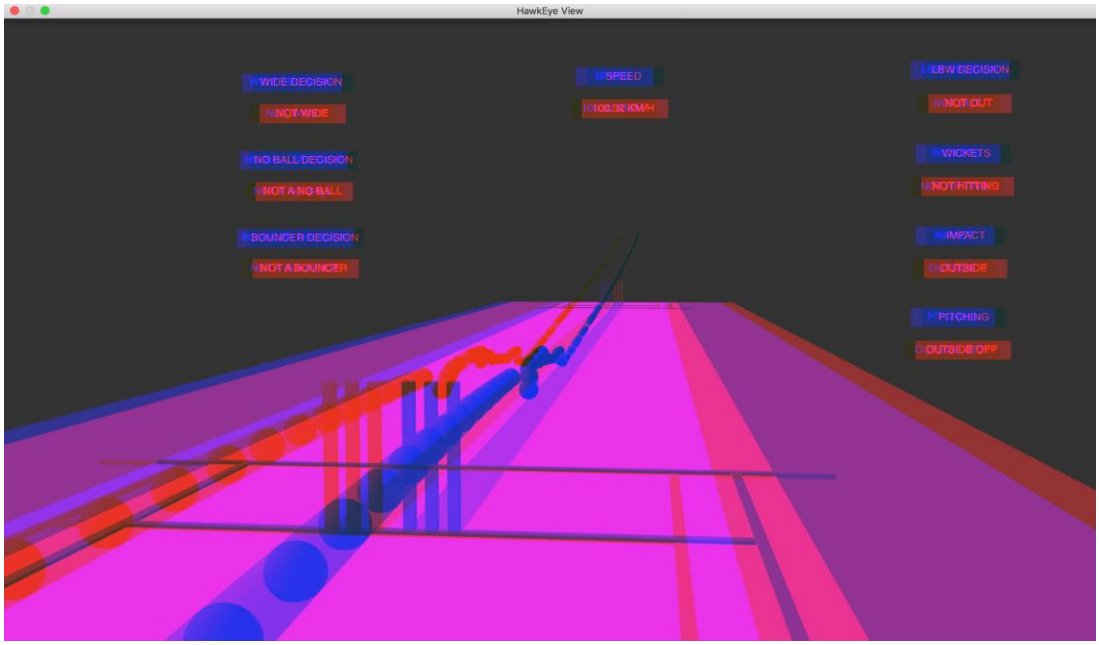


Fig. 12.19: Stereoscopic 3D visualization in ‘red-blue’

Graphical User Interface for the Application

A GUI for the project was developed using the Kivy python framework.

Kivy

Kivy is an open source (pronounced as Kiwi) Python library for developing mobile apps and other multi touch application software with a natural user interface (NUI). It can run on Android, iOS, Linux, OS X, and Windows. Distributed under the terms of the MIT license, Kivy is free and open source software.

Kivy is the main framework developed by the Kivy organization, alongside Python for Android, Kivy iOS, and several other libraries meant to be used on all platforms. Kivy also supports the Raspberry Pi which was funded through Bountysource.

The framework contains all the elements for building an application such as:

- extensive input support for mouse, keyboard, TUIO, and OS-specific multitouch events,
- a graphic library using only OpenGL ES 2, and based on Vertex Buffer Object and shaders,
- a wide range of Widgets that support multi touch,
- an intermediate language (Kv) used to easily design custom Widgets.

Kivy is the evolution of the PyMT project, and is recommended for new projects.

Kv Language

The Kivy language (Kv) is a language dedicated to describing user interface and interactions. As with QML, it is possible to easily create a whole UI and attach interaction. For example, to create a Loading dialog that includes a file browser, and a Cancel / Load button, one could first create the base widget in Python, and then construct the UI in Kv.

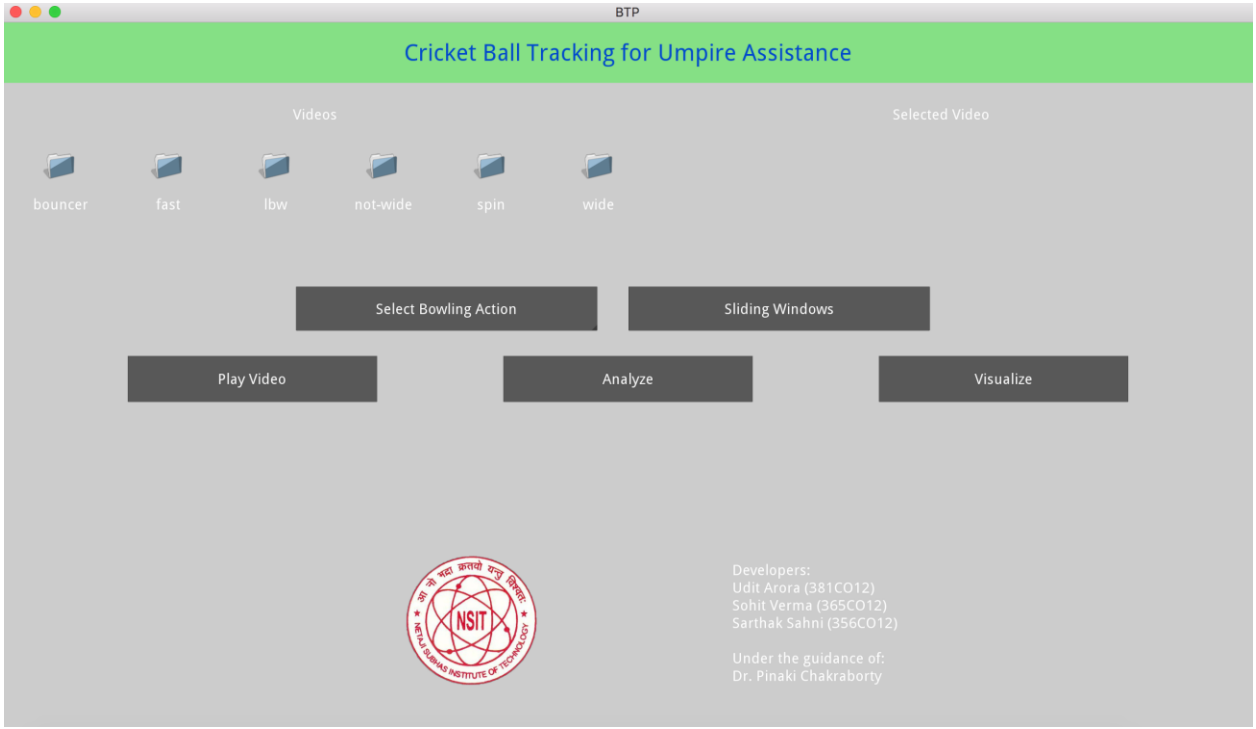


Fig. 12.20: The GUI on first load

Implementation

- The **Videos** section allows selection of a video to analyze and visualize.
- The selected video's full name is displayed under the heading **'Selected Video'**
- The **Select Bowling Action** dropdown button is used to choose the action between fast, slow and no-action for debugging purposes.

- The **Sliding Windows** toggle button when toggled shows the sliding windows in progress during ball detection.
- The **Play Video** button plays the selected video using a video player.
- The **Analyze** button activates the script that analyzes the video and finds the image coordinates of the tracked ball and player across frames.
- The **Visualize** button activates the script that visualizes the results of the analysis in 3D.

Fig. 12.21: Video analysis in progress

CONCLUSION

The main goal of this thesis was to develop a product for assisting the umpire in the sport of cricket while making decisions, using a single camera. The thesis involved the development of algorithms using computer vision and machine learning techniques for ball detection and tracking, along with various cricket decision making rules.

In this thesis a thorough overview of the fundamentals of computer vision was presented, including the conclusions of historical research and newly proposed techniques. The thesis discusses the use of computer vision to detect, identify and track the cricket ball (and other relevant objects in the context of cricket), and machine learning techniques to optimize and further predict various results and decisions.

Through our investigations we have observed that in the past few decades, researchers and engineers have developed quite a few technologies for assisting the umpire in making decisions. But the high cost and heavy technological requirements (high quality hardware like expensive high resolution cameras, microphones etc.). of these technologies restricts their use in any matches, competitions and training academies other than the ones operating at an international level. The use of just one camera, which may be of a quality equivalent to modern day smartphone cameras, along with the various algorithms and techniques of Computer Vision and Machine Learning helped us achieve a system that reliably assists the umpire and operates at a cheap cost.

The feature Histogram of Oriented Gradients (HOG) is implemented along with a Support Vector Machine (SVM) model to classify and detect the ball and batsman in a window frame. OpenCV and its various image processing features, with algorithmic techniques like frame

subtraction, sliding windows, CLAHE, minimum enclosing circle and machine learning techniques of weighted regression are implemented to achieve accurate tracking of the ball in motion.

Various umpiring decisions are made by checking specific conditions on the data obtained in accordance with the rules of the sport of cricket and the decisions involved.

VPython module is used to represent the decisions in the form of interactive 3D visualisations showing the ball trajectory path and information about the umpiring decisions.

Kivy is used to provide an interactive graphical user interface to access and use the tool.

FUTURE WORK

In our thesis, we have observed that the HOG based SVM classifier coupled with regression optimisations and computer vision techniques provide fairly accurate results.

However, from a practical point of view perhaps a serious problem with the project is that the tracking results take a long time to compute. To reduce the computational overload due to raw image processing, we can use better object detection algorithms and tools.

The capabilities of the project may be enhanced such that it may produce accurate results in varying environments. Depth analysis may also be done using stereo cameras, which will increase the functionality of the product and make it useful in real time cricket matches and practice sessions. Another of our future objectives is to try to use multiple cameras and microphones to include decisions like ‘front foot no-ball’, ‘run out’, ‘edged and caught behind’, without increasing the computational and cost overheads significantly.

To improve the tracking prediction results, Kalman Filters can be applied. The calculation method of batsman’s height can be bettered and it can also be detected whether the batsman is right handed or left handed. Mapping of 2D image coordinates to real world 3D coordinates can be improved by using better camera calibration methods.

REFERENCES

[1] Kumar, P. Ashok. "Automated Decisions in Aware Cricket Ground: A Proposal." *International Journal of Information and Education Technology* 2.3 (2012): 188.

[2] Khan, Wazir Zada, Mohammed Y. Aalsalem, and Quratul Ain Arshad. "The Aware Cricket Ground." *arXiv preprint arXiv:1109.6199* (2011).

[3] Kumar, Ajit, et al. "3D Estimation and Visualization of Motion in a Multicamera Network for Sports." *Machine Vision and Image Processing Conference (IMVIP), 2011 Irish*. IEEE, 2011.

[4] Bristow, Hilton, and Simon Lucey. "Why do linear SVMs trained on HOG features perform so well?." *arXiv preprint arXiv:1406.2419* (2014).

[5] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[6] http://web.mit.edu/vondrick/ihog/#

[7] Yan, Fei, W. Christmas, and Josef Kittler. "A tennis ball tracking algorithm for automatic annotation of tennis match." *British machine vision conference*. Vol. 2. 2005.

[8] Rock, R., et al. "The 5 th umpire: Automating crickets edge detection system." *Journal of Education, Informatics & Cybernetics* 11.1 (2013).

[9] Mao, Jinzi. "Tracking a tennis ball using image processing techniques." (2006).

[10] http://www.cse.psu.edu/~rtc12/CSE486/lecture12.pdf

[11] http://www.cse.psu.edu/~rtc12/CSE486/lecture13.pdf

[12] Rothe, Rasmus, Matthieu Guillaumin, and Luc Van Gool. *Non-maximum suppression for object detection by passing messages between windows*. Springer International Publishing, 2014.

[13] docs.opencv.org

[14] http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

[15] Efrat, Alon, Micha Sharir, and Alon Ziv. "Computing the smallest k-enclosing circle and related problems." Computational Geometry 4.3 (1994): 119-136

[16] Martelli, Alberto. "An application of heuristic search methods to edge and contour detection." Communications of the ACM 19.2 (1976): 73-83