Remote Summer Studentship Report

A photogrammetry system for an accurate Indoor Positioning System for Field mapping experiments

Robert Hudson

---

1 Introduction

The scope of this project is a feasibility study for a photogrammetric system to be used in magnetic field mapping experiments.

In the CERN Detector Technology group a team which is tasked with measuring magnetic fields for the magnets of different experiments. Examples of such magnets are solenoids, dipoles, toroids, etc.

In order to measure the magnetic field, one makes use of a moving gantry. A beam is usually made to move on rails. On the beam a gantry is installed, which sweeps the volume of the magnet. On the gantry several magnetic field (hall) sensors are mounted. As the volume is swept, a map of the magnetic field is generated. Two information are needed for generating a field map. The first one is the magnetic field intensity, which is provided by the hall sensors, the second is the position (x, y, z) or (z, r, θ) depending from the magnet geometry.

This position is provided by encoders mounted on the gantry. In order to have a good accuracy, the whole mechanical support needs to be machined with a very high precision. Small positioning errors might importantly affect the final magnetic field map.

In this project we explore the possibility to drop the accuracy requirement on the mechanical structure, and make the sensor assembly "position aware". This belongs to the realm of "indoor positioning systems". The accuracy required is better than 1 mm.

One important constraint is that all parts of the system need to be compatible with strong magnetic fields ($>$ 1 T). This rules out most electric motors (with the only notable exception of piezoelectric motors).

We explore here the idea to use photogrammetry in order to estimate the pose of the magnetic field probe. We assume that a target is placed somewhere at the end of your magnet (assuming for simplicity a solenoid topology, e.g. the solenoid of the CMS experiment). The volume is swept with magnetic field sensors, with a fixed focus camera mounted with the sensors. The camera is looking at a target at the end of the magnet, with some pattern on it, and is thus able to determine its position in space. Aim of this project is to quantitatively assess the resolution and accuracy limits for such a system.

2 Instrumentation and Software

Some of the results reported here were obtained using a simplified setup made of a chessboard target, printed on paper, and a standard webcam, for image acquisition. The image acquisition, pattern recognition and analysis was performed using the tools included in the OpenCV machine vision toolset [1].

3 Camera Calibration

Camera calibration was performed using the method suggested by OpenCV documentation [2]. Python code is also provided in `camera calibration.py` file attached to this report. A summary of the code is presented below:

```
# Import training image dataset
init images
```

```
# Set up list of chessboard corner coordinates (i.e. a 9x6 grid)
init objpoints

# List to store coordinates of chessboard corners found in images
imgpoints = []


for img in images:
        # Find the chessboard corners using the inbuilt function, then get a more accurate
            ↪ estimate by the cornerSubPix function
        corners = cv2.findChessboardCorners(img, (9,6), None)
        corners2 = cv2.cornerSubPix(img, (9,6), corners2)
        imgpoints.append(corners2)


# Perform the camera calibration algorithm
# mtx is the camera's intrinsic matrix and dist is the vector of distortion coefficients
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, img.shape[::-1],
    ↪ None, None)
```

There was little information available for what makes a good set of training images, other than that they should be pictures at varying positions and angles. In order to try to get an accurate calibration matrix, 255 training images were used (available in `images/camera calibration 2`), with the target moved in the xy plane at 3 different depths, and rotated through different angles at the corners and centre of the camera frame. Tutorials suggest only 20 images are needed for calibration, but as they do not specify which poses should be used a much larger dataset was collected. This made the calibration runtime extremely long, but hopefully makes the calibration better. The camera matrix $K = \begin{bmatrix} 853.55 & 0 & 342.75 \\ 0 & 854.66 & 250.20 \\ 0 & 0 & 1 \end{bmatrix}$ and distortion vector $d = \begin{bmatrix} 0.183 & -0.773 & 0.007 & 0.001 & 1.365 \end{bmatrix}$

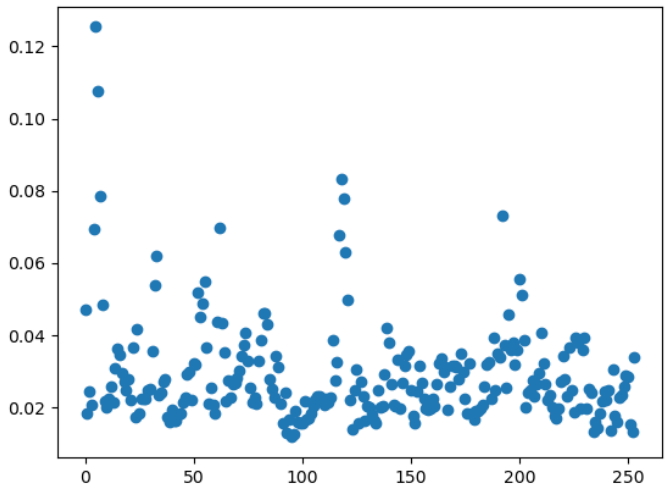


Figure 1: Plot of average reprojection error (y-axis) for each image in training dataset (x-axis)

Figure 1 shows the average reprojection error for each image in the dataset. The reprojection error is the distance between the location of the detected corner, and the expected location of the corner when projected using the calibrated camera parameters. Most of these errors are low so the calibration is likely to have been successful.

The calibration is checked by running the training images through a different calibration algorithm, this time found in the Computer Vision Toolbox from Matlab. The Matlab computed camera matrix was $K = \begin{bmatrix} 859.76 & 0 & 340.31 \\ 0 & 860.26 & 236.10 \\ 0 & 0 & 1 \end{bmatrix}$. This is similar to the OpenCV calibration matrix, however there are some differences. Without knowing the camera parameters, it is unknown which matrix is more correct, which highlights a flaw in this method of calibration as we cannot easily find out whether our calculated camera matrix is correct. Nevertheless, the OpenCV output was saved and used in future experiments.

3.1 Camera Accuracy Experiments

Once the camera was calibrated, the accuracy of the camera was tested by measuring how well it could locate known points. The camera was positioned at a measured distance away from a whiteboard, on which a target was moved in the xy plane. The experiment setup is shown in figure 2. The target position was measured relative to the edges of the whiteboard using a tape measure. The z distance to the camera was also measured with a tape measure, although it should be noted that this measurement could not be very accurate, and as moving the camera in the z direction would also slightly change its angle and xy position, the images taken at different z positions were not compared with each other, only with images taken at the same distance.

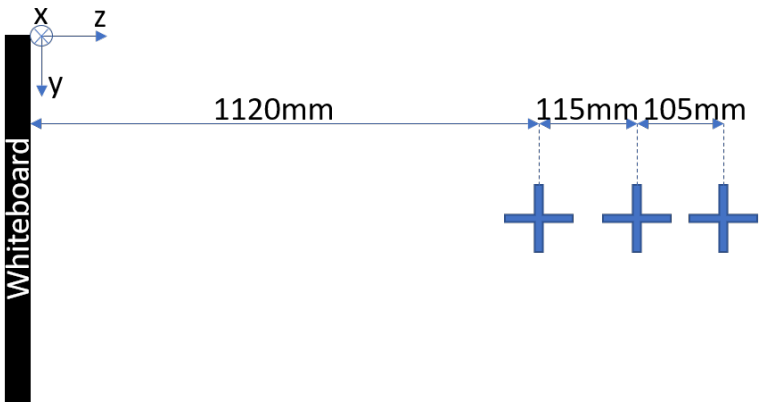


Figure 2: Experiment setup. Crosses mark the position of the camera

Because the camera position could not be measured accurately relative to the whiteboard, the origin was chosen to be the position of the target in the first image. The `cv2.solvePnPRansac` was used to get 3D position from the detected features in the 2D image plane. This solves the perspective n point problem, where camera pose is estimated from n image points which can be mapped to their real world equivalents [3]. As camera pose has 6 degrees of freedom and the chessboard has 54 matched features, this is an over-constrained problem so an optimisation is used to match the data best. The use of RANSAC allows for some resistance to outliers [4], which may be encountered if the corners are not detected properly, for example for images which are too far away so cannot be resolved properly. The algorithm used to predict pose is presented below:

1. Import images and camera matrix.
2. Generate grid of chessboard corner points.
3. Find location of chessboard corners in each image using `cv2.findChessboardCorners` and `cv2.cornerSubPix`.
4. Estimate pose using `cv2.solvePnPRansac`.
5. Import measured location of target (stored in `points.csv`).
6. Set first image as origin, and calculate all positions relative to this.
7. Calculate distance between the measured points and points predicted from image. This is the error.
8. Repeat for each camera z position.

The code is attached in `position.py`. In this code, there are some extra manipulations of the data to calculate the errors properly. In the following description, world points will be used to denote the set of measured target coordinates, and image points will denote the predicted position of the target calculated from the input image. When the camera was set up, it was not perfectly perpendicular to the whiteboard, so there is an extra rotation which must be accounted for by our world points in order to calculate the error properly. Instead of having no change in z, there was a 60mm change in the z coordinate across the whiteboard. By small angle approximations, this should not have too big an effect on the xy positions however it was decided to remove this source of error. As it was too difficult to measure this angle properly, it was done in Python. To do this, a rotation matrix was calculated which would map the skewed plane onto a plane with normal pointing along the z axis. The algorithm is outlined below:

1. Import image points. These are on a skewed plane.

2. The image points are observed in noise so need to be mapped to a plane as the matrix is currently rank 3. Compute SVD of the image points matrix. The smallest singular value is 100x smaller than the other two singular values, so can be removed to make the image points rank 2 and form a proper plane.

3. Find two unit vectors, $\mathbf{a}$ and $\mathbf{b}$ present on the plane. These cannot be co-linear.

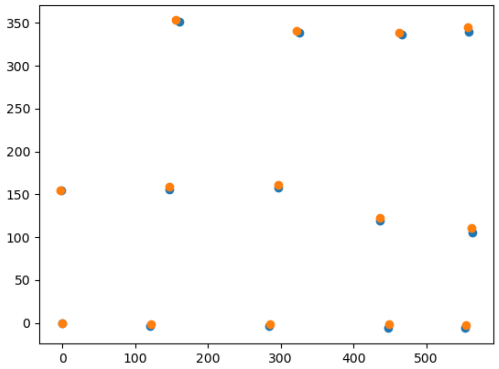
4. Make $\mathbf{b}$ orthogonal to $\mathbf{a}$ by removing its component in the $\mathbf{a}$ direction.

5. Compute normal to plane $\mathbf{n}$ by the cross product of $\mathbf{a}$ and $\mathbf{b}$.

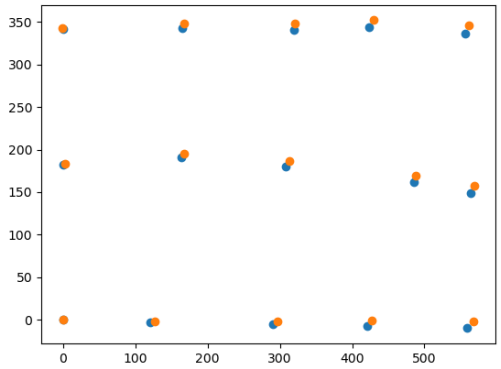
6. Form rotation matrix $\mathbf{Q}$. We want to map $\mathbf{a}$ to the x-axis, $\mathbf{b}$ to the y-axis and $\mathbf{n}$ to the z-axis. Therefore, $\mathbf{Q}\begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{n}\end{bmatrix} = \mathbf{I} \Rightarrow \mathbf{Q}^T = \begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{n}\end{bmatrix}$.

7. We can now project our measured world points (which assumed a plane perfectly aligned to the camera) to the plane the camera actually observes, by $\hat{\mathbf{p}} = \mathbf{Q}^T\mathbf{p}$. This puts both image and world points in the same reference frame so the error can be calculated.

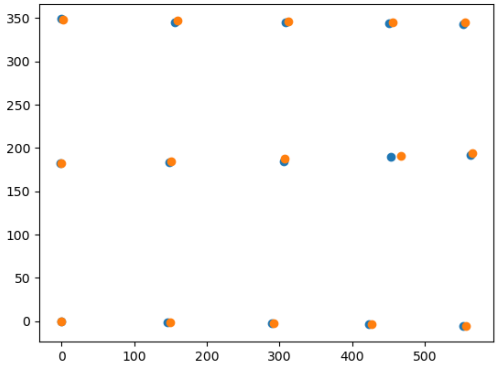
8. Calculate the distance between matched points to find the error.



(a) Plot of world and image points for z=1120mm. Mean error 3.5mm



(b) Plot of world and image points for z=1235mm. Mean error 6.73mm



(c) Plot of world and image points for z=1340mm. Mean error 3.35mm

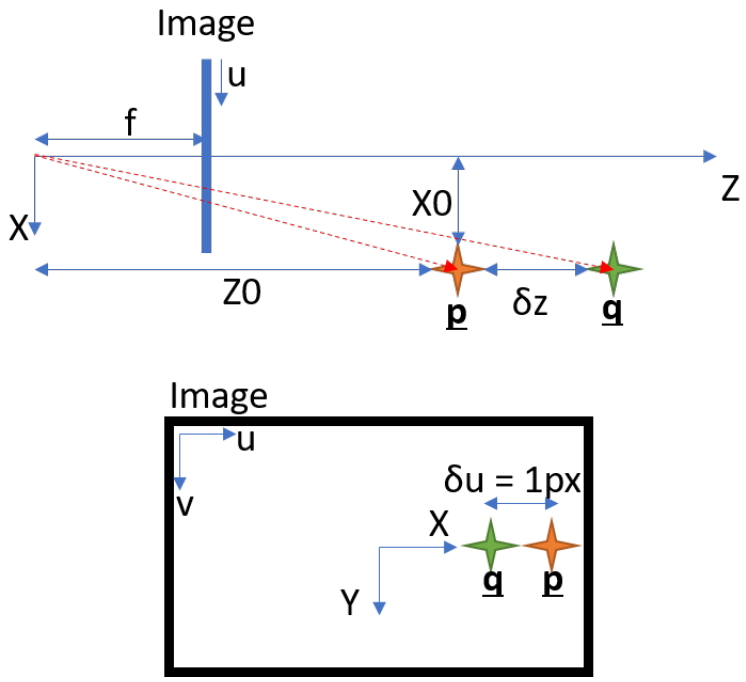
Figure 3

Figure 3 shows the measured and predicted locations of the target for three different z-distances. The results are of relatively good agreement given the inaccuracies in measuring the position by hand, so the results look promising. However, one would expect predictions to get worse with increased distance, which is not the case. This will be investigated further to see if it is due to poor quality images being taken or errors in measurements of some positions. The camera accuracy also needs to be measured along the z-axis, which will require better calibration of the test equipment.

4 Theoretical Limits

In this section, the theoretical limits for a single and stereo camera system are estimated. In particular the resolution and accuracy are estimated for a Cartesian system, in the three coordinates.

4.1 Single Camera Analysis



The scenario considered is shown in the above figure. It involves a point p observed in the XZ plane, which is moved a small distance δZ until it moves a small distance $\delta u = 1$pixel in the image plane (point q). This is the smallest δZ the camera can resolve. The points are taken at $Y = 0$ so only movements in u need to be considered in the first instance.

Real world coordinates XYZ are converted to pixel coordinates uv by the perspective camera model, namely $u = \frac{k_u f X + u_0 Z}{Z}$ and $u = \frac{k_v f X + v_0 Z}{Z}$. f is the focal length and $x = \frac{fX}{Z}$ models the perspective projection to the image plane, and k_u and u_0 convert from a coordinate system centred in the image plane (xy) to the pixel coordinate system by $u = k_u x + u_0$.

Therefore, we can set up two equations:

$$p = \frac{k_u f X_0}{Z_0} + u_0$$

$$q = \frac{k_u f X_0}{Z_0 + \delta Z} + u_0$$

Subtracting these gives:

$$\delta u = p - q = \frac{k_u f X_0}{Z_0} - \frac{k_u f X_0}{Z_0 + \delta Z} = 1$$

px

Rearranging:

$$k_u f X_0 \frac{\delta Z}{Z_0(Z_0 + \delta z)} = 1$$

$$k_u f X_0 \delta Z = Z_0^2 + Z_0 \delta z$$

$$\delta Z(k_u f X_0 - Z_0) = Z_0^2$$

$$\delta z = \frac{Z_0^2}{k_u f X_0 - Z_0} \tag{1}$$

So this should give us the resolution δz as a function of X and Z. Increasing X makes the resolution better, as should be expected as having a greater X gives a greater angle from the normal for the light ray to follow, so a change in Z has a greater effect on changing u.

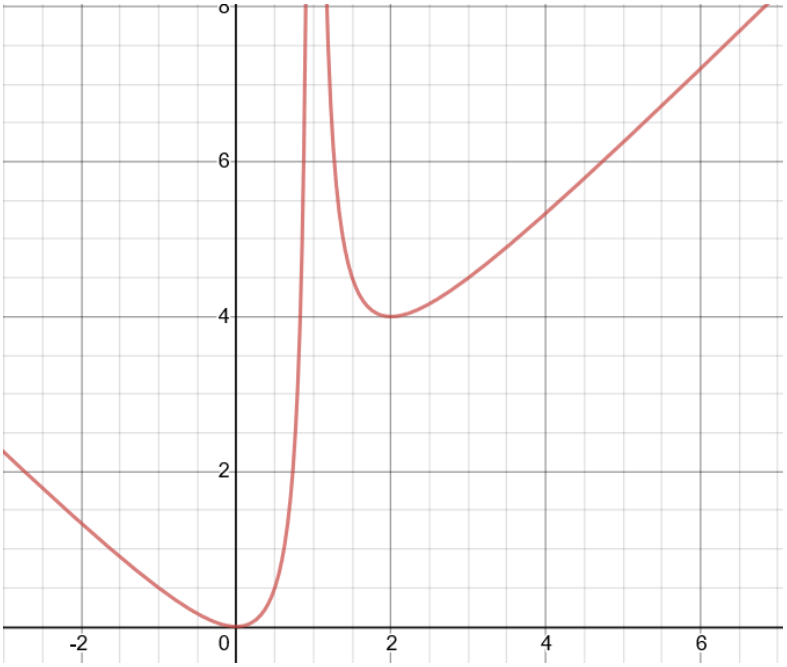


Figure 4: Plot of $|\delta Z|$ against Z, with $k_u f X = 1$

The effect of changing Z is somewhat stranger. One would expect that the resolution of the camera should be worse at large distances, as features will be closer together in the XY plane so give smaller changes in u. This behaviour is observed as $Z \to \infty$. The resolution asymptotes at $Z = k_u f X$, which is strange. One would expect an asymptote at $X = 0$, as the point would be on the camera's Z-axis, which would result in no change in image coordinates if the point was moved in the Z direction, as the angle it makes to the camera origin will not change so we cannot resolve Z at all. Another method for calculating the resolution was therefore proposed.

One can express a small, finite change in terms of partial derivatives:

$$\delta u = \frac{\partial u}{\partial Z} \delta Z$$

where

$$\frac{\partial u}{\partial Z} = -\frac{k_u f X}{Z^2}$$

Therefore, setting $\delta u = 1$px,

$$\delta Z = \frac{Z^2}{k_u f X} \tag{2}$$

which has the expected asymptote at $X = 0$. This is different from the equation 1 as it does not have Z subtracted from the denominator. The two are in relatively good agreement for the length scales required in this project ($Z < 10$m, $X < 2m$) so this is not a huge problem. Clearly equation 2 will only work for small δX and δZ, as it is a linear approximation, so because δZ becomes very large for large Z, this may be why the two equations begin to differ at large distances.

The resolution of changes in X can also be calculated by both methods, with them both yielding the same result:

$$\delta X = \frac{Z}{k_u f} \tag{3}$$

The same analysis can then be carried out for the v coordinate, yielding the same equations but with X replaced by Y and k_u replaced by k_v. As we only need to detect a difference in one of u or v, the final camera resolution is given by:

$$\delta Z = \min\left\{\frac{Z^2}{k_u f X}, \frac{Z^2}{k_v f Y}\right\}$$

To evaluate the performance of the webcam used in experiments, $k_u f$ and $k_v f$ were determined from the camera calibration matrix, which was generated by OpenCV. This may have some error depending on how the camera was calibrated, however it should give an estimate to the expected accuracy. The camera matrix gave $k_u = k_v = 854$.

X (mm)	Z (mm)	δX (mm)	δZ (mm)
100	100	0.12	0.12
100	1000	1.17	11.71
100	10 000	11.71	1170.96
1000	100	0.12	0.01
1000	1000	1.17	1.17
1000	10 000	11.71	117.10
10 000	100	0.12	0.00
10 000	1000	1.17	0.12
10 000	10 000	11.71	11.71

Table 1: Camera accuracy for a range of X and Z

Table 1 presents the theoretical minimum resolution of the camera for a variety of different (X, Z) positions. The camera performance varies greatly across the length scales studied, with completely unsatisfactory performance at Z=10m.

4.2 Stereo Camera Analysis

An alternative scheme is to use a stereo camera system to determine position. This should remove the issue of not being able to resolve changes in Z for a point aligned with the camera axis, as in the other camera it will not lie on the axis. The system considered is presented in figure 5.

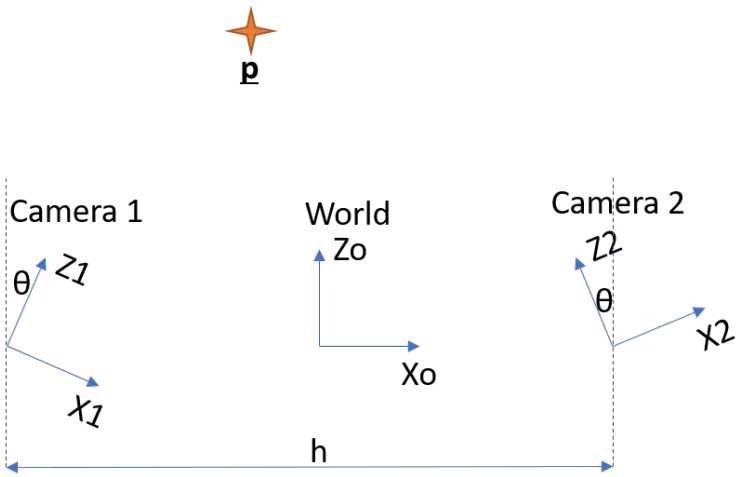


Figure 5: Stereo camera coordinate system

The cameras are set up such that they are equidistant from the world coordinates origin, with a common rotation θ. This analysis assumes all Y axes are parallel, so everything of interest occurs in the X-Z plane. This is not true in general, however it simplifies the problem so is a useful case to consider in the first instance. The two camera coordinate systems are related to the world coordinates by:

$$p_1 = R^T p - t$$

$$p_2 = Rp + t$$

where $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ and $t = \begin{bmatrix} h/2 \\ 0 \end{bmatrix}$.

To get pixel coordinate coordinates, we multiply the point by the camera matrix , $w_1 = \begin{bmatrix} su_1 \\ s \end{bmatrix} = Kp_1$ where $K = \begin{bmatrix} k_u f & u_0 \\ 0 & 1 \end{bmatrix}$. As w_1 is in homogeneous coordinates, we can find the pixel location $u_1 = \frac{su_1}{s}$.

$$\Rightarrow u_1 = \frac{k_u f(X\cos\theta + Z\sin\theta - \frac{h}{2})}{Z\cos\theta - X\sin\theta} + u_0$$

The errors can then be found by the partial derivatives method used for the single camera.

$$\Rightarrow \delta X_1 = \frac{\partial X}{\partial u} = \frac{(Z\cos\theta - X\sin\theta)^2}{k_u f(Z - \frac{h}{2}\sin\theta)}$$

$$\delta Z_1 = \frac{\partial Z}{\partial u} = \frac{(Z\cos\theta - X\sin\theta)^2}{k_u f(\frac{h}{2}\cos\theta - X)}$$

If we align camera 1 with the world coordinates ($\theta = 0$, $h = 0$) then we get the same result as the single camera situation, which is expected. A similar method can be used to find the errors for the second camera:

$$\delta X_2 = \frac{\partial X}{\partial u} = \frac{(Z\cos\theta + X\sin\theta)^2}{k_u f(Z - \frac{h}{2}\sin\theta)}$$

$$\delta Z_2 = \frac{\partial Z}{\partial u} = -\frac{(Z\cos\theta + X\sin\theta)^2}{k_u f(X + \frac{h}{2}\cos\theta)}$$

As we only need to detect a pixel change in one image to resolve a change in distance, the system resolutions are:

$$\delta X = \min(\delta X_1, \delta X_2)$$

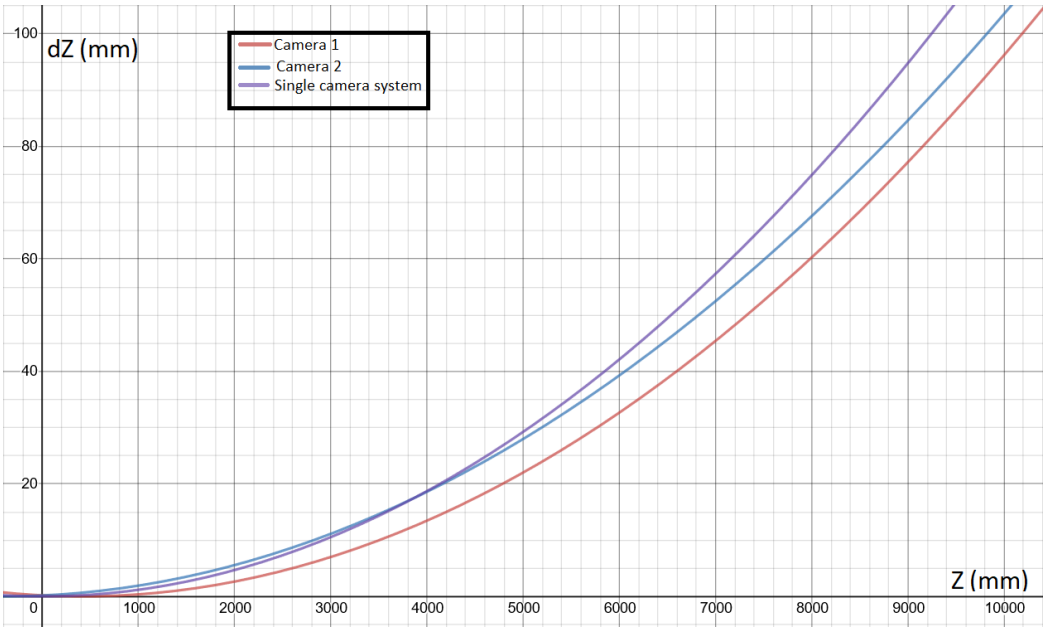
$$\delta Z = \min(\delta Z_1, \delta Z_2)$$

Figure 6 shows how the X and Z resolution vary as a function of Z distance, with all other variables fixed. The single camera system is also shown, and for virtually all cases the stereo system performed better. Nevertheless, figure 6a shows that the Z resolution quickly grows to unacceptable values at distances greater than 1m. The X resolution remains below 1cm for Z up to 10m. Increasing X betters the Z resolution of both cameras, so a target far from the origin is better. Increasing X also improves the X resolution of camera 1, but worsens camera 2. A large X is then still preferred, as we only use the smallest resolution for the overall system. Changing h and θ vary the resolutions of all cameras, so this becomes an optimisation problem to minimise the total error over the entire state space, which could be solved to design an optimal stereo system.

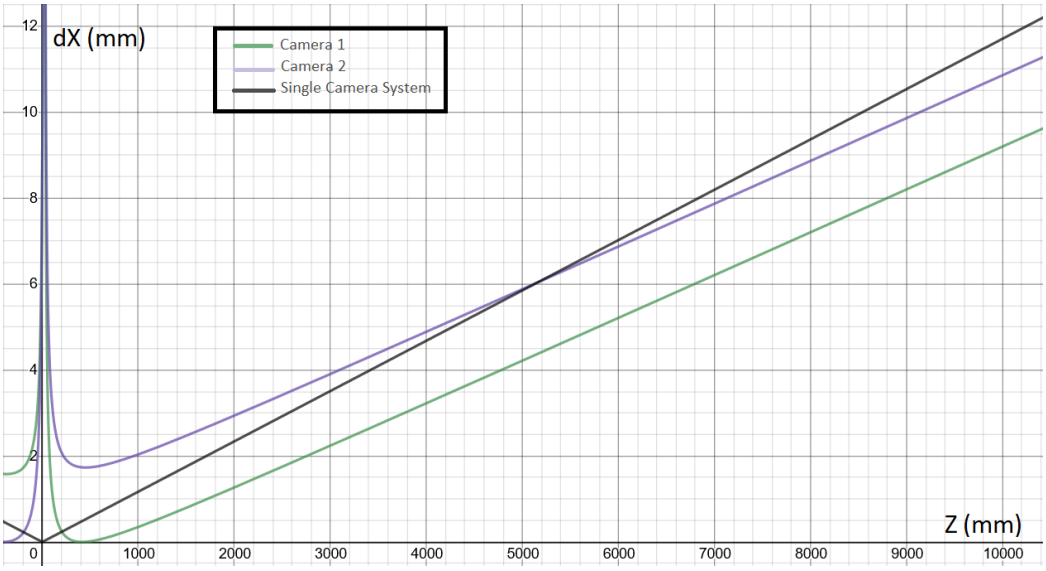
The improvement the stereo system gives to the resolution is modest, with a maximum improvement of around 2mm for the X resolution and 20mm for the Z resolution. This suggests that a stereo system may offer improvements on the single camera system, however these improvements will have to be balanced against the cost of a new camera.

One flaw of the above stereo analysis may be that is only considers the two cameras separately, instead of using information from both of them to infer location. If the two cameras were considered together, further improvements to resolution may be able to be made. Furthermore, the analysis assumes all the cameras are aligned in the same Y direction, so if a rotation in this plane is allowed as well further improvements may be made. In the current planar analysis, the two camera Z axes are coplanar and intersect at a point, so if they are allowed to rotate so they do not intersect this may improve resolution. Finally, only one point is considered in this analysis, whereas the actual target will have many corners which it can detect. Having a larger set of points to match may be able to better resolve position by detecting a change in a subset of points, instead of requiring all of them change by 1 pixel. Conversely, if an optimisation scheme is used to detect position, these individual changes may be averaged out. Another thing to note is that the algorithm used to detect the features of the target uses sub-pixel accuracy, so assuming this algorithm provides correct results then the assumption that a 1 pixel change is needed to detect a change in position may be an overestimate.

The analysis was repeated using the first method of calculating the position change which would result in a 1 pixel change in the image, with the final equations presented below. Again, these are similar to the partial derivative equations, but with extra terms added to the denominator. The results from the twot different methods are in agreement with each other.



(a) Z resolution



(b) X resolution

Figure 6: Comparison of stereo and single camera systems. Free parameters were chosen to be: X=1m, h=10cm, $\theta = \frac{\pi}{8}$

$$\delta X_1 = \frac{\partial X}{\partial u} = \frac{(Z\cos\theta - X\sin\theta)^2}{Z\sin\theta\cos\theta - X\sin^2\theta - k_u f(Z - \frac{h}{2}\sin\theta)}$$

$$\delta Z_1 = \frac{\partial Z}{\partial u} = \frac{(Z\cos\theta - X\sin\theta)^2}{X\sin\theta\cos\theta - Z\cos^2\theta + k_u f(X - \frac{h}{2}\cos\theta)}$$

$$\delta X_2 = \frac{\partial X}{\partial u} = \frac{(Z\cos\theta + X\sin\theta)^2}{k_u f(\frac{h}{2}\sin\theta - Z) - X\sin^2\theta - Z\sin\theta\cos\theta}$$

$$\delta Z_2 = \frac{\partial Z}{\partial u} = \frac{(Z\cos\theta + X\sin\theta)^2}{k_u f(X + \frac{h}{2}\cos\theta) - X\sin\theta\cos\theta - Z\cos^2\theta}$$

4.3 Example Application - CMS Magnet

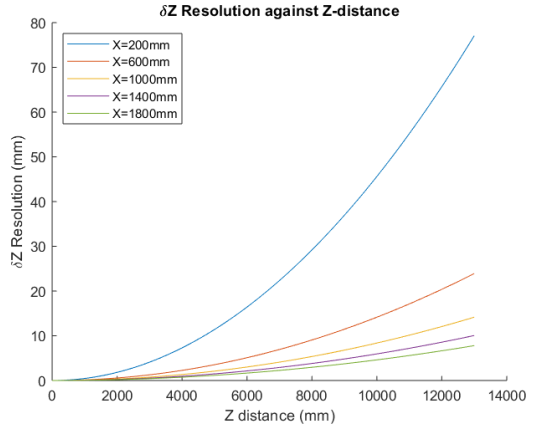
This section will present theoretical accuracy predictions for the mono camera system used to image the CMS magnet. The CMS solenoid was modelled as a cylinder of length 13m and inner bore radius of 2m. The camera considered was the Raspberry Pi High Quality Camera, as this was a high definition fixed focus camera, with a

datasheet which provided all the relevant parameters for modelling it. It is a 12.3MP camera, with a maximum focal length $f = 22.4$mm. It uses the Sony IMX477 sensor, and the datasheet allowed k_u to be calculated at $k_u = 538339$ pixels per metre. The X and Z resolutions were calculated using the equations:

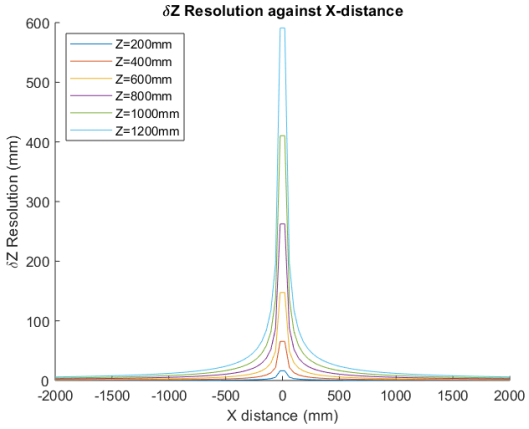
$$\delta X = \frac{Z}{k_u f}$$

$$\delta Z = \frac{Z^2}{k_u f X}$$

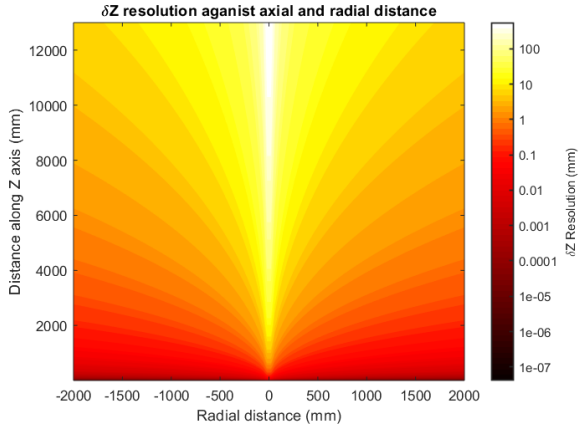
Symmetry was assumed in the X and Y axes of the camera, so only one needed to be considered and plotted.



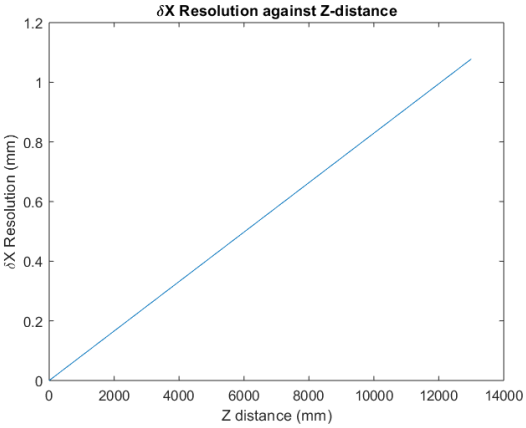
(a) Plot of Z resolution along Z axis, for different values of X



(b) Plot of Z resolution along X axis, for different values of Z



(c) Plot of Z resolution along X and Z axes



(d) Plot of X resolution along Z axis

Figure 7

Figure 7 shows various plots of the X and Z resolutions along the different axes. Figure 7c is useful for visualising the Z resolution as it varies across the cross section of the CMS magnet. The further away from the camera, the worse the resolution. Also, the closer to the Z axis, the worse the Z resolution. The resolution changes many orders of magnitude across the volume of the magnet. These are theoretical minimums however, so the resolution achievable in a practical setting could be a lot worse. The X resolution is simply linear in Z, and achieves a maximum value of 1.08mm at a Z distance of 13m, which is a good precision. Therefore, this camera will resolve X and Y well enough, but another method will need to be employed to get an accurate measure of the Z coordinate.

5 Conclusions

This study has partially confirmed the theoretical feasibility of using photogrammetry for the pose estimation of a field probe. The results are considered satisfactory for the estimation of in-plane (X, Y) coordinates, with a properly calibrated single-or-stereo camera system. However, the performance in "Z" (distance between camera and target) is only sufficient for low-Z values (< 1 m), making photogrammetry unapplicable alone for the pose estimation in long structures (e.g. the CMS solenoid). Further studies are to be foreseen to address this particular

limitation, integrating the photogrammetry systems with another ranging system for measuring the distance to the target, like for example laser ranging.

References

[1] "OpenCV Library." https://opencv.org. Accessed on 05-11-2020.

[2] "Camera calibration." https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html. Accessed on 02-07-2020.

[3] "Perspective-n-point." https://en.wikipedia.org/wiki/Perspective-n-Point. Accessed on 02-07-2020.

[4] "Camera calibration and 3d reconstruction." https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#boolsolvePnP(InputArrayobjectPoints,InputArrayimagePoints,InputArraycameraMatrix,InputArraydistCoeffs,OutputArrayrvec,OutputArraytvec,booluseExtrinsicGuess,intflags). Accessed on 02-07-2020.