TECHNICAL SPECIFICATION

SKETCHUP BLOCKS – ANDREW SMITH - CSIR

Version 1.7	11081092 – ET, van Zyl
13/10/2013	11211548 – WH, Oldewage
	11183153 – JL, Coetzee

CONTENTS

CHANGE LOG

Version 0.1	14/09/2013	Document created.
Version 1.0	15/09/2013	First published
Version 1.1	12/10/2013	Replaced formulas with text
Version 1.5	13/10/2013	Added Pseudo Physics section
Version 1.5	13/10/2013	Added lambda calculation details
Version 1.6	13/10/2013	Added screenshots
Version 1.7	13/10/2013	Revised cover page

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to give an overview of the mathematical models and methods that are made use of in the Sketchup Blocks project. This document aims to provide clear motivation for the chosen mathematical methods as well as an explanation of how and why they work.

1.2 PROJECT SCOPE

Sketchup Blocks aims to provide the client with a system that will convert physical 3D models into digital form. The system will include software and hardware components.

Hardware components:

- The physical blocks with fiducial markers attached (called ‘smart blocks’).
- The construction floor on which the smart block constructions are built.
- Approximately five web cameras.

The scope of the software components will broaden from release to release. The client only requires the first release to consider the project successful. The releases following the first release will address additional concerns and implement extra features mentioned by the client.

Release 1:

The initial system will be able to recognize a small set of geometrically simple objects without any time constraints. After the blocks have been transformed into a digital model, the vertices of the model will be output to an object file for import into Google Sketchup.

Release 2:

This release aims to limit the time taken to recognize a fiducial and recreate the model it represents. The amount of constructed and recognizable blocks will also be increased. The setup of the system will be simplified to improve mobility and allow for fast setup. The removal of blocks will be detected without extra input from the user.

Release 3

This release aims to add a viewing window that will display the model as it being created thereby providing visualization before exporting to a model. The detection of the fiducials should be possible in suboptimal lighting conditions.

Optional release

Custom blocks may be imported to the smart block database. Users can view the construction of other users over the network.

1.3 RELATED DOCUMENTS

Project Vision and Scope (Sketchup Blocks project, by CICO).

Project Design Specifications (Sketchup Blocks project, by CICO).

Project Architectural Requirements (Sketchup Blocks project, by CICO).

Project Management Document (Sketchup Blocks project, by CICO).

Testing Document (Sketchup Blocks project, by CICO).

Sketchup Blocks Developer Blog (http://sketchupblocks.wordpress.com/).

2. SYSTEM DESCRIPTION

This project aims to create a system that can convert physical 3D models into digital models that can be viewed and edited in Google Sketchup. The physical model will be built with blocks that have fiducial markers attached. The locations of these blocks will be picked up on web cameras by ReacTIVision software and the resulting data will be used to create digital 3D models that can be viewed and exported to Google Sketchup.

The project subscribes to the concept of the Internet of Things. The Internet of Things is centred around the idea of ‘smart objects’ which are physical objects and devices that are given a virtual representation and linked to information networks. These smart objects will ideally be able to capture, react to and communicate about data captured from their physical environment autonomously.

3. DESIGNS

3.1 SYSTEM INTRODUCTION

The Sketchup Blocks system is centred about the Model, which stores the digital representation of the physical model. Particularly, it stores the positions and vertices of all the blocks on the construction floor.

The construction of this Model is very nearly a pipeline process, in which many different classes are involved in processing the camera input to cause a change in the Model's structure. This process will be described in detail below.

The lowest level of work is done by the Interpreter class. The Interpreter class receives data from ReacTIVision regarding the location and orientation of the fiducial markers recognized by the camera. It then passes this information along the pipeline for other classes to use.

The Model Constructor uses the data it receives to infer the positions of the physical building blocks in 3D space. The cameras only provide 2D information and the x,y co-ordinates that they provide are co-ordinates in camera screen space, and not a co-ordinate system that can be used without futher processing.

Consequently, multiple Camera Events from at least two different cameras are required to infer the position of a particular smart block in the model. This position data, along with the data about the block's shape, size, etc is wrapped in a Model Block.

The exact calulations hinge data produced by camera calibration. Consequently, the process of Camera Calibration will be described first.

3.2 CAMERA CALIBRATION

Camera Calibration is the process through the position of each camera is determined in 3D space.

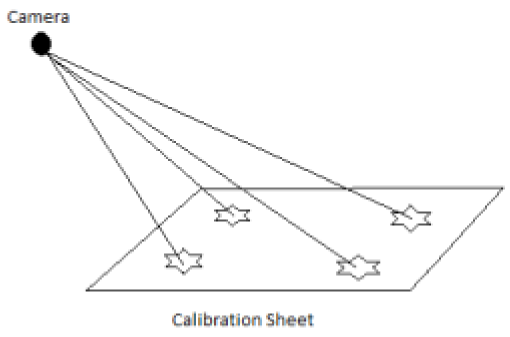
The cameras are not within one another's field of view nor are they able to recognize one another. Consequently data from multiple cameras cannot be used to calculate camera position (in the manner that block positions are calculated). Instead, a camera's position must be calculated by applying triangulation techniques to a single image provided by the camera itself. Particularly, the calibration process is done through a combination of Geometric Sphere Intersection methods and Newton's method for solving nonlinear equations.

ReacTIVision provides the positions of any fiducials that it picks up. These positions are specified as (x, y) co-ordinates in the image of a single camera frame ('pixel space'). The co-ordinates work similarly to that of a screen, in that (0, 0) are at the top, leftmost corner and (1, 1) in the bottom, rightmost corner of the image.

This data can be interpreted to provide the horizontal angles between two recognized fiducials by scaling the difference between their x co-ordinates with the camera's field of view. The vertical angle between two fiducials can be calculated by scaling the difference between their y co-ordinates with the camera's aspect ratio and field of view.

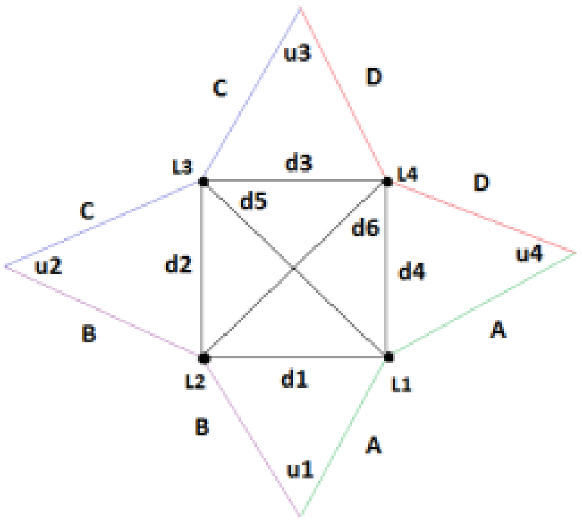
The process of camera calibration makes use of a 'calibration floor' which is simply a sheet on which four fiducial markers or 'landmarks' have been printed. The distances between the fiducial markers on the calibration sheet are known.

The mathematical model uses the pin hole camera model as a basis for reasoning. When viewing the four landmarks through a camera there are four imaginary lines, each originating from a landmark and crossing at one point. This point is the pin hole of the camera.



Pinhole Camera Model (Fig. 3.2.1)

This forms a rectangular pyramid of which the camera is the apex. Consider a 'flattened' view of this pyramid (see diagram below). Let the green and purple lines be labelled as A and B. Let the angle between them (measured from the camera) be μ_1. The four corners of the square below are the positions of the fiducials markers. Let the line opposite μ_1 be d_1. d_1 is the real-world distance between Landmark 1 and Landmark 2. As mentioned previously, the point of intersection of A, B, C and D is the camera's position (see diagram above). The landmarks are labelled from L_1 to L_4.



Flattened Pyramid
(Fig. 3.2.2)

The law of cosines produces equations that contain the sides of the pyramid as unknowns. If this is done for all the angles and lines, the following system of equations is found:

$$\begin{aligned} d_1^2 &= A^2 + B^2 - A * B * \cos(\mu_1) \\ d_2^2 &= B^2 + C^2 - B * C * \cos(\mu_2) \\ d_3^2 &= C^2 + D^2 - C * D * \cos(\mu_3) \\ d_4^2 &= D^2 + A^2 - D * A * \cos(\mu_4) \\ d_5^2 &= A^2 + C^2 - A * C * \cos(\mu_5) \\ d_6^2 &= B^2 + D^2 - B * D * \cos(\mu_6) \end{aligned} \quad \textit{(3.2.1)}$$

This provides six equations, one for each line between the four landmarks. The last two equations are found by considering the angle between fiducials that are on opposite corners of the calibration page. The angle between line A and line C has been named μ_5 and the angle between line B and line D has been named μ_6 .
The lengths d_1 to d_6 are the real world lengths of the lines between the landmarks and are known.
The angles, as previously mentioned, are those as measured by the camera. These are calculated as follows:

$$\mu_1 = \sqrt{\left(L_{1_x} - L_{2_x}\right)^2} * fov_h + \sqrt{\left(L_{1_y} - L_{2_y}\right)^2} * \frac{fov_v}{aspectRatio} \quad \textit{(3.2.2)}$$

Where fov_h and $fov_v$are the camera's horizontal and vertical fields of view.

Clearly, this systems of equations is non-linear. It is solved by means of Newton's Method for solving systems of non-linear equations. This method can approximate the solution to a nonlinear system, given some initial approximation. In order to solve for A, B, C and D, the system of equations is first manipulated to that each equation is equal to zero.

$$\begin{gathered}
0 = A^2 + B^2 - A * B * \cos(\mu_1) - d_1^2 \\
0 = B^2 + C^2 - B * C * \cos(\mu_2) - d_2^2 \\
0 = C^2 + D^2 - C * D * \cos(\mu_3) - d_3^2 \\
0 = D^2 + A^2 - D * A * \cos(\mu_4) - d_4^2 \\
0 = A^2 + C^2 - A * C * \cos(\mu_5) - d_5^2 \\
0 = B^2 + D^2 - B * D * \cos(\mu_6) - d_6^2
\end{gathered} \tag{3.2.3}$$

These equations then define the vector function f
where $f = (f_1, f_2, f_3 , f_4, f_5, f_6)^t$ and each component is defined as

$$\begin{gathered}
f_1 = A^2 + B^2 - A * B * \cos(\mu_1) - d_1^2 \\
f_2 = B^2 + C^2 - B * C * \cos(\mu_2) - d_2^2 \\
f_3 = C^2 + D^2 - C * D * \cos(\mu_3) - d_3^2 \\
f_4 = D^2 + A^2 - D * A * \cos(\mu_4) - d_4^2 \\
f_5 = A^2 + C^2 - A * C * \cos(\mu_5) - d_5^2 \\
f_6 = B^2 + D^2 - B * D * \cos(\mu_6) - d_6^2
\end{gathered} \tag{3.2.4}$$

The nonlinear system is then solved for

$$f = 0 \tag{3.2.5}$$

Solving the system described above provides the lengths of the lines between the landmarks and the camera.

By using these four lengths as radii for four spheres their point of intersection, which is the camera's position, can be found. The general form of these equations is shown below:

$$(L_x - C_x)^2 + \left(L_y - C_y\right)^2 + (L_z - C_z)^2 = A^2 \tag{3.2.6}$$

Where A is, as previously defined, the line between Landmark 1 and the camera. C_x , C_y and $C_z$are the (x,y,z) co-ordinates of the camera in 3D space. There is a similar equation for each line between a landmark and the camera. This system of equations is also solved by means of Newton's method to produce the positions of the cameras in 3D space.

Newton's method converges quadratically in most cases provided that the inverse of f's inverse exists. Unfortunately, it must be calculated with every iteration, but Newton's method converges fast for its applications in Sketchup Blocks and provides significantly more accurate results in a shorter time than the PSO that was also experimented with.

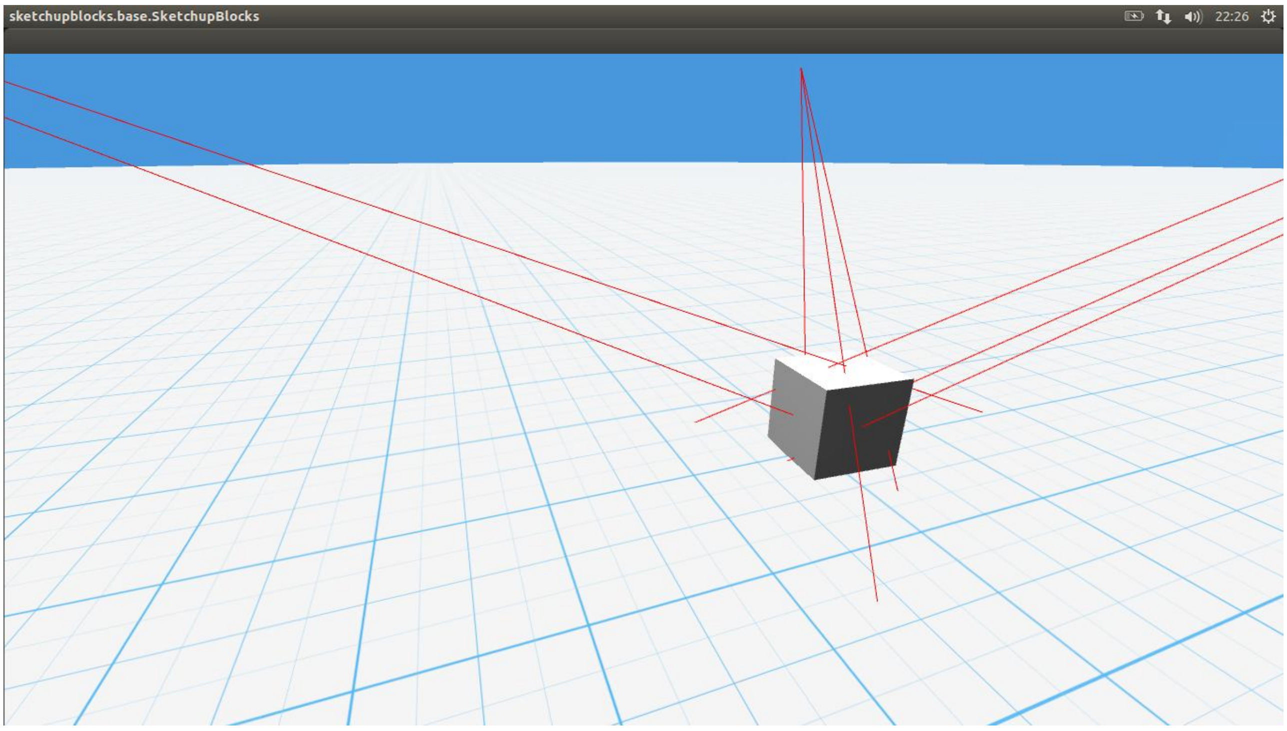
3.3 FIDUCIAL POSITION CALCULATION

The calculation of a fiducial marker's position in 3D space comes down to first finding a line between the fiducial's center and the camera. Let this line be L. Since the position of the landmarks used for calibration is known, a line can be defined between each landmark fiducial and the camera. Let these lines be V_1 to V_4.

The angle between the 'calibration lines' and L can be found by comparing their x and y co-ordinates in the camera's pixel space. In particular, the dot product between L and each calibration line yields a linear system of equations which can be solved for the direction between the camera and the fiducial.

$$\begin{aligned} V_1 \,.\, L &= \cos(\alpha_1) \\ V_2 \,.\, L &= \cos(\alpha_2) \\ V_3 \,.\, L &= \cos(\alpha_3) \\ V_4 \,.\, L &= \cos(\alpha_4) \end{aligned} \qquad (3.3.1)$$

This system is solved using Gauss's method with scaled partial pivoting.



Lines from cameras to fiducials (Fig.3.3.1)

Now the only question that remains is how far along in this direction this fiducial is. Since the Sketchup Blocks system requires that at least three fiducials be visible to determine a block's location, it is fair to suppose that three such fiducials are visible.

The standard 3D line equations below describe the lines between the observed fiducials and the camera that observes them. $v^1, v^2, v^3$are the direction vectors that were solved for in the previous step. c^1, c^2, c^3 are the positions of the cameras observing the fiducials (it is not necessary that they be different, but it does improve accuracy significantly). p^1, p^2, p^3 are the positions of the observed fiducials in 3D space. $\lambda^1, \lambda^2, \lambda^3$ are the unknowns line lengths.

$$p^1 = c^1 + \lambda^1 v^1$$
$$p^2 = c^2 + \lambda^2 v^2 \qquad (3.3.2)$$
$$p^3 = c^3 + \lambda^3 v^3$$

The distance between p^1 and p^2 is a physical property of the block and thus, is known. This provides the following equation:

$$\|d_{12}\| = \|p^1 - p^2\| \qquad (3.3.3)$$

Similarly, d_{13} and d_{23} can be used to obtain more equations containing $\lambda^1, \lambda^2, \lambda^3$. These equations are solved by means of Newton's method.However, Newton's method requires the calculation of the Jacobian matrix for the system of equations. This calculation is complicated by the varying number of fiducials that may be observed for a particular block at a given time (since it leads to a varying number of λ_i's to solve for).

Consequently, its calculation merits some explanation. First we consider how the function that represents the system of nonlinear equations is obtained.

The number of equations will vary with the number of inputs observed points.
Suppose n are observed. For any 2 points i and j

$$\|d_{ij}\| = \|p^i - p^j\| \qquad (3.3.4)$$

$$\|d_{ij}\|^2 = \|p^i - p^j\|^2 \qquad (3.3.5)$$

Where $p^i$and p^j is the ith and jth observed points (respectively) and d_{ij} is the distance between these two points. Each point can be expressed by means of a line equation in the following form:

$$p^i = c^i + \lambda^i v^i \qquad (3.3.6)$$

where c^i is the position of the camera observing the point, v^i is the direction between the camera and the point and λ^i is the distance between the camera and the point.

If we consider the right hand side of (3.3.5) we find

$$\left\|p^i - p^j\right\|^2 = \|(c^i + \lambda^i v^i) - (c^j + \lambda^j v^j)\|^2 = \left(\sqrt{\sum_{k=1}^{3}(c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j)^2}\right)^2$$

$$= \sum_{k=1}^{3}(c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j)^2 \qquad (3.3.7)$$

Where the last expression has been written explicitly as the sum of the squares of each component of $p^i - p^j$. If we now manipulate (3.3.5) so that the left hand side is equal to zero and substitute the expression from (3.3.7) we obtain

$$0 = \left\|p^i - p^j\right\|^2 - \left\|d_{ij}\right\|^2 \qquad (3.3.8)$$

$$0 = \sum_{k=1}^{3}(c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j)^2 - \left\|d_{ij}\right\|^2 \qquad (3.3.9)$$

Let (3.3.9) be defined as f_m, the mth component of the vector function f, which consists of $\frac{n(n-1)}{2}$ components for n observed points. f is constructed in the following manner

$$f = \begin{pmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_{\frac{n(n-1)}{2}} \end{pmatrix}$$

such that

$$f_1 = \|p^1 - p^2\|^2 - \|d_{12}\|^2$$

$$f_2 = \|p^1 - p^3\|^2 - \|d_{13}\|^2$$

$$\dots$$

$$f_n = \|p^1 - p^n\|^2 - \|d_{1n}\|^2$$

$$f_{n+1} = \|p^2 - p^3\|^2 - \|d_{23}\|^2$$

$$\dots$$

$$f_{2n-1} = \|p^2 - p^n\|^2 - \|d_{2n}\|^2$$

$$..$$

$$f_{\frac{n(n-1)}{2}} = \|p^{n-1} - p^n\|^2 - \left\|d_{(n-1)n}\right\|^2$$

Now that the construction of the input function to the Newton Method has been explained, the calculation of its Jacobean matrix will be considered. We start by considering the partial derivatives for its mth component, $\frac{\partial f_m}{\partial \lambda_i}$ and $\frac{\partial f_m}{\partial \lambda_j}$

$$\frac{\partial f_m}{\partial \lambda_i} = \sum_{k=1}^{3} 2v_k^i(c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j) \tag{3.3.10}$$

Similarly

$$\frac{\partial f_m}{\partial \lambda_j} = \sum_{k=1}^{3} -2v_k^i(c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j) \tag{3.3.11}$$

We define ω_k^{ij} as

$$\omega_k^{ij} = c_k^i - c_k^j + \lambda^i v_k^i - \lambda^j v_k^j \tag{3.3.12}$$

So that (3.3.10) and (3.3.11) can be simplified to

$$\frac{\partial f_m}{\partial \lambda_i} = \sum_{k=1}^{3} 2v_k^i w_k^{ij} \tag{3.3.13}$$

$$\frac{\partial f_m}{\partial \lambda_j} = \sum_{k=1}^{3} -2v_k^i w_k^{ij} \tag{3.3.14}$$

The Jacobean matrix has the following form:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial \lambda_1} & \cdots & \frac{\partial f_1}{\partial \lambda_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial \lambda_1} & \cdots & \frac{\partial f_n}{\partial \lambda_n} \end{bmatrix}$$

Fortunately, each component of f depends on only two variables, which considerably simplifies its calculation.

Let us consider an example with $n = 4$. The components of f will be:

$$\begin{aligned}
f_1 &= \|p^1 - p^2\|^2 - \|d_{12}\|^2 \\
f_2 &= \|p^1 - p^3\|^2 - \|d_{13}\|^2 \\
f_3 &= \|p^1 - p^4\|^2 - \|d_{14}\|^2 \\
f_4 &= \|p^2 - p^3\|^2 - \|d_{23}\|^2 \\
f_5 &= \|p^2 - p^4\|^2 - \|d_{24}\|^2 \\
f_6 &= \|p^3 - p^4\|^2 - \|d_{34}\|^2
\end{aligned}$$

The general form of the Jacobean matrix of f is shown below. The final expression takes the fact that each component of f depends on only two variables into consideration.

$$J_f = \begin{bmatrix}
\frac{\partial f_1}{\partial \lambda_1} & \frac{\partial f_1}{\partial \lambda_2} & \frac{\partial f_1}{\partial \lambda_3} & \frac{\partial f_1}{\partial \lambda_4} \\
\frac{\partial f_2}{\partial \lambda_1} & \frac{\partial f_2}{\partial \lambda_2} & \frac{\partial f_2}{\partial \lambda_3} & \frac{\partial f_2}{\partial \lambda_4} \\
\frac{\partial f_3}{\partial \lambda_1} & \frac{\partial f_3}{\partial \lambda_2} & \frac{\partial f_3}{\partial \lambda_3} & \frac{\partial f_3}{\partial \lambda_4} \\
\frac{\partial f_4}{\partial \lambda_1} & \frac{\partial f_4}{\partial \lambda_2} & \frac{\partial f_4}{\partial \lambda_3} & \frac{\partial f_4}{\partial \lambda_4} \\
\frac{\partial f_5}{\partial \lambda_1} & \frac{\partial f_5}{\partial \lambda_2} & \frac{\partial f_5}{\partial \lambda_3} & \frac{\partial f_5}{\partial \lambda_4} \\
\frac{\partial f_6}{\partial \lambda_1} & \frac{\partial f_6}{\partial \lambda_2} & \frac{\partial f_6}{\partial \lambda_3} & \frac{\partial f_6}{\partial \lambda_4}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial f_1}{\partial \lambda_1} & \frac{\partial f_1}{\partial \lambda_2} & 0 & 0 \\
\frac{\partial f_2}{\partial \lambda_1} & 0 & \frac{\partial f_2}{\partial \lambda_3} & 0 \\
\frac{\partial f_3}{\partial \lambda_1} & 0 & 0 & \frac{\partial f_3}{\partial \lambda_4} \\
0 & \frac{\partial f_4}{\partial \lambda_2} & \frac{\partial f_4}{\partial \lambda_3} & 0 \\
0 & \frac{\partial f_5}{\partial \lambda_2} & 0 & \frac{\partial f_5}{\partial \lambda_4} \\
0 & 0 & \frac{\partial f_6}{\partial \lambda_3} & \frac{\partial f_6}{\partial \lambda_4}
\end{bmatrix}$$

If we substitute from expressions (3.3.13) and (3.3.14) we find

$$J_f = 2\sum_{k=1}^{3} \begin{bmatrix}
v_k^1 w_k^{12} & -v_k^2 w_k^{12} & 0 & 0 \\
v_k^1 w_k^{13} & 0 & -v_k^3 w_k^{13} & 0 \\
v_k^1 w_k^{14} & 0 & 0 & -v_k^4 w_k^{14} \\
0 & v_k^2 w_k^{23} & -v_k^3 w_k^{23} & 0 \\
0 & v_k^2 w_k^{24} & 0 & -v_k^4 w_k^{24} \\
0 & 0 & v_k^3 w_k^{34} & -v_k^4 w_k^{34}
\end{bmatrix} \qquad \textit{(3.3.14)}$$

Examination of this example shows that the calculation of the Jacobean can be done algorithmically as follows:

```
$r = 1;$
for $i$ = 1 … $\frac{n(n-1)}{2}$
        for $j$ = $i$ +1 … $n$

                $J_{ri} = 2\sum_{k=1}^{3}(v_k^i w_k^{ij})$
                $J_{rj} = -2\sum_{k=1}^{3}(v_k^i w_k^{ij})$
                $r = r + 1$
```

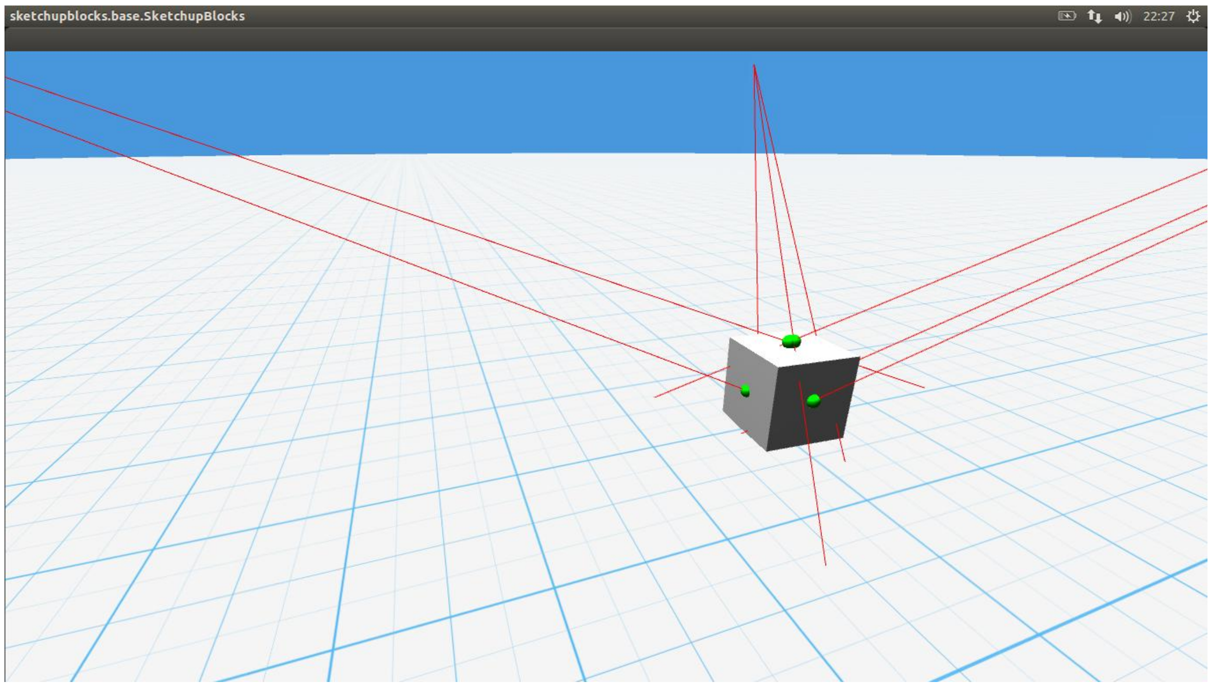
As has been mentioned before, Newton's method requires an initial estimate for the unknown – in this case, this estimate will be a vector of the form $(\lambda_0^1\ \lambda_0^2\ \dots\ \lambda_0^n)^t$. Since the cameras used by the Sketchup Blocks system have limited field of view, the distances between the cameras and the blocks fall in a fairly limited range. In addition, the positions of blocks are often recalculated even though the blocks' positions haven't changed. This is due to noisy input data. At any rate, this allows the system to make use of historical data as good initial estimates, thereby increasing the chances of convergence.

However, if the Newton method does not converge, then the Sketchup Blocks system falls back to PSO. The following error function is used:

$$Error = \left(\sqrt{\sum_{k=1}^{3} (p_k^1 - p_k^2)^2}^{\,2} - d_{12} \right)^2 \qquad (3.3.15)$$

PSO lends itself well to noisy data and can easily be extended to solve systems with a varying number of variables.

The calculated positions of the observed fiducials are shown in the screenshot below (Fig. 3.3.2)



Calculated Fiducial Points (Fig. 3.3.2)

3.4 BLOCK ORIENTATION CALCULATION

The previous sections have calculated the positions of the observed fiducials in 3D space. These positions must now be examined to find out where exactly to put the block and how to orientate it. This is describes by means of a transformation matrix. Since the Sketchup Blocks system stores the position of a block's vertices in model space (a frame of reference in which the block is at the origin), these vertices need only be multiplied with the transformation matrix to find their final position in 3D space. Since the Sketchup Blocks system does not perform any scaling, this transformation is a Rigid transform.

The optimal rigid transformation matrix is found by first calculating the centroids of the points (for the observed points and for the points in model space), translating the points so that they are at the origin, finding the optimal rotation matrix and then calculating the translation matrix. The translation and rotation matrices are then combined into the final transformation matrix. The process is explained below.

Let dataset A be the positions of the fiducial centers in model space. Let dataset B be the positions of the n observed fiducial centers in 3D space. Let the ith point be represented as p^i.

First the centroids of these datasets are calculated.

$$centroid_A = \sum_i^n p_A^i \tag{3.4.1}$$

$$centroid_B = \sum_i^n p_B^i \tag{3.4.2}$$

Both A and B are then translated so that their centroids are at the origin.

In order to calculate the rotation matrix, the covariance between the two datasets must first be calculated. The covariance matrix will be denoted by H.

$$H = \sum_{i=1}^n \left(p_A^i - centroid_A\right)\left(p_B^i - centroid_B\right)^t \tag{3.4.3}$$

The covariance matrix is then decomposed (by means of SVD) into the following three matrices:

$$H = USV^T \tag{3.4.4}$$

The rotation and translation of dataset B (denoted by R and t respectively) can then by found by these simple equations:

$$R = VU^T \tag{3.4.5}$$

$$t = -R * centroid_A^T + centroid_B^T \tag{3.4.6}$$

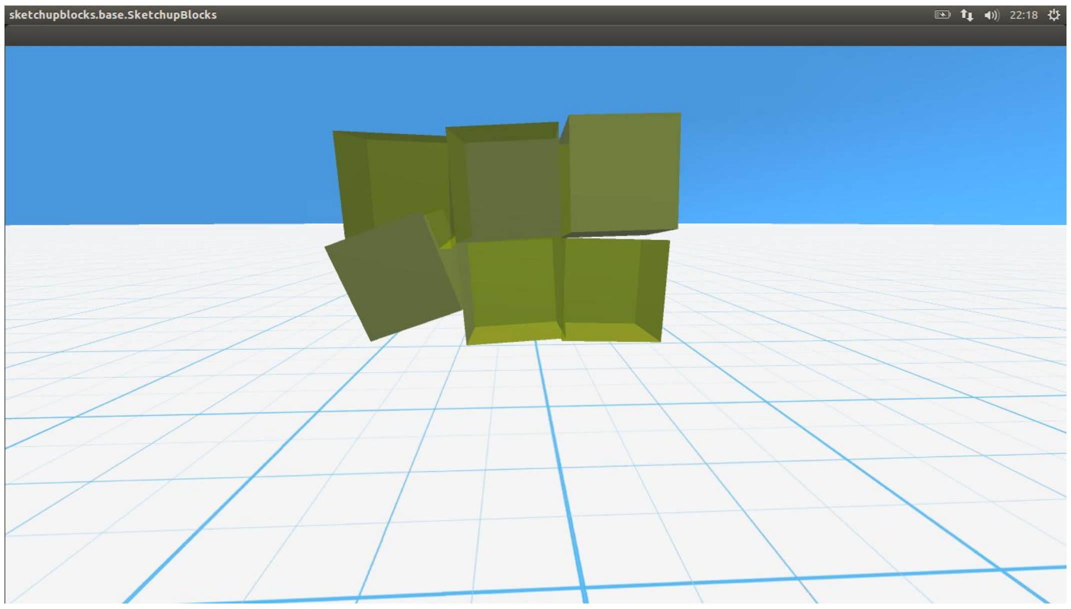
The rotation and translation can then be interpreted to mean:

$$B = R * A^T + t \tag{3.4.7}$$

Note that this method requires at least three points in each dataset, which poses a problem if only two fiducial markers are observed. This problem and its solution will be discussed further at a later stage.

3.5 PSEUDO-PHYSICS APPLICATION

Once enough blocks have been built, the introduction of new blocks prevents the cameras from picking up blocks that have already been built. This data scarcity coupled with the inherent noisiness of the input data causes varying degrees of error in the final placement and rotation calculated for a particular block. The visual effects of this can be seen in the screen shot below (Fig 3.5.1).



Construction without Pseudo Physics

(Fig. 3.5.1)

The Sketchup Blocks system compensates for the minor variation by applying 'Pseudo-Physics.' The Pseudo Physics engine applies a crude form of gravity to the digital model by ensuring that every block is placed flat on the surface it is placed upon.

Every time a new block is introduced, the bottom face of the block as well as the surface it is placed on is identified. Let the surface that the block is placed on be S. The normal of S is then calculated.

If the inverted surface normal, t, is parallel to the normal of the block's bottom face, s, then the block is placed flat on S. Otherwise, gradient descent is used to apply minor modifications to the block's transformation matrix until t and s are approximately parallel. Lastly, the new block is translated to match the height of the surface it is on. This entire process is discussed in more detail below.

The surface on which the block is to be placed is found by 2D bounding box methods. If the block is placed on another block, their projections onto the x-y plane will intersect. Consequently, each of the blocks that have already been placed are projected onto the x-y plane and are then tested for overlap with the projection of the new block onto the x-y plane.

This test consists of a broad phase and a narrow phase. The broad phase simply considers the maximum and minimum points of the blocks to test for possible overlap. The narrow phase makes use of the Separating Axis Theorem.

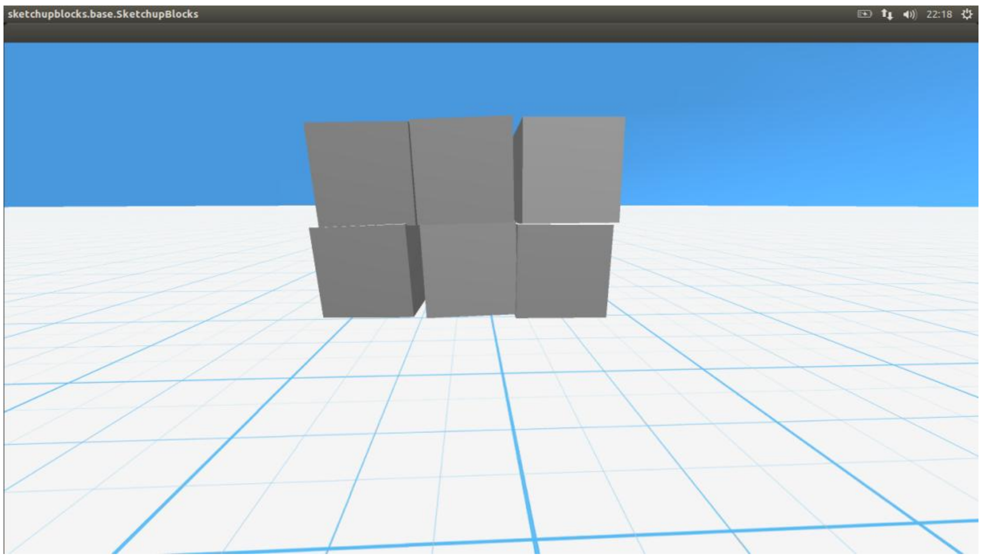
The highest surface that overlaps with the new block on the x-y plane is chosen as the surface on which the block has been placed. This allows blocks to be stacked.

A primitive form of gradient descent is applied to modify the transformation matrix of the block in order to make t and s parallel. This is equivalent to minimizing

$$error = s \cdot t - 1 \qquad (3.5.1)$$

The modification of the transformation matrix can be described as small steps of rotation that take place repeatedly around first the x axis and then the y axis until the error no longer improves.

The block is then translated down so that it is placed flat on top of S. The outcome of applying the Pseudo Physics method can be seen in the screen short below (Fig. 3.5.2)



Construction without Pseudo Physics

(Fig. 3.5.1)

This is the final step in calculating the position and rotation of a block.

7. GLOSSARY

Collada	Interchange file format for interactive 3D applications that is supported by Google Sketchup.
Command block	A smart block used to control the system (as opposed to model construction).
Construction floor	The flat surface on which the physical model is built and around which the web cameras are erected.
Fiducial markers	Small markers that will be attached to the physical blocks and used to identify them.
Google Sketchup	3D modelling program.
IoT	Internet of Things
MVC	Model-View-Controller, an architectural pattern
Object file	The Collada file that contains all the data of a particular model
PSO	Particle Swarm Optimization
ReacTIVision	Open source, cross-platform computer vision framework for tracking fiducial markers that are attached to physical objects.
Smart block	A physical block with an attached fiducial marker.
TUIO	A protocol that allows the encoding and transmission of a tangible interface component abstraction, such as tokens, pointers and geometries. This is the protocol by which ReacTIVision will communicate with the Sketchup Blocks system.