Xbox Kinect Tracking

Oliver Gordon & Lukas Rier

Abstract

There are many scientific applications involving motion tracking rigid bodies, such as the development of prototype MEG sensors. We demonstrate a robust method for tracking a rigid body under controlled conditions, and successfully apply Horn's quaternion based algorithm to determine the translations and rotations about the x, y and z axes. We accomplish this with a cheap Xbox 360 kinect sensor.

We find that the horizontal and vertical fields of view of the kinect to be $\alpha_x = 70.2 \pm 1.0^\circ$ and $\alpha_x = 72.2 \pm 0.8^\circ$, in contrast to the officially provided values. We also demonstrate that whilst translations could be determined to floating point precision, rotations above 10° can not be accurately calculated. However, we calculate that rotations of under 10°, can be calculated to over 90% accuracy. By recording the known movements of a solid object, we ultimately find our overall system to have an accuracy of $A = 4 \pm 2mm$.

Whilst these results are promising, they are limited by our simplistic experimental method and the low reliability of the kinect's depth sensor with respect to heat and noise.

Contents

1 Introduction

The ability to track three-dimensional objects both accurately and reliably is of great use inside and outside of physics, with examples including accounting for motion in PET scans[1,2] or detecting if elderly people fall[3]. Regardless, tracking in three dimensions with traditional cameras is made difficult by the fact that at least two cameras must be placed in at least two different locations. However, both cameras can be placed in only one location if a colour camera is paired with an infra-red depth camera.

There are many such dual sensor systems commercially available, with examples including the Microsoft Kinect[4,5], Intel's RealSense[5] and most recently the Apple iPhone X. These systems are affordable, widely available and simple enough for any consumer to use. The kinect used in this project was brought to market in 2010[4], and at the time of writing can be bought for under £15[1,2]. Despite this cheap price, it can locate an object in all three spatial dimensions to millimetre precision.

One major use of 3D object tracking involves prototype MEG sensors[6]. By allowing an individual to wear a mask, they do not need to lay fixed and stationary, but instead can move around. For these, the position of the head must be tracked. This is done by determining the translation and rotations about the x, y and z axes for a total of six degrees of freedom.

The aim of this project was to develop code to capture colour and depth data from a Microsoft kinect. The data from both cameras were registered together and used to track at least three marker points on a rigid mask. Distances in pixels from the image sensor were converted to mm, and the six manipulations of the helmet were calculated as a function of time. Finally, the accuracy of the system was investigated.

2 Theory & Code

2.1 Data Acquisition

To acquire data, a kinect v1 for xbox 360 was connected by USB to a computer with the Kinect 1.8 SDK[7] and MATLAB support package[8] installed. Although only one kinect was used at any one time, a total of two kinects were used over the course of the project. The setup is demonstrated in Figure 1.

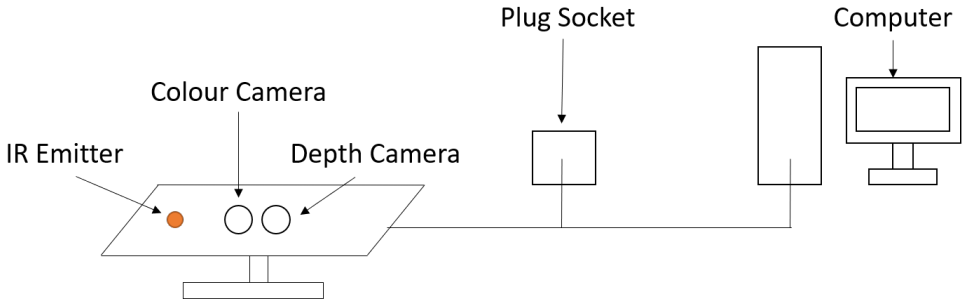


Figure 1: Figure to demonstrate the setup and sensors of the kinect v1 used in the project.

The kinect contained two sensors; a 640x480 colour sensor for image data, and a 640x480 IR sensor for depth. Colour information was transmitted in the RGB format, whilst the depth sensor directly transmitted the distance to an object in mm. The colour sensor had a horizontal field of view[4] of 62.0° and a vertical field[4] of view of 53.8°. The depth sensor operated between 0.8-4.0m[4,1], with depths outside these boundaries yielding a value of 0, and had vertical and horizontal fields of view of 58.5° and 48.6° respectively. Axes were defined to have colour information in the x (horizontal) and y (vertical) directions, with depth in z.

Data can be acquired from the kinect at a maximum of 30 frames per second[4], and was able to be captured at over 29 frames per second. This was key as the algorithm used to track data points benefited from minimising the displacement of each tracking point with each frame for obvious reasons. An example capture from the kinect is shown in Figure 2.

(a) Colour Sensor **(b)** Depth Sensor

Figure 2: Example figure to demonstrate colour (a) and depth (b) information captured by the kinect v1. The depth sensor is unable to distinguish the black dots in the paper, whilst the colour sensor is unable to distinguish the depth between the paper to the person. The front of the arm holding the paper is occluded as it is within 0.8m of the sensor. The windows, meanwhile, do not appear as the IR light passes through them without being reflected back.

2.2 Registration of Depth Image

Because the colour and depth sensors were offset from one another and had different fields of view, they needed to be co-registered[9] in order to determine the correct depth value for each pixel in the colour domain.

Mapping points of distinctive features in the colour domain, $X_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix}$, onto the respective points in the depth domain, $X'_n = \begin{pmatrix} x'_n \\ y'_n \end{pmatrix}$, was done by applying the affine transformation[9]

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \{1\}$$

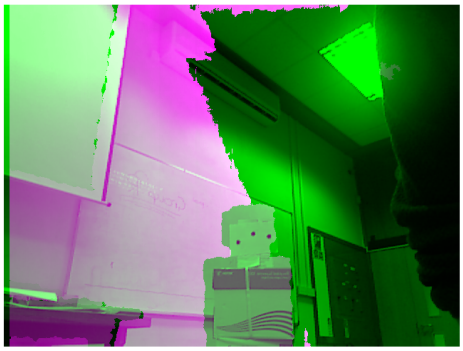
where a_2 and a_5 determine the translation and a_0, a_1, a_3 and a_4 specify the rotation, scale, stretch and shear transformations[9]. An example image with such distinctive points is shown in Figure 3.

(a) Colour Sensor

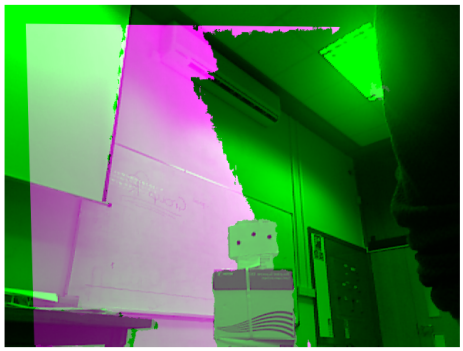
(b) Depth Sensor

Figure 3: Figure to demonstrate the colour (a) and depth (b) images of a stationary object used to register the two cameras together. The corners of the table, box and whiteboard were used as mapping points, as they are well defined on both shapes.

After applying the affine transformation, there was a far closer match between the colour and depth images. Figure 4a shows the mismatch of various objects between the sensors before registration, whilst Figure 4b shows the same images after registration.



(a) Before Registration



(b) After Registration

Figure 4: Figure showing a false colour depth image superimposed on a colour image to demonstrate the large mismatch of edges before registration (a), and subsequent improvement in (b) after registration.

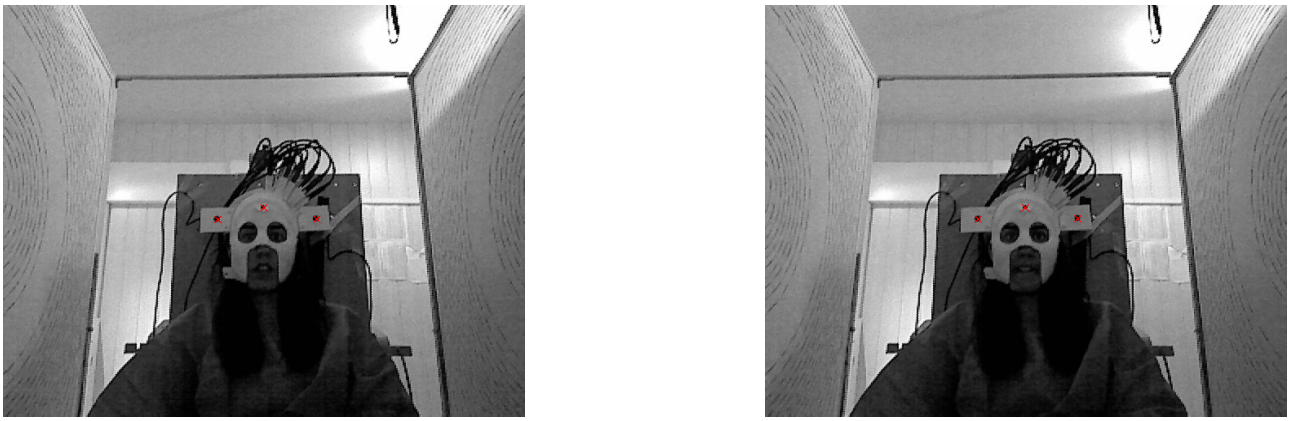
One limitation introduced during the registration process was the loss of useful depth information for part of the image. This is visible as the green bands in the edges of Figure 4b. This required us to limit the movement of each sample to the central section of the sensors. Furthermore, whilst distinguishing features were clearly visible in the colour image, noise made this difficult in the depth image. This resulted in imperfect registration. However, this calibration was sufficient for the purposes of this project.

2.3 Point Tracking

After registering the two cameras, a number, n, of circular black markers were placed on the object of interest to allow it to be tracked. Rough positions of the centroids of the markers in the x and y axes were determined manually for the first frame, and then algorithmically determined for all of the frames using information from the colour sensor.

First, n 15X15 pixel windows centred around the last known positions of the markers were created. A 4th degree Butterworth filter was then applied to smooth each window. From here, a binarising algorithm was applied at 25% of the average pixel brightness of each window. Anything above this threshold was defined as belonging to the markers. The centroids of the marker pixels were then calculated, and depth in z taken from an average of the points within the corresponding depth windows. The average depth position in mm was taken to help account for random noise and imperfect registration. These 3D centroid positions were stored and used to calculate the centroid positions of the next frame. An example of a tracked image is shown in Figure 5. A point was defined as lost if a centroid was not found or made a very large movement between frames, indicating that a random object was accidentally being tracked instead.

Finally, to convert the x and y information from the colour sensor from pixels to mm, the ratio of the field of view, α, to the horizontal/vertical resolution (640x480) was used to calculate the horizontal/vertical angles spanned per pixel. Trigonometry was then used to convert into Cartesian coordinates. At a typical operating distance of 1.5 meters, we found that 1 pixel in image space corresponded to about 3mm in real space both horizontally and vertically.



(a) Frame 1 **(b)** Frame 1000

Figure 5: Figure to demonstrate the ability of the tracking algorithm over 1000 frames (30 seconds). Detected tracking points are shown as red crosses. Although there is little movement here, if a tracking point was lost it would not show up in later frames. The program was therefore able to successfully track for at least 1000 frames.

2.4 Calculating Translations & Rotations

After the centroid positions of the n tracking points had been captured for all frames and converted to mm, the translations and rotations about the x, y, z axes were calculated. Scaling and shear transformations did not need to be calculated as the object being tracked was assumed to be rigid.

To calculate the potential 6 manipulations between pairs of two frames, Horn's quanternion based algorithm was used[10]. This allowed us to calculate the 3X3 rotation matrix, **R**, that mapped each of the 3Xn co-ordinate vectors from one frame to the next. One advantage of this algorithm is it can use more than three tracking points. This is important as tracking points were gradually lost over the course of a recording. It was also not affected by noise[11], and did not involve iterative calculations, so was computationally inexpensive[10]. Our code was able to take advantage of these features.

The first step in this algorithm involved assigning a "right side" and a "left side" to the calculations. The "left side" (referred to though the subscript l) is the first frame in a given pair of frames, and the "right side" the second (and referred with the subscript r). The mean centroids of the n tracking points in each frame, $\bar{\mathbf{r}}_l$ and $\bar{\mathbf{r}}_r$, were first calculated with the equations

$$\bar{\mathbf{r}}_l = \frac{1}{n}\sum_{i=1}^{n} \mathbf{r}_{l,i}, \qquad \{2\}$$

$$\bar{\mathbf{r}}_r = \frac{1}{n}\sum_{i=1}^{n} \mathbf{r}_{r,i}. \qquad \{3\}$$

Here, $\mathbf{r}_{l,i}$ referred to a set of co-ordinates that defined the centroid of the i^{th} tracking point of the left frame with the equation

$$\mathbf{r}_{l,i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \qquad \{4\}$$

and likewise with $\mathbf{r}_{r,i}$ for the right frame.

The co-ordinates of these mean centroids were then used to make the tracking points relative to each respective mean centroid with the equations

$$\mathbf{r'}_{l,i} = \mathbf{r}_{l,i} - \bar{\mathbf{r}}_l, \qquad \{5\}$$

$$\mathbf{r'}_{r,i} = \mathbf{r}_{r,i} - \bar{\mathbf{r}}_r, \qquad \{6\}$$

where $\mathbf{r'}_{l,i}$ denoted the adjusted position of a tracking point in the first frame that was originally at $\mathbf{r}_{l,i}$. $\mathbf{r'}_{r,i}$ was defined likewise.

From here, the rotation matrix, **R**, was calculated. Horn's algorithm did this by calculating the value of **R** that maximised the equation

$$\sum_{i=1}^{n} \mathbf{r'}_{r,i} \cdot \mathbf{R}(\mathbf{r'}_{l,i}). \qquad \{7\}$$

This was done by defining a 3X3 matrix, **M**, such that

$$\mathbf{M} = \sum_{i=1}^{n} \mathbf{r'}_{l,i} \mathbf{r'}^{T}_{r,i}, \quad \{8\}$$

$$\mathbf{M} = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}, \quad \{9\}$$

with T denoting the transpose of a matrix, and S the value of each element of $\mathbf{M}$ after its calculation in Equation 8.

Next, another symmetric matrix, $\mathbf{N}$, was calculated such that

$$\mathbf{N} = \begin{bmatrix} (S_{xx}+S_{yy}+S_{zz}) & (S_{yz}-S_{zy}) & (S_{zx}-S_{xz}) & (S_{xy}-S_{yx}) \\ (S_{yz}-Szy) & (S_{xx}-S_{yy}-S_{zz}) & (S_{xy}+S_{yx}) & (S_{zx}+S_{xz}) \\ (S_{zx}-Sxz) & (S_{xy}+S_{yx}) & (-S_{xx}+S_{yy}-S_{zz}) & (S_{yz}+S_{zy}) \\ (S_{xy}-Syx) & (S_{zx}+S_{xz}) & (S_{yz}+S_{zy}) & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix}. \quad \{10\}$$

The eigenvalues, λ, were then calculated by solving the equation

$$\left|\mathbf{N} - \lambda \mathbf{I}\right| = \mathbf{0}, \quad \{11\}$$

and taking the highest positive value of λ. Here, $\mathbf{I}$ represents a 4X4 identity matrix. The unit eigenvector that corresponded to λ equated to the 1X4 quaternion, $\mathbf{Q}$, that represented the manipulation of the helmet. $\mathbf{R}$ was then found by calculating the 4X4 matrix QQ^T, and taking the lower right 3X3 sub-matrix.

After calculating $\mathbf{R}$, the translations in x, y and z, $\Delta x, \Delta y$ and Δz, were calculated with the equation[12]

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \bar{\mathbf{r}}_r - (\mathbf{R}\bar{\mathbf{r}}_l). \quad \{12\}$$

The rotations in about each axis, $\theta x, \theta y$ and θz, meanwhile, were calculated with the equations[13]

$$\theta_x = atan2(\mathbf{R}_{3,2}, \mathbf{R}_{3,3}), \quad \{13\}$$

$$\theta_y = atan2(-\mathbf{R}_{3,1}, \sqrt{\mathbf{R}_{3,2}^2 + \mathbf{R}_{3,3}^2}, \quad \{14\}$$

$$\theta_z = atan2(\mathbf{R}_{2,1}, \mathbf{R}_{1,1}), \quad \{15\}$$

where a, b represent the a^{th} row and b^{th} column of $\mathbf{R}_{a,b}$. These values were then converted from radians to degrees.

After calculating $\Delta x, \Delta y, \Delta z, \theta_x, \theta_y$ and θ_z between the first and second frames, they were calculated for the second and third frames, third and fourth frames, and so on until all the frames had been analysed. Finally, the manipulations were cumulatively summed to calculate an overall manipulation as a function of frame number.

3 Validation of System

3.1 Field of View

To determine the accuracy of the sensor, the actual horizontal and vertical fields of view of the colour sensor, α_x and α_y respectively, were first calculated. Fields of view for the depth sensor were not tested as we did not use them; we instead automatically registered the depth image onto the colour image by using distinctive points. A test object consisting of a black piece of paper of width $L_x = 99 \pm 1$mm and height $L_y = 123 \pm 1$mm was attached to a heavy book and placed at various distances, r, away from the kinect, as measured with a tape measure. The height/width of the black paper was calculated in pixels, with an error of about 2-3 pixels due to parallax. The ratio of the height/width of the object in each image to the total height/width of the sensor was determined, and defined as s_x and s_y respectively. Using the small angle approximation, it was trivial to show that the ratio of the arc length across the field of view to the arc length of the object was equal to s. This defined the equation

$$s = \frac{L}{\alpha r} \tag{16}$$

A plot of s_x and s_y against $1/r$ was then plotted, and the gradient of a linear fit to the data, m_x and m_y, calculated, along with an associated error from the fit. This was used to calculate each value of α in radians with the equations

$$\alpha_x = \frac{L_y}{m_x}, \tag{17}$$

$$\alpha_y = \frac{L_x}{m_y}. \tag{18}$$

As shown in Figure 6, α_y of the colour sensor was found to be $52.2 \pm 0.6°$, and α_x to be $70.2 \pm 1.0°$. These values are significantly different to the official values for the kinect. However it is likely that this discrepancy is related to the sensor as our data closely matched the line of best fit, even when including errors. Because the lenses used in the kinect were plastic and not new, lens artefacts may have been introduced by multiple heating and cooling cycles, resulting in warping.

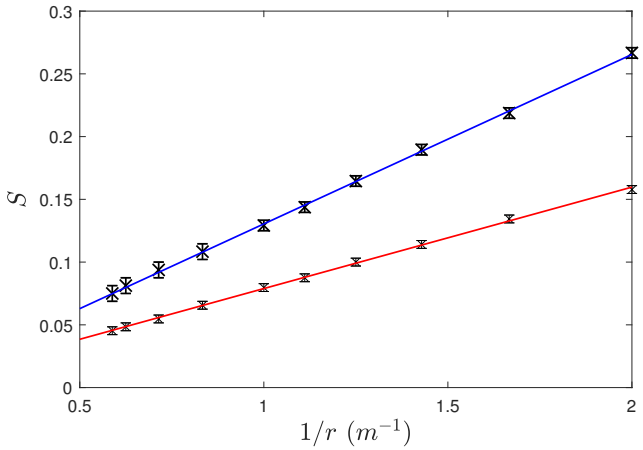


Figure 6: Figure to demonstrate the graph used to calculated the horizontal (blue) and vertical (red) fields of view. It was found that $\alpha_x = 70.2 \pm 1.0°$ and $\alpha_y = 52.2 \pm 0.6°$.

3.2 Horn's Algorithm

Following this, the accuracy of calculating rotations and translations was tested. This was done by taking the tracking coordinates from one frame of example data, and applying randomly chosen manipulations to generate new points in the second frame. **R** was calculated between the two frames, and Equations 12-15 used to reproduce the applied manipulations. Although this method ignored random noise, it had been shown elsewhere that noise has little effect [11].

The results for rotations and translations about a single axis are shown in Tables A.2 and A.1. From here, it is clear to see that the implementation of Horn's algorithm was correct for all 6 degrees of freedom. For these simple cases, the calculations were correct to floating point precision.

The ability of the algorithm to correctly determine simultaneous manipulations in all 6 degrees of freedom was then investigated. The percentage error for each manipulation was defined by taking the calculated value of each manipulation as a function of the applied value of each manipulation, taking this away from 100 and then taking the modulus. The errors were then averaged for each of the 6 manipulations, and converted to accuracy by taking the error away from 100. These results are shown in Table A.3.

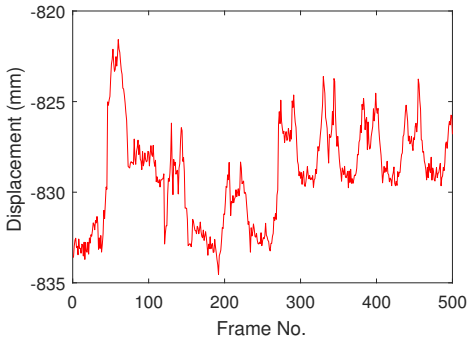
When both random rotations and translations were applied, accuracy did not decrease in comparison to only rotations. Translations of beyond 100mm were still calculated correctly to floating point accuracy as before. Because it would be difficult to move over 100mm in 1/30th of a second, the algorithm can be considered to be robust with respect to translation. There is therefore no issue with allowing a subject to walk about a room, provided the tracking algorithm can keep up. We therefore say that the error when calculating translations is negligible.

However, the accuracy is extremely poor for rotations of greater than 10° about all axes, with accuracies of 40-60%. This error was due to the algorithm, as it did not find an accurate value of **R** in these cases. However, for rotations of under 10° the error was far smaller, with accuracies of 90-98%. In these cases, rotations could be approximately calculated to within $\pm 0.5^\circ$. It was therefore important to keep rotations between frames small, which would require increasing the number of frames per second captured. The kinect was therefore a limiting factor.

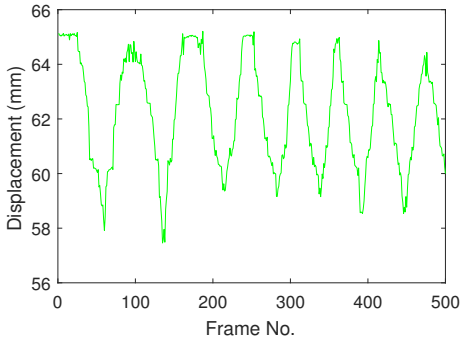
3.3 Rotation

After confirming that Horn's algorithm was able to calculate translations to an extremely high precision, the ability of the system to determine rotations in reality was investigated. Three tracking markers were placed on the same book used in Section 3.1, before moving the book whilst tracking its motion. The translations and rotations determined by the algorithms outlined in Section 2, with our calculated values for α.

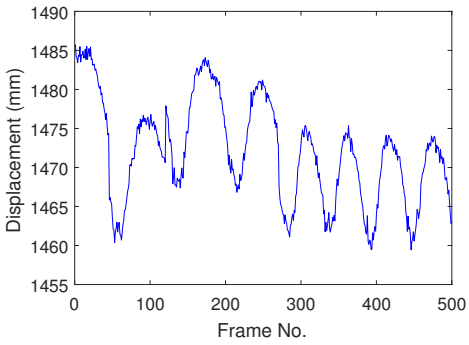
Without equipment enabling rotation about a point equidistant to all the markers, rotations were difficulty to accurately apply about specific axes. We could therefore only test for composite motions involving some unwanted translations and rotations about all axes.
Simulating a nodding motion, the three markers were tracked and their individual and collective motions analysed. Inspecting the tracked motion of one marker yielded the results seen in Figure 7. Given that the nodding motion was repeated 7 times, the number of cycles observed in the oscillatory motion of the tracking point matched our expectations.



(a) Motion of one tracking point in x.



(b) Motion of one tracking point in y.



(c) Motion of one tracking point in z.

Figure 7: Figure showing the motion of one tracking point in each dimension during the simulation of a nodding motion

The values for rotation of the whole body as given by Horn's algorithm were less easily analysed. Figure 8 shows the values calculated for the motion in each degree of freedom using the tracking data shown in Figure 7.

Although the number of cycles in the rotation about x and z as well as the translation in y matched the cycles seen in Figure 7, the translational amplitudes yielded clearly incorrect values of over 0.5m. However, we previously demonstrated that Horn's algorithm can calculate translations to floating point precision. The discrepancy observed could therefore be attributed to the fact that the book was not rigid. A change in the morphology of the object would correspond to a combination of scaling or shearing, which Horn's algorithm does not account for, thus yielding nonsensical results. Future research therefore requires working with a perfectly rigid object.

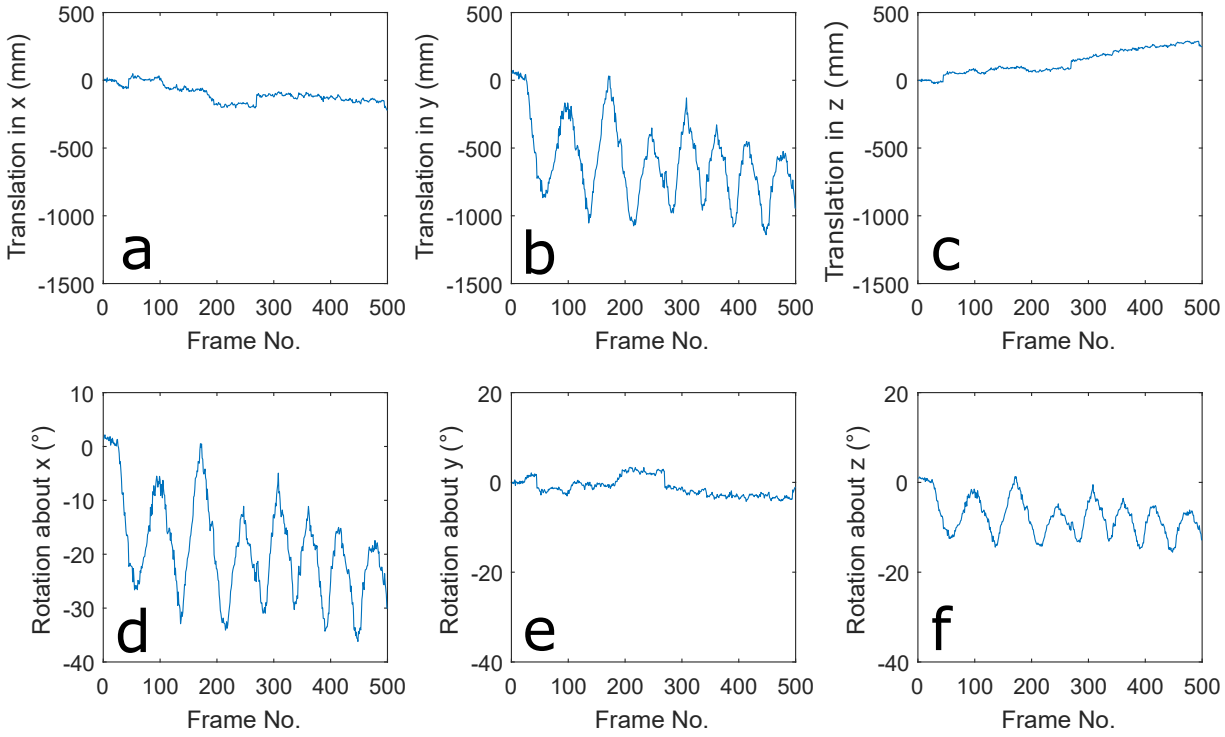


Figure 8: Figure showing the motion of our test object in each degree of freedom. The motion of the object simulated nodding, with the motion being repeated 7 times. This is demonstrated via the 7 oscillations seen in (b), (d), and (f).

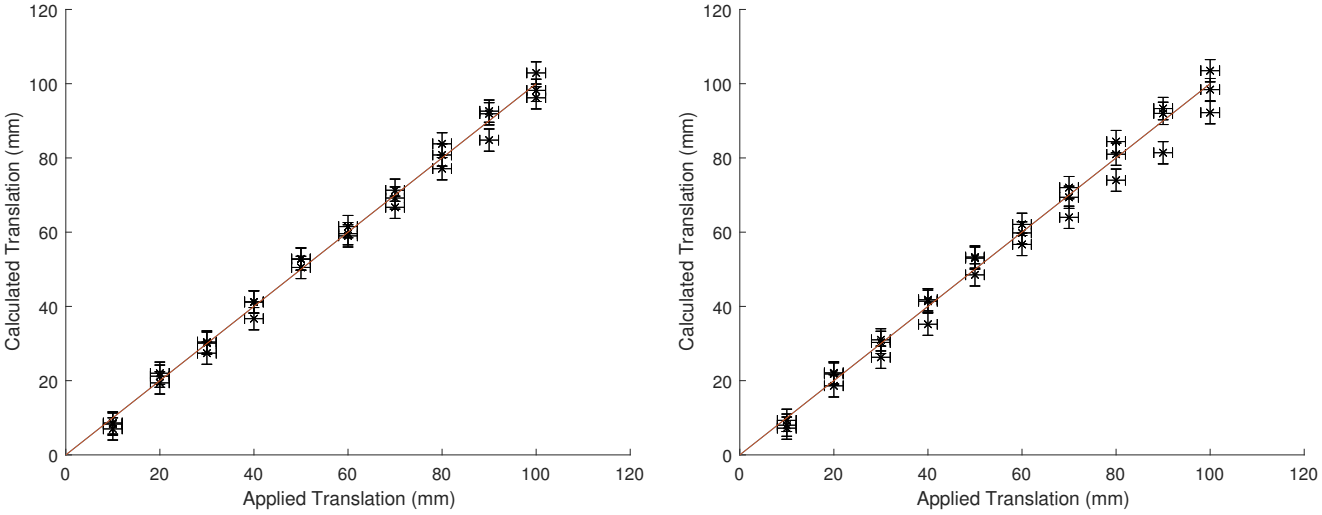
3.4 Translation

Finally, the ability of the system as a whole to calculate translations in real world scenarios was investigated. This was achieved using the same method as in Section 3.3. The tracking of one marker was sufficient for this.

The book was translated by known distances in the x-z plane, and all translations repeated three times. Translations in the vertical y direction were not applied, due to time constraints, but were still measured. Simple trigonometry was used to calculate the translation of the object.

The accuracy of the entire system, A, was calculated by taking the standard deviation of the differences between the applied and calculated translations, and adding in quadrature to the errors on the calculated transformations due to the kinect, δ_{kinect}. The error on this accuracy, δ_{app}, was equal to the error in our ability to move the book. The calculations of these errors are discussed below in Section 4.

For applied displacements in multiple directions, the calculated results are graphed in Figure 9 both for the values of α calculated in Section 3.1, and for those from the official documentation.



(a) Calculated Values of α **(b)** Official Values of α

Figure 9: Figure to demonstrate the accuracy of the system to detect lengths when using (a) our manually calculated values of field of view, and (b) official values of field of view given in the documentation. It was found that the accuracy of the system in each case was (a) $A = 4 \pm 2mm$ and (b) $A = 5 \pm 2mm$.

As shown in Figure 9a, when using our values for α we found that $A = 4 \pm 2mm$. In contrast, as seen in Figure 9b the inaccuracy was greater when using the officially provided values of α, with $A = 5 \pm 2mm$. As expected from Equation 16, calculated values became much further away from the expected values at greater distances for the official values as opposed to ours, with a wider spread of results in comparison to 9a. This provided further evidence that the field of view calculations performed in Section 3.1 were correct.

4 Major Sources of Error

One major source of error in the z direction involved heat, as both the kinect and the tracking mask heated up over time. Because the depth sensor worked using IR, warmer items were detected as being closer to the sensor in z. This was investigated by placing a hot mug of water at a fixed distance from the sensor and the mean recorded distance to all corresponding pixels measured as a function of time. This is shown in Figure 10. Errorbars were from the precision of the kinect's depth sensor at $\pm 1mm$. The result is shown in Figure 11.

Figure 10: Figure to demonstrate the error due to heat in the depth sensor output by calculating the mean distance to a warm mug of water over timer.

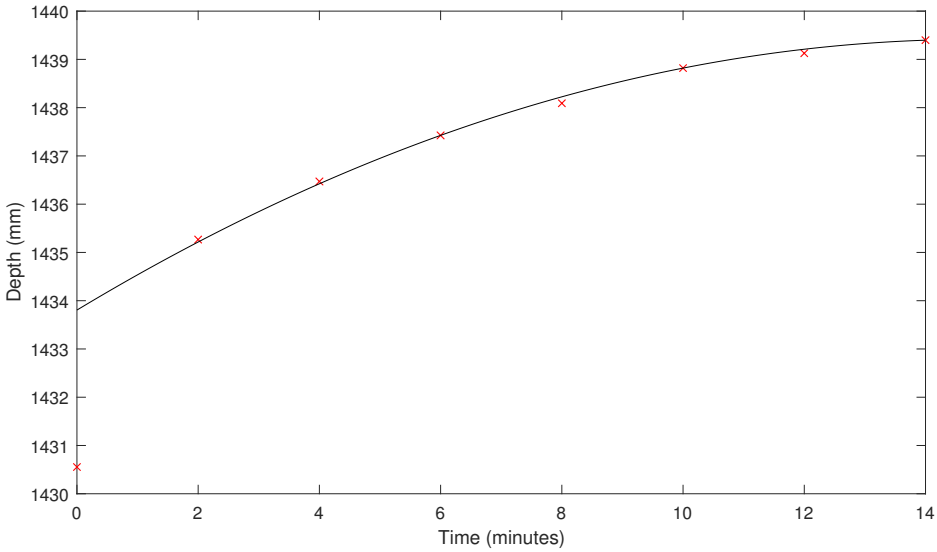


Figure 11: Figure to demonstrate the depth sensor output when calculating the mean distance to a warm mug of water. This setup was left fixed in place, and over the course of 15 minutes the depth of the stationary object increased by 5mm

It was found that after 14 minutes the mean distance of the cooling water increased from 1434mm to 1439mm. Although repeats were limited due to time constraints, a 14mm increase from 1913mm to 1927mm was also seen. There was too much variance between the two repeats to accurately calculate this error. However, it should be noted that the mask will heat up far less than a cup of boiling water will cool. This heating error should be more thoroughly investigated in the future by using a thermometer and plotting the change in depth over time, and taking multiple repeats over a variety of distances, as the IR sensor may be more sensitive in certain locations. In future, depth should be calibrated based on object temperature.

Furthermore, the depth sensor was also found to be highly susceptible to noise at ambient temperatures. Long after the mug had cooled to room temperature, it was left static for 10 minutes. It can be seen in Figure 12 that the depth of this spatially and thermally static object fluctuated by a standard deviation of $\pm 1mm$. When including the effects of heating, there was clearly a significant error in depth of approximately $2mm$.

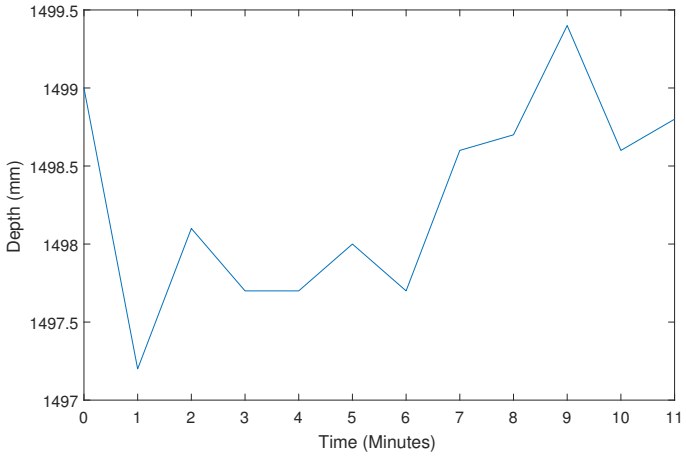


Figure 12: Figure to demonstrate the large effect of noise in measurements made by the depth sensor of the kinect. Here, a thermally and spatially static object was calculated as moving to a standard deviation of approximately 1mm.

Other sources of error included lens distortions, which may explain the majority of the errors when calculating the positions of objects the x and y axes. Most errors in calculations were likely due to the kinect, as Horn's algorithm was shown to correctly calculate translations to floating point accuracy, and small rotations to over 90% accuracy.

There were multiple major sources of error when determining the accuracy of the system in Section 3.4. First, there was the human error when applying a known translation to the book in Section 3.4, estimated as $\pm 1mm$ for all axes. The errors for all axes were added in quadrature to give an overall applied translation error, δ_{app}, of $\pm 2mm$. Next, there were three errors making up δ_{kinect}. First, there were errors in our measurements of horizontal/vertical field of view in Section 3.1 that were propagated through Equation 16. There were errors in depth measurement due to the $\pm 1mm$ precision of the kinect and the $\pm 2mm$ due to noise and heat. These errors were then added in quadrature and propagated through the same calculations used to calculate applied translations, giving $\delta_{kinect} = \pm 3mm$.

5 Further Discussion

There were also many limitations with the algorithms used. For example, fast movements were often poorly tracked, as the tracking point moved out of the search window between frames, so was a flaw inherent to the algorithm. Without creating a new algorithm, this could be improved upon by increasing the framerate of the sensor. However, this would require replacing the kinect. Alternatively, motion between frames could be interpolated with an algorithm such as the one created by Tan et. al.[5] for purposes similar to ours. This algorithm is also able to deal with multiple cameras and would allow us to eliminate the issue wherein an individual could obscure tracking points behind their head by over-rotating, causing tracking points to be lost.

It was also not possible to accurately determine if tracking points were lost. On some occasions, the algorithm would see a random dark area in a tracking window and believe it to be the tracking point, causing it to be miscalculated. This was of particular issue at larger depths, as the low resolution of the kinect made it difficult to resolve the tracking points. This was crudely accounted for by manually increasing the search window size in these cases, but at the expense of some performance. In future, this could be improved with more sophisticated point tracking algorithms or by searching the entire image for a more distinct tracking shape. Sensors with greater resolutions, such as the kinect sensor v2[2], could also be used.

Furthermore, the kinect v1 had a relatively low field of view. This was made even smaller due to the need to co-register the colour images with the smaller depth images. As the prototype MEG system could eventually be employed with individuals walking around a room, tracking points would quickly leave the field of vision of the sensors. The kinect had a vertical tilt motor from -27° to +27°, so could be placed on its side and tilted based on facial detection algorithms[14]. This would increase the vertical field of view from 54° to 62°, and the horizontal from 62° to (54+27)-(-54-27) ≈ 100°. This was not done here as it was difficult to simultaneously acquire data and move the motor.

It was also incorrect to make the assumption that the mask could directly correspond to the wearer's head. This was as the skin and/or mask may slip or warp slightly, which would introduce unexpected translations and shearing movements in the calculations. This was particularly true when a hardback book with tracking points taped on was used. In future, the mask should be tightly attached to the head, or trackers could be stuck directly to the face.

There were also numerous extensions that could be made. For example, the code could be extended to track and calculate manipulations in real time instead of capturing all frames and then performing all calculations. Because the algorithms discussed in Sections 2.3 & 2.4 were able to run in significantly under $1/30^{th}$ of a second per two frames, there would be no performance penalty. It would be relatively simple to add a two frame buffer and thus perform calculations in real time.

We also have an issue regarding file sizes. All data was stored to disk, including depth and colour information in all colour channels. This resulted in the production of 500MB files in approximately 15 seconds of recording. For the prototype sensor, recordings may last over an hour. With our system, data files of well over 20GB would be created. In future, only a single colour channel could be saved, or better yet compressed. Another alternative would be to implement object detection[15] around the face mask, and only data around the face stored.

Finally and most importantly, our testing methodology was extremely brutish. The manually displaced and rotated book was by no means a perfectly rigid body, was not perfectly square, and had taped on tracking points that wafted slightly with movement. This significantly impacted our ability to detect rotations, and introduced incalculable errors into A. In future, a far more rigid object is required. It would also be of significant use to mechanically rotate the object about the centroids of the tracking points, as this would avoid inaccurate manual manipulations.

6 Conclusion

To conclude, we we were successfully able to capture data from a kinect, co-register its two sensors, robustly track points in three dimensions over time and thus determine the manipulations applied to an object. We were able to calculate the fields of view of the sensor at $\alpha_x = 70.2 \pm 1.0^\circ$ and $\alpha_y = 52.2 \pm 0.6^\circ$. We were also able to demonstrate the ability of Horn's algorithm to determine translations of rigid bodies to floating point accuracy, and rotations of under 10° to under 10% error. This enabled us to track an object to an accuracy of $A = 4 \pm 2mm$.

However, the usefulness of our results are limited by our testing methodology. Our decision to stick tracking points on a book was very crude and could easily have introduced extra displacements and rotations of far more than the $\pm 1mm$ quoted due to compression of the book. This lack of rigidity particularly affected our ability to calculate rotations. To obtain a more rigorous estimation of system accuracy, a better testing method is necessary.

Our ability to determine the overall accuracy of the system was also limited by several sources of error. Alongside the human error when manipulating test objects, the depth sensor was affected by $\pm 1mm$ by noise, and up to around 5mm by heating effects. The depth sensor was also not perfectly co-registered with the colour sensor, owing to their different fields of view and generic registration algorithm, adding further incalculable errors. Furthermore, Horn's algorithm could also only achieve sub degree accuracy when dealing with rotations of under 5°, with errors increasing with rotation size.

In ideal lighting conditions in which an object was placed close to the centre of the sensor and barely moved, it was shown that near millimetre accuracy can be obtained. However, when considering the above limitations, it can be said with more confidence that the system achieved sub centimetre accuracy.

Ultimately, however, the kinect was created seven years ago as a gaming accessory for people sitting on a sofa. Whilst it is cheap, it is completely unoptimised for environments where consistent millimetre precision to a high accuracy is critical. Changing to a more advanced sensor system (especially depth) is vital for further development.

7 References

[1] Philip J Noonan, Jon Howard, Tim F Cootes, William A Hallett, and Rainer Hinz. Realtime markerless rigid body head motion tracking using the microsoft kinect. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pages 2241–2246. IEEE, 2012. doi:10.1109/nssmic.2012.6551510.

[2] PJ Noonan, J Howard, WA Hallett, and RN Gunn. Repurposing the microsoft kinect for windows v2 for external head motion tracking for brain pet. *Physics in medicine and biology*, 60(22): 8753, 2015. doi:10.1088/0031-9155/60/22/8753.

[3] Georgios Mastorakis and Dimitrios Makris. Fall detection system using kinects infrared sensor. *Journal of Real-Time Image Processing*, 9(4):635–646, 2014. doi:10.1007/s11554-012-0246-9.

[4] Microsoft Developer Network. Kinect sensor. `https://msdn.microsoft.com/en-gb/library/hh438998.aspx`, .

[5] David Joseph Tan, Federico Tombari, and Nassir Navab. Real-time accurate 3d head tracking and pose estimation with consumer rgb-d cameras. *International Journal of Computer Vision*, pages 1–26, 2017. doi:10.1007/s11263-017-0988-8.

[6] Elena Boto, Sofie S Meyer, Vishal Shah, Orang Alem, Svenja Knappe, Peter Kruger, T Mark Fromhold, Mark Lim, Paul M Glover, Peter G Morris, et al. A new generation of magnetoencephalography: Room temperature measurements using optically-pumped magnetometers. *Neuroimage*, 149:404–414, 2017.

[7] Microsoft Developer Network. Kinect for windows sdk. `https://msdn.microsoft.com/en-us/library/hh855347.aspx`, .

[8] Mathworks. Microsoft kinect for windows support from image acquisition toolbox. `https://uk.mathworks.com/hardware-support/kinect-windows.html`.

[9] Mosab Bazargani, Antnio dos Anjos, Fernando G. Lobo, Ali Mollahosseini, and Hamid Reza Shahbazkia. Affine image registration transformation estimation using a real coded genetic algorithm with sbx. *Arxiv*, 2012. URL `https://arxiv.org/abs/1204.2139`.

[10] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987. doi:10.1364/josaa.4.000629.

[11] David W Eggert, Adele Lorusso, and Robert B Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5-6):272–290, 1997. doi:10.1007/s001380050048.

[12] Gregory G Slabaugh. Computing euler angles from a rotation matrix. ., 6(2000):39–63, 1999. doi:10.1002/0470845767.ch15.

[13] Herbert Goldstein, Charles Poole, and John Safko. Classical mechanics, 2002.

[14] Mayank Agarwal, Nikunj Jain, Mr Manish Kumar, and Himanshu Agrawal. Face recognition using eigen faces and artificial neural network. *International Journal of Computer Theory and Engineering*, 2(4):624, 2010. doi:10.7763/ijcte.2010.v2.213.

[15] Jong-Min Jeong, Tae-Sung Yoon, and Jin-Bae Park. Kalman filter based multiple objects detection-tracking algorithm robust to occlusion. In *SICE Annual Conference (SICE), 2014 Proceedings of the*, pages 941–946. IEEE, 2014. doi:10.1109/sice.2014.6935235.

Author Contributions

Lukas Rier: Responsible for registration code, point tracking code and translation and rotation tests, and subsequent writing.

Oliver Gordon: Responsible for setting up of kinect, data acquisition code, coding and testing of Horn's algorithm, cover image, and most leftover writing.

Both authors were responsible for the abstract, generating figures and captions, proofreading, and suggesting points of discussion and errors.

A Tables

A.1 Single Axis Rotations

Table 1: Table to demonstrate correct programming of Horn's quaternion based algorithm for calculating rotations of an object. Rotations about a single axis were manually applied to a vector, and Horn's algorithm was able to detect these rotations to floating point precision.

Applied Manipulations						**Calculated Manipulations**					
Rotations (°)			Translations (mm)			Rotations (°)			Translations (mm)		
x	y	z	x	y	z	x	y	z	x	y	z
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10.00	0.00	0.00	0.00	0.00	0.00	**10.00**	0.00	0.00	0.00	0.00	0.00
25.00	0.00	0.00	0.00	0.00	0.00	**25.00**	0.00	0.00	0.00	0.00	0.00
40.00	0.00	0.00	0.00	0.00	0.00	**40.00**	0.00	0.00	0.00	0.00	0.00
0.00	**10.00**	0.00	0.00	0.00	0.00	0.00	**10.00**	0.00	0.00	0.00	0.00
0.00	**25.00**	0.00	0.00	0.00	0.00	0.00	**25.00**	0.00	0.00	0.00	0.00
0.00	**40.00**	0.00	0.00	0.00	0.00	0.00	**40.00**	0.00	0.00	0.00	0.00
0.00	0.00	**10.00**	0.00	0.00	0.00	0.00	0.00	**10.00**	0.00	0.00	0.00
0.00	0.00	**25.00**	0.00	0.00	0.00	0.00	0.00	**25.00**	0.00	0.00	0.00
0.00	0.00	**40.00**	0.00	0.00	0.00	0.00	0.00	**40.00**	0.00	0.00	0.00

A.2 Single Axis Translations

Table 2: Table to demonstrate correct programming of Horn's quaternion based algorithm for calculating translations of an object. Translations about a single axis were manually applied to a vector, and Horn's algorithm was able to detect these to floating point precision.

Applied Manipulations						**Calculated Manipulations**					
Rotations (°)			Translations (mm)			Rotations (°)			Translations (mm)		
x	y	z	x	y	z	x	y	z	x	y	z
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	**25.00**	0.00	0.00	0.00	0.00	0.00	**25.00**	0.00	0.00
0.00	0.00	0.00	**50.00**	0.00	0.00	0.00	0.00	0.00	**50.00**	0.00	0.00
0.00	0.00	0.00	**75.00**	0.00	0.00	0.00	0.00	0.00	**75.00**	0.00	0.00
0.00	0.00	0.00	0.00	**25.00**	0.00	0.00	0.00	0.00	0.00	**25.00**	0.00
0.00	0.00	0.00	0.00	**50.00**	0.00	0.00	0.00	0.00	0.00	**50.00**	0.00
0.00	0.00	0.00	0.00	**75.00**	0.00	0.00	0.00	0.00	0.00	**75.00**	0.00
0.00	0.00	0.00	0.00	0.00	**25.00**	0.00	0.00	0.00	0.00	0.00	**25.00**
0.00	0.00	0.00	0.00	0.00	**50.00**	0.00	0.00	0.00	0.00	0.00	**50.00**
0.00	0.00	0.00	0.00	0.00	**75.00**	0.00	0.00	0.00	0.00	0.00	**75.00**

A.3 Simultaneous Manipulations

Table 3: Table to demonstrate success of Horn's algorithm to calculate simultaneous manipulations.

Applied Manipulations						**Calculated Manipulations**						
Rotations (°)			Translations (mm)			Rotations (°)			Translations (mm)			
x	y	z	x	y	z	x	y	z	x	y	z	**Accuracy (%)**
20.00	**20.00**	**20.00**	0.00	0.00	0.00	**26.03**	**10.66**	**26.03**	0.00	0.00	0.00	**64.33**
30.00	**30.00**	**30.00**	0.00	0.00	0.00	**40.89**	**7.18**	**40.89**	0.00	0.00	0.00	**50.44**
40.00	**40.00**	**40.00**	0.00	0.00	0.00	**54.04**	**-2.06**	**54.04**	0.00	0.00	0.00	**41.55**
20.00	**20.00**	**10.00**	0.00	0.00	0.00	**23.97**	**14.90**	**16.74**	0.00	0.00	0.00	**62.40**
40.00	**20.00**	**10.00**	0.00	0.00	0.00	**43.31**	**8.42**	**20.69**	0.00	0.00	0.00	**42.30**
3.00	**2.00**	**1.00**	**25.00**	**50.00**	**10.00**	**3.04**	**1.94**	**1.10**	**25.00**	**50.00**	**10.00**	**95.21**
2.00	**4.00**	**6.00**	**50.00**	**45.00**	**25.00**	**2.41**	**3.77**	**6.15**	**50.00**	**45.00**	**25.00**	**90.35**
3.00	**1.00**	**2.00**	**90.00**	**30.00**	**40.00**	**3.03**	**0.89**	**2.05**	**90.00**	**30.00**	**40.00**	**95.24**
0.50	**1.00**	**0.20**	**30.00**	**80.00**	**60.00**	**0.50**	**1.00**	**0.21**	**30.00**	**80.00**	**60.00**	**98.24**
1.00	**0.30**	**2.00**	**40.00**	**30.00**	**85.00**	**1.01**	**0.26**	**2.00**	**40.00**	**30.00**	**85.00**	**95.69**
20.00	**20.00**	**20.00**	**25.00**	**50.00**	**10.00**	**26.03**	**10.66**	**26.03**	**25.00**	**50.00**	**10.00**	**64.33**
30.00	**30.00**	**30.00**	**50.00**	**45.00**	**25.00**	**40.89**	**7.18**	**40.89**	**50.00**	**45.00**	**25.00**	**50.44**
40.00	**40.00**	**40.00**	**90.00**	**30.00**	**40.00**	**54.04**	**-2.06**	**54.04**	**90.00**	**30.00**	**40.00**	**41.55**
20.00	**20.00**	**10.00**	**30.00**	**80.00**	**60.00**	**23.97**	**14.90**	**16.74**	**30.00**	**80.00**	**60.00**	**62.40**
40.00	**20.00**	**10.00**	**40.00**	**30.00**	**85.00**	**43.31**	**8.42**	**20.69**	**40.00**	**30.00**	**85.00**	**42.30**