Novel Methods for Reflective Symmetry Detection in Scanned 3D Models

Matthew Stephenson
Computer Science and Software Engineering Dept.
University of Canterbury
Christchurch New Zealand
mjs357@uclive.ac.nz

Adrian Clark
Computer Science and Software Engineering Dept.
University of Canterbury
Christchurch New Zealand
adrian.clark@canterbury.ac.nz

Richard Green
Computer Science and Software Engineering Dept.
University of Canterbury
Christchurch New Zealand
richard.green@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 the early work focuses on identifying symmetry in noiseless 3D models with most existing 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 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.**

Keywords—symmetry; detection; scanned; models; reflective

I. INTRODUCTION

Symmetry is a mathematical concept that exists in many man-made objects, as well as being widely prevalent in nature [1]. 3D geometrical models are typically represented as a mesh created from small polygons, usually triangles, with little to no information about their higher level structure. Determining these additional properties of a model, including its global reflective symmetry, is an important task within computer graphics and computer vision.

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 geometry and problematic restrictions on the properties the models must possess before the algorithm can be applied to them, 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 3D models distorted by small amounts of noise (see Figure 1). One key application of this research is the capability to detect symmetry within scanned models of 3D objects, which are typically 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 potential 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.

II. BACKGROUND

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 [2]. This approach can be adapted and improved further, with two notable extensions being the ability to detect partial symmetry within a 2D image [3], and the ability to detect approximate symmetry by utilising a hierarchy that defines symmetry as a continuous feature [4]. However, both of these additions are very computationally expensive and rely on

Figure 1: Example of a scanned 3D model containing one plane of global approximate reflective symmetry.

the algorithm's ability to establish correspondence between points within the image. The original idea of reducing 2D symmetry detection to a 1D pattern matching problem can then be expanded to detect symmetry in 3D point sets [5], as well as the ability to detect approximate symmetries using similar principles [6].

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. These include, identifying automorphisms of planar triply connected graphs [7], octree representation [8], extended Gaussian image [9] and spherical harmonic coefficients of generalised moments [10]. Several algorithms for partial symmetry detection are also suggested, such as Gaussian Euclidean distance transform [11] [12] and stochastic clustering to find pairs of vertex groups [13]. Whilst it is possible to detect partial, rotational and reflectional symmetry using a combination of these methods, none of them are designed specifically for use on scanned 3D models. Our methods are designed and evaluated specifically around these types of mesh.

III. 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 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. 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 [14]. For this reason it was selected as the method by which to derive the hypothesis planes. Using PCA we determine two eigenvectors representing the directions of maximum variance. From these two eigenvectors 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.

Now that the hypothesis planes have been 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.

B. 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. 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 within the mesh and allocating it to one of two sets based on which side of the hypothesis plane it is on. The vertices within one of these sets are then reflected about the hypothesis plane. If the two meshes are now approximately the same then we conclude that the hypothesis plane is indeed a plane of reflective symmetry.

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 $\boldsymbol{Y}$.

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

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 [15, 16]. Whilst this is good for measuring error during simplification [17] 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. 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 [18, 19].

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 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.

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.

C. Ray Casting Method

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. 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. 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. 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 which calculates the distance that the ray has travelled before each triangle intersection [20, 21].

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.

The total deviation $\boldsymbol{S}$ 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 $\boldsymbol{P}$.

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

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 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 collision 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 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.

D. 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, we estimate the model's volume by calculating the signed volume of a tetrahedron based on each triangle within the model [22]. These individual volumes are then summed together and the absolute value of this is used as an estimate $\boldsymbol{E}$ for the model's volume.

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

E. Additional Variations

While the general frameworks for the proposed symmetry detection algorithms have been described, several additions are proposed which can often provide better results.

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 [23] (see Figure 2). This simplifies the mesh by iteratively contracting edges until the desired number of vertices or faces remains. 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.

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 increases 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.

Figure 2: A collection of simplified models, the number of faces reduced by approximately half each time

2) K-d tree:

A *k*-d tree is a space partitioning data structure for organising points in *k*-dimensional space [24]. For our method, we use a *k*-d tree to 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 can improve this even more.

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.

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. 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 this variation 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.

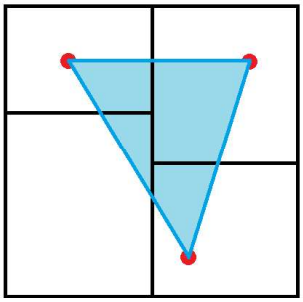

Figure 3: The problem with *k*-d tree method, the triangle's vertices are only located within three sections but the triangle overlaps four sections

3) 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 evenly along the model, based on the number of vertices rather than the total length. 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 and the other for y-axis ordering) is then split into subsets based on the number of rays 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.

IV. RESULTS

The testing for both the Hausdorff distance and ray casting methods, as well as all applicable variations, was initially performed using the Princeton Shape Benchmark (PSB) [25]. 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. These models were obtained using the scanning program 123DCatch [26]. This set of models was also used to determine a suitable threshold for the symmetry measure of each method.

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.

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. 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 (see Figures 4 and 5). We used 90% polygon reduction on all models and 400 rays for the ray casting approach.

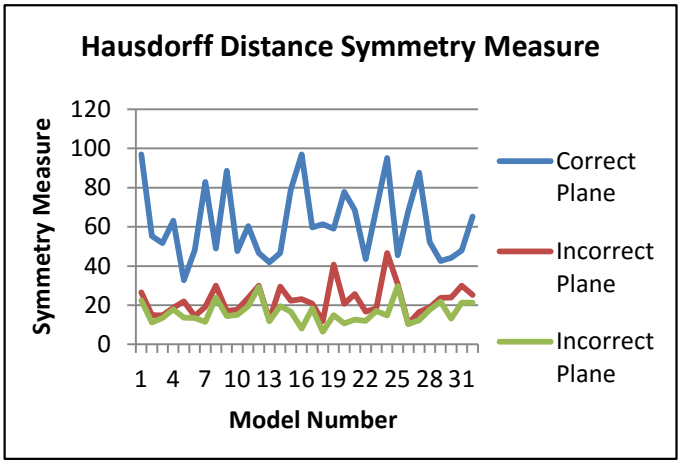

Figure 4: The reflective symmetry measure given to each model's hypothesis planes using the Hausdorff distance based method

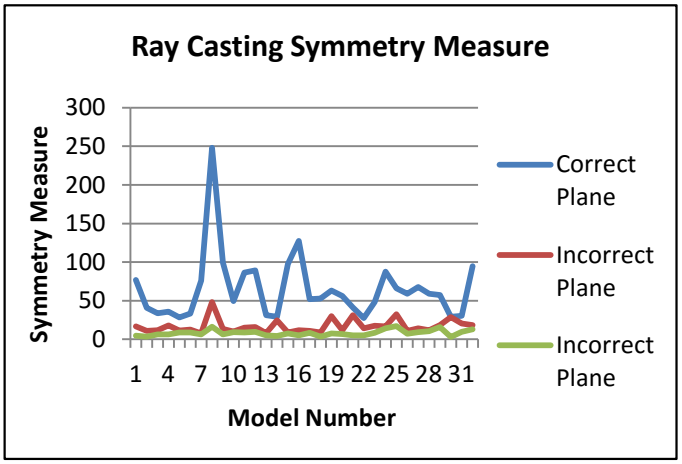

Figure 5: The reflective symmetry measure given to each model's hypothesis planes using the ray casting based method

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. (see Figure 6).

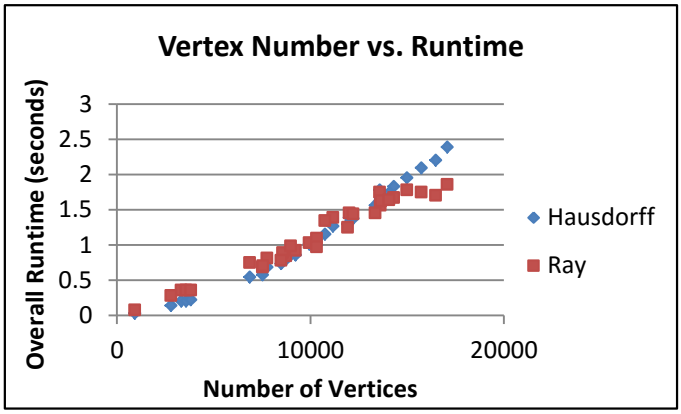

Figure 6: The total runtime for each method relative to the number of vertices in the model

A. 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. The thresholds for each method that produced the fewest misclassifications are as follows:

Hausdorff Distance Method:

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

Ray Casting Method:

- Threshold ≈ 29
- **Accuracy per plane = 94.62%**

These thresholds 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 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.

V. DISCUSSION

Whilst both of these methods have demonstrated their effectiveness at detecting global reflective symmetry within scanned 3D models, the choice of which approach to use depends greatly upon the situation and the types of models they are to be applied to. Originally, the Hausdorff distance method gave lower accuracy than the ray casting method. However, after the inclusion of polygon reduction the accuracy of the Hausdorff distance method increased dramatically, to above that of the ray casting method. 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, for models with a large number of vertices, and more customizable than the Hausdorff distance method. This would therefore make it more suitable for 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, for situations where the accuracy of detection is very important (e.g. take the average symmetry measure of both methods).

VI. CONCLUSION

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 94.62%. In addition, both methods 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.

REFERENCES

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

[2] M. J. Atallah, "On Symmetry Detection," *IEEE Transactions on Computers,* vol. 34, pp. 663-666, 1985.

[3] A. B. S. Parry-Barwick, "Symmetry analysis and geometric modelling," *Digital Image Computing Techniques and Applications,* vol. 1, pp. 39-46, 1993.

[4] H. Zabrodsky, S. Peleg, and D. Avnir, "Symmetry as a continuous feature," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 17, pp. 1154-1166, 1995.

[5] J. Wolter, T. Woo, and R. Volz, "Optimal algorithms for symmetry detection in two and three dimensions," *The Visual Computer,* vol. 1, pp. 37-48, 1985/07/01 1985.

[6] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl, "Congruence, similarity and symmetries of geometric objects," *Discrete Comput. Geom.,* vol. 3, pp. 237-256, 1988.

[7] X.-Y. Jiang and H. Bunke, "Determination of the Symmetries of Polyhedra and an Application to Object Recognition," presented at the Proceedings of the International Workshop on Computational Geometry - Methods, Algorithms and Applications, 1991.

[8] P. Minovic, S. Ishikawa, and K. Kato, "Symmetry Identification of a 3-D Object Represented by Octree," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 15, pp. 507-514, 1993.

[9] S. Changming and J. Sherrah, "3D symmetry detection using the extended Gaussian image," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 19, pp. 164-168, 1997.

[10] l. Martinet, C. Soler, N. Holzschuch, and F. X. Sillion, "Accurate detection of symmetries in 3D shapes," *ACM Trans. Graph.,* vol. 25, pp. 439-464, 2006.

[11] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Symmetry descriptors and 3D shape matching," presented at the Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Nice, France, 2004.

[12] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A planar-reflective symmetry transform for 3D shapes," *ACM Trans. Graph.,* vol. 25, pp. 549-559, 2006.

[13] N. J. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3D geometry," *ACM Trans. Graph.,* vol. 25, pp. 560-568, 2006.

[14] D. Dimitrov, *Geometric Applications of Principal Component Analysis*: VDM Publishing, 2012.

[15] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH: measuring errors between surfaces using the Hausdorff distance," in *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, 2002, pp. 705-708 vol.1.

[16] M. a. B. Guthe, Pavel and Klein, Reinhard, "Fast and accurate Hausdorff distance calculation between meshes," *Journal of WSCG,* vol. 13, pp. 41--48, 2005.

[17] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," Centre National de la Recherche Scientifique1996.

[18] R. Straub, "Exact Computation of the Hausdorff Distance between Triangular Meshes," *Proceedings of Eurographics 2007,* pp. 17-20, 2007.

[19] M. Barton, I. Hanniel, G. Elber, and M.-S. Kim, "Precise Hausdorff distance computation between polygonal meshes," *Comput. Aided Geom. Des.,* vol. 27, pp. 580-591, 2010.

[20] J. A. K. a. K. Choi, "Ray Tracing Triangular Meshes," *In Western Computer Graphics Symposium,* 1995.

[21] T. Moller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *J. Graph. Tools,* vol. 2, pp. 21-28, 1997.

[22] Z. Cha and C. Tsuhan, "Efficient feature extraction for 2D/3D objects in mesh representation," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, 2001, pp. 935-938 vol.3.

[23] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," presented at the Proceedings of the 24th annual conference on Computer graphics and interactive techniques, 1997.

[24] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM,* vol. 18, pp. 509-517, 1975.

[25] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," presented at the Proceedings of the Shape Modeling International 2004, Genova, Italy, 2004.

[26] D. Catch. (2015). *Autodesk 123D Catch | 3d model from photos*. Available: http://www.123dapp.com/catch