Nott-Hawkeye

Samuel Palmer, Graeme Russell

November 10, 2017

Contents

Abstract

This report focuses on using video in object tracking technology, recreating the system "Hawk-Eye", known for it's use in a variety of sports such as cricket and tennis. In order to do this we used a foreground detection algorithm to detect moving objects in video frames from two cameras and used a triangulation technique to place them in 3D. Relative to known positions, our technique is accurate to (17 ± 3)mm in a region of interest the size of an approximately 30cm cube, compared to Hawk-Eye's 3.6mm, which has a region of interest the size of a tennis court.

1 Introduction

Techniques to digitally track objects have been around for a long time, traditionally used in making animations for films and video games. More recently it has increasingly been used in a variety of sports in order to achieve more accurate refereeing decisions. Sports such as cricket, tennis and more recently football have taken up various forms of this technology to recreate events digitally, in order to inform officials and help coach players.

Object tracking techniques include the use of magnetic fields, radio-frequency identification, or video. This report will focus on video as this is the technique Hawk-Eye uses; Hawk-Eye is a technology that uses a network of high-speed cameras to track players and ball(s). It is able to distinguish between them accurately whilst ignoring background objects. The network of cameras are usually placed around the top of sporting venues to eliminate the issue of occlusion, which is where the object is obscured from view. Hawk-Eye has now been developed to the extent that, even in sub-optimal conditions such as snow, fog or mud, it performs well.

When considering the influence of modern technologies on traditional sports, Hawk-Eye has to overcome the social push-back to rule changes and arguments against its economy[2]. To do this it must have a solid argument for its robustness and prove that it aids in making better judgements over just employing more referees. What is the point of using it if it is not as accurate as an umpire? This report will explore some of the challenges facing using video to track objects.

The report starts by explaining the theory behind triangulation, the method we used to place objects in 3D. It then describes how we specifically implemented triangulation and our set-up. The report then ends with a discussion on the pitfalls of our system and how it could improve.

2 Theory

2.1 Field of View

To find the position of the object in 3D space, a method of triangulating the object was chosen. Since this was the case, it was important to find the range of angles horizontally and vertically that the camera was able to capture, this is known as the field of view (FOV). The vertical field of view angle Δ_v was found by taking photos of an object which was systematically moved further away from the camera along its principle axis. From each image produced, a ratio s, such that the number of pixels making up the height of the object relative to the number of pixels making up the height of the entire image was calculated, where

$$s = \frac{\text{Pixel Height of Object}}{\text{Pixel Height of Image}} \tag{1}$$

This ratio was also equivalent to the ratio of the vertical angle of the object δ relative to the vertical angle making up the field of view of the camera Δ_v

$$s = \frac{\delta}{\Delta_v} \tag{2}$$

Since the distance of the object r, relative to its height L was large, an approximation for δ could be made (see Fig. 1)

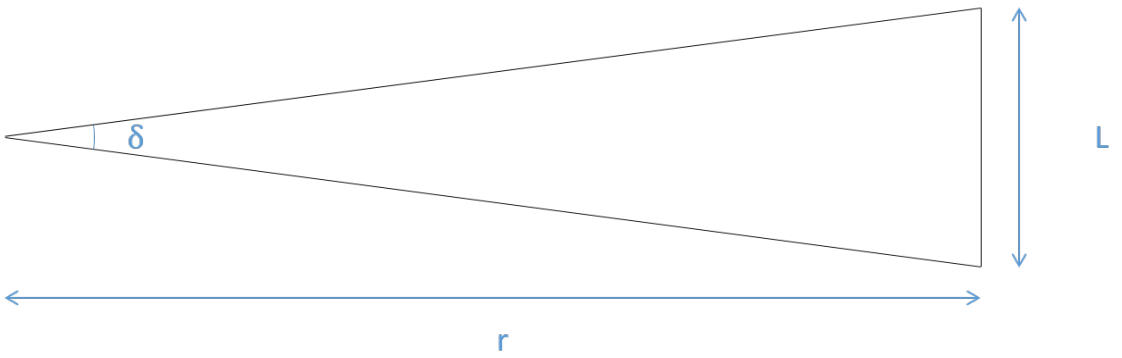


Figure 1: Diagram showing the small angle approximation for δ, where L is the object's height and r is the distance of the object to the camera.

hence,

$$\delta = \frac{L}{r} \tag{3}$$

By combining equations 2 and 3 the relationship

$$s = \frac{L}{\Delta_v r} \tag{4}$$

was found. Therefore, a measurement of different values of r and corresponding values of s for each image of the object were made and a plot of s against r^{-1} was created. From the gradient of this graph a value for Δ_v was deduced.This method was then repeated horizontally, to find the horizontal field of view angle Δ_h.

2.2 Finding the Position of an Object in 2D

To find the position of the object in 2D, the values α and β (see Fig. 2) were found from images taken from camera 1 and 2 respectively.

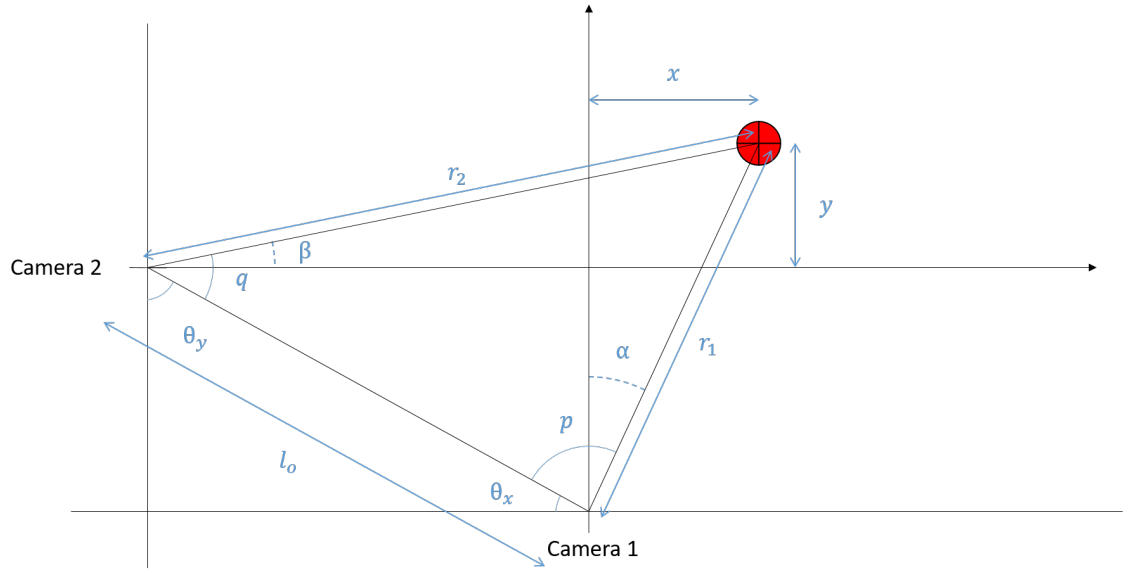


Figure 2: Diagram showing the angles and lengths used to find the position of the object in the x-y plane.

To do this, the value calculated for the horizontal field of view angle Δ_h was divided by the horizontal resolution of the image produced by each of the cameras. This gave the angle rotated in the x-y plane per pixel. The middle column of pixels in the image was set to be a rotation of zero. By counting the number of pixels away the centre of the object was from this middle line and multiplying this value with the angle rotated per pixel, a value for α from camera 1 and β from camera 2 was determined. The angles p and q were then found by using the values of θ_x and θ_y, which were inherent to the positioning of the cameras, and the values of α and β found from the image

$$p = \frac{\pi}{2} - \theta_x + \alpha \tag{5}$$

$$q = \frac{\pi}{2} - \theta_y + \beta \tag{6}$$

The values for x and y were then determined using the following equations, which were determined by the geometry of the set up

$$x = \frac{l_0 sin(q) sin(\alpha)}{sin(p+q)} \tag{7}$$

$$y = \frac{l_0 sin(p) sin(\beta)}{sin(p+q)} \tag{8}$$

2.3 Finding the Position of an Object in 3D

To find the vertical position z,the angle γ was found (see Fig. 3) using the same method for finding the angles α and β. However, the middle horizontal line of pixels in the image was set to be a rotation of zero degrees, and the rotation per pixel was found by dividing the vertical field of view angle Δ_v by the vertical image resolution.

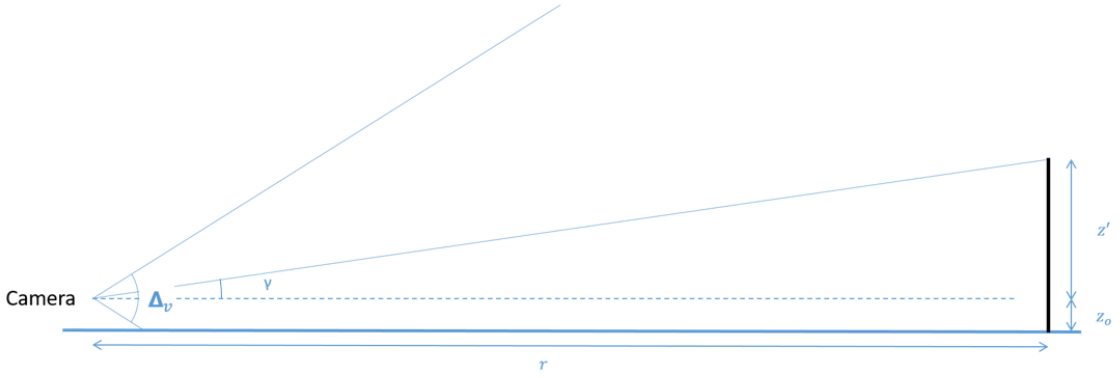


Figure 3: Diagram showing γ, the angle used to find the object's position in 3D.

To find the value for z, the following equation was used for each camera respectively.

$$z = rtan(\gamma) + z_0 \tag{9}$$

Since two cameras were used, two values for z could be determined. An average of these two values was taken to be the position of the object 3D.

3 Method

3.1 Object Positioning

The set up was made up of two Logitech HD C270 webcams, with fixed foci and a frame rate of 30 fps,which were perpendicular to each other and positioned on a right angled triangle (see Fig. 2).The horizontal and vertical distances between cameras as viewed from camera 1 were 257mm and 225mm respectively. A sheet of 5mm gridded paper was placed on the x-y plane to help find the position of the object relative to the origin more easily. Two screens were put up around the system to eliminate the background.

To find the field of view angle Δ, both vertically and horizontally, an object was placed in front of one of the cameras and six photos were taken of it, within a range of 120mm to 300mm from the camera lens along its principle axis. For each picture, the object's position r relative to the camera was recorded. A calibration code was created in Matlab, which then prompted the user to click on the object in each image to find the pixel coordinates of its top, bottom, left and right sides. To improve the precision and accuracy of these values, the windows magnifying tool was used to zoom 600% on the image so that the values used were correct ± 1 pixel. The number of pixels making up the height of the object was found by subtracting the top and bottom pixel values from each other and the horizontal pixel width was found in the same way by using the left and right side pixel values. The pixel height and width values were then divided by the corresponding vertical and horizontal pixel resolutions of the image, which gave a vertical and horizontal value for s, for that image. The values of s for each image were then plotted with their respective r^{-1} values and the gradients of these graphs were calculated.By dividing the actual height and width

of the object with its respective gradients, the vertical and horizontal field of view angles Δ_v and Δ_h were calculated.

In this case, the vertical pixel resolution was 480 pixels and the horizontal pixel resolution was 600 pixels. A cylindrical object was used with a height of 24mm and a width of 14mm. It's worth noting that depth does play a role in the measurement of the field of view angle, so the four pixel positions chosen for the measurement from an image, should all lie in the same plane of the centre of mass, so that the object's position r relative to the camera remains constant.

Having found values for the field of view angles, measurements of the position of the object could be calculated. To do this, a grid of nine points was chosen in the x-y plane with distances of 50mm between each point. A ruler, with error $\pm$1mm was then placed, perpendicular to the x-y plane, at each of these points and a photo was taken. The z values at each of these points were 30mm, 50mm and 70mm, which in turn produced a total of 27 images. The x,y and z values measured were recorded. Next the object's pixel position was found by finding the pixel position of the object in each of the 27 images, from which, values for α, β and γ could be determined. These values were then put into the equations from the theory (sections 2.2 and 2.3) and 3D plots of the actual position of the object and the calculated position of the object were made on top of each other, (section 4.1). The height of the camera relative to the x-y plane z_0 in this case was 15mm.

3.2 Object Tracking

In order to track the moving object the MatLab vision toolbox was used to find statistically significant changes in the image pixels. The method used by the toolbox was image segmentation using gaussian distributions. The first 5 frames from a video were taken and over the average histogram plot of these images, a mixture of different guassians was fitted. This mixed gaussian model was then compared with the rest of the frames from the video, adapting as it went. Any significant changes in the following frames' histogram relative to the model,depicted a new object entering the scene, opposed to changes in the lighting. Those pixels with a significant change were assigned a value of 1 and all others 0. By doing this, a video was then produced which was mostly black (since the background had been eliminated) interspersed white pixels representing moving objects.

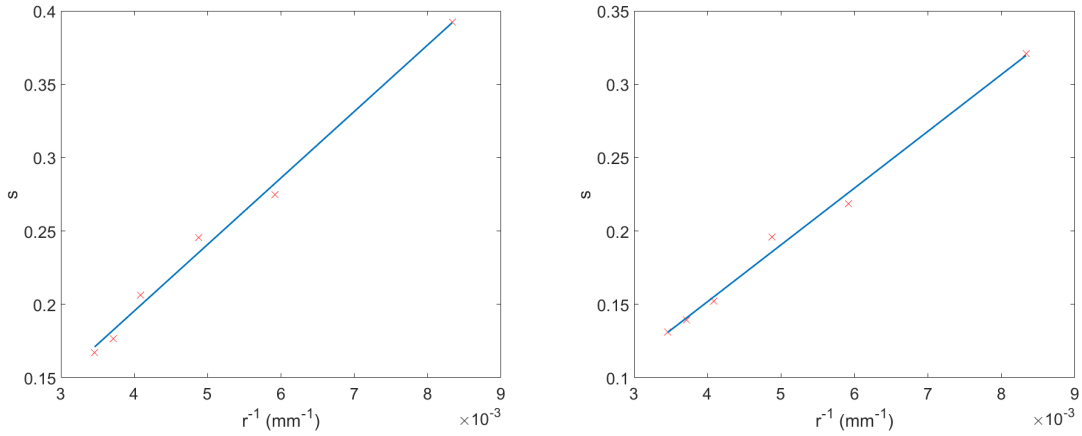
Having identified the foreground pixels, post processing was used to further reduce noise, eliminate unwanted detection of random objects and improve the shape of the expected object. This was first done by a morphological open (erosion followed by dilation) in order to eliminate groups of pixels which were too small (as defined by the structuring element), followed by a morphological close (dilation then erosion) which connects white pixels within a distance defined by a second structuring element in order to help with occlusion. The image was then filled to remove any remaining noise. In this case a disk shaped structuring element was used, since the object being tracked was a ball, a circular object. Depending on the size of the object the radius of this disk was adjusted.

The pixel co-ordinates of the centre of mass of the object were then used to find the objects angular position. Previously these positions were found by just clicking on still images, however with a video this wasn't possible. To automate this, the centroid of the object was set to be the co-ordinates of the centre of mass.

4 Results

4.1 Field of View

Using the method described in section 3.1 the horizontal and vertical field of view was calculated to be $\Delta_h = (51 \pm 3)^\circ$ and $\Delta_v = (36 \pm 2)^\circ$. This was determined using a linear least squares fit of the data (see Fig. 4) and the error propagation equation.



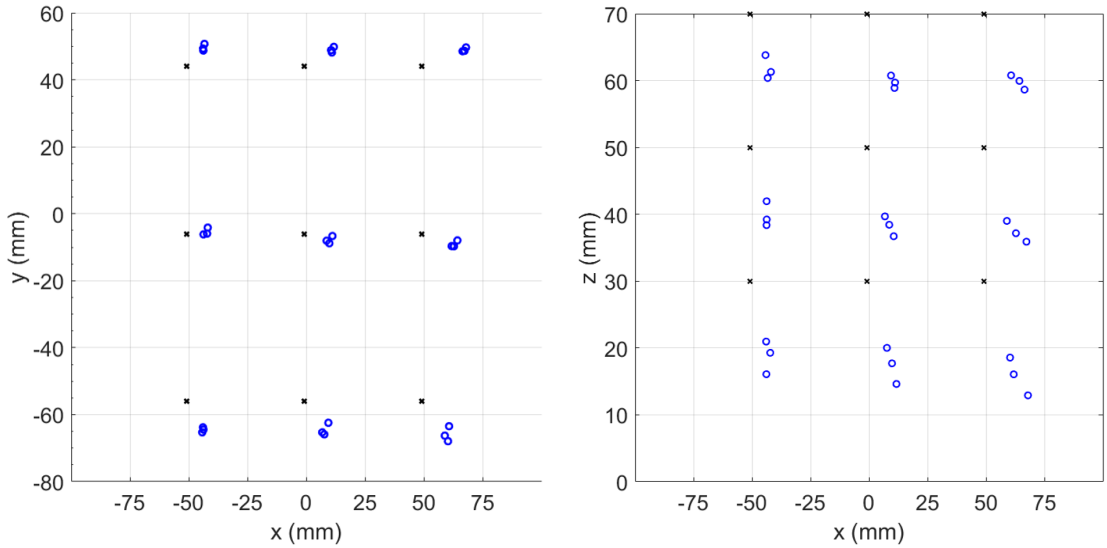
(a) Horizontal

(b) Vertical

Figure 4: Graphs showing best-fit lines used to find the horizontal and vertical field of view angle.

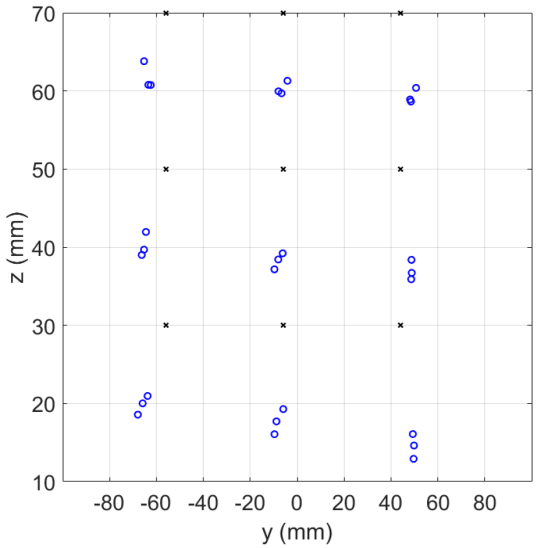
4.2 Finding the Position of an Object in 3D

Using the technique described in 3.1 it was found that on average we were able to calculate the position of a known point within (17 ± 3)mm in three dimensions and (13 ± 3)mm in two. The height difference in z calculated by each camera was (11 ± 3)mm (see Fig. 5).

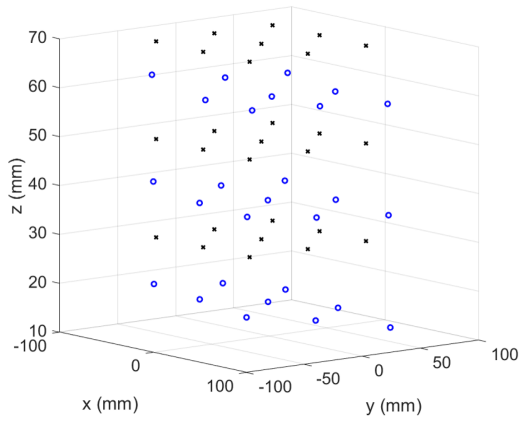


(a) x-y plane

(b) x-z plane



(c) y-z plane



(d) 3D

Figure 5: Graph showing calculated position (blue) versus known position (black) using the method described in part 3.1.

5 Discussion

5.1 Hardware

As mentioned in part 3.2 the cameras used, recorded at 30fps. The maximum speed v_{max} at which the system could detect an object in 3D is

$$v_{max} = fps \times (u + u_0) tan\left(\frac{FOV^\circ}{2}\right)$$

where $u+u_0$ is the distance from the camera to the object along the principle axis perpendicular to the direction of motion, and FOV° is the field of view along which the object is moving. With more information regarding the region of interest (area that the cameras field of views' overlap) bounds on v_{max} can be found. Hawk-Eye uses high-speed 340 fps cameras meaning it can track objects going over 10 times faster than our system. This also reduces motion blur, which, depending on the sensitivity of the foreground detection, may be falsely picked up. For example it is known that it

can track tennis serves as 150mph[5] however the robustness of the system under these conditions is not known. Collins & Evans [2] make good discussion of the point that Hawk-Eye is commercial and therefore doesn't need to be clear about their error, so long as it meets the specifications given by the officiating body (such as Fifa). In a 2012 video, they quote 3.6mm in tennis[3], information on their website states 2.6mm in 2006[5]. What they're quoted error actually means is largely not understood, for example the deviation of this number and under what conditions the deviation increases, making it difficult to make a comparison. Some sports federations/associations simply state it is within an agreed upon bound and leave it at that[1]. This makes the system difficult to compare to and to identify possible weaknesses.

As mentioned (section 1) Hawk-eye uses many cameras (up to 10 in tennis) compared to the two used. Not only does this guard against occlusion but it also creates redundancy; in the same way that we can take the average of two calculated z values, Hawk-Eye can take the mean over many x, y and z values, reducing it's random error.

Other than fps, the camera details are not known but it is reasonable to assume that another advantage Hawk-Eye has over Nott-Hawkeye is the camera resolution. From the error propagation equation: error on α and β is inversely proportional to resolution since

$$\delta_\alpha = \frac{\partial \alpha}{\partial FOV}\delta_{FOV} = \frac{\delta_{FOV}}{resolution}$$

It was only after FOV calibration and taking the images used in 3.1 that we noticed the resolution of the camera could be 1280x720 instead of 600x480. Using this higher resolution, the horizontal and vertical angle error could have been reduced from 0.0042° and 0.0043° to 0.0020° and 0.0029° respectively. However, considering the following error discussion this is probably not significant.

Lens distortion will inevitably have introduced errors which aren't seen using radio-frequency and magnetic field methods. MatLab is able predict and adjust for lens distortion, however this was discovered *after* data analysis and was not included in the report. Obviously an improvement over an estimate, would be taking the relevant preliminary tests (taking pictures of a grid) to get a specific profile for each lens used and would give a quantitative value of the error associated with lens distortion.

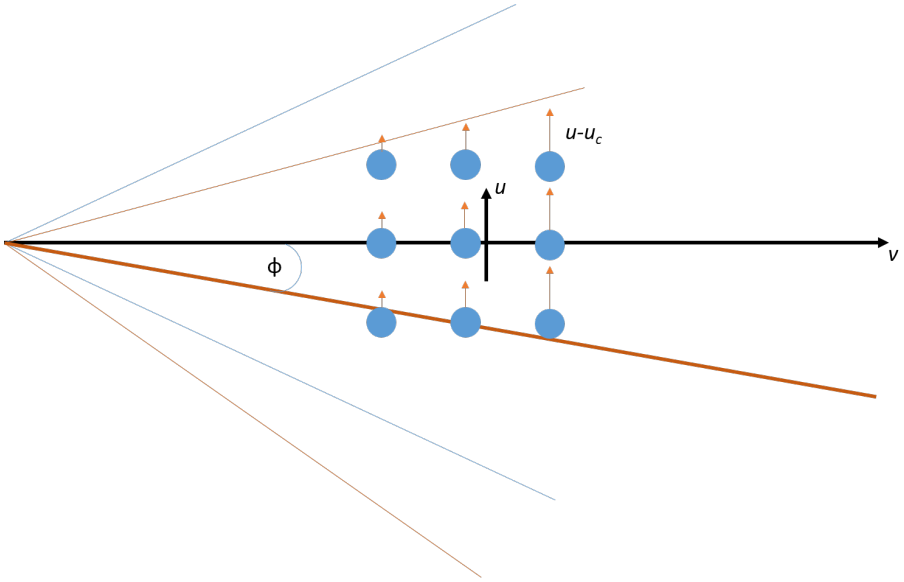
5.2 Calibration

For the following analysis "absolute" error wasn't used, instead the difference between a measured point and a calculated point was used, to make it clear whether the system was under/over shooting its position calculations. Sample correlation coefficients were used, due to time constraints, to identify where systematic errors might be. It is understood that it is not a measure of gradient, but it is a measure of statistical certainty that some variables may be related. This discussion then attempts to explain what the physical cause of this may be, and how, with more time, this could be better quantified. For the following the sample size is $n = 27$ as described in Part 3.1.

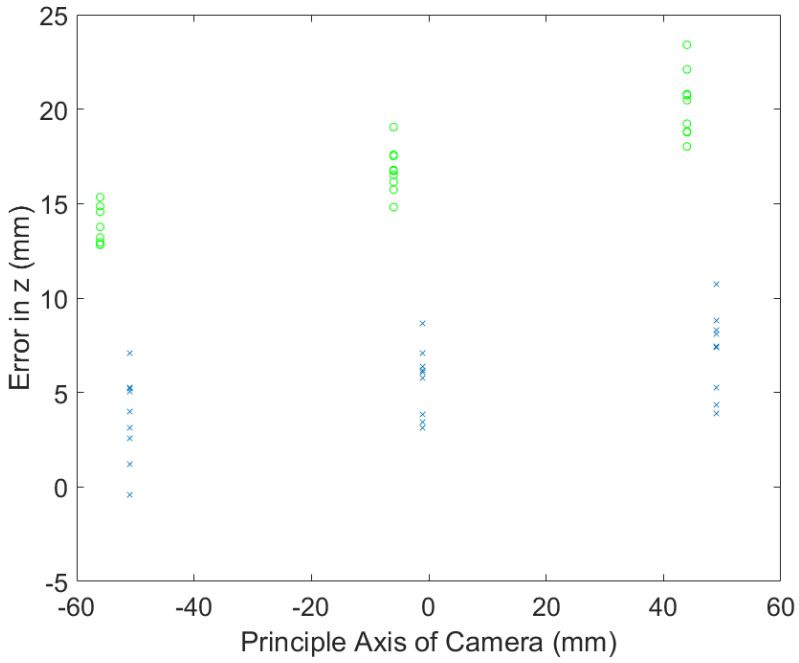
Figures 5b & 5c clearly show the system underestimating the height of the object on average by (-11 ± 2)mm. This is most likely due to a slight up turn of the cameras causing it to under estimate the height of the object i.e. the object would appear lower in the image than it should (see Fig. 6a) . This can be quantified by the statistically significant correlation coefficients r between the known x and y positions and the difference in calculated to known z positions, $(z - z_c)$. It was found that $[r_{xz} = 0.4381, n = 27, p = 0.0223]$ and $[r_{yz} = 0.6451, n = 27, p = 0.0003]$. Since r_{yz} and r_{xz} corresponded to the correlation of camera 1 and camera 2 respectively, this suggested that camera 1 was less aligned with the x-y plane than camera 2.

This was confirmed by comparing the calculation of z from each camera instead of the mean z used in the previous paragraph. The correlation between the difference of z from camera 2 and x $[r_{xz2} = 0.5732, p = 0.0018]$ is weaker than camera 1 difference in z and y $[r_{yz1} = 0.8978, p = 0.0000]$.

Currently, the geometry of the system assumes the camera origin to be parallel with the playing surface - in Hawk-Eye the cameras are usually pointed down from high - as described in Part 1, thus upon extending this project, an improvement would be to generalise the geometry to account for this zenith angle. Or the angle can be taken into account in post processing (as described below). One could reasonable argue after making this analysis that a weighted mean of z_1 and z_2 or even exclusion of z_2 would better find z. (See Fig. 6b)



(a)



(b)

Figure 6: a) Diagram showing the effect of misaligning a camera to the co-ordinate system. The brown principle axis is shifted by an angle ϕ away from the co-ordinate system. b) Graph showing y vs camera 1 error in z (green) and x vs camera 2 error in z. Used to argue that both cameras were tilted upwards - camera 2 more so than 1.

It was also assumed that the cameras were pointed perpendicular to each other. A similar analysis as before was made to test if this was a fair assumption. With increasing x there was a small correlation with the error in y, $(y - y_c)$ $[r = 0.1309, n = 27, p = 0.5152]$ however, due to

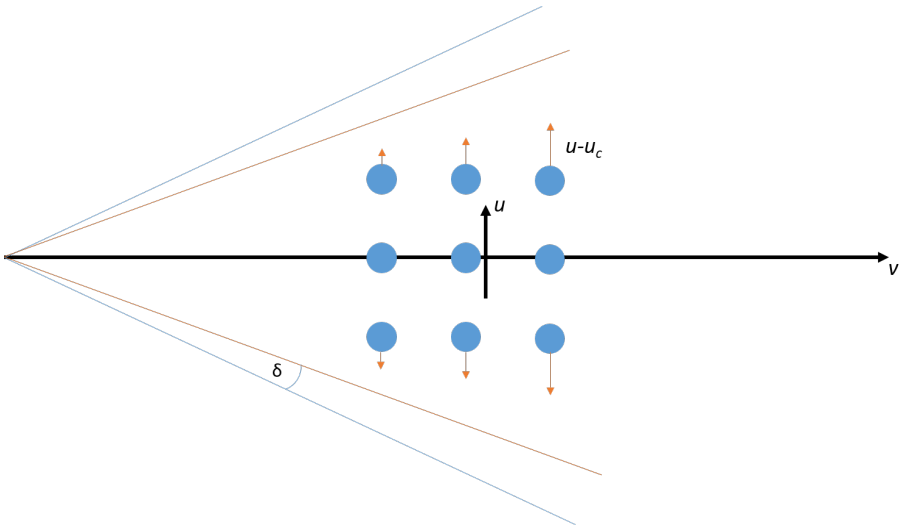
the statistical uncertainty of this correlation, it suggested there was a significant azimuthal angle in camera 2 relative to the proposed x axis. On the other hand, as y increased the difference in x decreased, $(x - x_c)$ such that: $[r = -0.4202, n = 27, p = 0.0291]$ suggesting camera 1 formed a slight negative angle relative to the y axis. Again, to solve this problem the azimuthal angle of the camera should be considered in the theory. Or the angle can be taken into account in post processing (as described below).

The pitch of the camera could also affect results (image is effectively rotated) under normal circumstance - for example a camera on a tripod could be tilted such that its horizon is not horizontal - however under test conditions the camera wass placed directly on the "court" therefore it is reasonable to assume this is negligible (as opposed to the other rotational freedoms).

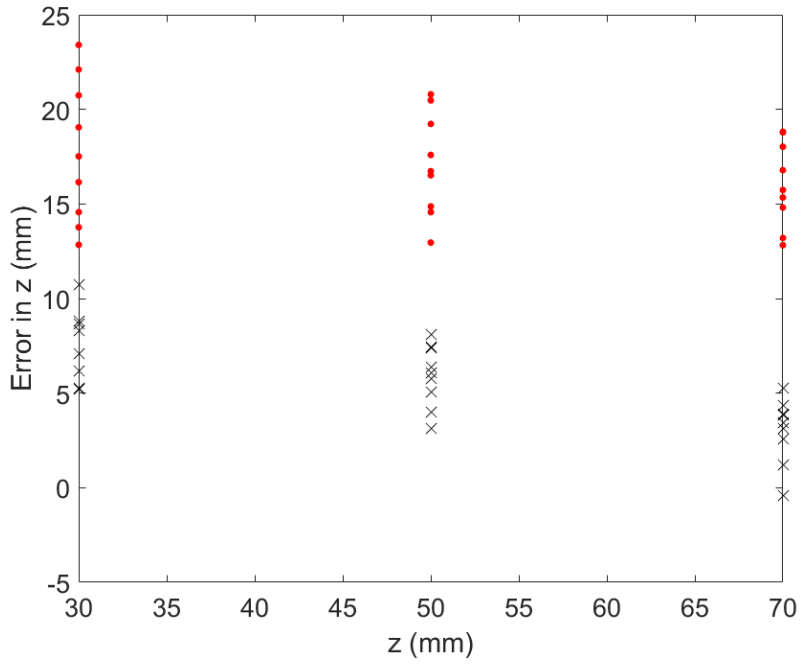
One could actually find out these angles using the non-normalised gradients (as opposed to correlations) and use them to adjust the results:

$$tan(\phi) = \frac{u - u_c}{v + v_0} \tag{10}$$

where u and v are perpendicular dimensions and ϕ is the angle the camera makes with v (see Fig. 6a) thus in a plot of $u - u_c$ versus $v + v_0$, $tan(\phi)$ is equal to the gradient m thus $arctan(m) = \phi$. This analysis along with the method described in Part 3.1 could be used to remove these error under unknown circumstances. Or they could be measured and included in the theory seen in part 2.2 & 2.3.



(a)



(b)

Figure 7: a) Diagram showing the effect of miscalculating field of view. If δ is negative the effect is reversed. This complicates the effect seen in 6a. b) Graph showing z vs camera 1 (red) and camera 2 (black) error in z. Used to argue that both cameras had different vertical field of views.

There was also a statistically significant correlation between each coordinate and the error associated with it. For x $[r = -0.8055, n = 27, p = 0.0000]$, remembering that by error I mean $u - u_c$. This can be interpreted as the calculated value increasingly over estimating to the right along x (see Fig.5a). By finding the mean on the error we can find out if x_0 is the error. The pattern in y, $[r = -0.9702, n = 27, p = 0.0000]$ is better explained by a miscalculation in the FOV of camera 2. This makes sense, as it was assumed that both cameras had the same FOV. This error analysis can be interpreted as under estimating in negative y and over estimating in positive y (See Fig.5a). This would suggest the horizontal field of view of camera 2 was larger than camera 1.

For z, each camera is now considered separately. There is no significant error in z as calculated by camera 1: $[r = -0.2473, n = 27, p = 0.2136]$ following from the interpretation of y this can also

be explained by our measured value of vertical field of view for camera 1. And again the error from camera 2 is significant $[r = -0.7387, n = 27, p = 0.0000]$ when calculating z (see Fig. 7b). This is analogous to the discussion associated with Fig.6b but it is harder to understand if the error is due to field of view overestimation or tilting, since it is likely to be a combination of both. To be more certain, the calibration points should include negative z values (or at least points smaller than z_0).

In general, the previous part of error analysis can be generalised, such that a plot of u versus u_c would give a translational error in camera position via the y-intercept and a error in FOV angle proportional to the gradient. This was not achieved due to time limitations. Extending this analysis would provide a fully fledged calibration technique. Which, after being applied would give the random error of the system.

Some of this analysis could also be attributed to lens distortion. However without the time to develop a lens distortion profile this is just conjecture.

5.3 Motion Tracking

Perhaps one of the largest weaknesses of this report is how the object tracking algorithm wasn't actually tested. By using known positions and manually selecting pixels - as described in Part 3.1 - the image processing described in part 3.2 is skipped. The errors quoted in Part 4.2 are thus an underestimate for the total system/process. Using FIFA's goal line technology testing manual [1] as a guide, the robustness of Nott-Hawkeye under realistic conditions could be tested quantitatively. Another way to test the Nott-Hawkeye system - even it's robustness for fast moving objects would be to trace around a known, still object with a object that can be tracked using both a magnetic or RFID system, and Nott-Hawkeye. The calculated dimensions of the object can then be compared to both the known object and alternative system.

For the algorithm to work in a more generalised sense (to ignore swaying tree's or moving crowds), a larger number of training frames would be necessary in order to widen the Gaussian model of the background pixels. If the object was already on scene when recording started, the algorithm would show where it began and where it moved to - this may be helpful but if not, a faster learning rate would be needed.

As described in the introduction (section 1) Hawk-Eye uses a network of cameras in order to improve the accuracy of the system. Depending on the sport this could be 5-10 cameras. Under the relatively controlled lab conditions occlusion was not much of an issue, except for how our cameras were positioned too low, so any creases in the paper could obscure the object preventing detection of foreground pixels (in the case of a ball rolling along the floor). This highlights the importance of not only having multiple cameras but also logical camera positioning. Upon extending this project it would be an improvement to position cameras in the top corners of the proposed region of interest.

In the theory, (section 2), the centre of the image is defined by half the resolution in each dimension. However the resolution of the image is even resulting in a off centre origin. This was assumed to have a negligible effect on results.

The algorithm assumed that only one object was being detected. By having the ability to track multiple objects and by using a feedback loop, which uses laws of motion as a predictor of the object location in next frame, known as a Kalman filter, the system would be adapted to handle multiple objects. Incidentally, this would also mitigate the error from occlusion as the shape and position of the object could be modelled/overlaid. To improve the robustness of this feedback, more geometric data about the object could be used. For example, does its area or diameter (major/minor axis) make sense, given it's distance from the camera. An issue arising from this would be ensuring that the same object was correctly labelled in each perspective. This could be done when the object moves out of the FOV of one camera into the overlapping FOV of both cameras. As long as the cameras "know" each others' fields of view, a region of pixels from one camera could correspond to a region of pixels to another so that, when the object is in both, it can be correctly labelled and tracked by either camera as the same thing [4]. Furthermore, colour

information could also help with tracking. Foreground detection could be performed separately in each colour space, one could even pre-process by binarising within a bound around the colour of the ball, great for snooker.

6 Conclusion

Our system is able to detect objects with (17 ± 3)mm accuracy in 3D space compared to Hawk-Eye, which inconsistently quotes "mm" accuracy - tennis requires less then 5mm error and in 2012 claim they are at 3.6mm (with no quoted standard deviation) average and FIFA require ± 1.5cm under various conditions. Considering this number is given without any calibration applied and the time and resources available for this project, this shows that Nott-Hawkeye has promise. The next step for this project is generalising/widening the lab conditions and testing the robustness of our system under a wider rage of conditions.

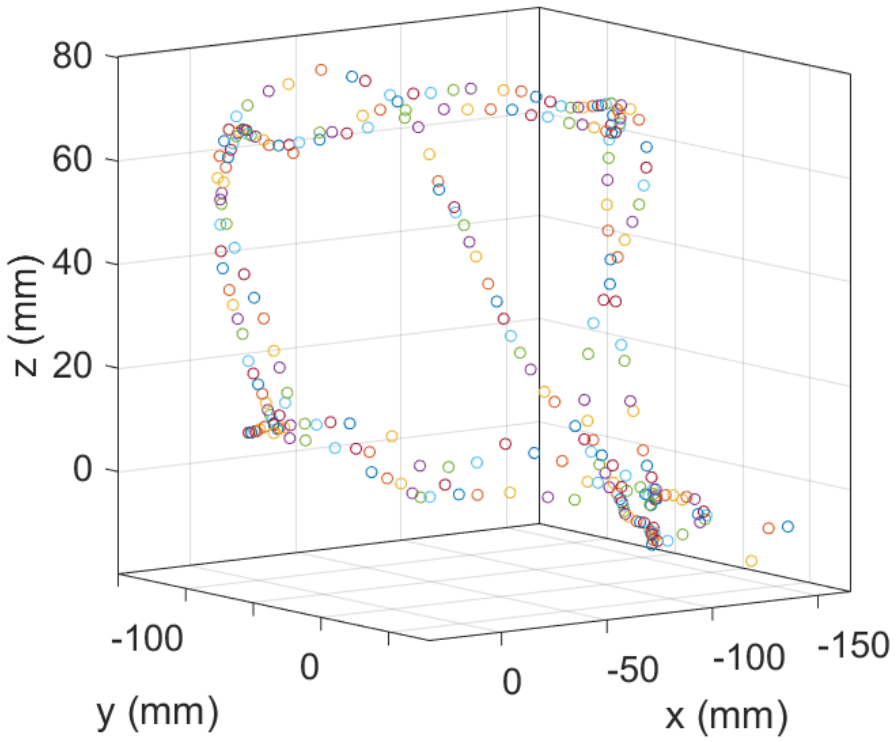


Figure 8: Graph showing tracing of 3D object

References

[1] FIFA QUALITY PROGRAMME, (2014) *Testing Manual - Goal Line Technology*, FIFA. `https://football-technology.fifa.com/media/1011/goal-line_technology_testing_manual_2014.pdf`

[2] Collins, H. Evans, R. (2008) *You cannot be serious! Public understanding of technology with special reference to "Hawk-Eye"* Public Understanding of Science, volume 17, pp.283-308.

[3] Australian Open TV (2012) *Inside Hawk-Eye*, YouTube, `https://youtu.be/XhQyVnwBXBs`

[4] Khan & Shah. (2003). *Consistent labelling of tracked objects in multiple cameras with overlapping fields of view*, Transactions on Pattern analysis and machine intelligence, volume 25, p1355.

[5] `https://www.hawkeyeinnovations.com/`