Stereo Tracking of Objects with respect to a Ground Plane

November 5, 2009

Greg Baguley
glb32@student.canterbury.ac.nz
Department of Computer Science and Software Engineering
University of Canterbury, Christchurch, New Zealand

Supervisor: Dr. Richard Green
richard.green@canterbury.ac.nz

Abstract

In this paper, a novel algorithm for tracking 3D cricket ball locations in relation to an estimated ground plane is proposed. This algorithm proposes using Random Sample Consensus to find a ground plane from disparity images, and then estimating 3D ball positions with respect to the ground plane using fast contour tracing and stereo depth values. In that no prior research has used stereo depth values to locate a ground plane and track an object with respect to that plane, the 90.7% successful tracking rate with only 1.95% false positives suggests that the proposed novel algorithm is a useful contribution for enabling automatic setup and ease of use for such tracking.

Keywords

Tracking, plane estimation, sports, disparity, stereo, OpenCV, image segmentation.

Acknowledgments

I would like to thank my supervisor, Richard Green for both suggesting this project and being extremely supportive of me the entire way through. I would like to thank Gary Bradski, Adrian Kaehler and Vadim Pisarevsky for their work with OpenCV and providing detailed images of the process behind stereo vision, which I have used in this report.

Contents

List of Figures

1 Introduction

The goal of this research is to automatically locate the ground plane and track an object with respect to this plane using depth values from a relatively low-cost stereo camera. One application would be the real-time 3D tracking of cricket balls like that of more expensive systems, such as Hawk-Eye [17]. These systems provide a wide range of feedback to the user, such as the path taken by the ball along with its velocity at any point on that path, enabling it to be displayed during action replays or for coaching purposes post-game. These systems are highly effective at tracking the entire sequence of ball movement, from the bowlers release to the path taken after being hit; however, they suffer from many drawbacks. Hawk-Eye's system is based on triangulation of visual images and timing data from at least four high-speed video cameras placed around the cricket field. This system costs the end user in excess of NZ$200,000 for the hardware alone, more for the human operators required each time it is used, and the ground plane needs to be manually calibrated. Our research aims to innovate an algorithm based on techniques used in previous research [7, 24, 30] of non-stereo based sports ball tracking, used in conjunction with additional techniques in computer vision to provide an accurate object-tracking system, which is able to work at near real-time speeds on low-cost hardware that is more readily available to end users.

Our algorithm uses a combination of techniques to provide the final cricket ball tracking in 3D: Disparity generation through the use of Sum of Absolute Difference (SAD) correspondence with additional interpolation techniques, ground plane estimation using Random Sample Consensus (RANSAC), and object tracking through foreground segmentation and contour detection and filtering.

To provide tracking in 3D, we look into disparity image generation and cover two low-cost hardware implementations to provide these images; two Logitech "Pro 5000" web cameras[1] in a stereo configuration, and a Point Grey Research Inc "Bumblebee 2"[2], which is built of two image sensors inside a single case, forming a conventional stereo pair. These hardware implementations were chosen because they are entry level cameras for beginners and professionals at NZ$80 and NZ$2000 respectively. Disparity images provide us with a means to calculate 3D depths of points within an image, relying on cameras set up in a stereo pair for its generation. Before disparity generation can be performed, a camera system needs to be calibrated to calculate its intrinsic and extrinsic parameters that define its internal geometry, and helps provide a relation between 3D world coordinates and 3D camera coordinates. We will discuss the method and theory behind our chosen calibration technique for each of the hardware implementations. We also discuss the process involved in generating disparity images with each system and the advantages, disadvantage, and effectiveness of using each system.

We will consider two techniques to estimating planes within a scene: image segmentation and Random Sample Consensus (RANSAC). Image segmentation provides a method to split an image up into segments based upon their colour or texture, and can be helpful in locating a specific plane. RANSAC is an iterative method used to estimate parameters of a mathematical model from a set of observed data that contains outliers. This technique is often used in mobility devices or for mapping outdoor areas, and we will discuss how it can be used in conjunction with our disparity data to automate the process of finding a ground plane.

Next we will discuss object tracking techniques and provide more detail into the inner workings and results of the Hawk-Eye system. We will discuss the research by Pingali et al [24] into tracking tennis balls through the segmentation of images based upon properties of the tennis ball, such as its colour, and how this does not hold well for cricket ball tracking due to its reflective texture. However, the idea of using foreground segmentation to locate the ball transfers well into our research, helping our approach in both

[1] http://www.logitech.com/index.cfm/435/243&hub=1&cl=us,en

[2] http://www.ptgrey.com/products/bumblebee2/index.asp

performance and accuracy of locating the cricket ball.

Finally, we created a test environment to calculate the accuracy of the 3D depth calculations, ground plane estimation, and cricket ball tracking from our algorithm. These initial results are proving positive for the concept of stereo object tracking.

1.1 Outline

Chapter 2 of this report provides background information into stereo-vision and how it is beneficial to our research. Related work for plane estimation and sports ball object tracking is covered in chapter 3. The design and implementation of our algorithm using the two camera systems is covered in chapter 4. An evaluation of our algorithms 3D depth accuracy and object tracking accuracy is discussed in chapter 5. Discussion of our work and future work is provided in chapter 6 with a final conclusion given in chapter 7.

2 Stereo Vision

In this chapter we will cover the theory behind using stereo vision systems over their mono counterparts, and discuss how it can be beneficial to our research. Stereo vision is a field with its own set of drawbacks and limitations, which will be discussed below. They differ from mono-based systems and require additional techniques in order to overcome these restrictions. We will briefly cover the general process behind stereo vision; a detailed discussion of how we implemented this process is provided in a later chapter.

2.1 Theory

Stereo vision, similar to human binocular vision, provides a method to extract 3D information from a pair of images. In computer systems this is done by stereo correspondence - comparing information about a scene from several camera perspectives of the particular scene to generate disparity images. These disparity images can then be re-projected to 3D depth information using triangulation techniques.

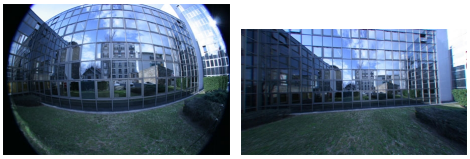
Stereo vision is very beneficial to our research for the 3D information it can provide, to enable locating the ground plane and tracking the 3D trajectory of a cricket ball. This 3D information can be gathered from mono systems through the use of triangulating information about objects gathered from multiple cameras placed in known fixed locations around the field of play, unfortunately they cannot automatically locate the ground plane without extra information. These systems are very expensive for the end user and often require extensive manual set-up and accumulated knowledge dealing with the object they are looking for to generate any tracking information. Using stereo vision allows us to provide the 3D information at a lower-cost, with no manual set-up nor extensive knowledge required.

2.2 Disparity Calculation

The basis of stereo systems is the ability to calculate disparity values for the scene you are viewing through stereo correspondence. There are multiple approaches to finding these disparity values; however there is a very large trade off between the accuracy of the disparity calculations, and the speed of which they take to do the calculations. An evaluation of stereo correspondence algorithm accuracy performed by Scharstein et al [26] led to the creation of the Middlebury Stereo Vision Web page [19] - here techniques are provided for researchers to compare their correspondence algorithm against previous work. At the time of writing this report, Klaus et al [11] have the most accurate correspondence algorithm listed on the Middlebury website, but their algorithm takes between 14-25 seconds per frame, which is no where near the real-time specification that we require. To get closer to this real-time requirement, accuracy has to be lowered for performance to increase, and in later chapters we discuss stereo correspondence algorithms that aim for this performance over accuracy factor.

Before stereo correspondence can be applied to an image pair, they must first go through undistortion and rectification. Undistortion is the process of removing any unwanted lens distortion apparent on the image as seen in Figure 2.1, this is due to the low focal length (wider angle of view) of a lens. Rectification is the process of reprojecting the two image planes from our camera system so that they both lie in the same plane. Doing this provides row-aligned images where feature points of both images reside on the same row, greatly speeding up the process of stereo correspondence, as it only needs to look over one row instead of the entire image to find corresponding feature points. As the rectification reprojects the image plane seen, information could be lost if the two cameras are not looking at the same scene from the same direction. To make this rectification retain as much of the original image planes as possible, it is best to set up the two cameras such that their image planes are coplanar and their principle rays are parallel. Figure 2.2 shows

this set up from a top down view where the two image planes (shown in dark horizontal lines) are aligned and the principle rays (c_x^{left} and c_x^{right}) are parallel to one another.



(a) *Lens with Fisheye distortion* ***(b)*** *Undistortion applied to (a)*

Figure 2.1: Example of using undistortion to remove lens distortion

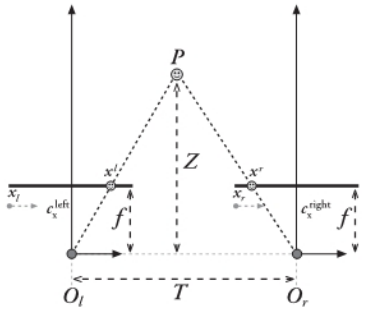


Figure 2.2: Coplanar Camera Image Planes with Parallel Principle Rays. Image by Bradski et al.

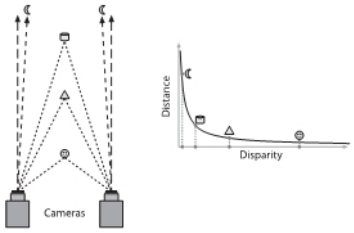


Figure 2.3: Inverse Relationship between Disparity and Depth. Image by Bradski et al.

2.3 Disparity-to-Depth

Due to the coplanar configuration of the cameras, there is an inverse relationship between disparity and depth as seen in Figure 2.3. Because of this relationship, accurate depth measurements can only be performed on objects close enough to the camera system so that they can both see in their respective image planes. To calculate this 3D depth we can use the principles of triangulation between two cameras as seen in Figure 2.2 using:

$$Z = fT/d \tag{2.1}$$

Where Z is the 3D depth to the object along the cameras Z axis, f is the focal length in pixels, T is the distance between cameras centres of project, and d is the disparity value:

$$d = x_l - x_r \tag{2.2}$$

3 Related Work

To provide object tracking in relation to a ground plane, we need to look into two areas of computer vision: plane estimation, and object tracking. In this section we will discuss related previous research into these areas, along with some background information on the Hawk-Eye system and what it offers.

3.1 Plane Estimation

As established in the introduction, we wanted to detect and calculate the ground plane to enhance tracking of the cricket ball. Providing this plane information will allow for more valuable data to be netted from the cricket ball tracking, such as bowling line and length - both of which are of particular use for training purposes. This section will review techniques available for plane detection, highlighting the advantages and limitations of each.

3.1.1 Plane Segmentation

Plane segmentation provides a method to split an image up into segments based upon their colour or texture, aiding in locating a specific plane. For the sport of cricket, this could be useful to find the regions of image containing either the cricket pitch or field that are both easily distinguishable by their colour and texture. In this section we will discuss two techniques for plane segmentation: using colour, and using a colour-texture mixture.

Colour-based segmentation

One approach to colour-based segmentation is presented in the research by Meyer et al [18], proposing applying an algorithm using the "watershed transform" which splits up images based on their topology. This segmentation is performed by a flooding technique where, from the starting pixels chosen by the user, neighbouring pixels are assigned into segments based on how similar they are from the starting pixel. While the results of his research in which he tested the method on paintings are promising (see Figure 3.1), the performance is fairly slow and would only get worse if the image size increased. Researchers have been improving the performance of the watershed transform over time by various techniques such as parallel processing [25], where, for example a number of processors have access to the same memory and can work on different sections of the image at the same time. However, the performance was still an issue and would take many frames to perform this segmentation - not at all practical if other algorithms are waiting for the segmentation to finish before beginning. Two approaches exist to overcome the performance issue; using new multi-core computers, or using a dedicated hardware approach. Using new multi-core systems would allow the plane processing to be performed on one core, parallel to that of the tracking algorithm on another, separate core. Using dedicated hardware to speed up the watershed algorithm is another approach, allowing separation of the segmentation code away from any other algorithms. Research by Trieu et al [28] proposed a parallel and pipelined watershed algorithm designed for hardware implementations that is fast enough to be near real-time.

Another technique for colour-based segmentation is the clustering of colour-similar regions using histogram-based methods. There are a number of histogram method choices available based upon the different representations of colour spaces, such as RGB (Red, Green, Blue) or HSI (Hue, Saturation, Intensity). Research by Delon et al [4] proposed an automated image segmentation technique using the HSI colour space with promising results. Using the HSI colour space allowed their approach to separate colour

in an image away from its intensity information - enabling it to better deal with varying shades of a colour on one plane from shadows or illumination levels much better than previous approaches that relied only on RGB. Segmentation of an image was performed by creating a 1D histogram of the entire image and splitting it into what they call “modes”. Delon et al define these modes as

> “an interval on which the histogram follows the increasing hypothesis on a first part and the decreasing one on the second part.”

This segmentation provided highly accurate segmentation of images taken of natural scenes, which they used in their tests as can be seen in Figure 3.2. This segmentation would work well when applied to cricket fields or pitches that follow a known colour and texture.

***(a)** Original Image* ***(b)** Segmentation of (a)*

Figure 3.1: Results of Watershed Image Segmentation by Meyer

***(a)** Original Image* ***(b)** Segmentation of (a)*

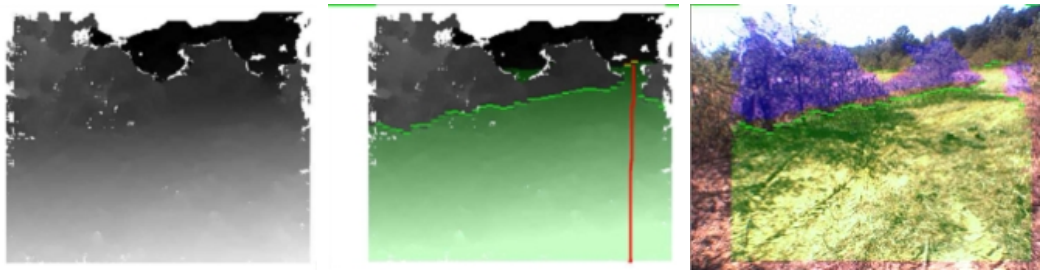
Figure 3.2: Results of Hue, Saturation, Intensity (HSI) Image Segmentation by Delon et al

Colour-Texture based segmentation

Research by Deng et al [5] into a hybrid colour-texture segmentation segments images into homogeneous colour-texture regions. Their approach is performed in two stages: what they call ‘colour quantization’ and ‘spatial segmentation’. This approach allows them to quantize colours in an image into several representative classes, enabling it so that they can be used to differentiate the regions in the image - effectively creating a class-map of the image. This class-map is viewed as a texture composition with which they apply the spatial segmentation. This spatial segmentation uses multi-scale measurements of the local homogeneities to find potential boundary locations. These boundary locations are then used in conjunction with a region growing technique to find final segmentation. This region growing approach could be beneficial for our research if we know of a location inside the image where the ground plane will always be visible, and can then grow the plane from this location using the available 3D information of pixels from their disparity values.

3.1.2 RANSAC

Originally proposed by Fischler et al [6] in 1981 as a method of best fitting a model to a set of points, such as finding a 2D line that best fits through a set of x and y data points, RANSAC has become a vital technique for the Computer Vision field, owing to its accuracy and performance. This technique can be easily extended to take a data set of 3D points and calculate the plane model that best fits the selection. Research by Se et al [27] into a mobility aid for the partially sighted uses RANSAC to great effect - enabling best fit of the visible ground plane from disparity values, even when there are obstacles in the way. In a similar manner, research by Konolige et al [14] into mapping and navigation of outdoor environments relied on RANSAC for finding the ground plane in a challenging terrain filled with obstacles (see Figure 3.3). This robustness to obstacles is useful to our proposed method, as players will normally be visible within the image, serving as obstacles that we do not want to be considered as part of the ground plane. The limitations of this technique are in its trade-off between accuracy and performance, based off how much of the ground plane is visible within the image frame when the calculation was processed. If very little of the ground plane is visible during calculation, then RANSAC will be unable to find a suitable plane that fits best with the data available. To overcome this, we could let the RANSAC method retry its calculation at a later frame if it was unable to get a sufficient plane estimation from the available data. If it remains unable to find a sufficient plane after a few retries, it could then take the best model it was able to get from those attempts. This approach allows us to combine the plane detection and calculation into a single step, providing a ground plane that can be related to the cricket ball tracking data.



(a) *Disparity Image* ***(b)*** *Extracted Ground Plane* ***(c)*** *Ground Plane Overlayed on Original Image*

Figure 3.3: Ground Plane Detection Using RANSAC by Konolige et al

3.2 Object Tracking

Object tracking is a well researched field in computer vision for tracking humans in security systems [15, 20] with great results on low-cost hardware; however object tracking in sports still remains to be fully developed for low-cost hardware due to the extensive requirements of accurately tracking high-speed small objects. Problems occur from the small size of sports balls and the shutter speed of the camera used, causing issues with motion blur and ghosting of objects. For these reasons we required object tracking techniques that were computationally simple and still provided reasonably accurate results. The two techniques available to us for this were segmentation of objects based upon their properties and contour tracing.

3.2.1 Foreground segmentation using Object Properties

Research by Pingali et al [24] into tennis ball tracking proposed using the yellow colour of the tennis ball to segment the image and help with object tracking. As the yellow colour had a very strong peak in the Hue scale this allowed it to be easily segmented from the rest of the scene. This approach assumes that there is a high contrasting background for his tracking to work, the problem being if any other object in the scene was the same colour as the tennis ball, it would become part of the segmented result. To overcome this, they used the properties of the ball, such as its size, to make their algorithm as accurate as possible for tracking only the tennis ball and nothing else. After each frame containing a tracked ball, the centre of

the tracked ball is joined to previous matches to create or update the trajectory and path it has taken. This trajectory and path taken data is then used in a virtual replay system as seen in Figure 3.4

Pingali later went on to improve his research [23] by providing real-time 3D tracking through the use of multiple cameras placed around the tennis court. For this he used six monochrome progressive scan cameras operating at 60Hz each, divided into four pairs. Each pair of cameras views a small section of the court and only performs tracking within that section. To perform the object tracking in real-time, Pingali used a multi-threaded approach with a one thread to one camera pair mapping, so that the entire tennis court is being viewed by their algorithm. The results shown are extremely accurate, able to track tennis serves up to 225km/h and have been used for international broadcasts because of this success. The drawback to this approach is the cost involved, the cameras on their own are in the range of NZ$2000+, and as six of them are required, the total cost increases fast. The computer that controls the system must be very powerful in order to control the tracking of six cameras, increasing the cost even further.

The advantage of Pingali's research is the foreground segmentation that, extended to work on disparity images, would be beneficial for our research, aiding in the locating of the ball. The disadvantage of this technique is relying on the properties of the ball. As Pingali et al used multiple cameras all recording at 60Hz, effects such as motion blur were not a limitation to their research, but would be for ours where we only have one stereo camera pair available. Another limitation is in the texture of cricket balls, which, due to the nature of their surface, cause colour bleeding to occur on the ball, causing the Hue reliance to break down dependant on illumination levels. This may also occur as the cricket balls texture gets scuffed over time as a result from hitting hard surfaces at high speeds, losing its red colour.

(a) *Tennis Serve Path* ***(b)*** *Serve landing points*

Figure 3.4: Virtual Tennis Replay system by Pingali et al.

3.2.2 Contour tracing

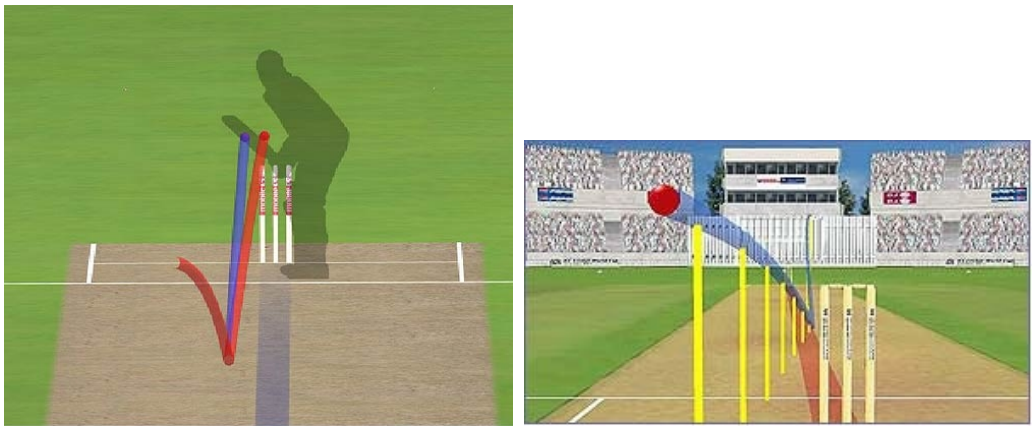
Research by Chang et al [3] into linear contour tracing has shown very fast results for detecting connected components of a scene - as fast as 60miliseconds to process an 1600x1200 resolution image. Chang's algorithm works by using a contour tracing technique to locate external and internal contours of connected components and from this, identifies and labels the interior area of each connected component. This approach is very robust as it does not rely on the properties of the components, instead relying only on the principle that a component is fully determined by its contours. Such robust component tracing is an advantage to our approach, as the the cricket ball may appear to change shape due to motion blur, where the ball travels at a speed faster than the shutter of the camera can capture. The limitation of this algorithm is that it will find the contours for every component within the scene, and an additional technique will still be required to find the cricket ball from the components. To overcome this we could apply this algorithm on to a foreground segmented image, so as to only locate only those objects that are moving.

3.3 Hawk-Eye

The research behind Hawk-Eye was developed by engineers at Roke Manor Research Limited in the UK in 2001, and later patented for use in broadcasting through a separate spin-off company, Hawk-Eye Innovations Ltd. Today it is used in broadcasting for cricket, tennis, and other ball sports to track and visually display the path that the ball has taken, along with the most likely path a ball would take if left unfettered. This visual display is used to enhance replays (see Figure 3.5) of sport and provide the broadcast viewers with possibilities of what could have happened, including use in cricket for leg-before-wicket (LBW) decisions, determining if the cricket ball would have hit the wickets if the batters leg had not stopped it.

The process behind Hawk-eye is based upon the principle of triangulation using visual images and timing data from at least four high-speed high-priced video cameras, all located around the sporting field at different locations and angles. In each frame from each of the cameras, the system locates the group of pixels which corresponds to the known image of the ball. From this location inside each image of the cameras surrounding the field, triangulation can be used to calculate the 3D position of the ball.The system used to capture each camera requires dedicated video processors, databases, and multiple processing computers to rapidly process the video feeds at high speeds. All of this equipment raises the price exponentially and this is not factoring in the costs related to the human operators and calibration, which when totaled can cost in the millions.

The limitations of this system are: the cost of required hardware, the required predefined model of the playing area, and the need for manual intervention in ball tracking, all of which we intended to avoid in our approach. As Hawk-Eye is intended for professional use in television broadcasting, they need to guarantee a 100% accuracy for their object tracking and to maintain this, they require the best of the best hardware. For our approach of sports coaching, the accuracy of the tracking should be suitable at lower levels, and techniques can be used to interpolate between readings. Even using this hardware, there may be times in which a cricket ball cannot be found by the algorithm and so manual intervention by the human operators is required to locate where the ball should be. Doing this does not allow the process to be automated and would not work well for our idea of creating an automated setup and track approach. To aid in the accurate calibration of the Hawk-Eye setup, the operators must create a predefined model of the playing area, including everything from the size of the field to the orientation of the cameras in relation to each other and where they are in location to the ground plane. Our approach aims to automate this process so that the user of the system does not need to define such details.



(a) *Hawk-Eye displaying the path of the last two bowls*

(b) *Hawk-Eye displaying a cricket balls estimated trajectory*

Figure 3.5: Examples of Hawk-Eye Innovations Incs Hawk-Eye System

4 Design and Implementation

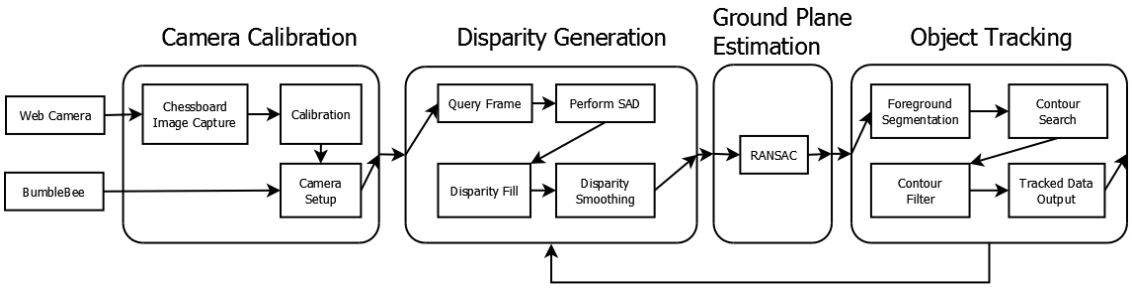


Figure 4.1: The Process of our Algorithm

4.1 Overview

To tackle the problem of stereo tracking high-velocity small objects we have designed an innovative algorithm based on a combination of the advantages of previous research with additional techniques to help overcome the limitations they face. This design and implementation section will be in three sections; a discussion of the test environment we designed to rigorously test our algorithm, the ground plane estimation we performed using RANSAC, and the cricket ball detection using foreground segmentation and contour searching. Our test environment covers the entire process seen in Figure 4.1 along with providing quantitative results for measuring against other algorithms. But we also learnt many important lessons from the camera calibration and disparity generation sections after they were implemented for the two sets of chosen hardware. While the ground plane estimation and cricket ball detection make up our innovative algorithm, they both require the preceding steps of camera calibration and disparity generation to correctly function, therefore we will discuss those first.

As established in the introduction, the aim for our research was to provide a low-cost alternative to tracking systems already available on the market. To provide this low-cost requirement, we initially designed the system to work off the two Logitech Pro 5000 web cameras in a stereo configuration. However, after many weeks of trials we found that the limitations of both the calibration and disparity generation from these cameras to be too strong to continue and required a change to the more expensive Bumblebee 2. These limitations are discussed in detail in the sections below. The specifications of both camera systems are also provided in Table 4.1.

4.2 Stereo Tracker Environment

To rigorously test our algorithm, we have created a C++ test environment called *Stereo Tracker* built upon OpenCV's[22] image processing libraries. OpenCV is a library of programming functions for real time computer vision. Our *Stereo Tracker* environment is a complete solution for the tracking of stereo images as it provides all the necessary tools for stereo tracking; from the initial camera calibration setup, to object tracking inside disparity images. This section will provide a brief overview of how *Stereo Tracker* works and then go into detail of the first two sections of the algorithm that are tied directly to the hardware used - the camera calibration and disparity generation.

(a) Two Logitech Pro 5000s *(b) Bumblebee 2*

Figure 4.2: The Two Camera Systems we have Implemented

	Logitech Pro 5000	Bumblebee 2 (BB2-08S2)
Interface	USB	IEEE-1394a
Max Resolution	640x480 (Interlaced)	1032x776
Horizontal FOV	49°	66°
Frame Rate	29	20
Price (NZD)	$80	$2000

Table 4.1: Camera Specifications

4.2.1 Process

The entire process that *Stereo Tracker* provides can be seen in Figure 4.1 and is split up into four key sections: Camera calibration, Disparity Generation, Ground Plane Estimation, and Object Tracking. The first two sections are required steps to generate the disparity images for the 3D data and rely heavily upon the hardware used. The last two sections, ground plane estimation and object tracking, rely on these generated disparity images and are what makes up our innovative algorithm.

4.2.2 Camera Calibration

$$S \underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{Image Plane}} = \underbrace{\begin{bmatrix} \alpha & -\alpha\cot\theta & u_0 \\ 0 & \frac{\beta}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{Extrinsic Parameters}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}}_{\text{3D world point}} \tag{4.1}$$

$$\alpha = f \cdot m_x$$

$$\beta = f \cdot m_y$$

Camera calibration is based upon the principals of the Pinhole Camera Model where a scene is formed on the image plane by projecting 3D world points using a perspective transform (see Equation 4.1). This calibration provides a method to compute the cameras internal configuration and, if multiple cameras are used in conjunction with each other, their relation to each other - these are the intrinsic and extrinsic parameters. The intrinsic parameters define both the geometry inside the camera and the distortion model of the lens and holds in any situation where the resolution or lens zoom of that camera is held. The geometry of the camera is made up of two factors: the focal length (f) of the lens represented in terms of pixels, where m_x and m_y are the scale factors relating rectangle pixels to distance and θ defines the skew in the cameras coordinate system, and the principal point (u_0,v_0) in pixel units which defines the intersection of the optical axis and the image plane. Calibration may also find the distortion parameters that define the cameras lens distortion. There are five distortion parameters, three that cover radial distortion such as barrel distortion, and two that cover tangential distortion caused by the imperfect centreing of the lens components.

The extrinsic parameters define the rotation and translation transforms from 3D world coordinates to 3D camera coordinates, specifying the cameras position and orientation in space. These transforms combined

with the intrinsic parameters provide the final projection matrix which can be used to associate points in a camera's image space with locations in 3D world space.

Two Logitech Pro 5000 Web Cameras

To perform stereo vision tasks on the Logitech web cameras, we required an easy to use calibration tool that could be used to provide us with the intrinsic and extrinsic parameters required for disparity generation. For this we have built a calibration toolbox into *Stereo Tracker*, based upon OpenCV's calibration functionality that performs two steps of calibration: chessboard image capture, and calibration calculation. The approach used by OpenCV to represent camera intrinsics is based upon research into calibration by Heikkila et al [9] and is performed by viewing a 2D object with a known structure that has many individual and identifiable points from multiple angles. The relative location and orientation of the camera can then be computed from viewing this structure from a variety of angles. For this OpenCV uses a chessboard pattern, as the corners of the chessboard squares are easy to identify from multiple angles and the pattern can easily be printed onto a hard cardboard backing.

To get the best disparity results, the two web cameras need to be set up with their image planes coplanar to each other and their principle rays parallel with each other (see Figure 2.2 for an example) so that both cameras are viewing as much of the same scene, from the same orientation, as possible. Once the user of our toolbox has positioned their camera system as required and has printed off the chessboard pattern onto a flat surface, a four-step process occurs as follows:

1. The user enters details about the chessboard pattern: the number of inner vertical and horizontal squares (the MxN pattern), and the width of each square.

2. The user then holds the chessboard pattern in front of the two web cameras so that the chessboard is visible in both views (see Figures 4.3a and 4.3c).

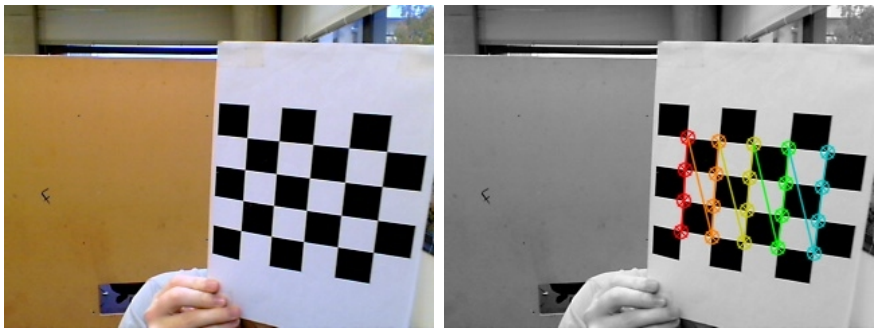
3. When the toolbox is able to locate the specified MxN chessboard pattern with sub-pixel accuracy (see Figures 4.3b and 4.3d), using OpenCV functions, in both camera views a screenshot from both cameras is taken and stored into a local directory and a three second countdown starts. This countdown allows the user to move the chessboard pattern to another position and angle relative to the camera system before it performs this step again.

4. Once ten pairs of screenshots have been taken by the toolbox, they are passed to the calibration method to perform calibration. Each pair of screenshots are passed through to OpenCV's stereo calibration function which uses its knowledge of the chessboard pattern, with the user specified details, to calculate the intrinsic and extrinsic parameters of the two web cameras, providing an estimated transformation between the two cameras and thus making a stereo pair.

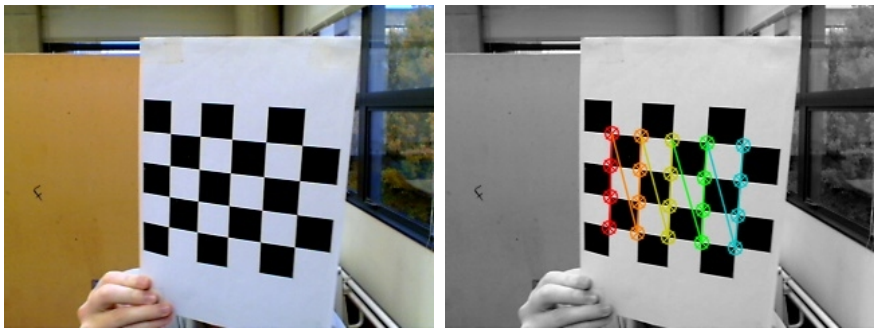
We decided to take ten screenshots based on a book into the inner workings of OpenCV by Bradski et al [2]. Here they explained that, due to the calibration method used in which homography matrices are fitted to each chessboard pattern, no matter how many chessboard corners were detected in an image, only four corners of information are needed to generate this homography. Bradski later goes on to suggest that using enough images so that $2NK \geq 6K + 4$ holds, where N = number of corners and K = number of images, should be sufficient to get an accurate calibration as long as the chessboard is moved enough between images.

As discussed previously, the intrinsic and extrinsic parameters hold only as long as the two cameras are in the exact configuration they were in when calibration was performed. This becomes the major limitation of using low-cost web cameras as our aim was to provide a user-friendly system accessible to any computer literate person; the technical understanding required to get the system up and running may be too much for the end-user. This calibration will only work for the configuration that the two cameras are in and if either of the cameras move in the slightest amount, the calibration no longer holds and any later techniques that rely on this calibration, such as the disparity generation, will return unusable results. The only way to overcome this issue would be to mechanically place the two cameras into a perfect stereo configuration where a single calibration will hold no matter the position and orientation of the cameras. Another issue found with using two web cameras, was how important it is to have their image planes coplanar such that

their views overlapped at much as possible to provide better disparity generation. When the cameras were not in this coplanar configuration, the resulting calibration would be fairly inaccurate, causing the need for re-orienting the camera pair and trying calibration again. The accuracy of the calibration was measured by the dot product of the epipolar lines after rectification, the closer the dot product is to 0, the more accurate the calibration would be. The best accuracy we could get for this hardware implementation was a dot product of 0.2.



(a) *Left Camera Image* ***(b)*** *Located chessboard pattern of (a)*



(c) *Right Camera Image* ***(d)*** *Located chessboard pattern of (c)*

Figure 4.3: Examples of our Chessboard Calibration Technique

Bumblebee 2

The Bumblebee 2 is a stereo vision camera developed by Point Grey Research Inc., designed as an entry level camera for professional use. The main advantage and our reason for using this camera system is that it comes mechanically pre-calibrated for lens distortion and camera misalignments, with images aligned within 0.05 dot product error after rectification - a much better result than that of the Logitech implementation. This removes the need to calibrate the system each time it is moved to a new location and provides the end-user with a highly accurate calibration straight away. With this need for manual calibration removed, the camera can pass the tedious "chessboard" calibration step of our environment and move straight to the camera setup section. This section gathers data about the camera setup required for the disparity-to-depth calculation, that would normally be estimated during calibration: baseline distance (the distance between the two lenses), focal length, and the principle point. This data can be automatically retrieved from the Bumblebee 2s internal memory, or entered by the user if they are using recorded footage.

4.2.3 Disparity Generation

Once the calibration of the camera system is done, the generation of disparity images can be performed. As established in the Stereo Vision chapter, before disparity generation can be performed the image pair needs to go through undistortion and rectification to fix lens distortion and align the two cameras images

planes. This section covers the implementation of these three steps for each hardware implementation.

Two Logitech Pro 5000 Web Cameras

For the Logitech system, we have implemented the process of undistortion and rectification together into two steps: rectification transform calculation, and image remapping. Rectification transform calculation uses the already calculated intrinsic and extrinsic matrices from calibration to compute the rotation and projection matrices for each camera to correctly align their image planes. These rotation and projection matrices are then used in conjunction with the distortion coefficients from calibration to create a 2D transformation map, which can be applied on an image to perform both undistortion and rectification on it.

OpenCV provides three correspondence techniques to perform disparity generation: Block Matching based on Konolige's research [13], stereo correspondence by scanline pair matching proposed by Birchfield et al [1], and graph cuts proposed by Kolmogorov et al [12]. As established in the stereo vision chapter, there is a large trade off between performance and accuracy of correspondence algorithms, and this is especially true for these three implementations. Graph-cut is the most accurate with 8.26% average error on Middleburys stereo image test and performs the slowest at, on average, 10seconds per 320x480 image pair tested on our system. Scanline pair matching lies in the middle for performance and provides the worst accuracy with 19.0% errors on Middleburys stereo test at, on average, 0.5seconds per 320x480 image pair tested on our system. Block matching is the least accurate with 15.7% errors on Middleburys stereo tests but performs the best, on average 0.031seconds per 320x480 image pair tested on our system.

As we wanted our algorithm to run at real time, we had to focus on performance first before looking at the accuracy, so we implemented the block matching technique by Konolige. This block matching technique is a one-pass stereo matching algorithm that uses a sliding sums of absolute differences window between pixels in the left image and the pixels in the right image. This algorithm computes disparity images at $O(X \cdot Y \cdot N)$ time for a pair of XxY images with a pixel search range of N. A problem we faced with this algorithm is that it only finds strongly matching high-texture points between the two images. Thus, to produce a disparity image that is highly populated, the camera system had to be used outdoors where natural light would enhance the texture of the scene. When the camera system was not used outside, a large percentage of the disparity image was not generated due to insufficient texture. To improve the quality of the generated disparity image we applied pre-filtering to the input image pairs first before disparity generation was performed.

Pre-filtering was performed by normalising the image brightness and enhancing texture of an image by running an 21x21 window over the image where the centre pixel I_c of the window is replaced by:

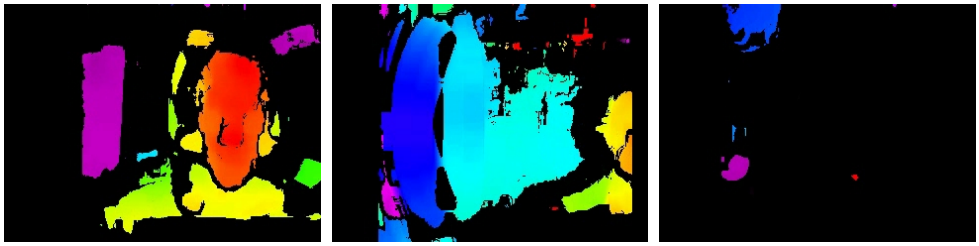
$$min[max(I_c - I_a, -I_{cap}), I_{cap}] \tag{4.2}$$

Where I_a is the average value in the 21x21 window, and I_{cap} is a positive limit of 30.

Correspondence was then performed by passing a sliding SAD window over the images. For each highly textured (x,y) point in the left image, the right image is scanned across the same row y from position x to position $(x-68)$, where 68 is our pixel search range, to find a strongly corresponding point. This technique works while the assumption of the two cameras image planes being coplanar is held. The larger the search space used, the greater the region of depth the algorithm will cover (as seen in Figure 2.3); however the higher the time taken to perform correspondence. The pixel search range of 68 was chosen as testing we had performed found this sufficient to provide us with depth information up to 10metres away from the camera, and larger search ranges had a dramatic impact on speed taken to perform correspondence. As rectification has projected each image so that they are row-aligned, this search only needs to be performed along the same row in the right image as the row that the feature was found in the left image. Without the rectification stage, this correspondence could not be performed.

As established in the camera calibration section, techniques that rely on the calibration of the camera system provide unusable results if that calibration no longer holds. As seen in Figure 4.4 where the left camera undergoes a rotation and a translation, this calibration problem became a strong limitation of this hardware implementation, where the disparity generation breaks down as the calibration it relies on no longer holds. Even with good calibration, the amount of unknown disparity values, where the block matching algorithm was unable to calculate a disparity value for a pixel due to lack of texture, or unable to find

feature points in both images remained fairly high. We performed tests on a selection of frames with good calibrations to find that, on average 30% of an image was taken up by unknown disparity values.



(a) *Initial Disparity Image* ***(b)*** *Disparity After Left Camera Rotation* ***(c)*** *Disparity After Left Camera Translation*

Figure 4.4: Disparity issues caused by Calibration

Bumblebee 2

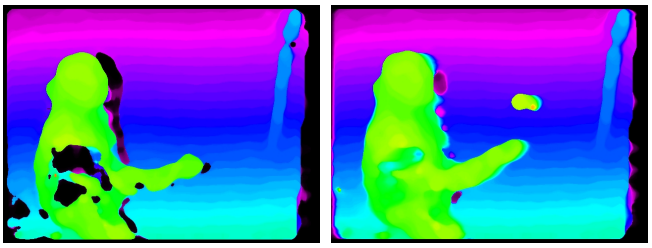
To perform the disparity generation steps on the Bumblebee 2, we use the "Triclops SDK", from Point Grey Research Inc, to define how we wanted the images to be returned from the sensor pair of the camera. The process implemented while using the Bumblebee 2 was very similar to that of the Logitech implementation, with a two-step process: pre-processing and stereo processing. Pre-processing performs the same action as pre-filtering for our Logitech implementation, enhancing the texture and normalising the brightness levels. Stereo processing is also performed using the same sliding SAD correspondence window technique as with our Logitech implementation. Because of the similarity between the two implementations we set up the Bumblebee 2 to use the same pixel range of 68 for the sliding SAD window as we had found successful in our Logitech implementation, pursuant to the distances we were looking at. After performing the same unknown disparity test as we did on the Logitech implementation, we found that on average the frames from the bumblebee would have 15% taken up by unknown disparity values. A limitation found with the model of Bumblebee 2 that we were using was that it only provides images at 20Hz, which only became an issue for tracking an object travelling at speeds over 30km/h.

Disparity Interpolation and Smoothing

As our chosen techniques for both hardware implementations are aimed at performance over accuracy, we have added some additional techniques to our algorithm to fill in and smooth regions of unknown disparity values. There were two cases where holes in disparities could occur: between two known disparity values, and between a known disparity value and the edge of the image. To fill in the regions of unknown disparity values between two known disparity values, we applied a linear interpolation from the two known values, infilling the unknowns. To fill in the regions of unknown values between a known value and the edge, we extrapolated the values from the known side until it encountered the edge. This region interpolation was performed both horizontally and vertically with the final image averaging the pixel values of the two. Once the region interpolation was performed, a 21x21 median filter was passed over the disparity image to smooth the filled regions into the known disparity values. Applying this interpolation to an image is causing us to make an assumption that the points we are interpolating are valid. Because of the distances we are covering with our disparity images, the majority of unknown disparity values that are close to the camera system are from the left and right cameras viewing different sides of an object, giving a lack of correspondence. Disparity holes at later distances could become a problem with our interpolation if, when applied, it ends up interpolating between two foreground objects creating completely invalid results.

4.3 Ground Plane Estimation

As we were only looking at depth ranges of up to 10 metres, we decided to set up the camera so that it would be facing towards the ground, so that the ground took up the majority of the view. Because of this



(a) *Original Disparity Image* ***(b)*** *Next Disparity Image after Interpolation*

Figure 4.5: Example of our Disparity Interpolation over two Image Frames

camera set up, the successful RANSAC techniques used in mobility research [14, 27] for ground plane estimation were best suited to our approach. This is because of RANSAC's ability to work, as long as the desired plane takes up the majority of the image. Using this assumption, there was no need to perform any image segmentation first to find where the ground plane may be located within the image. As obstacles such as the batter are often visible within the image, it might cause our implementation to be unable to find enough data to fit a plane. Because of this, our algorithm will repeat the RANSAC test at a later frame, if indeed a plane was not located.

The RANSAC process is iterative and works on disparity images, and has been implemented into our algorithm as follows:

1. Randomly select 300 points within the disparity image(See Fig 4.6).
2. Create a test plane from three randomly chosen points of the initial 300.

$$Ax + By + Cz + D = 0 \tag{4.3}$$

   where (A,B,C) is the planes nonzero normal vector through the point (x,y,z)

3. Test the remaining 297 points against this fitted plane and count how many fit the plane within a given error threshold - these are the hypothetical inliers. The equations used for determining if a given point is within the threshold is:

$$Di(x,y,z) = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{4.4}$$

$$Inlier(x,y,z) = \begin{cases} 1 & \text{if } Di(x,y,z) \leq T_D; \\ 0 & \text{if } Di(x,y,z) > T_D. \end{cases} \tag{4.5}$$

   where Di is the distance point (x,y,z) is from the fitted plane and T_D is a distance threshold.

4. Repeat steps 2 and 3 until the amount of hypothetical inliers found for a test plane is greater than a set threshold. The plane that covers this threshold is considered the ground plane.
5. If no plane has more hypothetical inliers than the threshold, the plane that comes the closest is, for now, considered the ground plane and the process is repeated again at a later time to try find a better fitting plane model.

Testing was performed to find which values for the thresholds resulted in the best detection rate for the ground plane. From this we set the distance error threshold to 0.5, such that a point (x,y,z) has to be within 0.5 metres of the test plane to be considered a hypothetical inlier. For the set threshold that decides if enough hypothetical inliers have been found, we used 70% of the original count of points, giving us 208 points required for the test plane to be considered good. With these threshold values set, and the ability to retry if the thresholds are not met, this technique will calculate a suitable ground plane, even with interfering obstacles.

Limitations

The limitation of the RANSAC technique is in its reliance on the error and inlier count thresholds, which are used to govern when a point is considered a hypothetical inlier and when there are enough hypothetical inliers to consider the plane 'good'. If a scene becomes highly cluttered for a long period of time, such that the repetition technique mentioned in Step 5 is not able to find a plane that surpassed the inlier count threshold (theoretically a case in which segmentation of the ground, discussed in previous sections as an alternative method to RANSAC, would succeed), RANSAC's only alternative is to relax its thresholds which could cause more problems. More relaxed thresholds will lower the accuracy of the ground plane and increase the possibility of predicting a wrong plane, for example from foreground objects creating their own visible plane that the algorithm picks up.

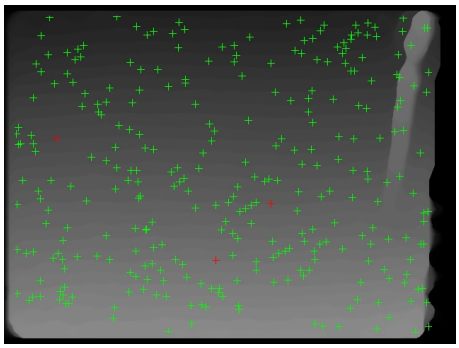


Figure 4.6: Our RANSAC Algorithm during Processing

4.4 Cricket Ball Detection and Tracking

To provide object detection and tracking, our algorithm performs two steps: foreground segmentation, and object tracking using contour tracing.

4.4.1 Foreground Segmentation

Our algorithm performs foreground segmentation first to help speed up the process of object tracking. Many foreground segmentation algorithms [10, 16] are so computationally inefficient as to be unlikely to support real-time processing for the speeds that the cricket ball will be travelling at. Because of this we decided to use a very computationally simple technique based upon absolute differences between frames. First, a background reference frame is created early on, around 30 frames in (See Fig 4.7a); any subsequent frame (See Fig 4.7b) is then subtracted from the background reference image to provide us with a difference mask.

$$M_{p(i)} = |S_{p(i)} - B_{p(i)}| \tag{4.6}$$

Where M = Difference Mask, S = Subsequent Image, B = Background Reference Frame, and $_p(i)$ = pixel i within the images. The reason behind taking the background reference frame after 30 frames have occurred instead of the very first frame is to allow for frames to pass where the cameras will still be initialising and changing any brightness or contrast settings due to auto-balancing features. A drawback of setting up the SAD implementation for performance settings over accuracy is that the disparity values provided can jump up or down from frame to frame. To account for this problem we only keep values of the difference mask that are greater than or equal to 30pixels, This value was chosen after testing the average size of jitters for footage up to 10m distance away from the camera.

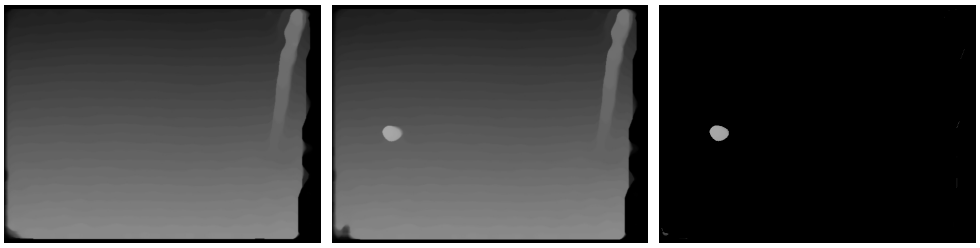
$$M_{p(i)} = \begin{cases} 1 & \text{if } i \geq 30; \\ 0 & \text{if } i < 30. \end{cases} \tag{4.7}$$

This step provides us with a binary mask that we can apply back onto the subsequent image by binary ANDing, resulting in the segmented foreground image.

$$F_{p(i)} = \begin{cases} S_{p(i)} & \text{if } M_{p(i)} = 1; \\ 0 & \text{if } M_{p(i)} = 0. \end{cases} \tag{4.8}$$

Where F = Segmented Foreground Image (see Figure 4.7c).

The drawback of applying equation 4.7 is that our result will now remove valid foreground objects from the segmented foreground image that are close to the background ground plane. This will cause issues for our cricket ball tracking if the ball is captured while touching the ground at the furthest range away from the camera and thus being ignored as a valid foreground object. However, this is also an advantage, as the movement of grass by wind will no longer be shown on the foreground segmented image. The limitation of the foreground segmentation occurs if the camera is moved after the background reference frame has been generated, as every pixel in the subsequent frames will be considered foreground. To overcome this, the background reference frame can be reset at any time.



(a) *Background Reference Frame* ***(b)*** *Subsequent Image* ***(c)*** *Segmented Foreground*

Figure 4.7: Foreground Segmentation Process

4.4.2 Object Tracking

To perform object tracking on the segmented foreground images we have implemented Chang's [3] contour tracing algorithm using the *CvBlobsLib* [29] library. CvBlobsLib provides two basic functionalities: Extracting 8-connected components, referred to as blobs, in binary or grayscale images using Chang's contour tracing algorithm, and filtering the obtained blobs to get the objects in the image that we are interested in. This library allowed us to apply Chang's algorithm on our segmented foreground images to locate and track the cricket ball. Due to the nature of the sport we are doing our ball tracking in, there is a high possibility of a player, usually the batter, being in the disparity image and then showing up in the segmented foreground image. Because of this we can use the second feature of the CvBlobsLib library to filter out unwanted foreground objects in the hopes of leaving only a blob for the cricket ball. This filtering is built up of a set of rules based upon factors such as the blobs area, perimeter, width and height. To build a filter model that best models properties of the cricket ball we recorded several videos of footage involving the cricket ball passing through the cameras view, where the camera was set up to follow our previously mentioned assumptions - three metres above the ground angled slightly so that the view covers the same area as a cricket pitch. We then manually processed the videos with debug information about the cricket ball, gathered by CvBlobsLib, to find the range that the ball falls into for area and perimeter properties. The results of this testing led us to our filter model as follows:

1. Remove any objects that have an area smaller than 50 pixels or larger than 1000 pixels.

2. Remove any objects that have a width or height larger than 200 pixels.

Any frame that results in a single blob after going through the filter model is considered a *tracked* frame by our algorithm. To provide reusable results, information about the object inside a *tracked* frame is stored in a file along with the calculated ground plane for later use by other tools. The information stored includes

the time in milliseconds and the 3D coordinates. These coordinates are calculated using the disparity-to-depth equation as discussed in the Stereo Vision chapter. This stored data could then be passed to a 3D plotting tool or a virtual replay system such as that used by Pingali et al [23] where tennis tracking data was used in international television broadcasts to provide immersive replays of footage including paths taken by the ball and its speed at any point in those paths.

5 Evaluation and Results

The evaluation of our research focuses on two accuracy measures: 3D depth calculations and object detection. The accuracy of our 3D depth calculation from the disparity values is important, to know how valid our tracking and ground plane results are, and to find out if our disparity interpolation and smoothing had any effect on this. The accuracy of our object detection allows us to know how confident we can be with the tracking results provided by our algorithm.

5.1 System

These evaluations were done using the Bumblebee 2 and our Stereo Tracker test environment running on a desktop machine with an Intel Core 2 Quad at 2.40GHz, 4GB ram, and an NVIDIA 9500GT graphics card. The Bumblebee 2 was connected to this machine via firewire and was set to operate at a resolution of 400x300. As established in the design and implementation section, there were a few limitations we faced with the chosen hardware that had to be accommodated for in our evaluation. Firstly the model of Bumblebee 2 we used was manufactured to only operate at 20Hz, this does not allow us to evaluate the object tracking stage of our algorithm on cricket balls at speeds they would normally travel. Because of this we were required to scale the problem down to tracking a cricket ball travelling at slower speeds over an area equivalent in size to a cricket pitch, in order to give a fair evaluation for the frame rate we had available.

5.2 Depth Estimation

To determine the accuracy of our stereo vision technique, calculation between known physical distance and our algorithm's calculated distance is performed, so as to test the results of the conversion from disparity values to 3D depth. To perform this evaluation we placed a square box in front of the Bumblebee camera, with a corner directly facing the camera as to be the closest point, at 20 different measured distances from the camera. The square box was chosen as it provided easy to distinguish corners that could be located by the human operating the test. The 20 distances were taken from a range of 50cm to 950cm from the camera; however it was not possible for the human operating the test to accurately locate the closest corner of the box at the later distances, so the closest point of the object as shown in the data was assumed to be the corner.

To gather our algorithm's calculated distance, the test environment was set up to perform the necessary steps to generate disparity images using the Bumblebee 2 and display them on screen. Both disparity interpolation and smoothing were applied to accurately test the final disparity image that we would be performing tracking on. Mouse interaction was added to the displayed disparity image, so that clicking on any point would provide the calculated 3D position of that point. This calculation was performed using the disparity-to-depth equation shown in the Stereo Vision chapter. The calculated z value of the 3D position was then compared to the measured distance as seen in the results section below.

Results

The results of this experiment proved to be highly accurate for the distances measured, with calculated distances up to 5 metres being within 2% of the measured distance, and calculated distances up to 10 metres being within 5% of the measured distance. The decrease in accuracy as distance increased can be accounted for by two factors: the smoothing filter we applied and the inverse relationship between disparity

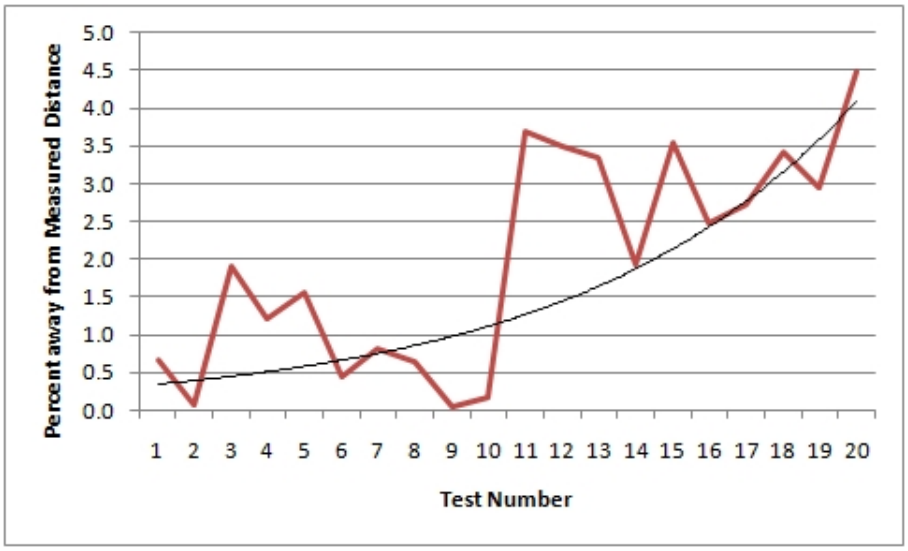


Figure 5.1: Accuracy results of 3D depth calculations for 20 tests, 10 within 1-5 metres from the camera and 10 within 5-10 metres. Vertical scale displays how far off, as a percentage, the calculated 3D depth was from the measured 3D Depth

and depth. As our smoothing filter applies a median filter over the image using a 21x21 window, as objects get further away from the camera they will cover less pixel area. As this area tends towards the size of the median window, the smoothing effect causes more of the object to be blurred into its surrounds, and in our case, the corner of the box facing the camera would blend into the flat surfaces surrounding this corner, changing its disparity value. The trendline of our accuracy results corresponds to the inverse relationship between disparity values and 3D depth values as Bradski et al explains:

> "Since depth is inversely proportional to disparity, there is obviously a nonlinear relationship between these two terms. When disparity is near 0, small disparity differences make for large depth differences. When disparity is large, small disparity differences do not change the depth by much. The consequence is that stereo vision systems have high depth resolution only for objects relatively near the camera"

That is, as the disparity values decrease for distances away from the camera, small changes, such as the corner of the box we measured the distances to blurring into its surrounds, make for larger distance differences as seen in Figure 5.1.

5.3 Object Detection

As discussed in the Design and Implementation chapter, the Bumblebee 2 camera that we used, along with the size of the ball to be tracked provided many limitations that we had to keep in mind. Because of these limitations, we had to consider the speed of the ball and the distance of the ball away from the camera, and scale our evaluation to properly test our algorithm with these considerations in mind. As the Bumblebee 2 only operates at a maximum of 20Hz we were unable to perform an evaluation on balls travelling at speeds over 30km/h. Because of the small size of the cricket ball to be tracked, the Bumblebee 2 is unable to distinguish between the ball and surrounding background objects when it creates its disparity image and the ball is over 10 metres away.

Because of these hardware limitations, we devised a scaled down evaluation to better test our algorithm without the limitations being as much of an issue. This scaled down evaluation consisted of applying our algorithm to four videos, with a total frame count of 2905 frames, and manually comparing the algorithm's results with the actual data. These videos each contained a cricket ball passing in front of the camera between two people in different ways. First video involves the two people passing the ball directly across

the camera view with a slight down curve as gravity takes hold to represent when the ball would first be released from the bowler. The second video involves the two people bouncing the ball between each other at various throw strengths to represent the ball bouncing after bowler release. The third and forth videos involve the people throwing the ball towards or away from the camera to represent the ball being hit by the batter and suddenly changing direction.

To test the accuracy of our tracking algorithm, we have manually processed each video in comparison to the tracking data provided by our algorithm to provide two accuracy values - detected and false positive. Detected accuracy constitutes where the algorithm correctly located the cricket ball within a video, where higher is better. False positive accuracy constitutes where the algorithm tracked an object that was not the cricket ball, where lower is better. Nearly all of the false positives encountered were due to the disparity values around the border of the image jumping up and down as discussed in the Design and Implementation section. Further work on the filtering model would help eliminate these.

Results

The results for these four videos are shown in Table 5.1 below:

Video	Detected %	False Positive %
1	92.0	1.2
2	90.4	1.5
3	89.4	2.8
4	90.9	2.3
Overall	90.7	2.0

Table 5.1: Stereo Tracking Accuracy

Each of these videos have been taken from the same camera height of three metres off the ground at varying angles of view of the ground. A sequence of five correctly detected cricket balls can be seen in Fig 5.2

As we are using a different approach to gather object tracking aimed towards automated low-cost systems for sports coaching, than those used by previous research aimed at broadcasting or using high-cost systems, it is difficult to make a direct comparison to the level of accuracy that they were able to get. For this reason we cover the aspects and techniques of these broadcast systems and compare them to our approach.

High-cost broadcast systems

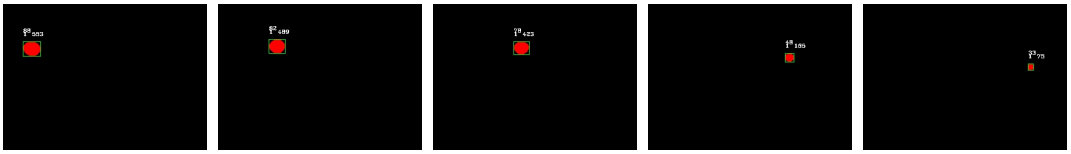
High-cost broadcast systems, such as Hawk-Eye Innovations Incs Hawk-Eye [17] and Pingali et als tennis tracker [23], are developed for commercial use where price is no problem and accuracy is key. Both systems rely on at least four high-speed, high-resolution cameras placed at fixed locations around the sporting field, to provide as much detail as possible about the scene viewed and to allow for triangulation. In comparison our system uses one, lower speed and resolution, stereo pair that only needs to be facing towards the ball to be tracked, at a slight down facing angle.

Both broadcast systems require a predefined model of the relationship between each fixed camera and details about the object being tracked. For Hawk-Eye, this is a complete model of the sporting grounds that their system is placed at, including the positioning of their fixed cameras in relation to each other and in relation to the ground plane. This requires many hours of set-up and calibration so that they are able to accurately calculate 3D positional information using triangulation. Our approach uses an automated approach to calibration to generate both the cameras relationship to one another and the ground plane.

Both broadcast systems say that they can produce extremely accurate object tracking. Hawk-Eye provides 100% accuracy through manual intervention by human operators watching over the system, who can fix any frames in which the system incorrectly tracked an object. Pingali et al produce their accuracy by maintaining a trajectory model of the tracked tennis ball as their predictive filter as to place the search space in the next frame where the algorithm believes the ball should be. Unfortunately no accuracy results were

provided by Pingali et al, only information retaining that the system had been used in professional tennis broadcasts with extremely high accuracy.

The speed for both of these systems is near or above real time due to the amount of processing power they have available. Our approach is able to run at near real-time on an average computer system.



(a) Frame 56 *(b) Frame 57* *(c) Frame 58* *(d) Frame 59* *(e) Frame 60*

Figure 5.2: Correctly Detected Cricket Ball Sequence

5.4 Performance

To test the performance of our algorithm, we implemented timing output for each of the sections inside the disparity generation, ground plane estimation, and object tracking stages shown in Figure 4.1. We then let our algorithm run through the entire process for 3000 frames at 20 frames per second of 400x300 resolution input. The average performance of each section can be seen in Table 5.2.

Section	Time Taken(msec)
Query Frame	50
Perform SAD	30
Disparity Fill	2
Disparity Smoothing	1
RANSAC	10
Foreground Segmentation	2
Contour Search	3
Contour Filter	1

Table 5.2: The Performance of each section of our algorithm

These results show that our algorithm performs faster than the time it takes for the Bumblebee 2 to return a frame. As the RANSAC function does not require being performed on each frame the total time, on average, our algorithm takes without frame querying is 39ms. This could provide us with a theoretical performance of just over 25frames per second without the frame query limitation. This is a great result for the concept of object tracking in stereo and with further optimisations, such as implementing Intel's "Integrated Performance Primitives"[1] libraries which suggests improvements of up to 50% in performance, the frame rate could be increased even higher.

[1] http://software.intel.com/en-us/intel-ipp

6 Discussion and Future Work

This research aimed to provide a method to perform an automated low-cost, real-time 3D tracking system for cricket balls in relation to the ground plane, which could be used for coaching purposes. As a result of our evaluation, we have proven that while object tracking accuracy of our algorithm is over 90%, using low-cost hardware is too much of a limitation when performing stereo-processing. As such real-time is not possible for tracking high-velocity objects using our current hardware with this approach for small objects. Instead, we have been able to create a proof of concept that will work for slower paced sports, with larger balls such as soccer that, when better hardware comes out, can be extended to work for cricket.

Our evaluation has shown highly accurate results for a proof of concept, with over 90% of cricket balls being tracked at a 3D depth accuracy of 2% up to 5metres away and 5% over larger distances. Improving on the extensive setup time needed in prior research, our approach had zero set-up time by comparison, automatically locating the ground plane. While our tracking accuracy is not as high as previous research, including Pingali's [23] tennis ball tracking and commercial systems like Hawk-eye [17], they are aimed at commercial broadcasting. Because of this requirement, they guarantee 100% accuracy for ball tracking. This is provided through manual intervention of their algorithm's object tracking, or through using multiple, expensive high-resolution high-speed cameras, which provide a much better input of the scene in view. Other approaches to small ball tracking, such as tennis ball tracking by Yan et al [30] on recorded video sequences from the Australian Open, aim for accuracy over performance. With the need for performance removed, they are able to implement computationally expensive techniques that take as long as required to finish, without the need of attempting to work in real-time that we were aiming for.

Hardware

While implementing our algorithm on different hardware, we have learnt how important and fragile calibration is to a stereo vision system. Our approach using two Logitech Pro 5000 cameras was rarely successful at providing a good calibration, due in large to problems with illumination and the cameras image planes not being perfectly coplanar. Even with a good calibration, any slight movement from either of the cameras would cause the calibration to become invalid, producing bad disparity images. For this reason we moved on to using Point Grey Research Inc's Bumblebee 2, which comes mechanically pre-calibrated for lens distortion and camera misalignments. This pre-calibration allowed us to have an accurate calibration no matter where the camera was used, enabling us to skip the tedious calibration stage each time the camera system was moved. The result of moving from the Logitech web cameras to the Bumblebee 2 gave us a decrease in unknown disparity points, before our disparity interpolation was applied, of 30% for the Logitech web cameras to 15% for the Bumblebee 2, greatly increasing the amount of valid information we were able to gather from the scene.

Using disparity as our method of gathering 3D information limits the range at which the camera system can be placed from the cricket pitch while still being accurate. Because of this restriction, our camera system must be very close to the cricket pitch - impossible without interfering with the players, unless it is used in training nets. The only other approach would be to use two expensive cameras with high zoom capabilities, placed at the sides of the playing field facing the cricket pitch side-on. This would cause many issues with set-up, as both of their views would need to cover the entire pitch, while overlapping each other as much as possible. Ultimately, this system would require expensive camera equipment and extremely accurate calibration to use.

Ground Plane

With our assumption in place that the camera would be oriented to look down towards the ground plane, allowing it to take up the majority of the view, RANSAC was able to accurately estimate the position of the ground in relation to the camera. As our RANSAC implementation would retry at a later frame when unable to find a sufficient data to fit a plane, it was able to work well with obstacles by waiting until they took up a smaller portion of the view. Our approach to ground plane estimation provides automated results that do not require manual set-up, unlike Hawk-Eye's [17] system, which provides a more user-friendly system that is advantageous for sports coaching.

Object tracking

Our object tracking approach of foreground segmentation followed by linear contour searching has shown highly accurate results when applied to disparity images. The information gathered from the components found during the contour search provided us with the ability to filter objects based upon their visible size, which can help exclude unwanted objects. Applying foreground segmentation to the disparity images before performing the contour search helped narrow down the amount of unwanted objects found during contour search, displaying only the objects that moved within the scene.

6.1 Future work

Hardware

A simple low-cost approach to improving the disparity generation of stereo tracking would be to improve the hardware used either by building dedicated stereo camera pairs, or by using existing hardware to speed up stereo correspondence calculations. This can be done by building stereo camera systems from low-cost image sensors and hardware boards placed into a single case, similar to that of the Bumblebee 2. This would allow for accurate disparity generation with the ability to perform computationally expensive algorithms on dedicated hardware instead of software, increasing the overall speed of stereo processing. Research by Muehlmann et al [21] into a high speed CMOS camera for real-time tracking is a good example of this, where two image sensors are used to create a stereo pair and dedicated hardware on PCB boards attached to the cameras provide mono or stereo object tracking. Their dedicated hardware includes modules for performing tasks relating to calibration and synchronised image grabbing and is able to perform at over 30fps at 480x320 resolution. Another approach to using hardware is making use of graphics processing units (GPUs) - common inside computer systems that are able to perform vast amounts of number crunching in parallel, which could be applied to stereo correspondence. Research by Gong et al [8] proposed using this approach to speed up stereo matching for stereo correspondence with results showing over an 200% increase in fps when algorithms utilised both CPU and GPU in comparison to utilising the CPU only.

Ground Plane

The ground plane could be made more robust to different camera angle orientations by utilising image segmentation. Using the known properties of texture and colour for a cricket field or pitch would allow for selection of the image's correct segment, after the image segmentation technique has been performed. Once this segment has been located, the ground plane could be calculated using 3D information of the segments pixels in the disparity image and mathematical principles of how to calculate planes from 3D points.

Object Tracking

An improvement that could be applied to the object tracking would be to automate the blob filtering model based upon information gathered from previously tracked frames. With the aid of a particle filter, to predict positions of where the ball could be located in the next frame, and a trajectory model of the balls path, a confidence level could be formed to estimate the location and size of the ball based on previously tracked frames.

7 Conclusion

We have created an innovative algorithm using stereo vision that provides automated object tracking in relation to calculated ground plane. The results show that we successfully achieved our goal of low-cost real-time tracking at 20fps with respect to an estimated ground plane.

Accuracy of our approach was calculated using a test environment called *Stereo Tracker*, which we created to rigorously test all aspects of our algorithm. The results of our evaluation for object tracking show accurate tracking, with over 90% of frames involving the cricket ball being correctly detected by our algorithm. Stereo vision was used to be able to provide 3D location data of the viewed scene and was accurate to within 2% for objects up to 5 metres from the camera and within 5% for objects up to 10 metres from the camera.

In that no prior research has used stereo depth values to locate a ground plane and track an object with respect to that plane, the results suggest that the proposed novel algorithm is a useful contribution for enabling automatic setup and ease of use for such tracking.

Bibliography

[1] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

[2] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[3] F. Chang, C.J. Chen, and C.J. Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93(2):206–220, 2004.

[4] J. Delon, A. Desolneux, J.L. Lisani, and A.B. Petro. Color image segmentation using acceptable histogram segmentation. In *Second Iberian Conf. on Pattern Recognition and Image Analysis, IbPRIA*, pages 239–246. Springer, 2005.

[5] Yining Deng and b.s. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/34.946985.

[6] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/358669.358692.

[7] H Gao. Novel algorithms for tracking small and fast objects in low quality images. Master's thesis, University Of Canterbury, 2005.

[8] M. Gong and Y.H. Yang. Near real-time reliable stereo matching using programmable graphics hardware. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, volume 1, 2005.

[9] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1997.

[10] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *2nd European Workshop on Advanced Video-based Surveillance Systems, Kingston upon Thames*, 2001.

[11] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, 2006.

[12] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions via graph cuts. In *International Conference on Computer Vision*, volume 2, pages 508–515. Citeseer, 2001.

[13] K. Konolige. Small vision systems: Hardware and implementation. In *ROBOTICS RESEARCH-INTERNATIONAL SYMPOSIUM-*, volume 8, pages 203–212. Citeseer, 1998.

[14] K. Konolige, M. Agrawal, R.C. Bolles, C. Cowan, M. Fischler, and B.P. Gerkey. Outdoor mapping and navigation using stereo vision. In *Int. Symp. on Exp. Robotics (ISER), Rio de Janeiro, Brazil*. Springer, 2006.

[15] J.H. Lee, Y.S. Kim, B.K. Kim, K. Ohba, H. Kawata, A. Ohya, and S. Yuta. Security Door System Using Human Tracking Method with Laser Range Finders. In *Proc. 2007 IEEE International Conference on Mechatronics and Automation*, pages 2060–2065, 2007.

[16] L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10. ACM New York, NY, USA, 2003.

[17] Hawk-Eye Innovations Ltd., Jun 2009. `http://www.hawkeyeinnovations.co.uk/`.

[18] F. Meyer. Color image segmentation. In *4th International Conference on Image Processing and its Applications*, pages 303–306, 1992.

[19] Middlebury. Stereo vision page, Feb 2009. `http://vision.middlebury.edu/stereo/`.

[20] T.B. Moeslund, A. Hilton, and V. Kr
"uger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2-3):90–126, 2006.

[21] U. Muehlmann, M. Ribo, P. Lang, and A. Pinz. A new high speed CMOS camera for real-time tracking applications. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 5195–5200. Citeseer, 2004.

[22] OpenCV. Open source computer vision library, Oct 2008. `http://opencv.willowgarage.com/wiki/`.

[23] G. Pingali, A. Opalach, and Y. Jean. Ball tracking and virtual replays for innovative tennis broadcasts. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, 2000.

[24] GS Pingali, Y. Jean, and I. Carlbom. Real time tracking for enhanced tennis broadcasts. In *1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998. Proceedings*, pages 260–265, 1998.

[25] J.B.T.M. Roerdink and A. Meijster. The watershed transform: Definitions, algorithms and parallelization strategies. *Mathematical morphology*, page 187, 2000.

[26] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[27] S. Se and M. Brady. Ground plane estimation, error analysis and applications. *Robotics and Autonomous Systems*, 39(2):59–71, 2002.

[28] D.B.K. Trieu and T. Maruyama. Real-time image segmentation based on a parallel and pipelined watershed algorithm. *Journal of Real-Time Image Processing*, 2(4):319–329, 2007.

[29] OpenCV Wiki. cvblobslib, Jun 2009. `http://opencv.willowgarage.com/wiki/cvBlobsLib`.

[30] F. Yan, W. Christmas, and J. Kittler. A tennis ball tracking algorithm for automatic annotation of tennis match. In *British Machine Vision Conference*, volume 2, pages 619–628, 2005.