COSC460

Novel Methods for Reflective Symmetry Detection in Scanned 3D Models

October 16, 2015

Matthew Stephenson
mjs357@uclive.ac.nz
Supervisor: Richard Green
richard.green@canterbury.ac.nz
Co-Supervisor: Adrian Clark
adrian.clark@canterbury.ac.nz

Abstract

The concept of detecting symmetry within 3D models has received an extensive amount of research within the past decade. Numerous algorithms have been proposed to identify reflective symmetry within 3D meshes and to extract a quantitative measure for the model's level of symmetry. Much of this existing work focuses on identifying symmetry in noiseless 3D models with most methods unable to work effectively on models distorted by noise, such as those commonly obtained when scanning objects in the real world. This report details the design and implementation of two robust and fast algorithms, which can be used on a wide variety of models to identify global approximate reflective symmetry. These proposed methods are also able to identify likely planes of symmetry in models that have been distorted with noise or contain minor imperfections, making them ideal for scanned models of real world objects. The hypothesis planes are determined by principal component analysis, after which the proposed algorithms give each plane a numerical value corresponding to its likelihood of being a plane of global approximate reflective symmetry. The first algorithm uses the Hausdorff distance between vertices to estimate symmetry, whilst the second uses an approach based on ray casting. We estimate the accuracy of our proposed methods to be 96.88% for the Hausdorff distance method and 93.75% for the ray casting method.

Acknowledgements

I would like to thank Richard Green and Adrian Clark for all their supervision and assistance on this project, as well as providing feedback on my reports and presentation. In addition I would like to thank Andy Cockburn for offering additional insight on report writing style and to him and Tanja Mitrovic for examining this report. Lastly I would like to thank all my friends, family and lecturers for their emotional assistance throughout the year.

CONTENTS

1 Introduction

Symmetry is a mathematical concept that exists in many man-made objects, as well as being widely prevalent in nature (Liu 2010). These objects can be represented digitally as 3D geometric models, which are typically encoded as a mesh created from small polygons, usually triangles, with little to no information about their higher level structure. Determining additional properties of a model, including its global reflective symmetry, is an important task within computer graphics and computer vision. Many existing applications benefit greatly from the ability to identify and extract symmetry, such as 3D model retrieval, geometric problem solving, object recognition, robotic assembly, procedural modelling, segmentation and remeshing.

Many of the current methods for identifying global reflective symmetry planes suffer from a range of problems. These include the inability to detect approximate symmetry within complex geometry and restrictions on the model's structure, such as being convex or fully connected.

The aim of this research is to develop simple, accurate and fast algorithms that can be used to detect likely planes of global approximate reflective symmetry within scanned models of 3D objects, which are often distorted by noise. By first identifying potential planes of symmetry within the model, the algorithms calculate a measure for how likely each hypothesis plane is to be a plane of reflective symmetry. This value is then compared against a threshold to determine whether it is large enough for the given model.

Likely planes of reflective symmetry are determined using principal component analysis (PCA) and two proposed methods for measuring the planes likelihood of symmetry have been developed. The first method utilises the Hausdorff distance between vertices on either side of the hypothesis plane. The second method utilises ray casting to determine the deviation between mesh intersection points on either side of the hypothesis plane. Several variations to each method which improve their accuracy and runtime are also investigated.

1.1 Report outline

This report begins with an explanation about some of the key concepts within symmetry, followed by an overview of relevant prior work, including research done on both global and partial symmetry detection. Section 3 details the mathematical design and implementation of our proposed methods. Section 4 demonstrates how we have evaluated our proposed algorithms and presents the final results. Section 5 contains a discussion of these results and a comparison of the proposed algorithm's strengths and weaknesses, as well as a brief look into some of the applications that 3D symmetry detection has. Section 6 presents our final conclusion and outlines possible future work which could be conducted to improve our methods.

2 Background and Related Work

2.1 Key concepts within symmetry

There are many different categories of symmetry that any 2D image or 3D model may possess. These can be divided into basic types of symmetry (symmetry that most people are familiar with) and complex types of symmetry (symmetry that is less commonly known).

2.1.1 Basic types of symmetry

Reflectional symmetry

Reflectional symmetry, mirror symmetry or bilateral symmetry represents symmetry due to reflection (see Figures 2.1 and 2.3). In 3D objects there is a plane about which reflection takes place and for 2D images this is a vector. A plane of symmetry for a 3D object is any plane such that if the object was reflected about this there would be no visible change. A plane of reflective symmetry will always go through the centre of the object. This is the type of symmetry that our method is attempting to detect.

Rotational symmetry

Rotational symmetry or radial symmetry represents symmetry due to rotation (see Figures 2.2 and 2.3). In 3D objects there is a vector around which rotation takes place and for 2D images there is a point. A 3D object is said to have a vector of rotational symmetry if it can be rotated a certain amount around this vector and still look the same. The objective of rotational symmetry algorithms is to determine the direction and amount of rotational symmetry the object possesses.

Translational symmetry

Translational symmetry represents symmetry due to translation. In 3D objects the translation is

Figure 2.1: Shape with reflective symmetry

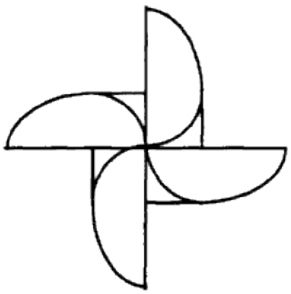


Figure 2.2: Shape with rotational symmetry

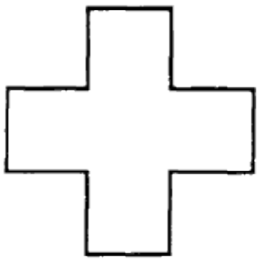


Figure 2.3: Shape with reflective and rotational symmetry

given as a vector in three dimensions (x, y, z) and for 2D images this is given in two dimensions (x, y). A 3D object is said to have translational symmetry if it is comprised of identical elements that can be separated using planes.

2.1.2 Complex types of symmetry

Global symmetry

An object that contains some form of symmetry throughout the entire model is said to contain global symmetry (see Figure 2.4). This is in contrast to the idea of an object having partial symmetry.

Partial symmetry

An object that does not contain some form of symmetry throughout the entire model may still possess partial symmetry. This occurs when some part of the model possesses a form of symmetry but another part of it does not. For example, this occurs in Figure 2.5 where the horse's legs are in different positions on either side of its body. The rest of the model apart from the legs is symmetrical so the model is said to have partial symmetry.

Approximate symmetry

For digital models created by a human it is easy to say whether an object has symmetry or not. However, it is extremely unlikely for a scanned real world object to contain perfect symmetry of the types discussed so far. It is more likely that an object would have approximate symmetry, where the symmetry is not mathematically exact but is close enough that we could identify it as such. A good example of this would be a person's head. For most people the two halves of their head are not identical but are close enough that we would state that there was approximate symmetry. For methods which attempt to detect approximate symmetry, this is usually achieved by calculating a numerical value representing the deviation between the actual and ideal symmetry of the model. If this value is small enough then the model is said to have approximate symmetry, although the threshold used for this decision can vary significantly.

Figure 2.4: Model that contains global reflective symmetry

Figure 2.5: Model that contains partial reflective symmetry

2.2 Early work within symmetry detection

2.2.1 2D symmetry detection

Early symmetry detection algorithms were only concerned with identifying exact symmetries within 2D images represented as a set of planar points. The most common way of achieving this is by reducing the 2D symmetry problem to a 1D pattern matching problem which works in O(n log n) time (Atallah 1985). This approach can be adapted and improved further, with two notable extensions being the ability to detect partial symmetry within a 2D image (S. Parry-Barwick 1993) and the ability to detect approximate symmetry by utilising a hierarchy that defines symmetry as a continuous feature (Zabrodsky, Peleg et al. 1995). However, both of these additions are very computationally expensive and rely on the algorithm's ability to establish correspondence between points within the image.

2.2.2 Primitive 3D symmetry detection

The original idea of reducing 2D symmetry detection to a 1D pattern matching problem can be expanded to detect symmetry in 3D point sets (Wolter, Woo et al. 1985) as well as the ability to detect approximate symmetries using similar principles (Alt, Mehlhorn et al. 1988).

2.3 Advanced methods for 3D symmetry detection

After this initial research had constructed the basis for more advanced 3D model symmetry detection algorithms, many improvements and variations were proposed in subsequent years. A comparison of these previous methods is presented at the end of this section (see Table 2.1).

2.3.1 Global symmetry detection

Identifying automorphisms of planar triply connected graphs

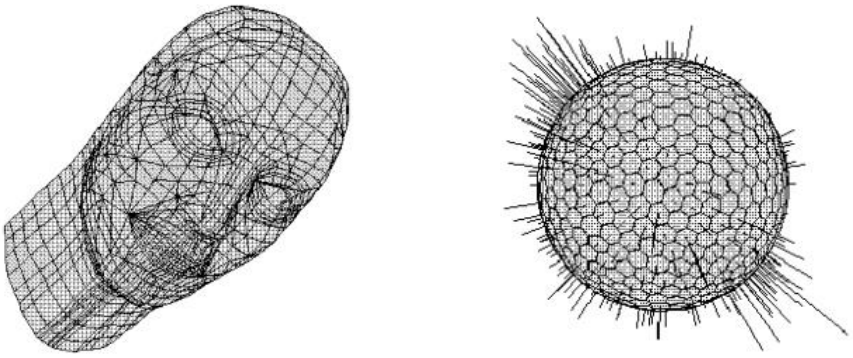
One of the earliest methods for detecting rotational symmetry in 3D models creates a graph-based representation of the solid object (Jiang and Bunke 1991). Hypothetical symmetry axes are then extracted, by finding automorphisms of the graph and a rotation matrix. This method can determine global and approximate symmetry for rotation. This method has many downsides however, as it is highly dependent on the topology of the model, requiring the mesh to be fully connected in order to generate the corresponding graph. It is also very susceptible to noise or other small imperfections within the object's geometry. The algorithm used has quadratic complexity and requires $O(m^2)$ time, where m represents the number of edges in the object. This means that whilst this method is simple to implement, it suffers from many geometry restrictions and computational inefficiency.

Using Octree representation to identify symmetry

This method creates an octree representation of the model which is then traced to identify likely planes of symmetry (Minovic, Ishikawa et al. 1993). This was one of the first papers to propose the use of the principle axis transform to help orientate the object before attempting to detect symmetry. This allows the input object to be in an arbitrary position and rotation. Many subsequent algorithms used this or similar methods to first orientate the object before identifying potential symmetry planes. This method can determine global and approximate symmetry for rotation and reflection. However, this method does become more computationally complex for larger models and has been shown to be sensitive to noise.

Extended Gaussian image of model

Another approach to identifying symmetry in 3D models centres around the use of the extended Gaussian image of a model (Changming and Sherrah 1997). This method works by creating a tessellated sphere of hexagons around the object, with the same centre of mass as the mesh. The algorithm then iterates though each face of the mesh and assigns it to the hexagon which intersects with the face's normal vector, creating an orientation histogram for the model (see Figure 2.6). The number of hexagons used to create the tessellated sphere can be altered based on the desired level of accuracy. This method can determine global and approximate symmetry for rotation and reflection. The main problem with this method is how it responds to small imperfections in the model. While these typically only cause minor changes to the positions of the model's faces, they can have a major influence on the normal of the faces. This would make this method very impractical for use on scanned 3D models, as these are frequently subject to noise distortions.



(a) (b)

Figure 2.6: Simple mesh model of a human head (a) and corresponding orientation histogram (b) (Changming and Sherrah 1997)

Spherical harmonic coefficients of generalised moments

This method detects global symmetries of 3D models by analysing the extrema and spherical harmonic coefficients of generalised moments (Martinet, Soler et al. 2006). This method utilises the fact that the even order moments contain the same symmetries as the model. The generalised moments are not computed directly; instead their spherical harmonic coefficients are computed using an integral expression. After this, the extrema of these functions are used to identify candidates for symmetries, which are then checked against the original shape using an appropriate geometric measure. When compared to the previous algorithms, this method computes a deterministically small number of surface integrals whilst still providing fairly accurate results. This method can determine global and approximate symmetry for rotation and reflection. Whilst this method can be shown to detect reflective symmetry within scanned 3D models it was not specifically designed for this purpose. Because of this, the method is very inaccurate when applied to scanned models containing large holes or other distortions (scans must be have very high resolution and accuracy). It is also far more complex than most other methods, making it difficult to integrate easily into other applications.

2.3.2 Partial symmetry detection

Gaussian Euclidean distance transform

By using the Gaussian Euclidean distance transform it is possible to determine a shape descriptor similarity, detailing the distance between the shape descriptor of an object and its perfectly symmetrical equivalent. This can be used to determine a measure of a model's symmetry with respect to every axis passing through the centre of mass (Kazhdan, Funkhouser et al. 2004). This method can determine global and approximate symmetry for rotation and reflection. This method can also be used to detect partial symmetry but relies on the algorithm's ability to find suitable pairs of vertices within the model (Podolak, Shilane et al. 2006). This makes the method good for shape identification but very costly for accurate results, as the algorithm has to compute the surface integration for each of the sampled directions.

Stochastic clustering to find pairs of vertex groups

By matching local shape signatures, followed by stochastic clustering in transformation space, it is possible to extract potential symmetry planes from a 3D model (Mitra, Guibas et al. 2006). The first part of this method works by computing simple descriptors at a set of chosen locations on the shape. These local descriptors are then used to pair up groups of vertices to form clusters that provide information about the symmetry relation between them. The second part of this method extracts the significant modes of this mass distribution and uses this to check the spatial consistency, verifying whether symmetry is present. This algorithm is similar in design to another method which uses a variation of the Hough transform to extract features (Cailliere, Denis et al. 2008). These methods can determine global, partial and approximate symmetry for rotation and reflection. The main problem with these methods is that they rely on the ability of the algorithm to identify suitable pairs of vertices within the model, which may not always be possible.

Using 2D depth images

If a 3D model has been orientated, either manually or by using PCA (or a similar method), it is possible to get a quick estimation of symmetry using 2D depth images. This method is much faster than any of the previous approaches but suffers from being far less accurate. After taking a 2D depth image of each side of the orientated model, the image is analysed to determine if any symmetry is likely to be present (using a 2D symmetry detection algorithm). Whilst a symmetry detection method of this nature is not mathematically valid, it can provide a reasonably accurate estimation of symmetry (Axenopoulos, Litos et al. 2011). These methods are typically only used for real-time symmetry detection, as their accuracy is much less than what would normally be desired.

2.3.3 Additional detection methods

This section describes several other symmetry detection algorithms that are designed for specific situations or requirements.

Detecting 3D symmetry from 2D images

All of the methods mentioned so far rely on the model to be fully accessible and manipulable by the algorithm. It is sometimes the case however, that symmetry must be estimated without the objects full mesh being available. Instead, a single or collection of 2D images of the object is provided, from which the symmetry of the original 3D object is estimated. This can be achieved either from a sketch of the model (Zou and Lee 2005), a single 2D image of a volumetric shape (Sawada and Pizlo 2008) or by using a view-based approach (Li, Johan et al. 2014). These methods have demonstrated reasonable accuracy under the right circumstances and are mainly used to detect global reflective symmetry in specialised situations.

Curved reflective symmetry detection

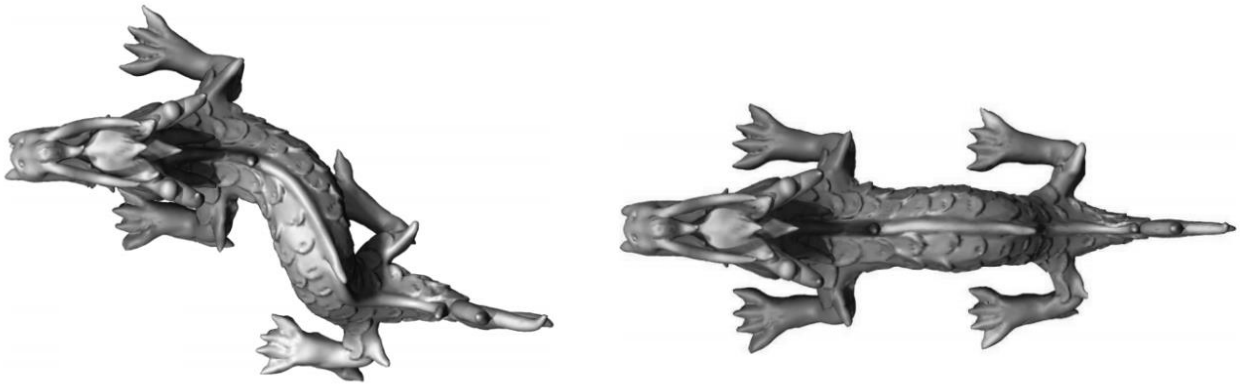
There are also some algorithms that attempt to detect, or correct, curved reflective symmetry within 3D models. These methods are used when an object contains reflective symmetry about a curved plane rather than a straight one. This symmetry is detected by identifying matching sections of partial symmetry and then connecting all these planes together (Liu and Liu 2011). The position of the objects vertices can then be adjusted so that their corresponding planes are parallel, giving the effect of symmetrizing the model (see Figure 2.7). (Mitra, Guibas et al. 2007).

Symmetries of non-rigid shapes

By extending the concepts of intrinsic symmetry for a non-symmetric model, it is possible to detect symmetry within a non-rigid shape (Raviv, Bronstein et al. 2010). Similar to the previous method of curved symmetry detection, symmetry within a deformed model can be identified by connecting many small planes of partial symmetry. From this the algorithm can then decide whether the model may possess intrinsic or extrinsic symmetry (see Figure 2.8).

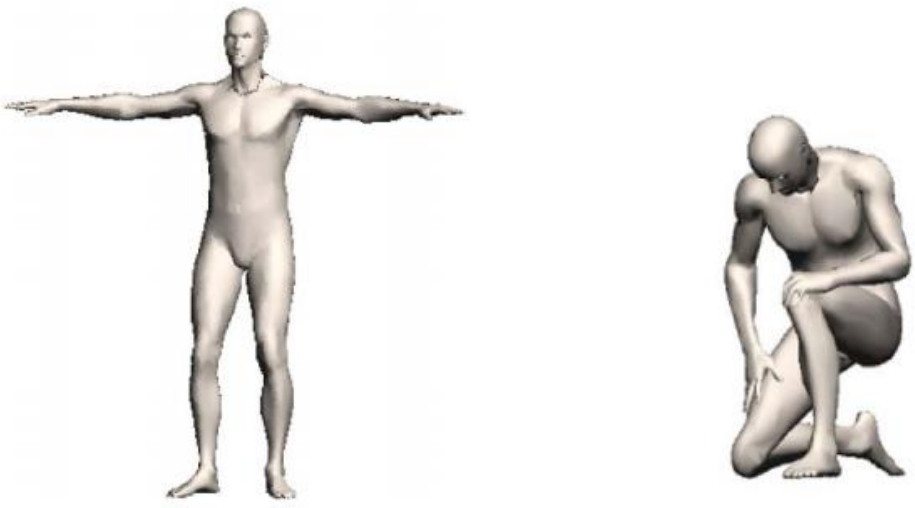
2.4 Summary

While there are many previous methods for symmetry detection they contain several limitations which make them ineffective on scanned 3D models, including being topology dependent, sensitive to noise and requiring vertex pairings. Our proposed algorithms are designed specifically for these types of models, with the goal of providing a robust and fast means of detecting global approximate reflective symmetry. The next chapter details the design and implementation of our algorithms.



(a) (b)

Figure 2.7: Model that contains curved reflective symmetry (a) which has then been symmetrized (b) (Mitra, Guibas et al. 2007)



(a) (b)

Figure 2.8: Model that contains extrinsic symmetry (a); model that contains intrinsic symmetry (b) (Raviv, Bronstein et al. 2010)

Reference	Reflection	Rotation	Global	Approximate	Partial	Designed for scanned models
(Alt, Mehlhorn et al. 1988)	✓		✓	✓		
(Axenopoulos, Litos et al. 2011)	✓		✓	✓		
(Cailliere, Denis et al. 2008)	✓		✓	✓	✓	
(Changming and Sherrah 1997)	✓	✓	✓	✓		
(Jiang and Bunke 1991)		✓	✓	✓		
(Kazhdan, Funkhouser et al. 2004)	✓	✓	✓	✓		
(Martinet, Soler et al. 2006)	✓	✓	✓	✓		
(Minovic, Ishikawa et al. 1993)	✓	✓	✓	✓		
(Mitra, Guibas et al. 2006)	✓	✓	✓	✓	✓	
(Podolak, Shilane et al. 2006)	✓		✓	✓	✓	
(Wolter, Woo et al. 1985)	✓		✓			
Proposed method	✓		✓	✓		✓

Table 2.1: General comparison of features between the most common 3D symmetry detection methods and our proposed method

3 Design and Implementation

Our algorithms for global reflective symmetry detection both have two distinct processes. The first process involves determining potential planes of reflective symmetry (hypothesis planes) by using principal component analysis (PCA). The second process involves calculating a symmetry measure for each of the hypothesis planes based on the level of reflective symmetry the model has with respect to it. Our two algorithms differ in this second process. One uses the Hausdorff distance and the other uses ray casting. A flowchart of the entire program is provided at the end of this section (see Figure 3.14).

3.1 Identifying hypothesis planes

Using PCA to orientate a model before attempting symmetry detection is a technique that has been implemented in many previous methods and has been shown to work effectively at determining potential planes of reflective symmetry (Dimitrov 2012). For this reason it was selected as the method by which to derive the hypothesis planes. In order to perform PCA on a model it is necessary to first estimate the model's centre of mass

3.1.1. Centre of mass approximation

There are two main methods for determining the centre of mass $\boldsymbol{M}$ for a model constructed using the set of vertices $\boldsymbol{V}$.

The first method is to use the mean position of each vertex within the mesh.

$$\boldsymbol{M} = \frac{\mathbf{1}}{|\boldsymbol{V}|}\sum_{\boldsymbol{v}\in\boldsymbol{V}} \boldsymbol{v}$$

The second method is to use the centre of the mesh's bounding box.

$$\boldsymbol{M} = \frac{\boldsymbol{\max_{v\in V} V} - \boldsymbol{\min_{v\in V} V}}{\mathbf{2}} + \boldsymbol{\min_{v\in V} V}$$

Although both of these methods have limitations, they are each suited to different types of models. The first method is more suited to models that may potentially contain noise or outliers. The second method is more suited to models where the vertices are not spread evenly throughout the mesh. In the case of scanned 3D models it is more often the case that the data is noisy or contains outliers, meaning that the first method would generally be the more suitable choice.

3.1.2. Principal component analysis

PCA is a method of determining, for a given dataset, the direction along which the data varies the most. The result of performing PCA on a 3D collection of points is two eigenvectors representing the principal components (see Figure 3.1). This is achieved through the concept of dimensionality reduction, where the number of dimensions $\boldsymbol{p}$ within a dataset $\boldsymbol{X}$ is reduced to a desired value $\boldsymbol{L}$.

Firstly, the data is arranged as a set of $\boldsymbol{n}$ vectors and placed in a matrix $\boldsymbol{X}$ of dimensions $\boldsymbol{n} \times \boldsymbol{p}$. The deviations from the centre of mass are then calculated by subtracting $\boldsymbol{M}$ from each row of the data matrix $\boldsymbol{X}$. This is then stored in a matrix $\boldsymbol{B}$ of size $\boldsymbol{n} \times \boldsymbol{p}$.

$$\boldsymbol{h}[\boldsymbol{i}] = \boldsymbol{1}, \boldsymbol{i} = \boldsymbol{1} \ldots \boldsymbol{n}$$

$$\boldsymbol{B} = \boldsymbol{X} - \boldsymbol{h}\boldsymbol{u}^{\boldsymbol{T}}$$

The covariance matrix $\boldsymbol{C}$ is then calculated from the outer product of matrix $\boldsymbol{B}$ with itself.

$$\boldsymbol{C} = \frac{\boldsymbol{1}}{\boldsymbol{n} - \boldsymbol{1}} \boldsymbol{B}^{*} \boldsymbol{B}$$

(Where $*$ is the conjugate transpose operator)

Lastly, the matrix $\boldsymbol{V}$ of eigenvectors that diagonalizes the covariance matrix $\boldsymbol{C}$ is calculated using the diagonal matrix $\boldsymbol{D}$ of eigenvalues of $\boldsymbol{C}$.

$$\boldsymbol{V}^{-\boldsymbol{1}} \boldsymbol{C} \boldsymbol{V} = \boldsymbol{D}$$

$\boldsymbol{D}$ is a $\boldsymbol{p} \times \boldsymbol{p}$ diagonal matrix, where $\boldsymbol{\lambda_k}$ represents the $\boldsymbol{k}$-th eigenvalue of the covariance matrix $\boldsymbol{C}$.

$$\boldsymbol{D}[\boldsymbol{k}, \boldsymbol{l}] = \begin{cases} \boldsymbol{\lambda_k}, \boldsymbol{k} = \boldsymbol{1} \\ \boldsymbol{0}, \boldsymbol{k} \neq \boldsymbol{1} \end{cases}$$

In effect, this allows us to determine the direction of maximum variation in the mesh and the direction of maximum variance perpendicular to this. These two vectors together form the principle components, from which the hypothesis planes for reflective symmetry can be derived.

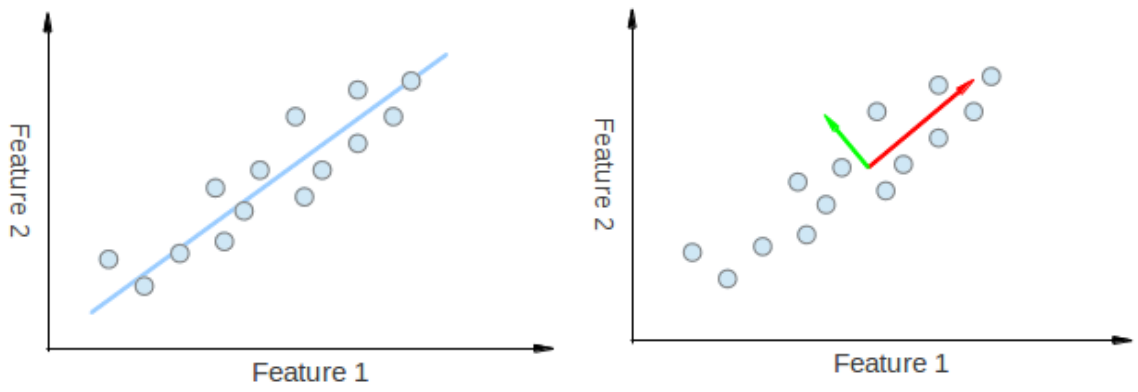


Figure 3.1: Example of PCA in 2D space

3.1.3. Hypothesis symmetry planes

Using the two eigenvectors calculated from PCA we then identify three hypothesis planes, defined as follows:

- Plane 1: The plane containing both PCA eigenvectors.
- Plane 2: Formed by creating a plane along the first PCA eigenvector and which is also orthogonal to Plane 1.
- Plane 3: Formed by creating a plane along the second PCA eigenvector and which is also orthogonal to Plane 1.

In many symmetrical models, simply using the two PCA eigenvectors to form a plane is a good method for finding the plane of reflective symmetry, yet in some models this is not the case. For the model in Figure 3.2 for example, the two eigenvectors found point in the correct directions to identify the plane of reflective symmetry. For the model in Figure 3.3 however, one of the eigenvectors points in an incorrect direction. This is because the low flat body type of the fly means that there is greater variation from left to right rather than top to bottom. For this reason we also calculate the two planes which are orthogonal to the first plane but parallel to one of the eigenvectors. This has been shown through experimentation to identify reflective symmetry in a large number of 3D models (Dimitrov 2012).

With the hypothesis planes identified, it is necessary to calculate a symmetry measure for determining whether or not each of the hypothesis planes is also a plane of reflective symmetry. Two alternative methods for calculating this measure are proposed in the following sections.

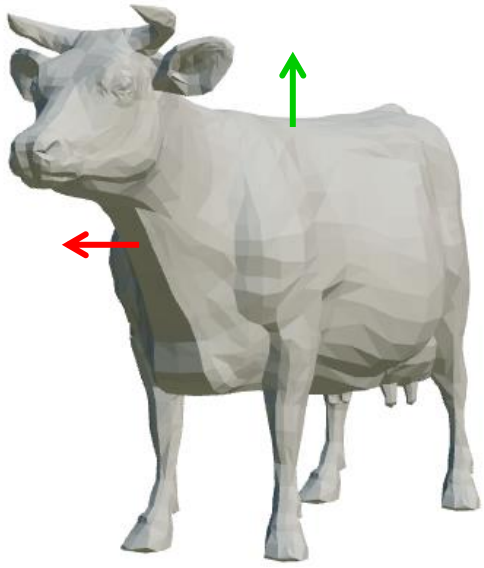


Figure 3.2: PCA eigenvalues for cow mesh

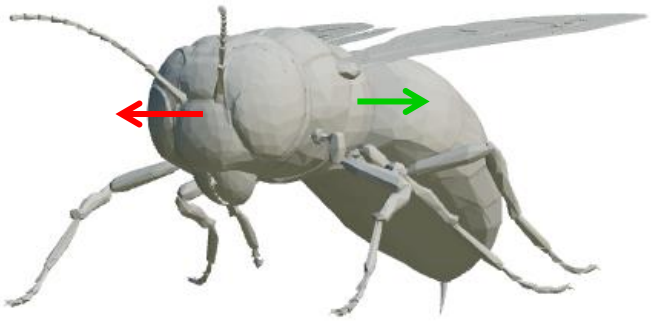


Figure 3.3: PCA eigenvalues for fly mesh

3.2 Hausdorff distance approach

The first method for calculating a symmetry measure uses a variation of the Hausdorff distance algorithm to estimate a symmetry measure for each of the model's hypothesis planes.

3.2.1. Mesh split and reflection

The mesh is first split into two smaller meshes using the hypothesis plane that is being tested. This is done by iterating through each vertex $\boldsymbol{v_i}$ within the mesh and allocating it to one of two sets $\boldsymbol{S_a}$ or $\boldsymbol{S_b}$ based on its position relative to the hypothesis plane $\boldsymbol{ax + by + cz = 0}$.

$$\boldsymbol{v_i} \in \begin{cases} \boldsymbol{S_a} & \boldsymbol{if\ \ av_x + bv_y + cv_z < 0} \\ \boldsymbol{S_b} & \boldsymbol{if\ \ av_x + bv_y + cv_z > 0} \end{cases}$$

(Note that if $av_x + bv_y + cv_z$ is equal to zero then the vertex lies on the hypothesis plane and is not an element of either set)

If the hypothesis plane is a global reflective symmetry plane then each of the meshes created using these new vertex sets will be mirror images of each other. In order to determine whether this is the case, the vertices within set $\boldsymbol{S_a}$ are reflected about the hypothesis plane.

$$\boldsymbol{v_x = (1 - 2a^2)v_x - (2ab)v_y - (2ac)v_z}$$
$$\boldsymbol{v_y = (1 - 2b^2)v_y - (2ab)v_x - (2bc)v_z}$$
$$\boldsymbol{v_z = (1 - 2c^2)v_z - (2ac)v_x - (2bc)v_y}$$

If the two meshes are now approximately the same it can be assumed that the hypothesis plane is a plane of reflective symmetry.

3.2.2. Hausdorff distance symmetry measure

The Hausdorff distance is a similarity measure that is predominantly used to calculate the error created by simplifying a mesh, but it can easily be modified for our purpose here. The Hausdorff distance $\boldsymbol{d_H}$ is defined between two non-empty datasets $\boldsymbol{X}$ and $\boldsymbol{Y}$.

$$\boldsymbol{d_H(X,Y) = max\left\{\max_{x\in X}\ \min_{y\in Y}\ d(x,y),\ \max_{y\in Y}\ \min_{x\in X}\ d(x,y)\right\}}$$

In context, this is calculated by taking each vertex within a mesh and finding the minimum distance between it and any vertex on the other mesh. The same is then done with the meshes swapped and the maximum of these minimal distances is defined as the Hausdorff distance (Aspert, Santa-Cruz et al. 2002, Guthe 2005). Whilst this is good for measuring error during simplification (Cignoni, Rocchini et al. 1996) it is not entirely effective for our purposes. This is mainly because it returns the maximum deviation between the meshes, meaning that if our model is perfectly symmetrical apart from a single outlier then this would result in a large

Hausdorff distance (see Figure 3.4). Instead, we are likely to get a better result if the average distance is used as the similarity measure, rather than the maximum.

$$d_H(X,Y) = max\left\{\frac{1}{|X|}\sum_{x\in X} \min_{y\in Y} d(x,y), \frac{1}{|Y|}\sum_{y\in Y} \min_{x\in X} d(x,y)\right\}$$

One disadvantage of this new approach is that it increases the time required to sample the mesh, as we cannot apply any vertex culling or other traditional improvements to increase the algorithm's efficiency (Straub 2007, Barton, Hanniel et al. 2010).

In context, this new method is performed by taking each vertex within one of the meshes and recording the shortest distance between it and any vertex on the other mesh. We then compute the average of all these distances. The same is then done but with the meshes swapped and the maximum of these two averages is taken as the total deviation. The inverse of this deviation can then be used as a similarity measure. This level of similarity between these two meshes can also be used to represent a measure of symmetry S that the hypothesis plane has with respect to the original model. If this value is above a pre-determined threshold, then we conclude that the hypothesis plane is likely to be a plane of reflective symmetry.

3.2.3. Potential limitations

Whilst this method is simple to understand and implement, it suffers from being extremely inefficient and overly reliant on the sampling resolution of the model. This algorithm can potentially require exponential time, meaning that this method is impractical for models where the number of vertices is very high. Scanned 3D models can potentially contain millions of vertices, necessitating the creation of an alternative method for detecting symmetry which avoids these problems.

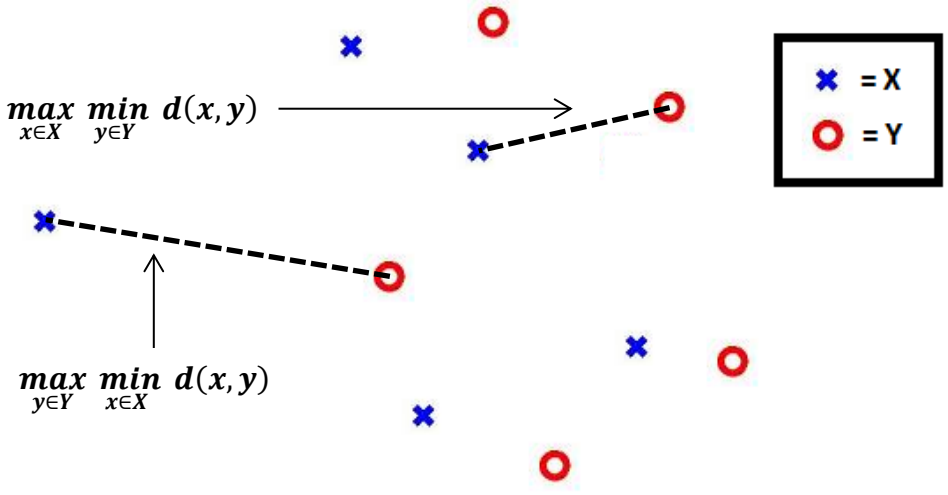


Figure 3.4: Hausdorff distance calculation

3.3 Ray casting approach

The second method for calculating a symmetry measure attempts to avoid the problem of sampling rate dependence by using ray casting to create a set of mesh intersection points.

3.3.1. Orientate plane and mesh

In order to simplify the ray casting algorithm, the model is first rotated so that the hypothesis plane aligns with the plane created by the x and y axis in world space. Firstly, the angle $\boldsymbol{\theta}$ and direction $\boldsymbol{D}$ by which to rotate the plane $\boldsymbol{P}$ are calculated.

$$\boldsymbol{Dot}(\boldsymbol{A},\boldsymbol{B}) = \boldsymbol{A_x B_x} + \boldsymbol{A_y B_y} + \boldsymbol{A_z B_z}$$

$$\boldsymbol{Cross}(\boldsymbol{A},\boldsymbol{B}) = \langle \boldsymbol{A_y B_z} - \boldsymbol{A_z B_y}\ ,\ \boldsymbol{A_z B_x} - \boldsymbol{A_x B_z}\ ,\ \boldsymbol{A_x B_y} - \boldsymbol{A_y B_x} \rangle$$

$$\boldsymbol{\theta} = \mathbf{cos}^{-1}\big(\boldsymbol{Dot}(\boldsymbol{P}, \langle \mathbf{0},\mathbf{0},\mathbf{1} \rangle)\big)$$

$$\boldsymbol{D} = \boldsymbol{Cross}(\boldsymbol{P}, \langle \mathbf{0},\mathbf{0},\mathbf{1} \rangle)$$

For every vertex $\boldsymbol{V}$ within the mesh, a new position is then determined.

$$\begin{aligned}
\boldsymbol{V_x} &= \big(\boldsymbol{D_x}^2(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \mathbf{cos}\,\boldsymbol{\theta}\big)\boldsymbol{V_x} + \big(\boldsymbol{D_x D_y}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) - \boldsymbol{D_z}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_y} \\
&\quad + \big(\boldsymbol{D_x D_z}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \boldsymbol{D_y}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_z} \\
\boldsymbol{V_y} &= \big(\boldsymbol{D_x D_y}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \boldsymbol{D_z}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_x} + \big(\boldsymbol{D_y}^2(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \mathbf{cos}\,\boldsymbol{\theta}\big)\boldsymbol{V_y} \\
&\quad + \big(\boldsymbol{D_y D_z}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) - \boldsymbol{D_x}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_z} \\
\boldsymbol{V_z} &= \big(\boldsymbol{D_x D_z}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) - \boldsymbol{D_y}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_x} + \big(\boldsymbol{D_y D_z}(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \boldsymbol{D_x}\,\mathbf{sin}\,\boldsymbol{\theta}\big)\boldsymbol{V_y} \\
&\quad + \big(\boldsymbol{D_z}^2(\mathbf{1}-\mathbf{cos}\,\boldsymbol{\theta}) + \mathbf{cos}\,\boldsymbol{\theta}\big)\boldsymbol{V_z}
\end{aligned}$$

We can now treat the hypothesis plane as simply the plane formed by connecting the x and y axis (the plane z = 0).

3.3.2. Ray casting and intersection

A set number of rays are then uniformly cast through the mesh along the z-axis. Due to the prior mesh rotation this has the effect of casting the rays through the mesh in the direction perpendicular to the hypothesis plane being tested. The origin points of the rays are set as one less than the lowest z-axis value of the mesh's vertices. The rays are linearly positioned along the x and y axis, determined by the newly rotated meshes bounding box. If a ray intersects with the mesh then the positions at which it intersects are recorded for use in calculating the symmetry measure. Intersections are determined using a simple ray-triangle intersection algorithm (see Figure 3.5) which calculates the distance $\boldsymbol{t}$ that the ray has travelled before each triangle

intersection (Choi 1995, Moller and Trumbore 1997). The ray is defined with an origin point $\boldsymbol{O}$ and a direction $\boldsymbol{D}$, with each triangle being defined in terms of the location of its corners $\boldsymbol{C_x}$, $\boldsymbol{C_y}$ and $\boldsymbol{C_z}$.

$$\boldsymbol{A = Dot\left(\overrightarrow{C_yC_x}, Cross\left(D, \overrightarrow{C_zC_x}\right)\right)}$$

$$\boldsymbol{U = \frac{1}{A} Dot\left(\overrightarrow{OC_x}, Cross\left(D, \overrightarrow{C_zC_x}\right)\right)}$$

$$\boldsymbol{V = \frac{1}{A} Dot\left(D, Cross\left(\overrightarrow{OC_x}, \overrightarrow{C_yC_x}\right)\right)}$$

$$\boldsymbol{t = \begin{cases} \frac{1}{A} Dot\left(\overrightarrow{C_zC_x}, Cross\left(\overrightarrow{OC_x}, \overrightarrow{C_yC_x}\right)\right) & if\ A \neq 0, U > 0, U < 1, V > 0, U + V < 1 \\ -1 & otherwise \end{cases}}$$

As the origin point of each ray is also known, it is a trivial calculation to determine the location of each intersection. These intersections are then split into two sets based on which side of the hypothesis plane they are on.

The efficiency of the ray-triangle intersection algorithm can be improved by first checking whether the ray being tested intersects any of the triangles' bounding boxes. Only if the ray intersects this bounding box will the normal ray-triangle intersection algorithm be carried out. This initial check is performed very quickly and as many of the triangles' bounding boxes will not intersect with the ray, this helps reduce the overall running time for the majority of models.

3.3.3. Ray casting symmetry measure

The total deviation $\boldsymbol{T}$ between the models on each side of the hypothesis plane is calculated based on the two sets of intersection points $\boldsymbol{A}$ and $\boldsymbol{B}$ for the set of all rays $\boldsymbol{R}$ and the hypothesis plane $\boldsymbol{P}$.

$$\boldsymbol{T = \sum_{r \in R} \begin{cases} max\left\{\sum_{a \in A} \min_{b \in B} d(a,b), \sum_{b \in B} \min_{a \in A} d(a,b),\right\} & if\ |A| > 0, |B| > 0 \\ \sum_{a \in A} d(a,P) & if\ |A| > 0, |B| = 0 \\ \sum_{b \in B} d(b,P) & if\ |B| > 0, |A| = 0 \\ 0 & if\ |A| = 0, |B| = 0 \end{cases}}$$

This means that for each ray there are three possible outcomes.

- No intersection points are found. The ray is ignored and no calculation is done.

- There are one or more intersection points on only one side of the hypothesis plane. The sum of the distances between each intersection point and the hypothesis plane is added to the total deviation.

- There are one or more intersection points on both sides of the hypothesis plane. The sum of the minimum distances between each of the points in one set and any point in the other set is calculated. The same is then done but with the two sets swapped. Whichever of these two "sums of minimum differences" is greater is then added to the total deviation.

Once all rays have been checked, the total deviation is divided by the number of rays which intersected the mesh. The inverse of this deviation is then used as a measure of symmetry ***S*** that the hypothesis plane has with respect to the original model. Much like the Hausdorff distance approach, if this value is above a pre-determined threshold we conclude that the hypothesis plane is likely to be a plane of reflective symmetry.

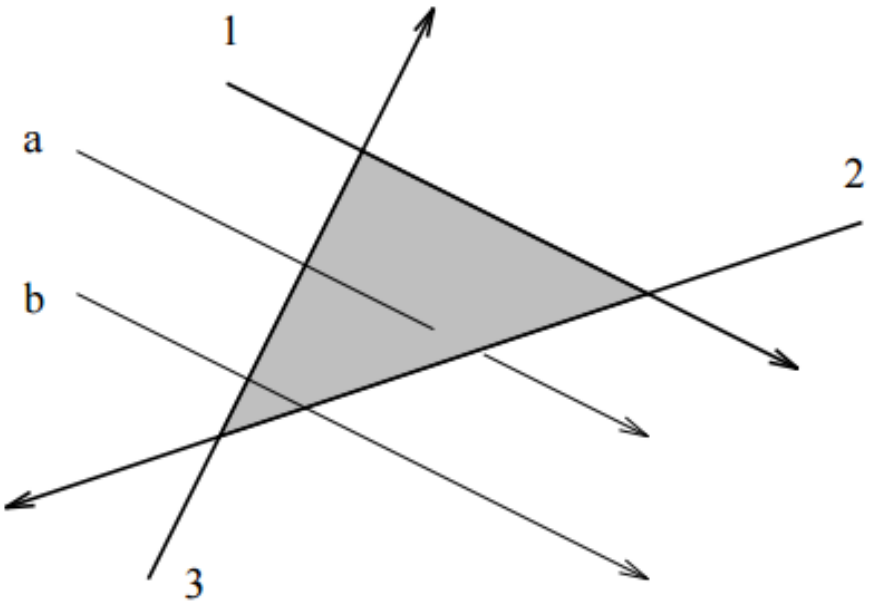


Figure 3.5: Ray 'a' intersects the triangle but ray 'b' does not
(Choi 1995)

3.4 Symmetry measure normalisation

Whilst both the Hausdorff distance and ray casting approaches calculate a measure of symmetry, this value is not normalised across models of different sizes. This is important for the determination of a suitable threshold to use when detecting approximate symmetry. The best way to normalise the symmetry measure is to multiply it by the cube root of the model's volume. This is very difficult to compute however, since many of the scanned models contain holes or other distortions. Instead, there are two main ways for estimating the volume of a model.

3.4.1 Bounding box

The easiest method for estimating the volume $\boldsymbol{E}$ of a model $\boldsymbol{M}$ with a set of vertices $\boldsymbol{V}$ is to simply use the volume of the mesh's bounding box.

$$\boldsymbol{E = \left(\underset{v\in V}{max}\, v_x - \underset{v\in V}{min}\, v_x\right)\left(\underset{v\in V}{max}\, v_y - \underset{v\in V}{min}\, v_y\right)\left(\underset{v\in V}{max}\, v_z - \underset{v\in V}{min}\, v_z\right)}$$

Whilst this method is both fast and simple it lacks accuracy, especially for models which do not sufficiently fill the bounding box.

3.4.2 Signed volume of a tetrahedron

A more complex alternative is to estimate the model's volume by calculating the signed volume of a tetrahedron based on each triangle $\boldsymbol{t}$ within the model(Cha and Tsuhan 2001). These individual volumes are then summed together and the absolute value of this is used as an estimate for the model's volume.

$$\boldsymbol{E = \left|\sum_{t\in M} Dot\ \left(v_a, \left(Cross\ (v_b, v_c)\right)\right)\right|}$$

(Note each triangle is made of three vertices $\boldsymbol{v_a}$, $\boldsymbol{v_b}$ and $\boldsymbol{v_c}$)

This second method generally provides a better estimate of the model's volume and was therefore chosen to normalise the symmetry measures.

For each model, the cube root of this volume estimate is multiplied by the symmetry measure to create a normalised value that could be compared against a symmetry threshold.

$$\boldsymbol{S_{normalised} = S\sqrt[3]{E}}$$

3.5 Additional variations

In the previous sections we have detailed the general frameworks for our two proposed symmetry detection algorithms. However, there are several additional variations that we have implemented which attempt to improve these methods further.

3.5.1 Polygon reduction

One of the main limitations with the two methods described is the large amount of time needed to analyse detailed models, particularly with the Hausdorff distance approach. In order to reduce the overall computation time we can reduce the number of polygons within the mesh before attempting symmetry detection. One of the main polygon reduction methods is to use quadric error metrics (Garland and Heckbert 1997). This simplifies the mesh by iteratively contracting edges until the desired number of vertices or faces remains (see Figure 3.6). The choice about which edge to remove is determined by approximating the error cost of each possible contraction between a pair of vertices. The algorithm then iteratively removes the pair with minimum cost, and updates any affected edges.

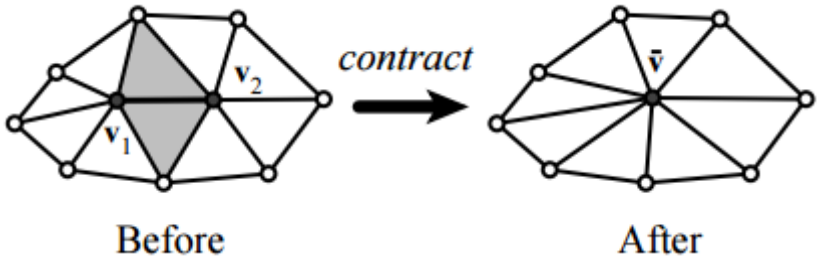


Figure 3.6: Edge contraction into a single point (Garland and Heckbert 1997)

The cost of contracting an edge is derived using quadrics, which are constructed by using a heuristic to characterise the geometric error. Firstly, the plane equation is determined for each triangle within the original model, defined by the equation $\boldsymbol{ax + by + cz = 0}$ where $\boldsymbol{a^2 + b^2 + c^2 = 1}$. The derived plane is then represented in the form $\boldsymbol{p = [a\ b\ c\ d]^T}$. The error for each vertex can then be defined with respect to the sum of squared distances to its intersecting planes.

$$\Delta(\boldsymbol{v}) = \Delta\left(\left[\boldsymbol{v_x\ v_y\ v_z\ 1}\right]^{\boldsymbol{T}}\right) = \sum_{\boldsymbol{p \in planes(v)}} (\boldsymbol{p^t v})^2$$

In addition, a matrix $\boldsymbol{K_p}$ is constructed for each triangle.

$$\boldsymbol{K_p = pp^T} = \begin{bmatrix} \boldsymbol{a^2} & \boldsymbol{ab} & \boldsymbol{ac} & \boldsymbol{ad} \\ \boldsymbol{ab} & \boldsymbol{b^2} & \boldsymbol{bc} & \boldsymbol{bd} \\ \boldsymbol{ac} & \boldsymbol{bc} & \boldsymbol{c^2} & \boldsymbol{cd} \\ \boldsymbol{ad} & \boldsymbol{bd} & \boldsymbol{cd} & \boldsymbol{d^2} \end{bmatrix}$$

Allowing the error metric to be written in a solvable quadratic form.

$$\Delta(\boldsymbol{v}) = \boldsymbol{v}^{\boldsymbol{T}}\left(\sum_{\boldsymbol{p}\in\boldsymbol{planes}(\boldsymbol{v})} \boldsymbol{K_p}\right)\boldsymbol{v}$$

The overall result of these edge contractions is that they can be used to greatly reduce the total number of computations required to detect symmetry within 3D models. This also potentially improves the algorithm's accuracy by making dense groups of vertices sparser, resulting in a more uniform spread of the model's vertices. However, too much simplification is likely to result in an increased error rate. A visual display of these reductions can be seen in Figure 3.7.

Figure 3.7: A collection of simplified models, the number
of faces reduced by approximately half each time (Garland and Heckbert 1997)

3.5.2 Laplacian smoothing

Another potential improvement which may improve our algorithms' accuracy is to smooth the mesh before performing symmetry detection. Whilst there are many different smoothing functions for 3D objects, the most common and simple of these is Laplacian smoothing. The Laplacian smoothing algorithm is a method designed to smooth a 3D polygonal mesh by changing the location of each vertex to the average of its adjacent vertices (see Figure 3.8). This can be achieved in O(n) time and space. The formal definition for the Laplacian smoothing operation can be defined per-vertex as,

$$\boldsymbol{p_i} = \frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{q_j}$$

Where $\boldsymbol{N}$ represents the number of vertices connected by an edge to the vertex $\boldsymbol{i}$ and $\boldsymbol{p_i}$ is the new position for vertex $\boldsymbol{i}$ based on the adjacent positions $\boldsymbol{q_j}$

There are two main variations for updating the positions of the vertices. The first updates the positions of the vertices in a single step and all vertices use the same original set of positions to update their locations. This is known as the simultaneous version. The second updates the positions of each vertex immediately after it is computed, meaning that early adjustments can influence later ones. This is known as the sequential version. Whilst the simultaneous version requires more memory space than the sequential version, it usually produces better results, meaning that this is the technique that is most commonly used.

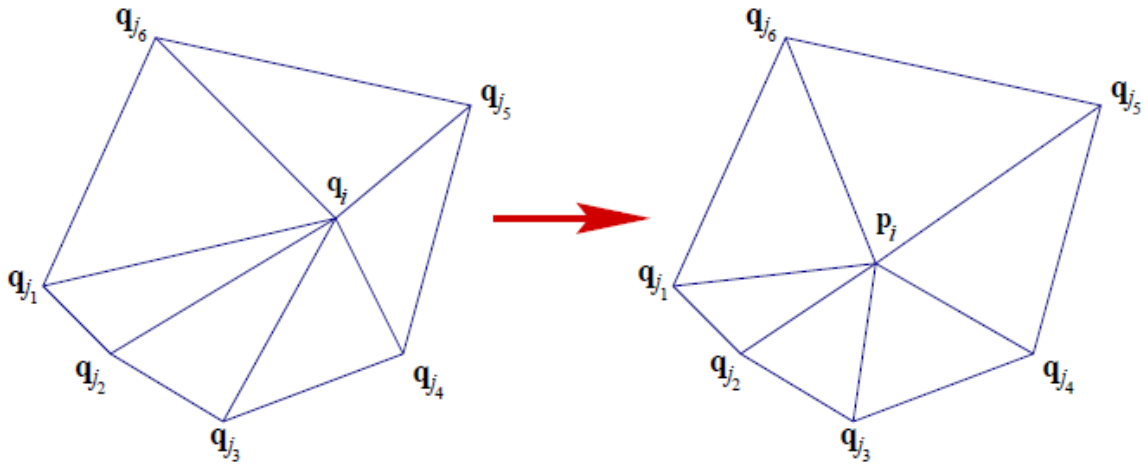


Figure 3.8: The basic Laplacian smoothing algorithm
(Vollmer, Mencl et al. 1999)

Whilst this algorithm is simple to understand and implement, it suffers from some key problems. The most significant is the deformation and shrinkage of the mesh after many repeated iterations. A popular variant of the Laplacian smoothing algorithm, referred to as the HC Laplacian algorithm, is designed specifically for noisy surface meshes and attempts to avoid these problems (Vollmer, Mencl et al. 1999). This algorithm reduces shrinkage by pushing the vertices that have been adjusted by the Laplacian smoothing iterations back towards their original location (see Figure 3.9). More specifically, the modified points $\boldsymbol{p_i}$ are moved towards the previous points $\boldsymbol{q_i}$ with a distance $\boldsymbol{d_i}$ equal to the average of the differences.

$$\boldsymbol{b_i = p_i - q_i}$$

$$\boldsymbol{d_i = -\frac{1}{N}\sum_{j=1}^{N} b_j}$$

Whilst this does not completely remove the problem of shrinkage it does dramatically reduce its effect, particularly for models that contain noise. This makes it an ideal candidate for testing the effects of smoothing a model before attempting to detect symmetry. Unlike polygon reduction, Laplacian smoothing only alters the positions of the model's vertices. This means that the time taken to analyse a smoothed model will be approximately the same as the original, although the smoothing takes a small amount of time. A visual comparison of the results of these two algorithms can be seen in Figure 3.10.

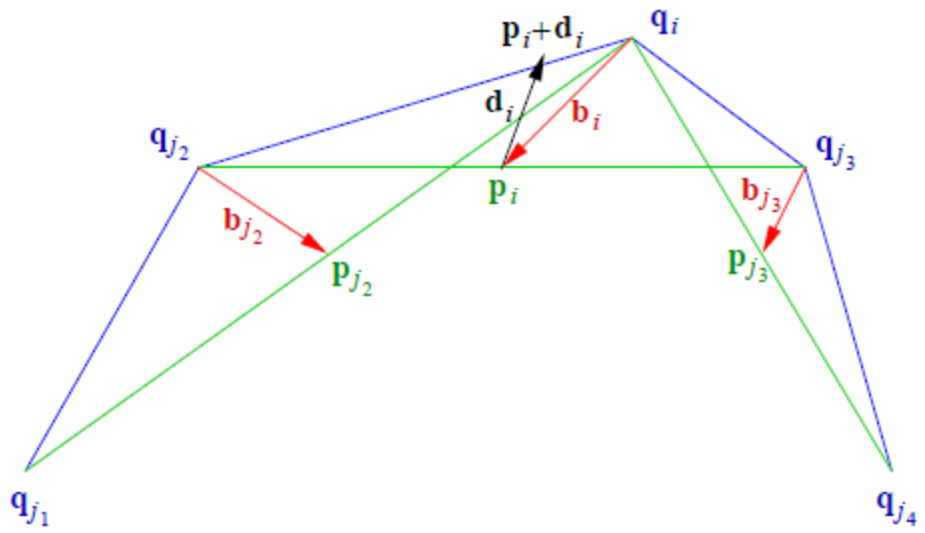


Figure 3.9: The HC Laplacian algorithm variation of the Laplacian smoothing algorithm (Vollmer, Mencl et al. 1999)

Figure 3.10: From left to right: original model, model smoothed using basic Laplacian smoothing algorithm, model smoothed using HC Laplacian smoothing algorithm

3.5.3 RANSAC

Although it has been demonstrated that using PCA to identify potential symmetry planes is robust against minor noise, for models with only approximate symmetry and a heavy leaning away from the most symmetrical plane, the PCA method may not provide the best estimation. There are no easy ways of resolving this without considerably reducing the efficiency of the overall method. However, if this is not a problem then using the RANSAC algorithm may reduce the influence of outliers. The RANSAC (RANdom SAmple Consensus) algorithm is a general method used to fit a model to data which is contaminated with gross outliers. In this case we can apply RANSAC to our method of identifying the PCA eigenvectors using the mesh's vertex set.

1. Firstly, the complete set of vertices is used for the determination of both the PCA eigenvectors and the symmetry measure for each hypothesis plane.
2. If no likely planes of reflective symmetry are found, then the PCA eigenvectors for the model are recalculated using a set percentage of randomly selected points.
3. The hypothesis planes for the new PCA eigenvectors are then calculated.
4. The symmetry measure for each of these planes is then determined using all the original vertices, not just those that were randomly selected.
5. If any of these planes are found to be likely planes of reflective symmetry then they are recorded, otherwise repeat from step 2.

While this is a very naïve and inefficient algorithm for improving symmetry detection, it works well in cases where computation time is not a major factor.

3.5.4 *k*-d tree

A *k*-d tree is a space partitioning data structure for organising points in *k*-dimensional space (Bentley 1975). Formally, a *k*-d tree is a binary partitioning where the the split operation is performed cyclically along each axis direction using axis-aligned hyper-planes.

For our method, we use a *k*-d tree to potentially improve the overall runtime of our ray casting approach by reducing the time taken to find ray-triangle intersections. Although we have already improved the runtime slightly by first performing an intersection test for each ray using the triangles' bounding boxes, using a *k*-d tree may improve this even more. In this case we set k=2 as we do not need to partition the mesh along the z-axis, as this is the direction along which the rays are cast.

The *k*-d tree is constructed by recursively splitting the sets of vertices (initially all vertices are in one set) into two smaller subsets based on the median of either the x or y value of their positions (the axis to split on is swapped for each iteration). Each iteration effectively doubles the number of vertex sets and once a desired depth is reached the algorithm halts (see Figure 3.11).

This *k*-d tree can now be used during ray casting to reduce the total number of ray-triangle intersection tests that need to be carried out. Firstly, we determine which of the *k*-d tree subsets the ray will intersect. We then perform our regular intersection algorithm but only for the triangles which have at least one of their vertices within the subset that the ray intersected. The triangles that each *k*-d tree subset is associated with are determined before any rays are cast to prevent repeat calculations. This method may be used alongside, or instead of, the original bounding box improvement.

There is a small issue with this method however, which may decrease the accuracy of the symmetry detection. It is entirely possible (especially for implementations where the depth of the *k*-d tree is high) for a situation to arise where none of a triangle's vertices are located within a particular subset but part of the triangle is. This results in an inaccurate reading for any rays that may pass through this portion of the triangle. This particular problem is demonstrated visually in Figure 3.12. Whilst there are more sophisticated methods for correctly identifying

which subsections intersect the triangle, they take much longer to run, counteracting any potential runtime reduction. This means that a *k*-d tree should only be used in situations where time is a critical factor and should ideally be paired with a low number of rays being cast.

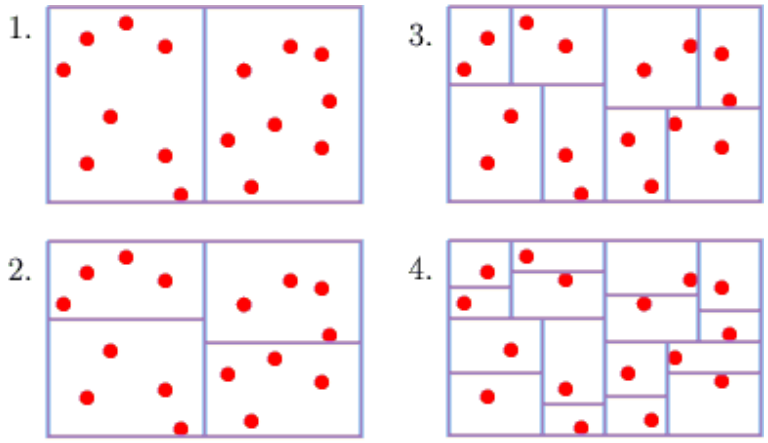


Figure 3.11: *k*-d tree partition over subsequent iterations (Standford 2000)

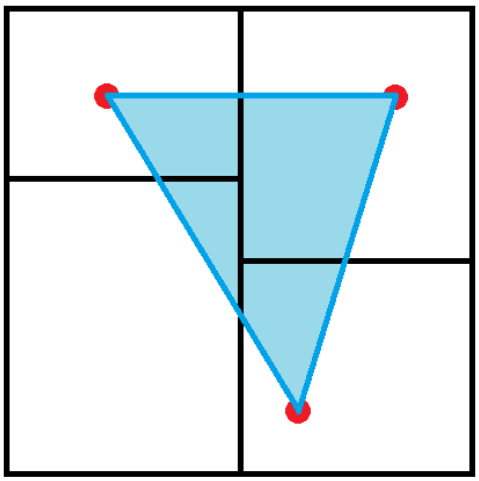


Figure 3.12: The problem with *k*-d tree method, the triangle's vertices are only located within three sections but the triangle actually overlaps four sections

3.5.5 Non-uniform casting

Another potential variation to the ray casting method is to improve upon the regular uniform casting to a more advanced non-uniform method. There are many different ways to perform this but we will only focus on one of them.

One of the simplest methods to improve upon uniform ray casting is to base the distribution of the rays on the distribution of the model's vertices. This is performed by spreading the rays along the model, based on the number of vertices rather than the total length (see Figure 3.13). This results in the algorithm obtaining a greater level of information for the sections of the model which contain more vertices and less information about the sections with fewer vertices.

In context, this is achieved by first sorting the vertices into order along the x and y axis (after the vertices and plane have been orientated correctly). Each of these sorted sets (one for x-axis ordering $\boldsymbol{X}$ and the other for y-axis ordering $\boldsymbol{Y}$) is then split into subsets based on the number of rays $\boldsymbol{R}$ that are being cast along each axis. The starting position for the rays are then determined by taking the first value of each set for all possible pairings of the x-axis ordered subsets and y-axis ordered subsets.

$$\boldsymbol{x\text{-}axis\ positions} = \boldsymbol{X_p} = \left\{\boldsymbol{x}: \boldsymbol{x} = \boldsymbol{X}_{\boldsymbol{n}\frac{|\boldsymbol{X}|}{\boldsymbol{R}}}\ \boldsymbol{for\ all\ n\ between\ 0\ and\ R-1}\right\}$$

$$\boldsymbol{y\text{-}axis\ positions} = \boldsymbol{Y_p} = \left\{\boldsymbol{y}: \boldsymbol{y} = \boldsymbol{Y}_{\boldsymbol{n}\frac{|\boldsymbol{Y}|}{\boldsymbol{R}}}\ \boldsymbol{for\ all\ n\ between\ 0\ and\ R-1}\right\}$$

$$\boldsymbol{Non\text{-}uniform\ ray\ positions} = \left\{(\boldsymbol{x},\boldsymbol{y})\ \boldsymbol{where}\ \boldsymbol{x} \in \boldsymbol{X_p}\ \boldsymbol{and}\ \boldsymbol{y} \in \boldsymbol{Y_p}\right\}$$

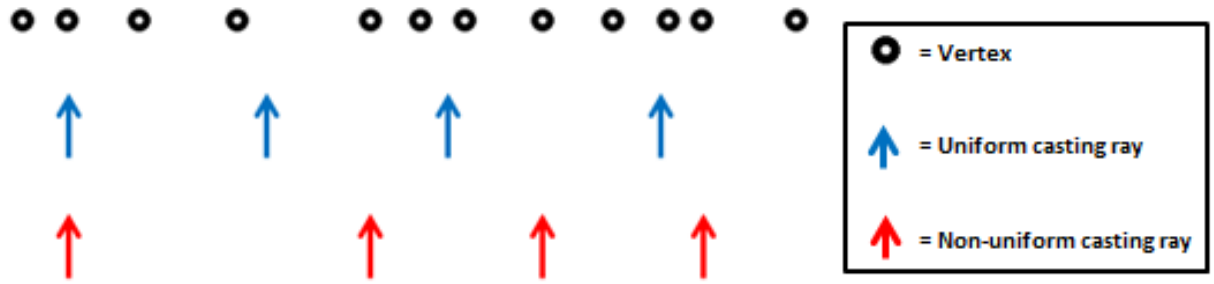


Figure 3.13: Comparison of uniform ray casting and non-uniform ray casting for a set of vertices in one direction

3.6 Summary

Within this chapter we have detailed the design and implementation of two general algorithms for detecting global approximate reflective symmetry within scanned 3D models (Hausdorff and ray casting), as well as several additional variations which attempt to improve their runtime and/or accuracy (polygon reduction, Laplacian smoothing, RANSAC, k-d tree, non-uniform casting). The next chapter details our evaluation of each method and variation.

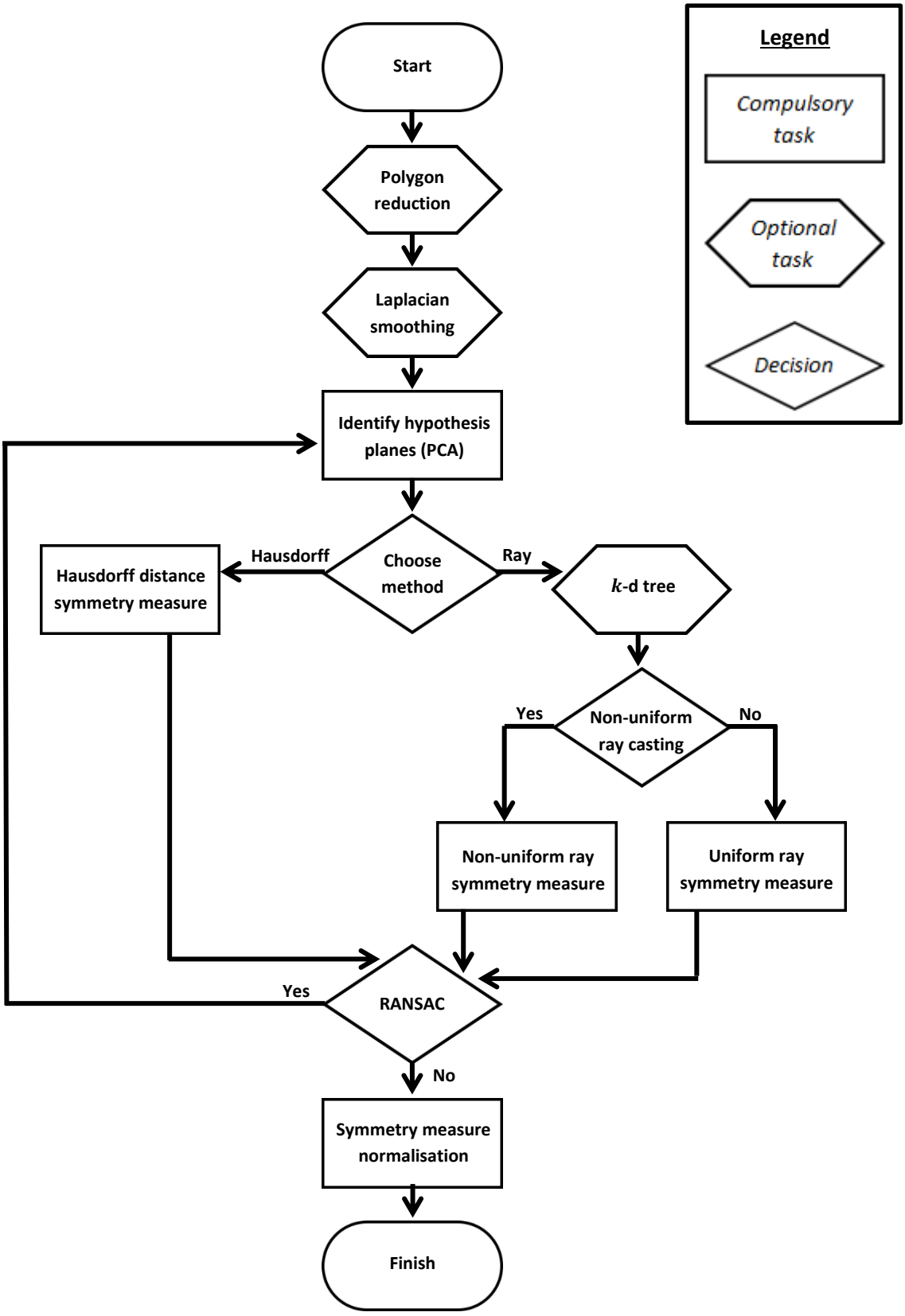


Figure 3.14: Flowchart of the entire program

4 Results

4.1 Experimental design

The testing of both the Hausdorff distance and ray casting methods, as well as all applicable variations, was initially performed using the Princeton Shape Benchmark (PSB) (Shilane, Min et al. 2004). This is a database containing 1814 3D polygonal meshes and has been used previously to test many model analysis programs. The models within this database vary greatly in terms of their size, detail and of course their symmetry.

After this, more specific tests were conducted using a smaller collection of 32 scanned 3D models (all of which can be viewed in the appendix). These models were obtained using the scanning program 123DCatch (Catch 2015). This set of models was also used to determine a suitable threshold for the symmetry measure of each method.

The lack of available source code for other prior methods made it difficult to compare our algorithms accuracy and runtime against them. As a result, we have only been able to compare our implementations against each other, for various different inputs and parameters.

Testing was performed on a machine running Windows 8.1 with an i7-4690 processor and 16GB of RAM. Both algorithms were developed in C++ using Microsoft Visual Studio 2013.

4.2 Overall accuracy

Whether an object has approximate symmetry depends very much on the desired level of accuracy. There is no mathematical definition of approximate symmetry and it is generally left for the user to decide whether the symmetry is sufficient enough for their purpose. Both our methods demonstrated 100% accuracy when applied to models containing perfect symmetry, but the accuracy for approximate symmetry detection is more difficult to quantify (see Figure 4.1). How much each half of a perfectly symmetrical model may differ before it is no longer considered approximately symmetrical is ultimately dictated by the desired application.

To gain a better measure of accuracy, each method was used to detect global reflective symmetry within the collection of 32 scanned 3D models. All of the real-world objects used in these scans had a high level of approximate reflective symmetry, although the models were distorted slightly by the scanning process. For each model, three hypothesis planes were identified by PCA and the correct plane of reflective symmetry was determined manually.

Each of our symmetry detection methods were then applied to each of the hypothesis planes identified by PCA. Each of these planes was then given a symmetry measure representing the level of reflective symmetry the model has with respect to this plane. The symmetry measure for the correct plane of reflective symmetry was then compared against the values for the other two incorrect planes. For the ray casting approach several variations on the number of rays cast were tested (25,100, 400 and 2500 rays).

The graphs for the Hausdorff distance and the ray casting variant with 400 rays are shown in Figures 4.2 and 4.3 respectively.

We have decided to show only certain key graphs for each results section. Graphs that are not given here can be found in Appendix B. (Note, although the data obtained in these experiments was discrete it is easier to understand and compare the methods if line graphs are used).

(a) (b)

Figure 4.1: Although no official definition is available we would typically consider model (a) to contain approximate reflective symmetry, but not model (b)

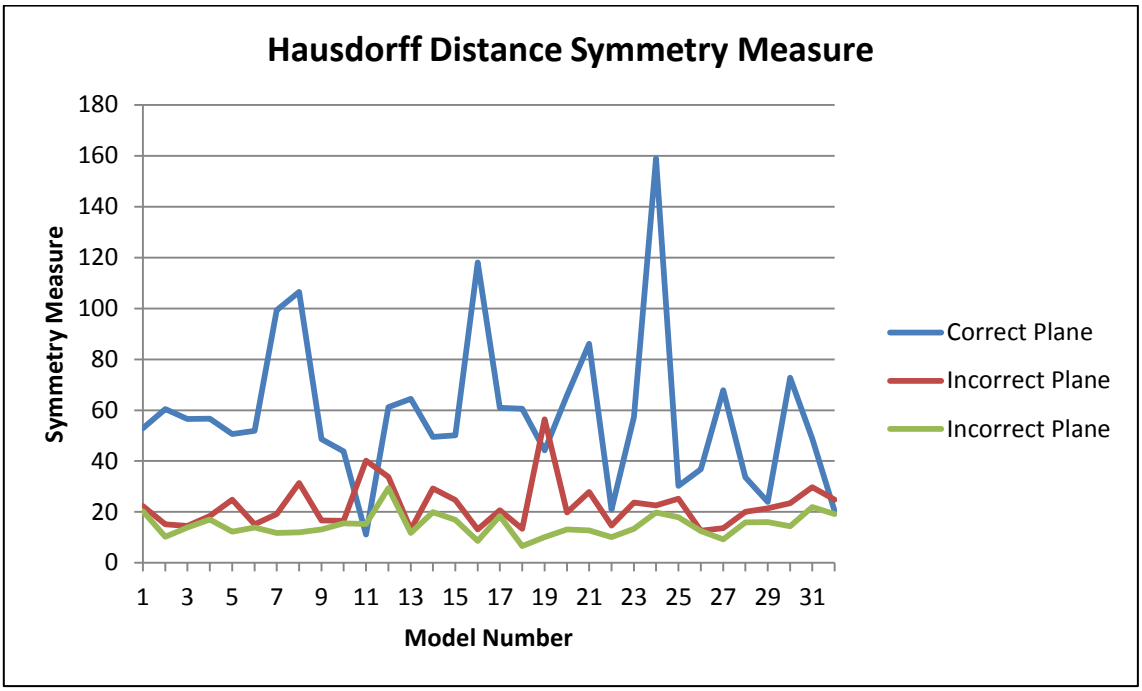


Figure 4.2: The reflective symmetry measure given to each model's hypothesis planes using the Hausdorff distance based method

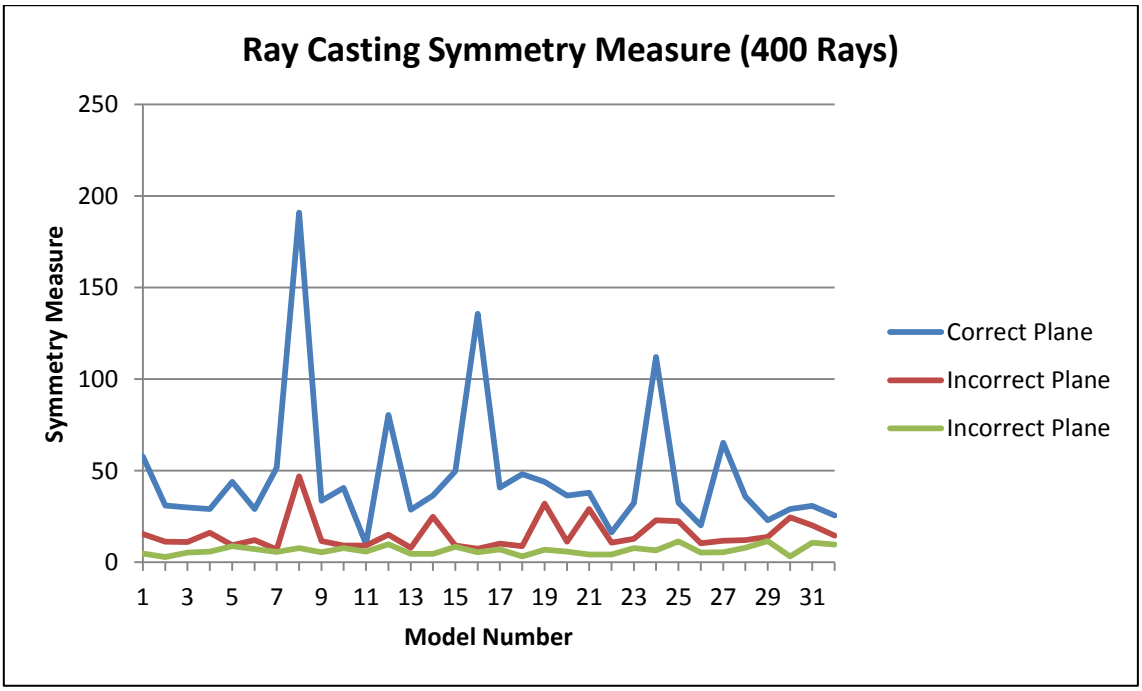


Figure 4.3: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method with a 20x20 grid of rays

The ideal result from these experiments would be to have all the values for the symmetry measure of the correct plane greater than any value from either of the other two incorrect planes (no overlap between these sets). This would allow us to easily derive a threshold value which could then be used to identify symmetry planes in unknown models. In order to determine an approximate level of overlap we can calculate the index of dispersion $\boldsymbol{D}$ for the difference between the symmetry measure of the correct hypothesis plane $\boldsymbol{C}$ and the greatest incorrect plane $\boldsymbol{I}$. A lower $\boldsymbol{D}$ value means less overlap and improved accuracy (see Figure 4.4).

$$\boldsymbol{D} = \frac{\boldsymbol{\sigma^2}}{\boldsymbol{\mu}}$$

$$\boldsymbol{\mu} = \frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{a=1}}^{\boldsymbol{N}} \boldsymbol{C_a} - \frac{\boldsymbol{1}}{\boldsymbol{N}}\sum_{\boldsymbol{b=1}}^{\boldsymbol{N}} \boldsymbol{I_b}$$

$$\boldsymbol{\sigma^2} = \boldsymbol{var(C)} + \boldsymbol{var(I)} + \boldsymbol{2(cov(C, I))}$$

$$\boldsymbol{var(X)} = \frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{(x_i - \mu)^2}$$

$$\boldsymbol{cov(X, Y)} = \frac{\boldsymbol{1}}{\boldsymbol{n^2}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{(x_i - x_j)} \cdot \boldsymbol{(y_i - y_j)^T}$$

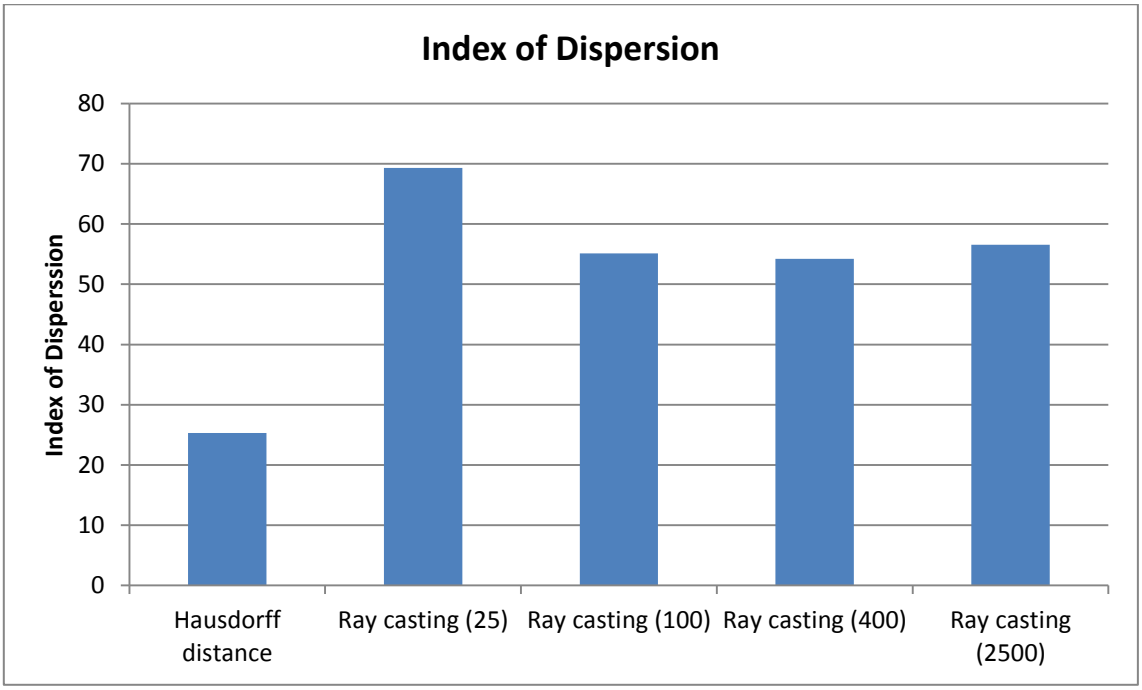


Figure 4.4: The index of dispersion for each method

The Hausdorff distance method gave the lowest index of dispersion although it did have several instances where the correct plane received a lower measure than an incorrect plane. The first of the ray casting algorithms, with only 25 rays, has a higher index of dispersion than the other three ray casting variations. This is likely due to the low number of rays that were cast, resulting in the algorithm not having sufficient data to make a good estimate of reflective symmetry. The other three ray casting algorithms all produced very similar results, indicating that it is unlikely that the estimates could be improved further by increasing the number of rays. As a result of this similarity, the 400 ray variant was chosen as a baseline for the potential improvements as it had the lowest index of dispersion. It is likely that any of the proposed enhancements would perform similarly with the 100 and 2500 ray variants. For the ray casting approach model 8 was a clear outlier, with a very large symmetry measure for one of its incorrect planes. This was due to our ray casting method failing to sample the model effectively and is further discussed in Section 5.3. Model 8 was therefore removed for subsequent calculations of the index of dispersion to give a better comparison between the ray casting and Hausdorff distance approaches (see Figure 4.5).

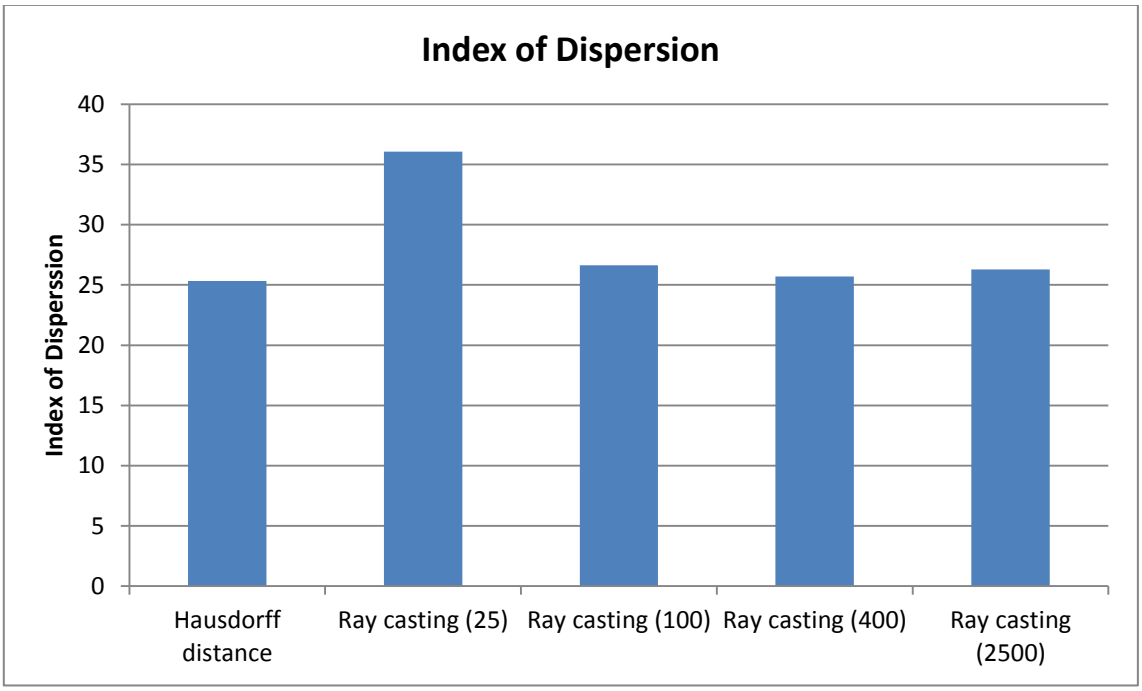


Figure 4.5: The index of dispersion for each method with model 8 removed for ray casting

4.3 Overall runtime

The speed of each of our methods varies depending upon different factors. For the Hausdorff distance method the runtime of the algorithm increases relative to the number of vertices the model has. For the ray casting method the runtime of the algorithm increases relative to both the number of faces the model has and how many rays are cast through it. Fortunately, the relationship between the number of vertices and number of faces within a model is typically very linear, with the number of faces approximately double the number of vertices. This allows us to compare the runtime of both methods against the number of vertices within the model, making it easier to compare each method's runtime (see Figures 4.6 and 4.7).

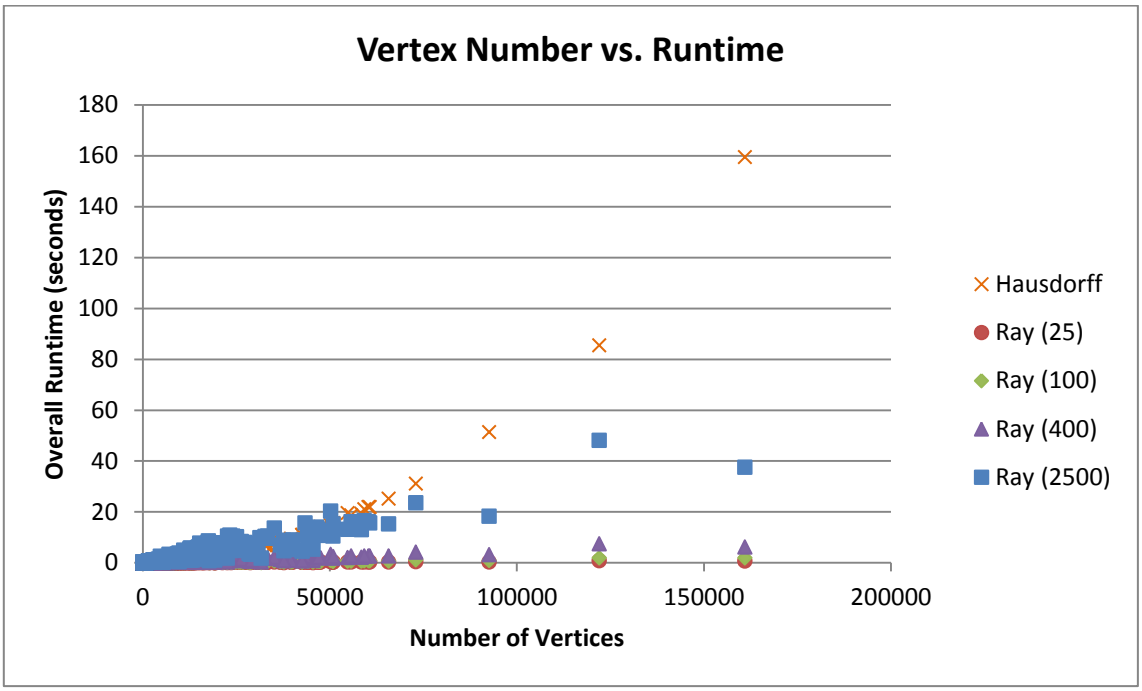


Figure 4.6: The total runtime for each algorithm relative to the number of vertices in the model for the all models within the Princeton Shape Benchmark

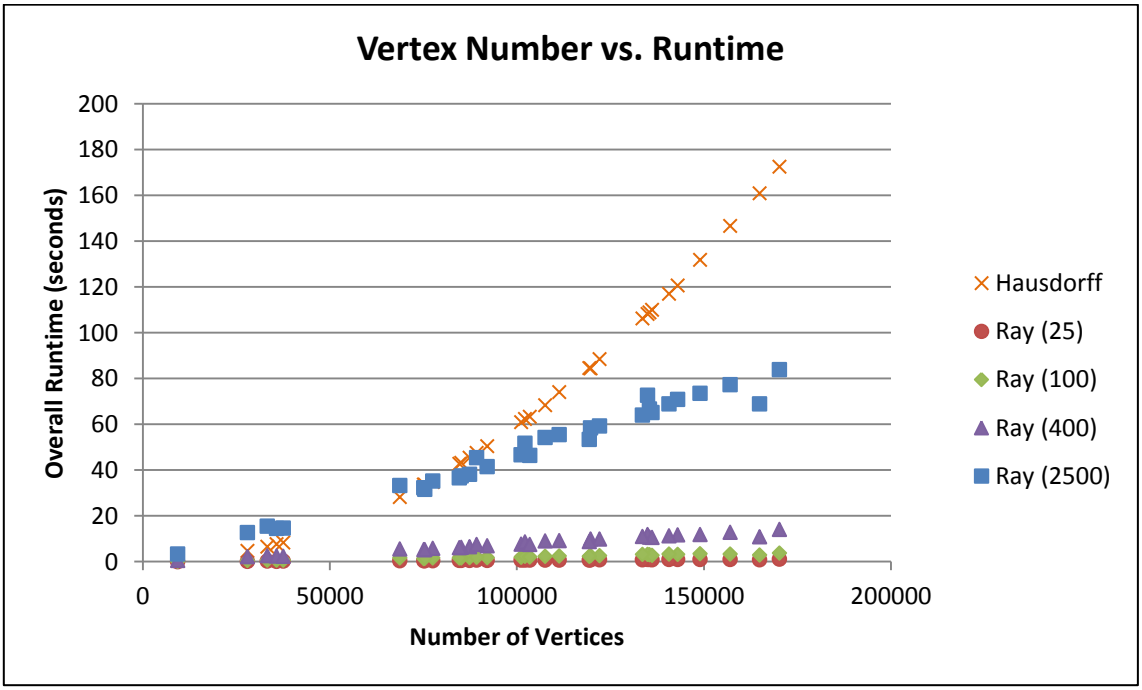


Figure 4.7: The total runtime for each algorithm relative to the number of vertices in the model for the set of scanned models

The previous graphs indicate that the time complexity for the Hausdorff distance method is approximately $O(n^2)$, whereas the time for each of the ray casting algorithms is closer to linear. Even the slowest ray casting method tested (50x50 rays) becomes faster than the Hausdorff distance method for a relatively low number of vertices (approximately 80,000 vertices). Variations using a lower number of rays become faster than the Hausdorff distance method for an even lower number of vertices. This means that for a large majority of models the ray casting approach is quicker than the Hausdorff distance approach, assuming the number of rays cast is sufficiently low. There also appears to be more variation in the runtime of the ray casting methods when compared to the Hausdorff distance method.

4.4 Additional Variations

4.4.1 Polygon reduction

Polygon reduction can be used to dramatically reduce the overall runtime of both our symmetry detection methods. However, if the amount of polygon reduction is too large the accuracy of our algorithms will suffer. Also, as many of the models within the PSB were designed to have a very low number of faces, polygon reduction would not be suitable. Because of this, the results for polygon reduction were only obtained using the set of 32 scanned models. Polygon reduction was tested using both 50% reduction (see Appendix B.) and 90% reduction (see Figures 4.8 and 4.9). The index of dispersion (see Figure 4.10) and runtime (see Figure 4.11) was also recorded.

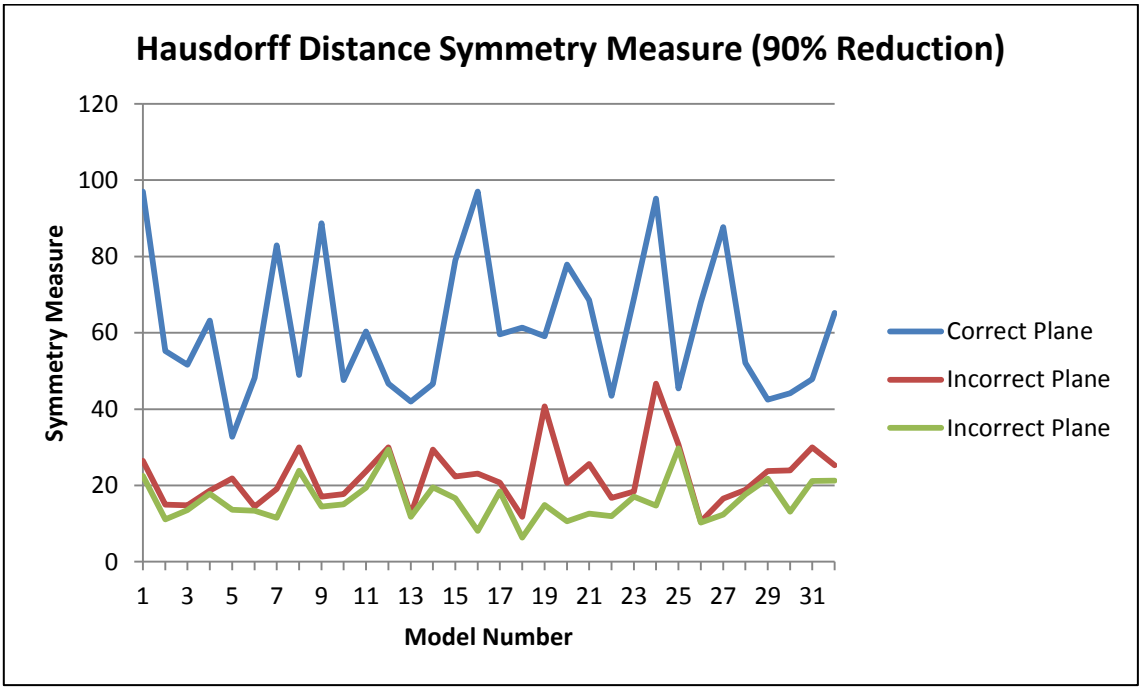


Figure 4.8: The reflective symmetry measure given to each model's hypothesis planes using the Hausdorff distance based method after the model has had 90% of its polygons removed

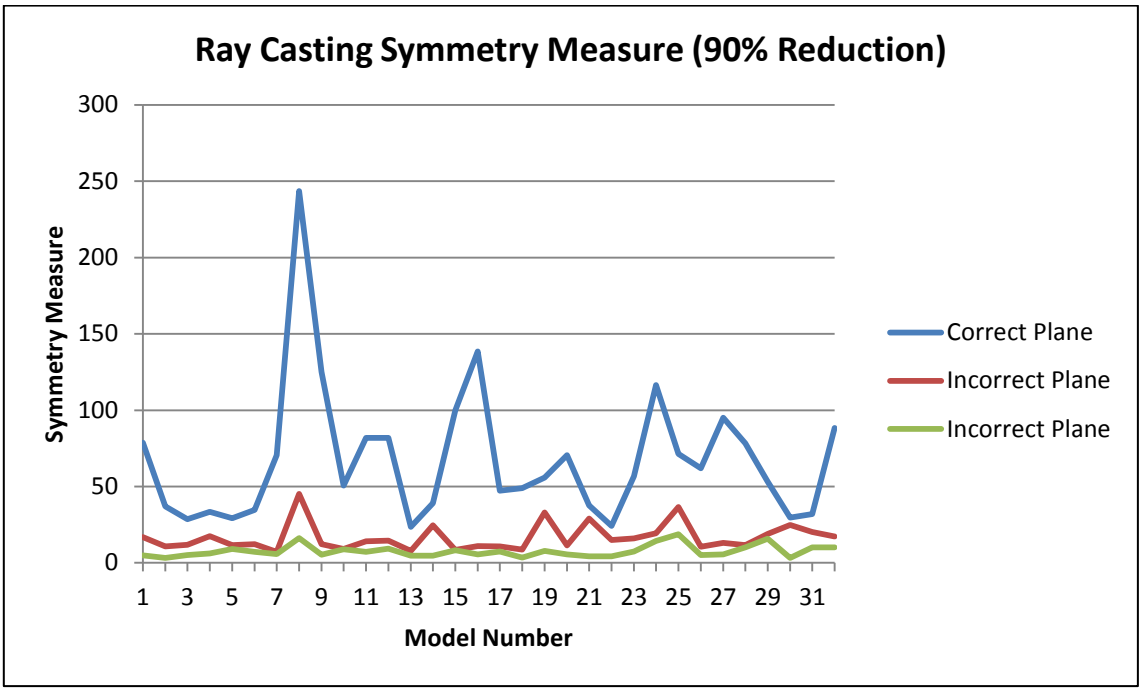


Figure 4.9: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method after the model has had 90% of its polygons removed

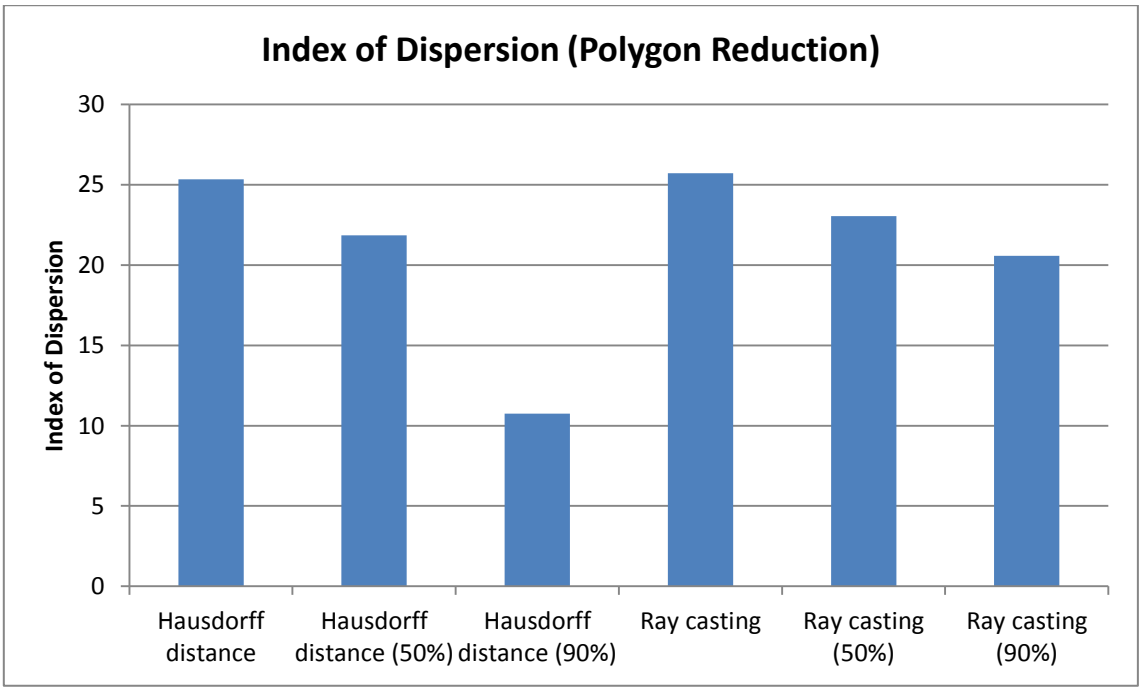


Figure 4.10: The index of dispersion for each method with and without polygon reduction

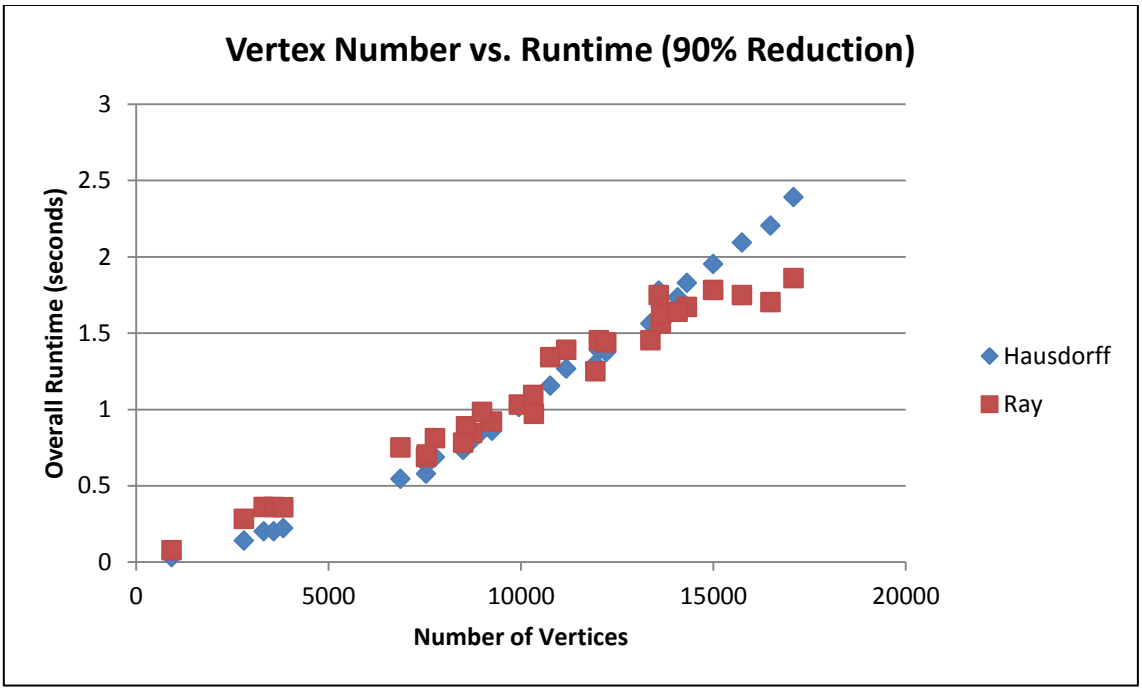


Figure 4.11: The total runtime for each method relative to the number of vertices in the model after it has had 90% of its polygons removed

From these results we can see that polygon reduction not only dramatically reduces the runtime of both methods (see Table 4.1) but also improved their accuracy for a large proportion of the models. 90% polygon reduction appears to be the best choice for both time and accuracy, with the lowest speed and index of dispersion for both methods. Whilst some extra time was taken to reduce the models beforehand, the time that was needed to analyse the reduced meshes was significantly reduced. This reduction in the number of polygons also has the effect of making the spread of the model's vertices more uniform, reducing the influence of any large vertex clusters or irregular sampling. This had a major effect not only on the accuracy of the Hausdorff distance method but also for the ray casting approach (although the impact was less pronounced). Although the accuracy and speed of both algorithms has been improved by polygon reduction, there are other potential improvements that may provide even more benefits.

	Original	50% reduction	90% reduction
Hausdorff distance	70.53	18.09	1.18
Ray casting (400)	7.92	4.16	1.11

Table 4.1: Average runtime (sec) for symmetry detection using scanned models in sample dataset

4.4.2 Laplacian smoothing

HC Laplacian smoothing was applied to each of the scanned models after they had been subject to 90% polygon reduction to see if this would provide even better accuracy. Laplacian smoothing is extremely quick and thus has little impact on the overall runtime of each method. Unfortunately, after comparing the results for Laplacian smoothing against the original data, it would appear that the improvements (if any) are minimal (see Figure 4.12).

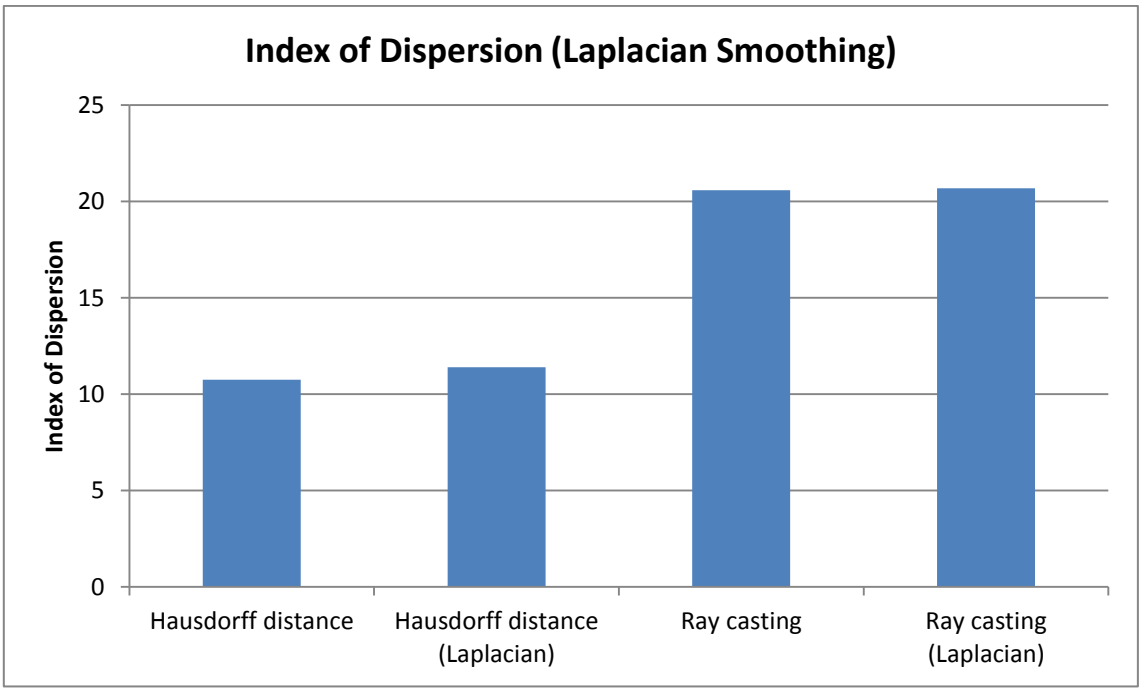


Figure 4.12: The index of dispersion for each method with and without Laplacian smoothing

4.4.3 RANSAC

Using RANSAC to improve the identification of hypothesis planes by PCA is an inefficient and often unnecessary variation, due to the large amount of time that it takes with no guarantee of improvement. It is therefore only advised for situations where the time taken is secondary to the detection accuracy. As RANSAC selects the vertices randomly, it is difficult to obtain any meaningful results from experimentation. Although RANSAC can improve symmetry detection accuracy, the time taken to find any improvements is always unknown. For each RANSAC iteration the base algorithm is started again from the beginning, greatly increasing the overall runtime (ten RANSAC iterations means the runtime is approximately ten times longer).

4.4.4 k-d tree

Using a k-d tree to divide up the model's faces before performing ray casting can potentially decrease the overall runtime, assuming the depth for the k-d tree is chosen correctly. The maximum depth that a k-d tree can theoretically have for a model with $\boldsymbol{V}$ vertices is between $\sqrt{\boldsymbol{V}}$ and $\boldsymbol{V}$. However, as our model's vertices are located in three dimensions and our k-d tree is only constructed in two dimensions the depth will likely be less than this. This is due to the fact that some of the model's vertices will have the same x and y axes coordinates but have a different z-axis value. This makes it difficult to establish a perfect depth that the k-d tree should have, even if the number of vertices is known. To gain a good estimate however, many different depths were

tested for a small collection of the scanned models (see Figures 4.13 and 4.14). Ray casting was performed using 400 rays.

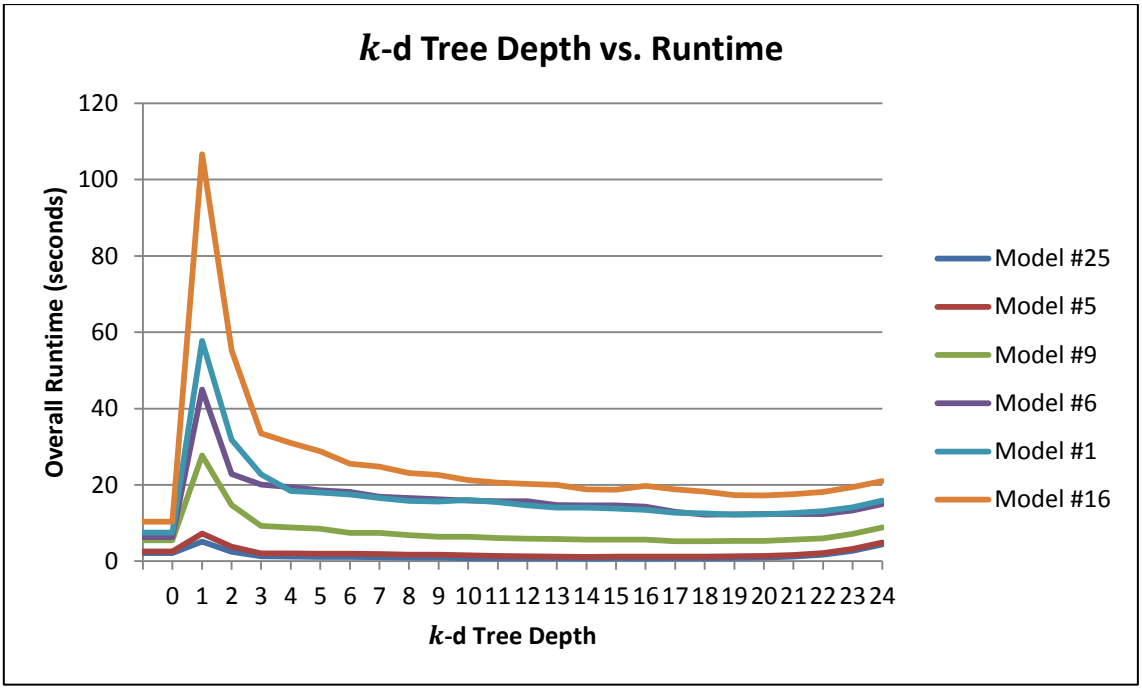


Figure 4.13: The total runtime for each model relative to the depth of the k-d tree

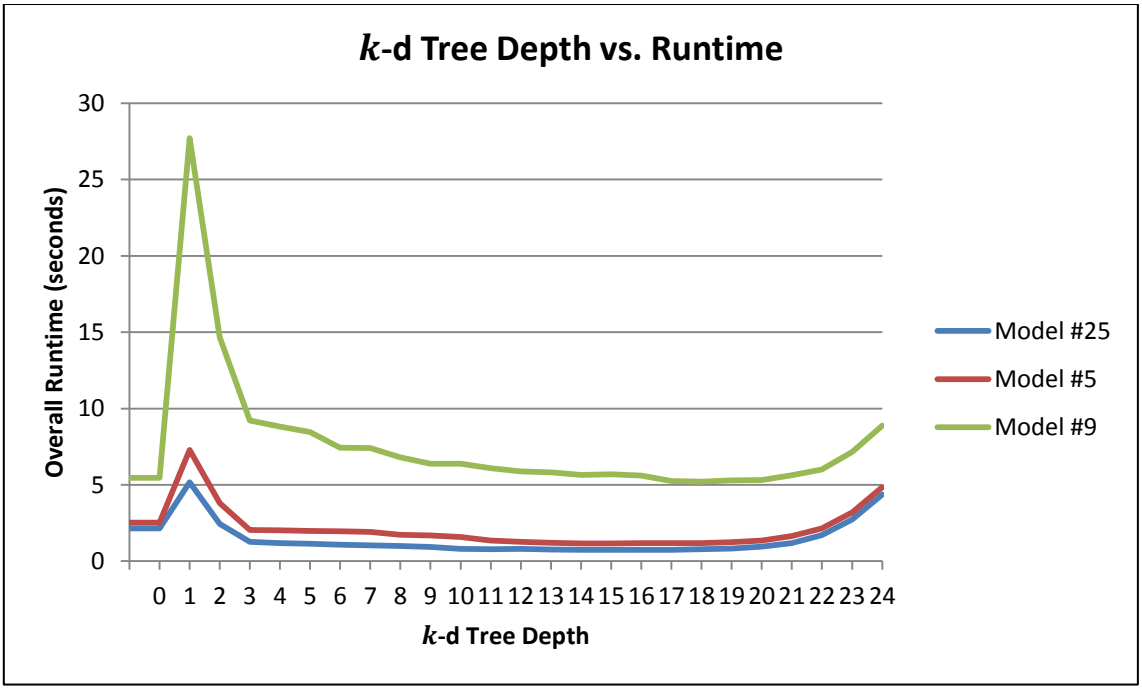


Figure 4.14: The total runtime for each of the smaller models relative to the depth of the k-d tree

Only the smaller models (less than 70,000 vertices) showed any improvement with respect to the original method (k-d tree depth equal to zero). The decrease in their runtime was also fairly small, with the peak improvement typically between depth = 10 and depth = 20. This means that whilst using a k-d tree can potentially reduce the algorithm's overall runtime for smaller models, it usually requires more time if used on larger models. This, coupled with the potential decrease in accuracy, means that using a k-d tree is generally not advisable.

4.4.5 Non-uniform casting

Using the distribution of the vertices to determine the positions from which to cast rays may also potentially improve the accuracy of the ray casting method. The ray casting method with 400 rays was tested using non-uniform casting to see if there was any improvement in accuracy (see Figures 4.15 and 4.16). While this new casting method does take slightly more time than uniform casting as the vertices need to be sorted beforehand, the impact of this on the total runtime is negligible.

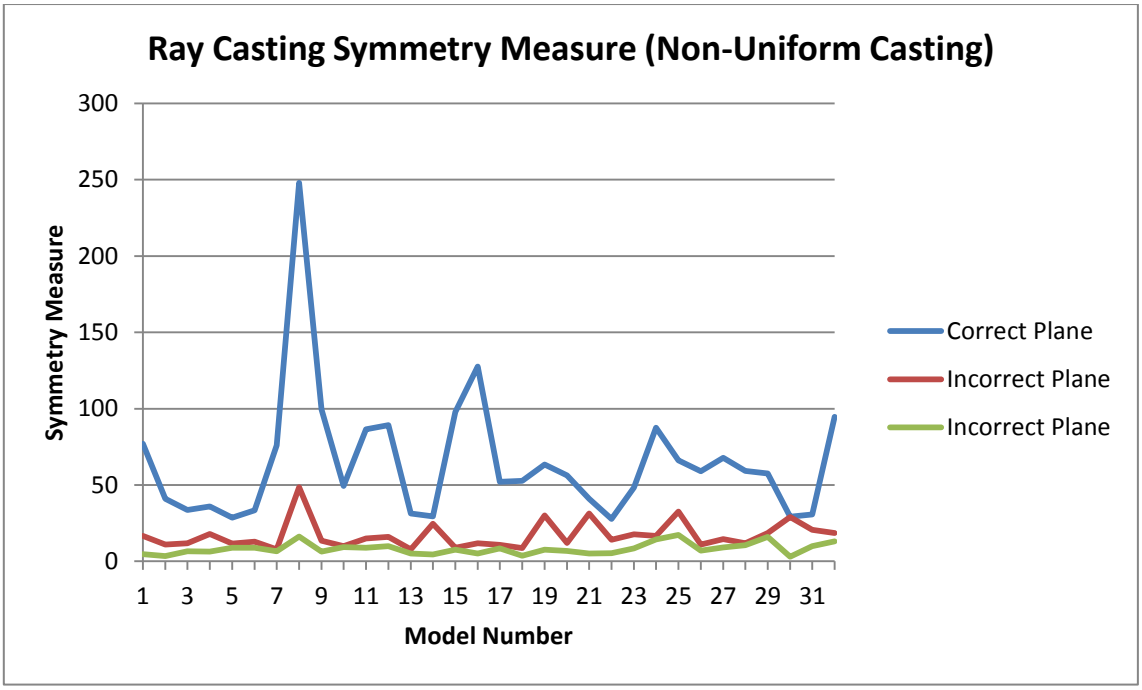


Figure 4.15: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method with non-uniform casting after the model has been reduced

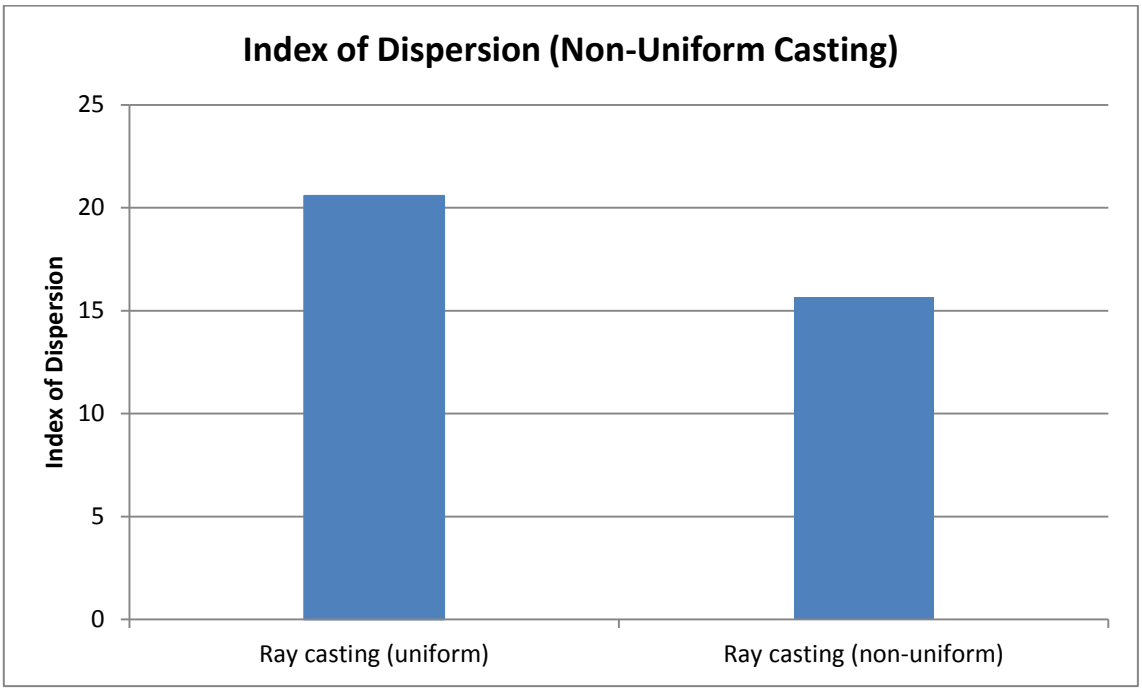


Figure 4.16: The index of dispersion for the ray casting method with uniform and non-uniform casting

From these results we can see that using non-uniform casting does appear to slightly improve the accuracy for the ray casting method. This is because non-uniform casting results in a greater focus on the sections of the model with more vertices. These sections are likely to contain more information about the model's symmetry allowing our algorithm to derive a more accurate symmetry measure.

4.4.6 Additional variations discussion

Out of the five additional variations tested only polygon reduction was shown to consistently improve both the runtime and accuracy of our methods. Non-uniform casting did not improve our runtime but did slightly improve our accuracy. The results of the Hausdorff distance and ray casting (20x20) methods, with 90% polygon reduction and non-uniform casting, were therefore used to determine a suitable threshold for reflective symmetry.

4.5 Threshold determination

In order to gain a measure of the accuracy of each method it is necessary to determine a suitable threshold for the symmetry measure of each method. This threshold value can then be used to identify reflective symmetry planes in unknown models. If the symmetry measure for a plane is

above this threshold then we classify it as a plane of global approximate reflective symmetry. For the ray casting approach model number 8 was removed as an outlier in order to calculate a more suitable threshold (as previously mentioned). The reasons for this anomaly are discussed in Section 5.3.

The thresholds calculated are based on the assumption that the cost of a false positive is the same as the cost of a false negative. These thresholds should therefore be tailored by the situation and conditions they are applied to. It is important to note that both the Hausdorff distance and ray casting methods (with 90% polygon reduction) always gave the correct plane of reflective symmetry a higher value than either of the incorrect planes. This means that if the user knows that there is a plane of reflective symmetry within a model our methods can determine this plane with a very high level of certainty.

The range of possible values was determined for each method, between the lowest value for the correct plane and the highest value of an incorrect plane.

Hausdorff Distance Method:

- Minimum Correct symmetry measure= 32.75
- Maximum Incorrect symmetry measure = 46.70
- Mid-range = 39.7
-

Ray Casting Method:

- Minimum Correct symmetry measure = 27.69
- Maximum Incorrect symmetry measure = 32.69
- Mid-range = 30.19

By using the mid-range value we can now determine the number of planes that would be misclassified if this was used as the threshold.

Hausdorff Distance Method:

- Number of misclassifications = 3
-

Ray Casting Method:

- Number of misclassifications = 6

The average symmetry measure of these misclassifications should be suitable as a threshold for each method.

Hausdorff Distance Method:

- Threshold ≈ 40
- **Accuracy per plane = 96.88%**

Ray Casting Method:

- Threshold ≈ 29
- **Accuracy per plane = 94.62% (without model 8)**
- **Accuracy per plane = 93.75% (with model 8)**

5 Discussion

After investigating these two methods and their possible variations we can see that they are both effective at detecting reflective symmetry within scanned 3D models. The limitations and outliers for each method as well as their overall accuracy and runtime are discussed in this section.

5.1 Principal component analysis limitations

Although PCA correctly identified (within a small margin of error) the reflective symmetry plane for all the scanned models tested, occasionally the symmetry plane for a model is not among those identified by PCA. This is usually the case for models with only a small level of approximate symmetry. An example of this is shown in Figure 5.1. The chair shown here has a large clustering of vertices in the height adjustment lever. This causes PCA to calculate eigenvectors that are more directed towards this lever, causing the algorithm to misidentify the ideal plane of reflective symmetry. This issue is very rare however and should not be a problem on good scans of symmetrical objects (as demonstrated by the models used in these experiments).

Figure 5.1: Model of chair, PCA identifies incorrect hypothesis planes

5.2 Hausdorff distance method limitations

As mentioned previously, the Hausdorff distance method is less efficient than the ray casting approach when dealing with larger models. It is also not suitable when the sampling resolution of the model is inconsistent on either side of the symmetry plane. This problem is largely present within the results for the Hausdorff distance method without polygon reduction. The most noticeable case where this occurs is for model 11. For this model the correct plane is given a symmetry measure lower than both of the other two incorrect planes. A brief analysis of this model demonstrates why this occurs (see Figure 5.2).

For model 11 the number of vertices on each side of the face differs significantly. This has a very large effect on the Hausdorff distance approach as it relies on the number of vertices (sampling resolution) to be roughly equal on each side of the reflective symmetry plane. Polygon reduction helps to alleviate this problem although the benefit obtained depends very much on the topology of the original model.

The Hausdorff distance approach also lacks the flexibility to adapt its speed and resolution to the desired situation, unlike the ray casting approach. It can however be used on models that consist only of vertices (e.g. point cloud models) as it does not require the mesh's faces to identify reflective symmetry.

Figure 5.2: The two halves of model 11, the left half contains far fewer vertices than the right half

5.3 Ray casting method limitations

The primary advantage of the ray casting method is that it is typically faster than the Hausdorff distance approach for models with a large number of vertices, as well as being more customisable. It does suffer from some limitations however, the most noticeable being its potential to ignore certain parts of the model when performing its calculations. Whilst the Hausdorff distance approach uses all the vertices in the model to perform its calculations, the ray casting method only uses the points at which the cast rays intersect the mesh. The problems this may cause are demonstrated by looking at the results for model 8. The symmetry measure of this model is very high for the correct plane as well as one of the incorrect ones. Closer analysis of the model indicates a likely cause of this abnormally high measure for the incorrect plane (see Figure 5.3).

The correct plane of reflective symmetry has been accurately identified as down the middle of the model's front view. However, the hypothesis plane that passes down the middle of the side view is also given a high symmetry measure by the ray casting method. This is likely a combination of two factors. Firstly, the majority of the jar, apart from the head and line at the back, has reflective symmetry with respect to both of these planes. Secondly, whilst the head of the jar was detected by the algorithm, explaining why the incorrect plane still has a lower symmetry measure than the correct one, the line at the back is very thin. This means that it is possible that the rays cast through the front view of the model did not intersect with this part of the object as much as would be desirable. This means that the algorithm would have little knowledge of this section of the model and would estimate a symmetry measure for the hypothesis plane without fully taking account of it. This issue does not occur when using the Hausdorff distance method as all the mesh's vertices are taken into account. This property of the ray casting method may in some cases be considered a benefit, depending on whether the user classifies the line at the back of the model as noise.

Figure 5.3: Front and side view of model 8

5.4 Choice of method

The choice of which approach to use depends greatly upon the situation and the types of models they are to be applied to. The Hausdorff distance method generally gave better accuracy than the ray casting method after the use of polygon reduction. This is largely due to the fact that this reduction made the sampling resolution more consistent throughout the model. The ray casting approach was typically both faster and more customizable than the Hausdorff distance method and would therefore be suitable to situations where time is a critical factor or if the models are known to have a very irregular sampling rate. It is also possible to use both methods in conjunction with each other (e.g. take the average symmetry measure of both methods) for situations where the accuracy of detection is very important.

5.5 Multiple symmetry planes

In our evaluation we have only considered models with one plane of reflective symmetry. Our methods also work on any models that contain multiple planes of reflective symmetry. However, they can only identify at most three hypothesis planes, due to the limitations of PCA. To confirm that our methods can detect multiple planes of reflective symmetry they were tested on several models contacting two or more reflective symmetry planes (see Figure 5.4). The results are very promising, with both methods identifying all planes of reflective symmetry (up to three) correctly, as well as a 0% false positive rate. This is a very preliminary result and requires further investigation. However, there is no reason to doubt that this result is incorrect and that our methods would perform any worse on models with multiple planes of reflective symmetry rather than just one.

Figure 5.4: Example of model containing two planes of reflective symmetry

5.6 Applications

Although the main focus of this report is on the design of the proposed symmetry detection algorithms, some of the potential applications of symmetry detection were attempted using our methods and are briefly described within this section.

5.6.1 Model remeshing

Once a reflective symmetry plane for a model has been identified it can be used to increase the symmetry of the model for further applications. One of the simplest ways of attempting to make a model more symmetrical with regard to its symmetry plane is described below.

Firstly, the model is reflected about its symmetry plane to create a new model, referred to as the reflected model. Each vertex in the original model is then moved to the midpoint between itself and the closest vertex in the reflected model. Whilst this is a very basic algorithm to accentuate a model's symmetry the effects are prominent enough to be observed on our scanned models (see Figure 5.5). This procedure can also be performed multiple times to increase the level of symmetry further, but the effect will decrease each time. More advanced algorithms for symmetry based remeshing could also be implemented (Podolak, Golovinskiy et al. 2007).

(a) (b)

Figure 5.5: Original model (a) and remeshed model (b) for model 32

5.6.2 Shape classification

Many existing methods for shape classification use symmetry as a key feature for distinguishing models (Kazhdan, Funkhouser et al. 2004). By using symmetry as a means to orientate an unclassified object it is then possible to identify similar models or shapes, which can then be used to classify the object. Reflective symmetry has also been shown to be the key factor in shape perception and viewpoint selection (Reisfeld, Wolfson et al. 1995). Knowing a model's symmetry planes can therefore help in selecting an optimum orientation and viewpoint when observing the model. This concept of model alignment can also be extended to many other applications, such as database matching or object identification.

5.6.3 Model compression

Another potential benefit of identifying symmetry within a 3D model is that is allows the model to be stored using less memory. If we know that a model contains a plane of reflective symmetry then only vertices and faces on one side of this plane need to be stored. Then, when the model is required, we can simply reflect the stored data about the symmetry plane to give the other half of the model. This is usually only viable with models that contain perfect symmetry or an extremely high level of approximate symmetry, as the uncompressed model may appear distorted if the original level of symmetry was too low.

6 Conclusion and Future Work

This report provides a detailed description and analysis of two novel methods, as well as several additional improvements, for global approximate reflective symmetry detection within scanned 3D models. These methods are both fast and robust, identifying planes of reflective symmetry correctly for the majority of 3D models tested. The first of these methods uses a variation of the Hausdorff distance to identify reflective symmetry, whilst the second method utilises ray casting and triangle intersection. When applied to our database of 32 scanned 3D models, the Hausdorff distance method had an accuracy of 96.88% whilst the ray casting method had an accuracy of 93.75%. In addition, both methods (with suitable variations) always assigned a symmetry measure to the correct plane that was larger than either of the other two incorrect planes. However, it is important to note that approximate symmetry is not an absolute property but rather a measure relative to the model's own perfect symmetry. Whilst approximate symmetry detection is difficult to quantify, we are confident that our methods provide a robust and fast approach for detecting global approximate reflective symmetry in scans of 3D models.

Future work

The area of symmetry detection within 3D models has received a large amount of prior research, yet there is still a lot of potential for future work. Whilst the methods proposed in this paper only investigated global reflective symmetry they could be extended to many other types of symmetry. These could include rotational symmetry, translational symmetry or partial symmetry along with many others.

The ability of PCA to identify potential planes of symmetry could also be extended to suit different types of symmetry, as well as being improved to provide better accuracy for the current system. More sophisticated methods for determining the models centre of mass may also prove helpful, such as using centralised moments.

.

Whilst the issue of having an irregular sampling resolution can be partially solved by polygon reduction, the Hausdorff distance method still suffers from having a greater runtime than the ray casting approach for the majority of scanned models. This is largely due to the need to compare every vertex point to every other vertex point within the mesh. An investigation into how this could be improved may provide a faster method which would greatly improve the speed of our algorithm.

The ray casting method's main weakness is that it may miss important sections of the model if the cast rays do not intersect it there. This issue may be reduced if a more sophisticated method for casting rays was developed, to put even greater focus on the important sections of the model.

Bibliography

Alt, H., et al. (1988). "Congruence, similarity and symmetries of geometric objects." Discrete Comput. Geom. **3**(3): 237-256.

Aspert, N., et al. (2002). MESH: measuring errors between surfaces using the Hausdorff distance. Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on.

Atallah, M. J. (1985). "On Symmetry Detection." IEEE Transactions on Computers **34**(7): 663-666.

Axenopoulos, A., et al. (2011). 3D model retrieval using accurate pose estimation and view-based similarity. Proceedings of the 1st ACM International Conference on Multimedia Retrieval. Trento, Italy, ACM: 1-8.

Barton, M., et al. (2010). "Precise Hausdorff distance computation between polygonal meshes." Comput. Aided Geom. Des. **27**(8): 580-591.

Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching." Commun. ACM **18**(9): 509-517.

Cailliere, D., et al. (2008). 3D mirror symmetry detection using Hough transform. Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on.

Catch, D. (2015). "Autodesk 123D Catch | 3d model from photos." from http://www.123dapp.com/catch.

Cha, Z. and C. Tsuhan (2001). Efficient feature extraction for 2D/3D objects in mesh representation. Image Processing, 2001. Proceedings. 2001 International Conference on.

Changming, S. and J. Sherrah (1997). "3D symmetry detection using the extended Gaussian image." Pattern Analysis and Machine Intelligence, IEEE Transactions on **19**(2): 164-168.

Choi, J. A. K. a. K. (1995). "Ray Tracing Triangular Meshes." In Western Computer Graphics Symposium.

Cignoni, P., et al. (1996). Metro: measuring error on simplified surfaces, Centre National de la Recherche Scientifique.

Dimitrov, D. (2012). Geometric Applications of Principal Component Analysis, VDM Publishing.

Garland, M. and P. S. Heckbert (1997). Surface simplification using quadric error metrics. Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co.: 209-216.

Guthe, M. a. B., Pavel and Klein, Reinhard (2005). "Fast and accurate Hausdorff distance calculation between meshes." Journal of WSCG **13**(2): 41--48.

Jiang, X.-Y. and H. Bunke (1991). Determination of the Symmetries of Polyhedra and an Application to Object Recognition. Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications, Springer-Verlag: 113-121.

Kazhdan, M., et al. (2004). Symmetry descriptors and 3D shape matching. Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. Nice, France, ACM: 115-123.

Li, B., et al. (2014). Efficient view-based 3d reflection symmetry detection. SIGGRAPH Asia 2014 Creative Shape Modeling and Design. Shenzhen, China, ACM: 1-8.

Liu, J. and Y. Liu (2011). Curved reflection symmetry detection with self-validation. Proceedings of the 10th Asian conference on Computer vision - Volume Part IV. Queenstown, New Zealand, Springer-Verlag: 102-114.

Liu, Y., Hel-or, H., Kaplan, C. S., and Gool, L. J. V. (2010). "Foundations and Trends in Computer Graphics and Vision." Computational symmetry in computer vision and computer graphics **5**(1-2): 1-195.

Martinet, l., et al. (2006). "Accurate detection of symmetries in 3D shapes." ACM Trans. Graph. **25**(2): 439-464.

Minovic, P., et al. (1993). "Symmetry Identification of a 3-D Object Represented by Octree." IEEE Trans. Pattern Anal. Mach. Intell. **15**(5): 507-514.

Mitra, N. J., et al. (2006). "Partial and approximate symmetry detection for 3D geometry." ACM Trans. Graph. **25**(3): 560-568.

Mitra, N. J., et al. (2007). "Symmetrization." ACM Trans. Graph. **26**(3): 63.

Moller, T. and B. Trumbore (1997). "Fast, minimum storage ray-triangle intersection." J. Graph. Tools **2**(1): 21-28.

Podolak, J., et al. (2007). Symmetry-enhanced remeshing of surfaces. Proceedings of the fifth Eurographics symposium on Geometry processing. Barcelona, Spain, Eurographics Association: 235-242.

Podolak, J., et al. (2006). "A planar-reflective symmetry transform for 3D shapes." ACM Trans. Graph. **25**(3): 549-559.

Raviv, D., et al. (2010). "Full and Partial Symmetries of Non-rigid Shapes." Int. J. Comput. Vision **89**(1): 18-39.

Reisfeld, D., et al. (1995). "Context-free attentional operators: the generalized symmetry transform." Int. J. Comput. Vision **14**(2): 119-130.

S. Parry-Barwick, A. B. (1993). "Symmetry analysis and geometric modelling." Digital Image Computing Techniques and Applications **1**: 39-46.

Sawada, T. and Z. Pizlo (2008). Detecting mirror-symmetry of a volumetric shape from its single 2D image. Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on.

Shilane, P., et al. (2004). The Princeton Shape Benchmark. Proceedings of the Shape Modeling International 2004. Genova, Italy.

Standford, G. D. (2000). "Search Structures." from http://graphics.stanford.edu/courses/cs368-00-spring/TA/manuals/CGAL/ref-manual2/SearchStructures/Chapter_main.html.

Straub, R. (2007). "Exact Computation of the Hausdorff Distance between Triangular Meshes." Proceedings of Eurographics 2007: 17-20.

Vollmer, J., et al. (1999). "Improved Laplacian Smoothing of Noisy Surface Meshes." Computer Graphics Forum **18**(3): 131-138.

Wolter, J., et al. (1985). "Optimal algorithms for symmetry detection in two and three dimensions." The Visual Computer **1**(1): 37-48.

Zabrodsky, H., et al. (1995). "Symmetry as a continuous feature." Pattern Analysis and Machine Intelligence, IEEE Transactions on **17**(12): 1154-1166.

Zou, H. L. and Y. T. Lee (2005). Skewed mirror symmetry detection from a 2D sketch of a 3D model. Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. Dunedin, New Zealand, ACM: 69-7

Appendix

Appendix A: Collection of scanned 3D models

Figure A.1: Model 1

Figure A.2: Model 2

Figure A.3: Model 3

Figure A.4: Model 4

Figure A.5: Model 5

Figure A.6: Model 6

Figure A.7: Model 7

Figure A.8: Model 8

Figure A.9: Model 9

Figure A.10: Model 10

Figure A.11: Model 11

Figure A.12: Model 12

Figure A.13: Model 13

Figure A.14: Model 14

Figure A.15: Model 15

Figure A.16: Model 16

Figure A.17: Model 17

Figure A.18: Model 18

Figure A.19: Model 19

Figure A.20: Model 20

Figure A.21: Model 21

Figure A.22: Model 22

Figure A.23: Model 23

Figure A.24: Model 24

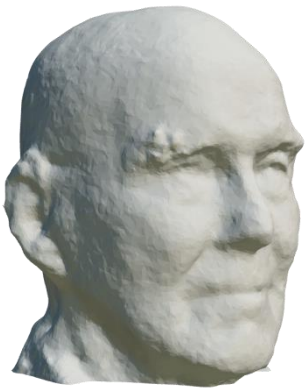


Figure A.25: Model 25

Figure A.26: Model 26

Figure A.27: Model 27

Figure A.28: Model 28

Figure A.29: Model 29

Figure A.30: Model 30

Figure A.31: Model 31

Figure A.32: Model 32

Bibliography for Appendix A:

Model 1: http://www.123dapp.com/obj-Catch/Mummy-Boston-Museum/2222854

Model 2: http://www.123dapp.com/catch/Alligator-skull-at-the-Skulls-exhibit-at-the-California-Academy-of-the-Sciences/2616526

Model 3: http://www.123dapp.com/catch/Marcus-Aurelius/2645898

Model 4: http://www.123dapp.com/catch/NASA-EVA-Space-Suit-model-from-Bandai-1-10-Scale/2671037

Model 5: http://www.123dapp.com/catch/Charles-darwin-statue/2748211

Model 6: http://www.123dapp.com/catch/ram-skull/2792731

Model 7: http://www.123dapp.com/catch/Egyptian-feline/2894209

Model 8: http://www.123dapp.com/obj-Catch/Canopic-jar/1621529

Model 9: http://www.123dapp.com/catch/R2-d2/2955339

Model 10: http://www.123dapp.com/catch/Matti-s-new-L-A-Gear-shoe/3115700

Model 11: http://www.123dapp.com/catch/Portrait-of-Antoni-Gaud-/3358833

Model 12: http://www.123dapp.com/catch/USF-Museum-Visualizations/3382857

Model 13: http://www.123dapp.com/catch/Drake/3389466

Model 14: http://www.123dapp.com/catch/Homer-wood-sculpture-at-Home-Depot/3560340

Model 15: http://www.123dapp.com/catch/Tapir/3567176

Model 16: http://www.123dapp.com/catch/The-Swan-/3582510

Model 17: http://www.123dapp.com/catch/Leopard-seal-skull/3585690

Model 18: http://www.123dapp.com/catch/Sarcophagus/3603386

Model 19: http://www.123dapp.com/catch/Happy-Chinese-goat-year/3657254

Model 20: http://www.123dapp.com/catch/zhangliang1/3778831

Model 21: http://www.123dapp.com/catch/-I-have-the-Power-/3834554

Model 22: http://www.123dapp.com/catch/Hulkbuster-Avengers-Movie/3852196

Model 23: http://www.123dapp.com/catch/beeld-3-Mooniq/3960506

Model 24: http://www.123dapp.com/catch/bumblebee-transformers-deluxe-class/4114483

Model 25: http://www.123dapp.com/catch/Dad/4125292

Model 26: http://www.123dapp.com/catch/Buddha-Statue/4136278

Model 27: http://www.123dapp.com/catch/Surprise-/4145623

Model 28: http://www.123dapp.com/catch/Turtle/4161963

Model 29: http://www.123dapp.com/catch/Terminator-Skull/4163065

Model 30: http://www.123dapp.com/catch/Naoya/4164048

Model 31: http://www.123dapp.com/catch/einstein/4164243

Model 32: http://www.123dapp.com/catch/toy-bird/4181176

Appendix B: Additional graphs for results

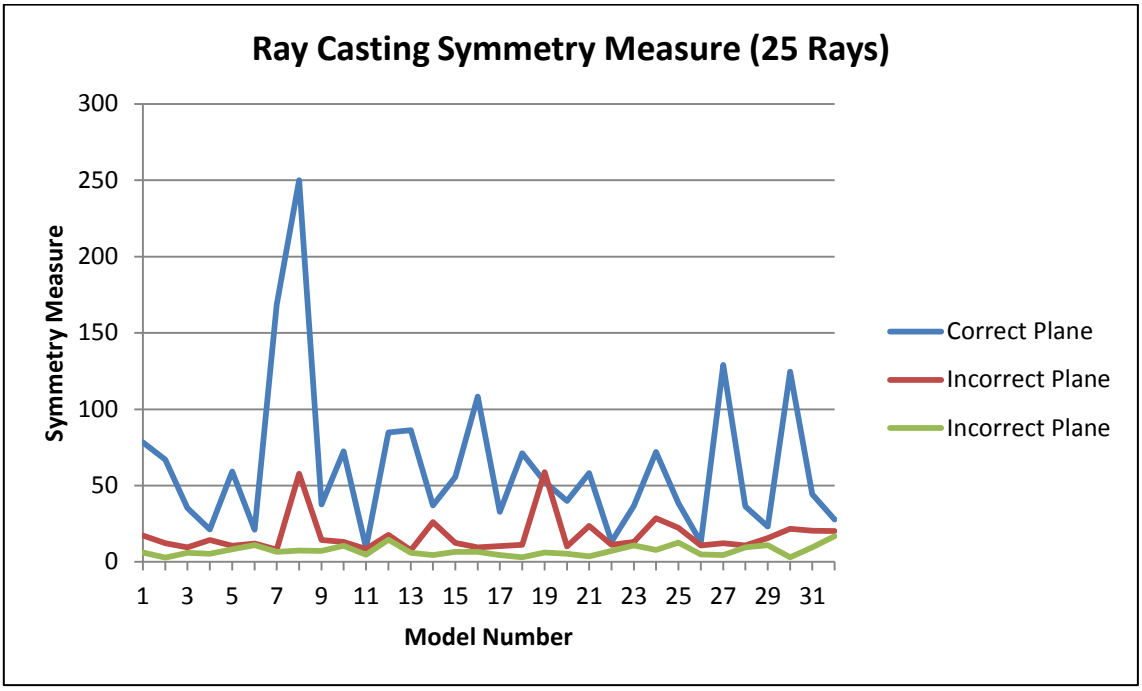


Figure B.1: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method with a 5x5 grid of rays

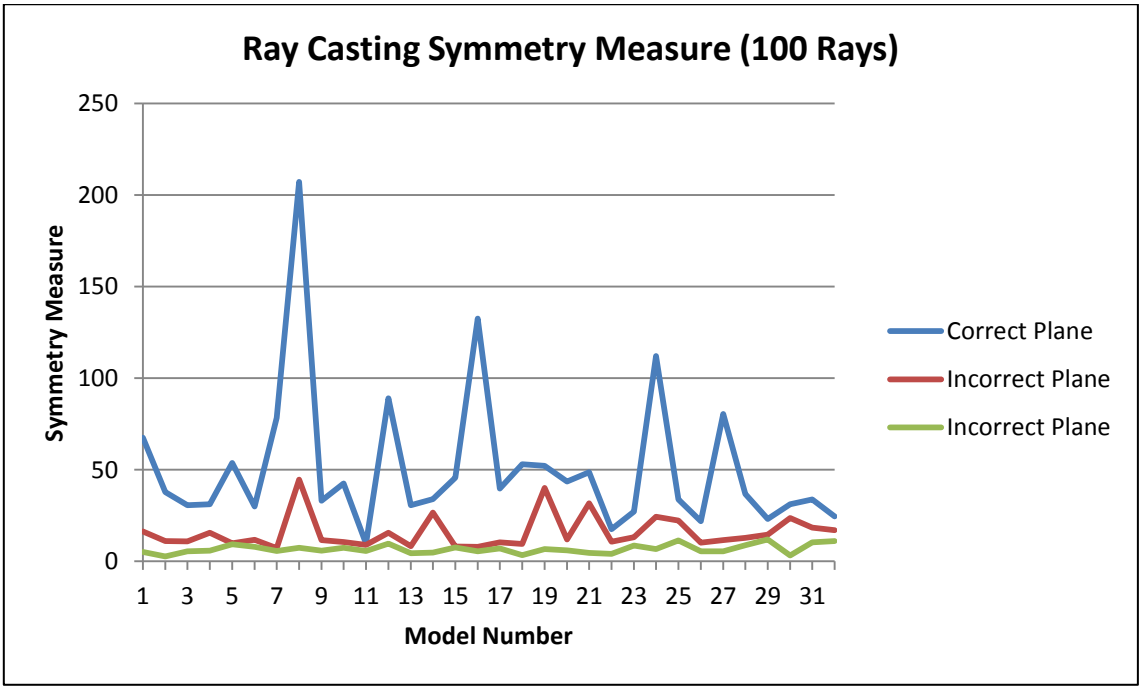


Figure B.2: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method with a 10x10 grid of rays

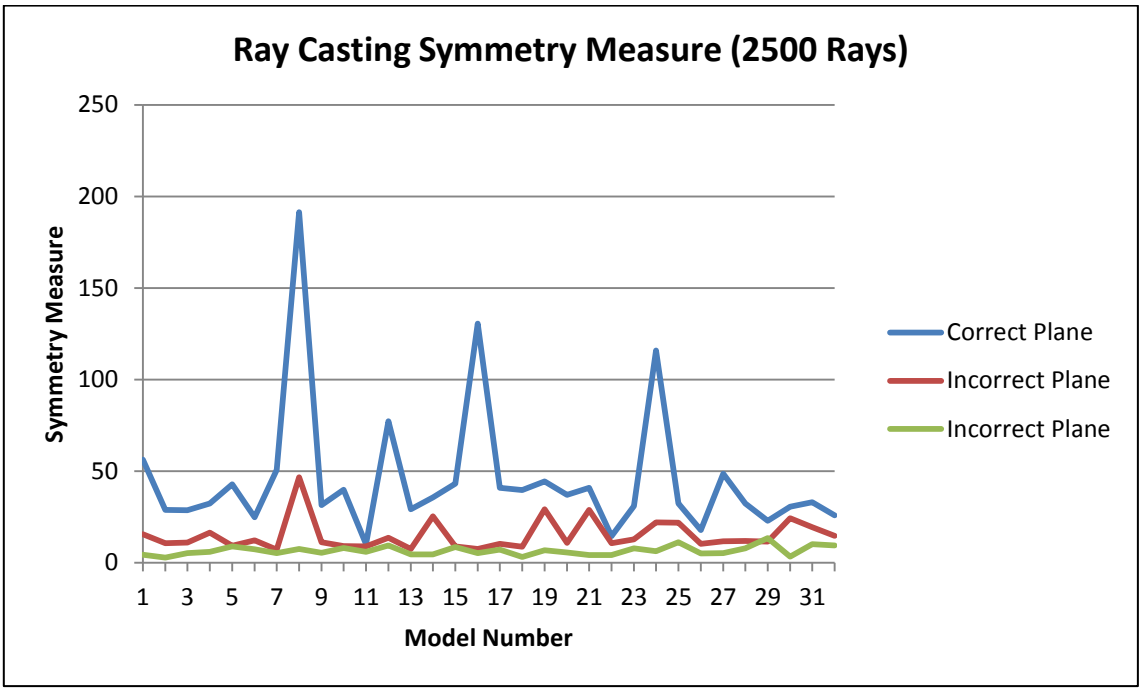


Figure B.3: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method with a 50x50 grid of rays

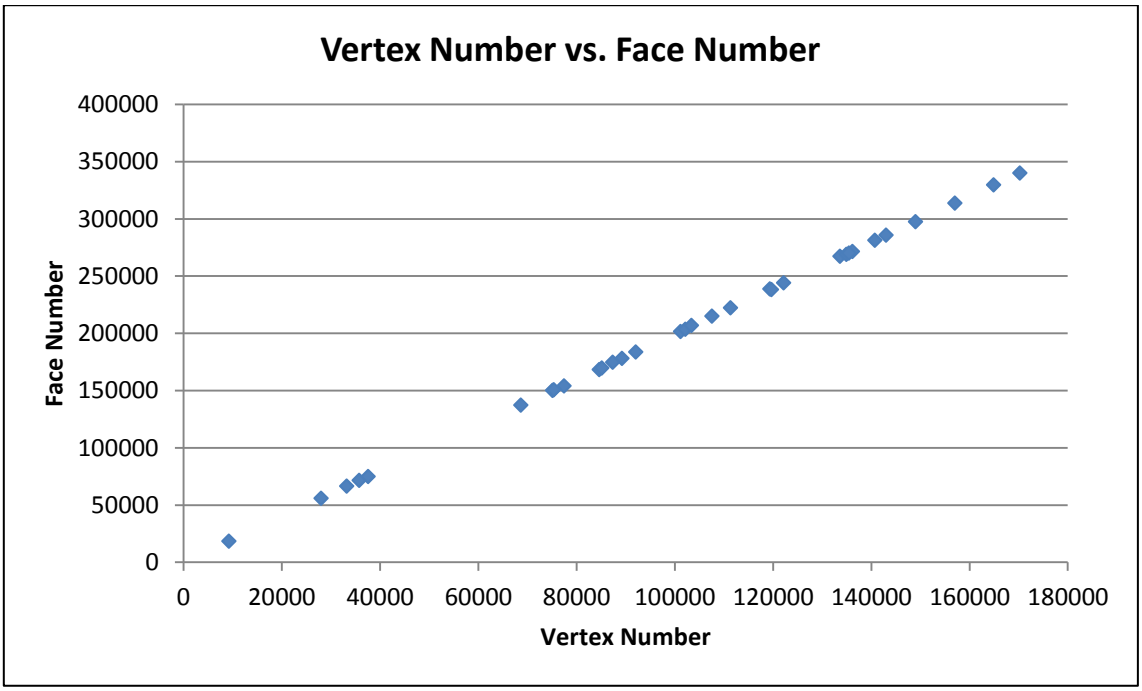


Figure B.4: The relation between vertex number and face number for each of the scanned models

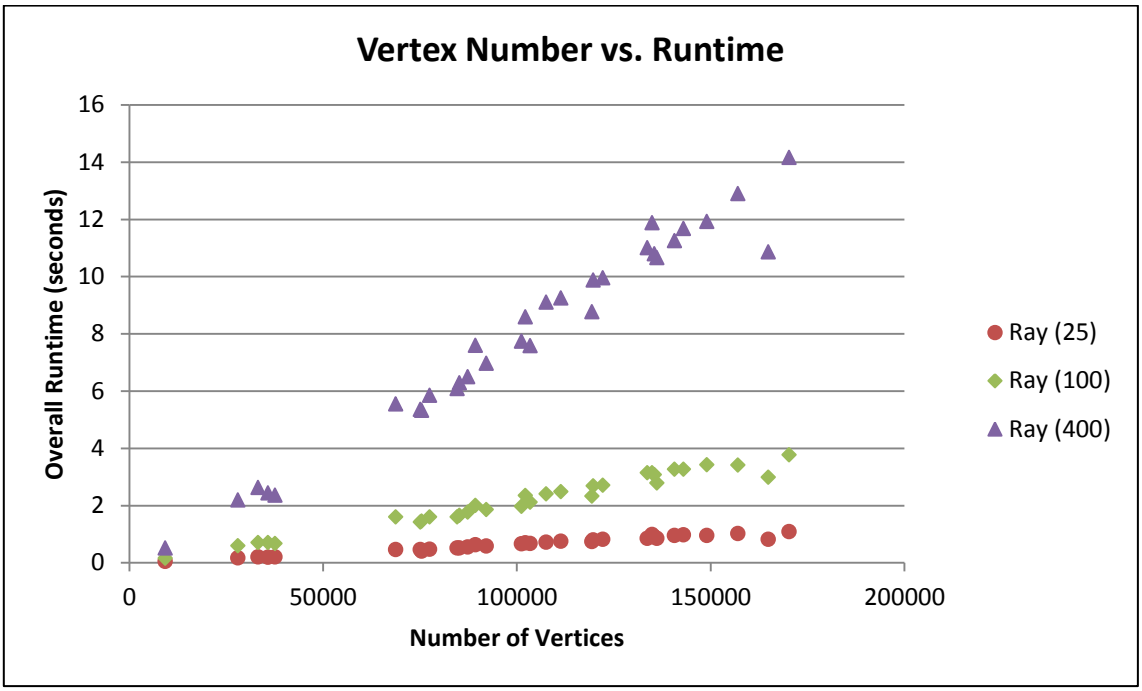


Figure B.5: The total runtime for each of the faster algorithms relative to the number of vertices in the model for the set of scanned models

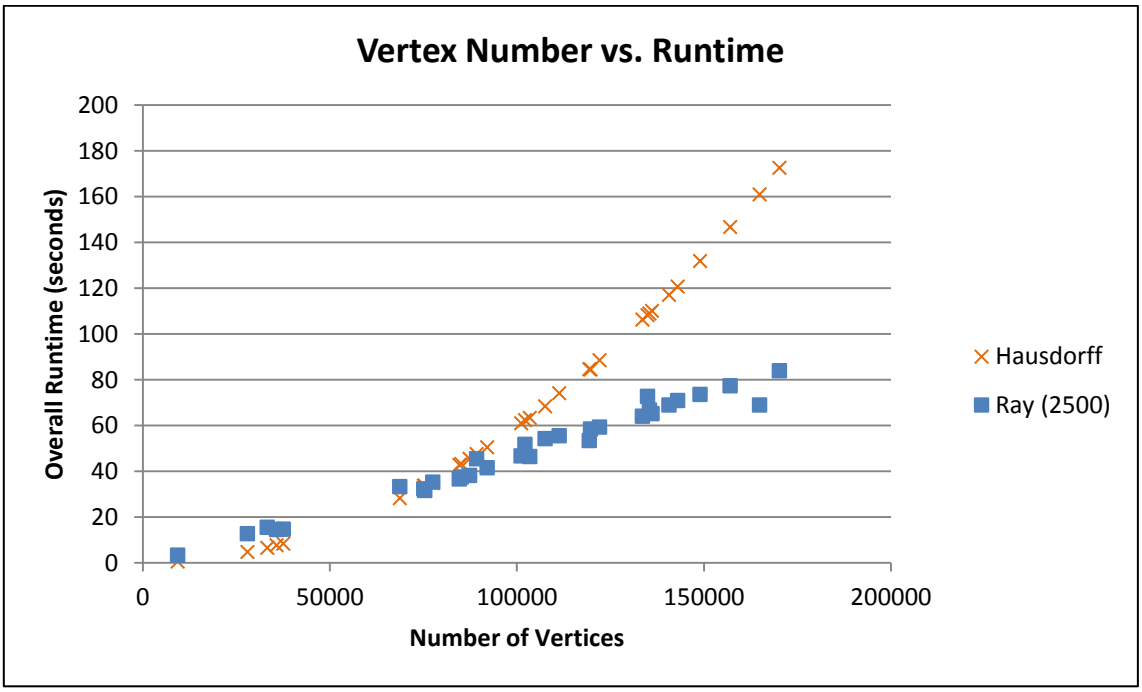


Figure B.6: The total runtime for each of the slower algorithms relative to the number of vertices in the model for the set of scanned models

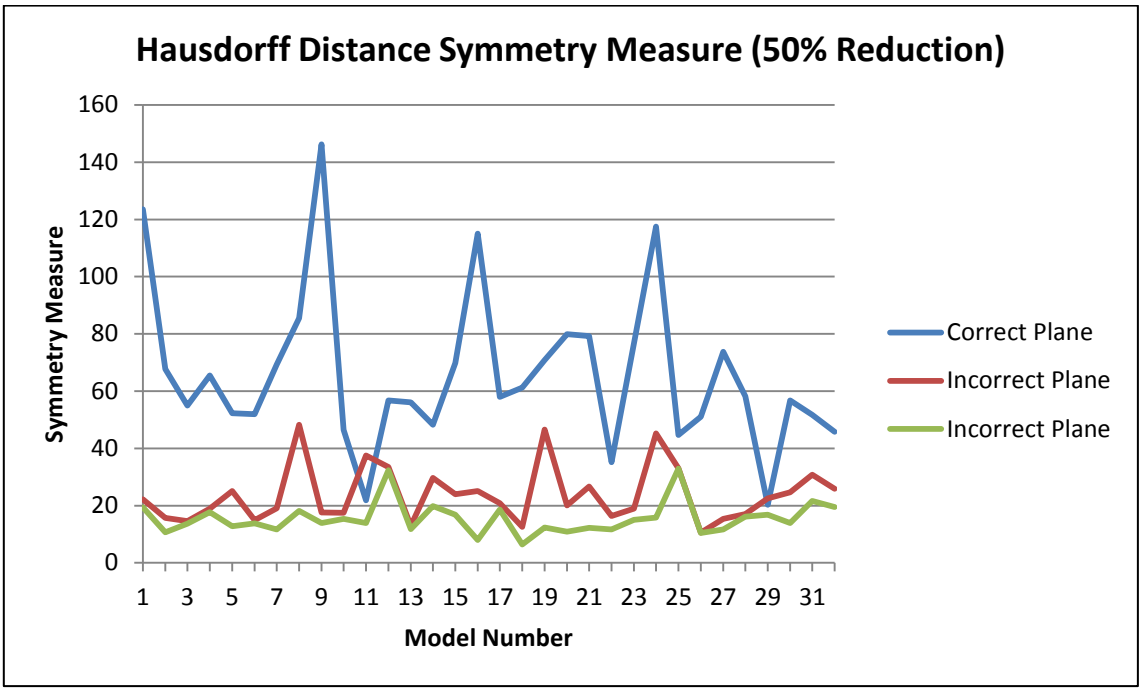


Figure B.7: The reflective symmetry measure given to each model's hypothesis planes using the Hausdorff distance based method after the model has had 50% of its polygons removed

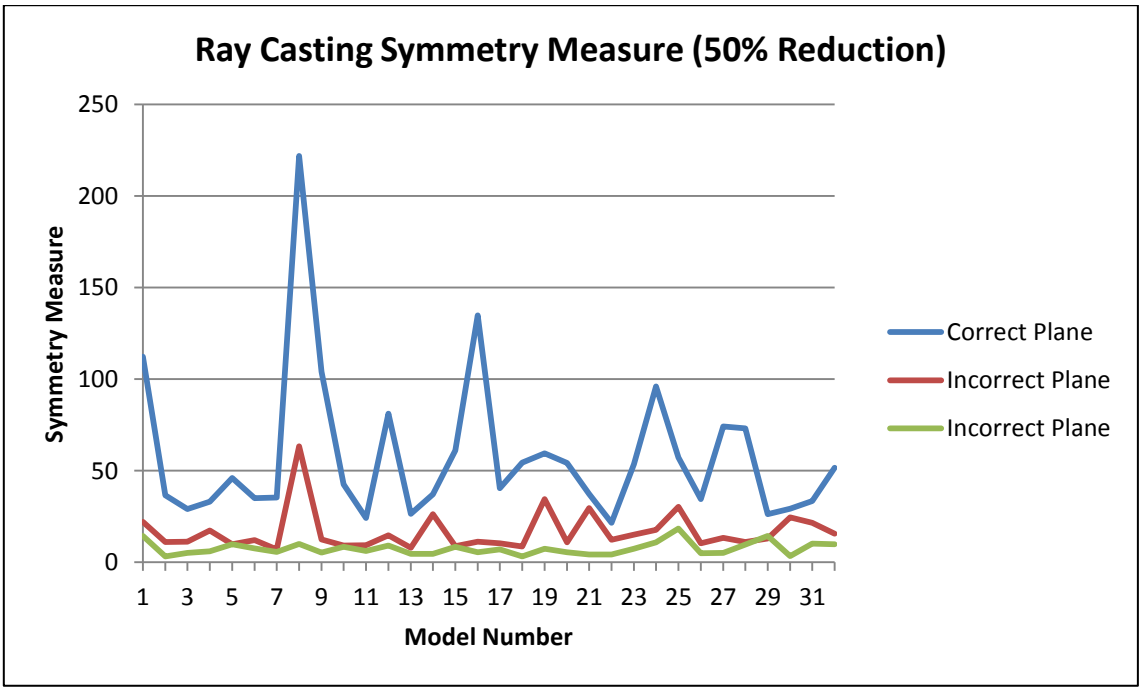


Figure B.8: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method after the model has had 50% of its polygons removed

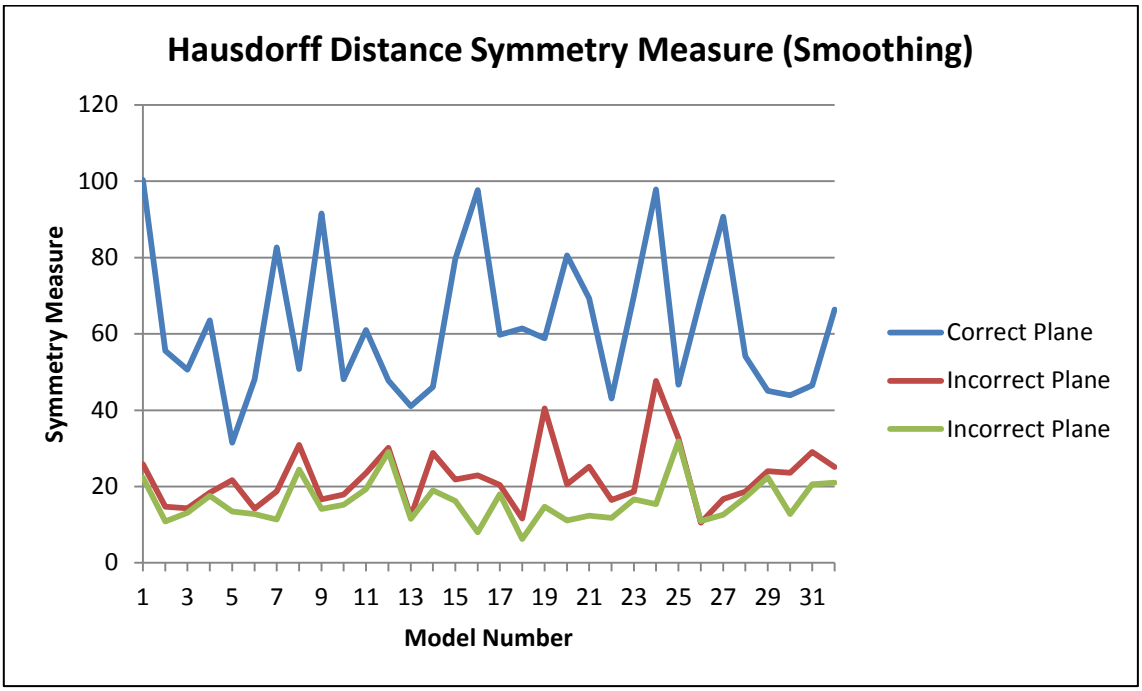


Figure B.9: The reflective symmetry measure given to each model's hypothesis planes using the Hausdorff distance based method after the model has been reduced and smoothed

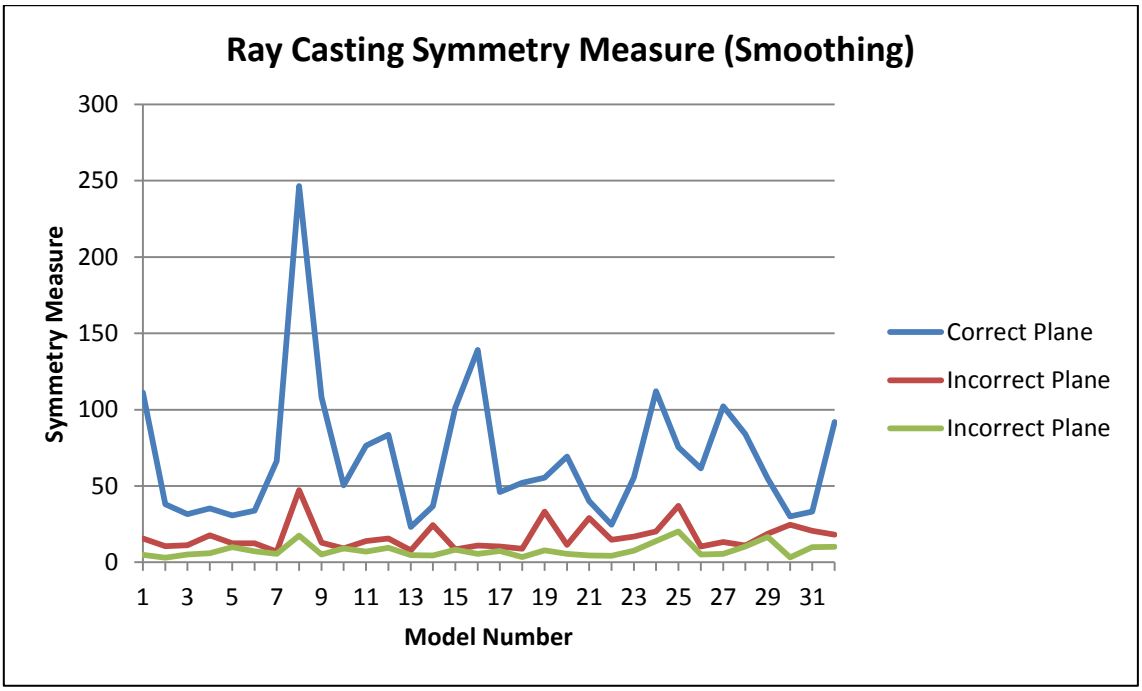


Figure B.10: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method after the model has been reduced and smoothed