In Proc. of the IEEE International Conference in Multimedia and Expo (ICME) 2002,
vol. 1, pp. 705-708, Lausanne, Switzerland, August 26-29, 2002, `http://mesh.epfl.ch/`.

MESH : MEASURING ERRORS BETWEEN SURFACES USING THE HAUSDORFF DISTANCE

Nicolas Aspert, Diego Santa-Cruz, Touradj Ebrahimi *

Signal Processing Institute (ITS)
Swiss Federal Institute of Technology (EPFL) - Lausanne - Switzerland

ABSTRACT

This paper proposes an efficient method to estimate the distance between discrete 3D surfaces represented by triangular 3D meshes. The metric used is based on an approximation of the Hausdorff distance, which has been appropriately implemented in order to reduce unnecessary computations and memory usage. Results show that when compared to similar tools, a significant gain in both memory and speed can be achieved.

1. INTRODUCTION

In the last decade, many new compression techniques adapted to images, sounds, and more recently 3D-models have been developed, and many more are foreseen in the next years. The growing amount of transmitted data through the Internet has also raised the problem of watermarking, which is usually achieved through modification of a content in a non-perceptible way to embed an information. Compression and watermarking share a common goal, which is to minimize the distortions added to the original signal while maximizing the compression ratio, the strength or the capacity of the watermark.

In the case of 1D-signals and images, many distortion measurements have been studied. They range from the simple analytic methods such as the *mean square error* (MSE) to much more elaborate techniques based on the characteristics of human perception [1]. Despite the constantly growing number of techniques related to 3D models, the distortion measurements for such data have only been sparsely covered. One of the simplest approaches in order to provide an MSE-like measurement for 3D models is to use the *Hausdorff distance*, which is a very generic technique to define a distance between two nonempty sets. The Hausdorff metric has already been used when addressing the problem of mesh simplification[2].

In this paper, we present an efficient tool to evaluate the distance between 3D models, similar to `Metro`[3]. The paper is organized as follows. Section 2 will present the Hausdorff distance, and the measurements based on this distance. Section 3 will present the details of the implementation of these metrics, and section 4 will focus on the analysis. Conclusions are drawn in section 5.

2. HAUSDORFF DISTANCE BETWEEN TRIANGULAR MESHES

While digital image/signal techniques require a one-to-one mapping between the samples of the original data and the samples from the modified version, such a constraint would practically be too restrictive in the case of 3D models. Many methods applied to 3D models imply a change of topology, in addition to purely geometrical distortions, which justifies the choice of the Hausdorff metric to perform measurements over this type of data, instead of a simple vertex to vertex metric.

2.1. Notations

For the sake of simplicity, we will only present the case of discrete 3D models represented by *triangular meshes*, since this is the most generic representation of such data. A triangular mesh $\mathcal{M}$ will be represented by a set $\mathcal{P}$ of points in $\mathbb{R}^3$ (vertices), and by a set $\mathcal{T}$ of triangles describing how the vertices from $\mathcal{P}$ are linked together. We will denote by $\mathcal{S}$ and $\mathcal{S}'$ two continuous surfaces.

2.2. Hausdorff distance

Let us first define the distance $d(p, \mathcal{S}')$ between a point p belonging to a surface $\mathcal{S}$ and a surface $\mathcal{S}'$ as:

$$d(p, \mathcal{S}') = \min_{p' \in \mathcal{S}'} \|p - p'\|_2, \tag{1}$$

where $\|.\|_2$ denotes the usual Euclidean norm. From this definition, the *Hausdorff distance* between $\mathcal{S}$ and $\mathcal{S}'$, denoted by $d(\mathcal{S}, \mathcal{S}')$ is given by:

$$d(\mathcal{S}, \mathcal{S}') = \max_{p \in \mathcal{S}} d(p, \mathcal{S}').$$

It is important to note that this distance is in general not symmetrical, i.e. $d(\mathcal{S}, \mathcal{S}') \neq d(\mathcal{S}', \mathcal{S})$. We will refer to $d(\mathcal{S}, \mathcal{S}')$ as *forward distance*, and to $d(\mathcal{S}', \mathcal{S})$ as *backward distance*. It is then convenient to introduce the *symmetrical Hausdorff distance* $d_s(\mathcal{S}, \mathcal{S}')$, defined as follows:

$$d_s(\mathcal{S}, \mathcal{S}') = \max\left[d(\mathcal{S}, \mathcal{S}'), d(\mathcal{S}', \mathcal{S})\right]. \tag{2}$$

The symmetrical distance provides a more accurate measurement of the error between two surfaces, since the computation of a "one-sided" error can lead to significantly underestimated distance values, as illustrated by figure 1.

The computation of the Hausdorff distance between two discrete surfaces $\mathcal{M} = (\mathcal{P}, \mathcal{T})$ and $\mathcal{M}' = (\mathcal{P}', \mathcal{T}')$ relies on the previous definitions. We will focus on the computation of the forward Hausdorff distance, i.e. $d(\mathcal{M}, \mathcal{M}')$, since the symmetrical distance can be simply obtained from the computation of the forward and backward distances. The distance between any point p belonging to $\mathcal{M}$ (p might not be in $\mathcal{P}$) and $\mathcal{M}'$ can be computed analytically, since it can be reduced to the minimum of the distances between p and all the triangles $T' \in \mathcal{T}'$. When the orthogonal projection p' of p on the plane of T' is inside the triangle, the

*Corresponding authors: {Nicolas.Aspert,Diego.SantaCruz}@epfl.ch

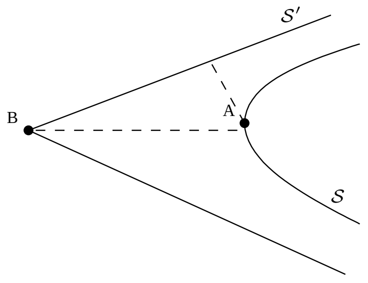

Figure 1: In this case, $d(\mathcal{S},\mathcal{S}')$ will remain much smaller than $d(\mathcal{S}',\mathcal{S})$, since here $d(A,\mathcal{S}') \ll d(B,\mathcal{S})$. Thus a small one-sided distance does not imply a small distortion.

point-to-triangle distance is nothing but a point-to-plane distance. When the projection lies outside T', the point-to-triangle distance is the distance between p and the closest point p'' of T', which is necessarily on one of the sides of T' (see figure 2). Although

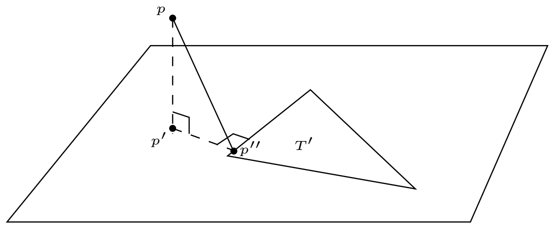

Figure 2: The distance between p and T' is the distance between p and the closest point of T', lying on one of its sides.

$d(p,\mathcal{S}')$ can be computed analytically for any point p, it is necessary to resort to sampling to obtain the maximum for $p \in \mathcal{S}$. Each triangle of $\mathcal{T}$ is sampled, and the distance between each sample and $\mathcal{M}'$ is computed. The sampling of each triangle is done in the following way : two sides of the triangle are considered as directions for the sampling lattice. According to a length criterion (see section 3 for more details), each side is sampled with n points. Using the directions, it is then easy to build a "regular" grid over the considered triangle (see figure 3). According to this sampling pattern, $n(n+1)/2$ samples are created in each triangle. One interesting property of this sampling is that the triangle can be easily split into smaller triangles having all the same area, which leads to significantly simpler computations of integrals over a surface (cf. section 2.3).

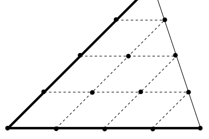

Figure 3: Illustration of the sampling performed on triangles, for $n = 5$. The sides used as main directions are drawn in bold lines and the samples are materialized by the black dots.

2.3. Mean and mean square error

The point-to-surface distance defined in (1) can be used to define a *mean error* d_m between two surfaces $\mathcal{S}$ and $\mathcal{S}'$:

$$d_m(\mathcal{S},\mathcal{S}') = \frac{1}{|\mathcal{S}|}\iint_{p\in\mathcal{S}} d(p,\mathcal{S}')dS, \tag{3}$$

where $|\mathcal{S}|$ denotes the area of $\mathcal{S}$. From this, the definition of a *root mean square error* d_{rmse} follows naturally :

$$d_{rmse}(\mathcal{S},\mathcal{S}') = \sqrt{\frac{1}{|\mathcal{S}|}\iint_{p\in\mathcal{S}} d(p,\mathcal{S}')^2 dS}. \tag{4}$$

Using equation (2), it is possible to define symmetrical versions of the mean and root-mean-square errors.

The computation of such quantities in the case of discrete models is rather simple, provided the error values (see figure 3) have been computed for each sample. The integral of the (squared) error over the whole surface is computed by summing the contributions of all the parallelograms formed by 4 samples (see figure 3), plus the boundary triangles. Let us denote by $x_{i,j}$, $x_{i+1,j}$, $x_{i,j+1}$, $x_{i+1,j+1}$ four samples inside a triangle, and by $e_{i,j}$, $e_{i+1,j}$, $e_{i,j+1}$, $e_{i+1,j+1}$ the error value associated with each sample. The integral of e over the parallelogram formed by the x samples can be divided into two triangles as shown in figure 4. Let us now focus on the integral of e over the triangle $T_{i,j} = (x_{i,j}, x_{i+1,j}, x_{i,j+1})$. The simplest way of computing the integral of the error is to linearly interpolate between the values $e_{i,j}$, $e_{i+1,j}$ and $e_{i,j+1}$. A property of the sampling method proposed in section 2.2 is that inside each triangle from $\mathcal{T}$, the samples are easily triangulated, as shown in figure 4, and all the resulting triangles have the same area. The value of the integral is then $|T_{i,j}| \cdot \frac{e_{i,j}+e_{i+1,j}+e_{i,j+1}}{3}$. The integral of e^2 over the same triangle is also computed using linear interpolation between the values of e (which leads to quadratic interpolation between the values of e^2), and finally the value of the integral is :

$$\frac{|T_{i,j}|}{6}\Big[e_{i,j}(e_{i,j}+e_{i+1,j}+e_{i,j+1})+e_{i+1,j}(e_{i+1,j}+e_{i,j+1})+e_{i,j+1}^2\Big].$$

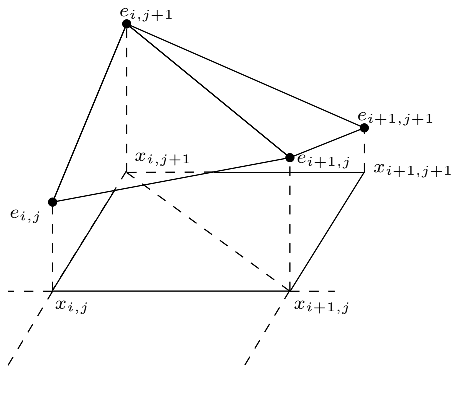

Figure 4: Computation of the integral of e over the surface.

3. IMPLEMENTATION

The algorithm outlined in the previous section has been implemented in the `Mesh` tool using the C/C++ language, using double precision floating-point arithmetic. Although straightforward, the algorithm becomes too complex if implemented naively. In fact, for each sample point p it would be necessary to calculate the distance to all triangles $T \in \mathcal{T}'$ in order to find the minimum. This leads to a complexity $\mathcal{O}(N_p N_{\mathcal{T}'})$, where N_p is the number of sample points taken on $\mathcal{M}$ and $N_{\mathcal{T}'}$ is the number of triangles of $\mathcal{T}'$. While this could be acceptable for small models, it becomes unmanageable for large models and a large number of samples. In order to reduce the complexity it is necessary to reduce the number of point-triangle distances that need to be evaluated, as explained below. The sample points are obtained by sampling the triangles $T \in \mathcal{T}$ according to the value n (cf. section 2.2). It is desired to have a uniform sampling density of $1/\delta^2$, where δ is the sampling step (fixed by the user as a percentage of the diagonal length of the model's bounding box). The sampling frequency should then be $n' = \sqrt{1/4 + 2|T|/\delta^2} - 1/2$, where $|T|$ is the area of T. In general n' is not integer. We thus randomly choose $n = \lfloor n' \rfloor$ or $n = \lfloor n' \rfloor + 1$ for each triangle, with probability ρ and $1 - \rho$, respectively. The probabilities are chosen such that the expected value of the resulting sampling density, $E\big[n(n+1)/(2|T|)\big]$, is $1/\delta^2$. Solving for ρ gives $\rho = \lfloor n' \rfloor/2 + 1 - |T|/\big[\delta^2(\lfloor n' \rfloor + 1)\big]$.

3.1. Hausdorff distance evaluation

One effective technique to achieve a large reduction of the number of point-triangle distance evaluations, also used in [3], is to use a uniform grid. The joint bounding box of $\mathcal{M}$ and $\mathcal{M}'$ is partitioned into non-overlapping cubic cells of side-length Δ. The set of triangles $\mathcal{T}'$ is indexed by constructing, for each cell C, the list of triangles intersecting it. For each sample point $p \in \mathcal{M}$ we need to calculate the minimum distance to the triangles $T' \in \mathcal{T}'$. Let $\bar{C}$ be the cell in which p is located and $\mathcal{D}_l(\bar{C})$, $l \in \mathbb{N}$, the set of cells that are at a distance $l\Delta$ from $\bar{C}$ along one of the coordinate axes. The sets $\mathcal{D}_l(\bar{C})$ can be thought of as layers of a cube centered around $\bar{C}$. First the distances from p to all the triangles intersecting $\bar{C}$ are calculated, and their minimum retained. Let $\hat{d}_p$ denote this current minimum. Then, for each value of l we consider each of the cells in $\mathcal{D}_l(\bar{C})$. If a cell C is farther from p than $\hat{d}_p$, it is skipped. Otherwise the distances from p to all triangles intersecting C are calculated and $\hat{d}_p$ updated accordingly. This procedure is performed for increasing values of l until $\hat{d}_p$ is smaller than $l\Delta$. At this point $\hat{d}_p$ is the minimum distance from p to all of the triangles $T' \in \mathcal{T}'$.

One important parameter is Δ, which determines the total number of cells. It should be chosen so that there is a good equilibrium between the speedup gained by using cells and the overhead of handling them. We define an average triangle $\bar{T}'$ that is the equilateral triangle having an area equal to the average triangle area of $\mathcal{T}'$. Then we set $\Delta = c\mu$, where μ is the side length of $\bar{T}'$ and c a fixed constant. Empirically we have found $c = 1$ to be a good value, suitable for a wide variety of models. Another important aspect is that, since $\mathcal{M}'$ represents a surface, the majority of cells are empty (i.e. they have no intersecting triangles). The emptiness of a cell is evaluated very often, in particular when $\mathcal{M}$ and $\mathcal{M}'$ differ significantly, and it is thus worthwhile having a fast method of doing it. To this end we construct a bitmap indicating empty cells. The bitmap has usually a small memory footprint, even for large models, allowing it to stay almost permanently in the computer's L2 memory cache and thus providing a very fast evaluation method. Using this approach, no more than 5 to 50 triangles are tested per sample, instead of thousands.

3.2. Point to triangle distance evaluation

The use of the cell partition greatly reduces the number of point-triangle distance evaluations. However, it is still necessary to be able to evaluate them in a fast way. Let a, b and c be the vertices of the triangle T' and p' the orthogonal projection of the sample point p on the plane of T'. If T' has an obtuse angle, let c denote the vertex at that angle. Let also $\mathbf{p}$ denote the vector from the origin to p and $\mathbf{ab}$ the vector from a to b. As explained in section 2.2, it is necessary to determine if p' is inside T' and if not, which side is the closest. To do this we consider the three planes through the sides of T' and perpendicular to T'. The orientation of the plane through the ab side is given by the normal vector $\mathbf{n}_{ab}$, perpendicular to ab and parallel to the plane of T'. Analogously, the other two perpendicular planes have normal vectors $\mathbf{n}_{ac}$ and $\mathbf{n}_{bc}$. For the general case where T' can have an obtuse angle we consider also the plane perpendicular to bc through c. With these four planes and their intersections we can define five regions of the space, as depicted in figure 5. If T' has an obtuse angle $\widehat{acb}$, any p in region 3 is closest to the ac side. Otherwise any p in region 3 is closest to the bc side and there is no need to distinguish between regions 2 and 3.

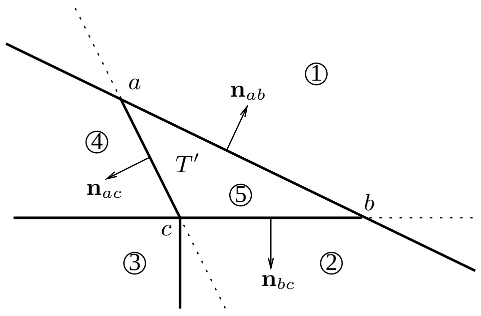

Figure 5: Top view of a triangle T', the side normals and the five regions (delimited by the solid lines) used to efficiently calculate the point to triangle distance.

The following procedure, requiring no more than three scalar products and three comparisons, can be applied to determine to which of the above mentioned regions the sample point p belongs. If $\mathbf{p}.\mathbf{n}_{ab}$ is greater than $\mathbf{a}.\mathbf{n}_{ab}$, p is in region 1. Otherwise, if $\mathbf{p}.\mathbf{n}_{bc}$ is greater than $\mathbf{b}.\mathbf{n}_{bc}$, p is in region 2 or 3. Otherwise, if $\mathbf{p}.\mathbf{n}_{ac}$ is greater than $\mathbf{c}.\mathbf{n}_{ac}$, p is in region 4 and else in 5. If required, the scalar product $\mathbf{cp}.\mathbf{cb}$ determines if p is in region 2 or 3. All the quantities, such as $\mathbf{n}_{ab}$ and $\mathbf{a}.\mathbf{n}_{ab}$, that do not depend on p are precalculated for each triangle before starting the Hausdorff distance computation, since the distance to any given triangle is computed many times for different sample points.

Now that we know to which region p belongs, we need to calculate the distance. If in region 5, $p' \in T'$ and thus the distance is the point to plane distance. Otherwise we need to determine the distance from p to one of the sides of T'. For example, for region 1 we must determine the distance from p to the ab side. Let p'' be the orthogonal projection of p on the ab line. If p'' is between a

and b, the distance is simply the point to line distance. Otherwise it is the distance to the closest point, either a or b. This can be determined by evaluating $\mathbf{ap}.\mathbf{ab}$. If negative, a is the closest point. If positive and larger than $\|ab\|^2$, b is the closest point. Otherwise we compute the point to line distance. For the other regions (2, 3 and 4) the procedure is analogous.

3.3. Results

After reading the two input models $\mathcal{M}$ and $\mathcal{M}'$, `Mesh` first outputs some simple information about them (number of triangles and vertices and bounding box diagonal length) as well as their topological nature (2-manifold, orientable, closed and number of disjoint parts). Then the forward Hausdorff distance is evaluated and the minimum, maximum, mean and root-mean-square error distances are reported. Optionally the backward and symmetric distances can also be evaluated. In addition, an OpenGL$^{\text{TM}}$ graphical user interface (GUI) displays the distribution of the Hausdorff distance on the model $\mathcal{M}$ and allows to interactively perform a visual comparison of the shape of the two models.

4. ANALYSIS

The `Mesh` tool has been evaluated on a variety of models, with good results. Figure 6 shows the results for the root-mean-square forward and backward distances, for two models: *horse* and *lion*. The *horse* model has 96966 triangles originally and has been simplified to 500 triangles using QSlim [4]. The *lion* model is a B-Spline parametric model that has been tessellated to 50820 and 7490 triangles. As expected, the sampling step δ plays a role in the precision of the measured distance. The measure is stable for values of δ below 0.5% or 0.4% of the bounding box diagonal. The difference between the forward and backward distances can also be observed, which demonstrates the need for a symmetrical metric. Another important point is that, for large δ, measures from a coarse to a fine model are less precise than the opposite. This can be explained by the fact that for models with a large number of triangles, a non-negligible proportion of the triangles are not sampled if a large δ is used, and thus relevant details in the models' shape are missed.

Figure 7 compares the execution time of `Mesh` and `Metro` (version 2.5) on a Silicon Graphics Onyx computer with R10000 194MHz MIPS processors. It can be easily observed that `Mesh` is between three to four times faster. In addition, the memory footprint is in general twice smaller. The figure also shows a limitation of `Metro`. If the sampled model has a relatively large number of faces, it is not possible to use a small number of samples. With `Mesh` it is possible to use a large δ, and thus a small number of samples, to get a rough distance estimate very quickly.

5. CONCLUSION

In this paper the Hausdorff distance and its application to distance measurements between 3D models have been introduced. Furthermore, `Mesh`, an efficient implementation of the Hausdorff distance for triangular meshes is presented and evaluated. The comparisons with `Metro` [3], a similar tool, show that `Mesh` is very fast, memory efficient and provides stable distance measures. `Mesh` is publicly available in source-code form and can be obtained on the World-Wide-Web at `http://mesh.epfl.ch`.

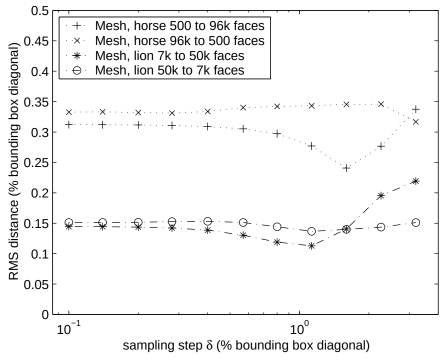

Figure 6: Evolution of the root-mean-square distance measure as a function of the sampling step δ. Forward and backward distances are reported. For each successive δ value, from left to right, the number of sample points is roughly halved.

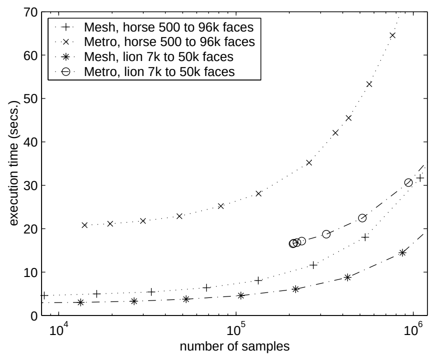

Figure 7: Comparison of the execution times of `Mesh` and `Metro`, as a function of the number of sample points.

6. ACKNOWLEDGMENTS

The authors would like to thank Davy Jacquet, who helped in developing the `Mesh` tool. The *horse* and *lion* models are courtesy of Cyberware, Inc. and the University of North Carolina, respectively. This research was partially supported by the EC-project Certimark.

7. REFERENCES

[1] S. Winkler, "Visual fidelity and perceived quality: toward comprehensive metrics," in *Human Vision and Electronic Imaging VI, Proc. of the SPIE*, 2001, vol. 4299, pp. 114–125.

[2] R. Klein, G. Liebich, and W. Strasser, "Mesh reduction with error control," in *IEEE Visualization '96 Proceedings*, 1996, pp. 311–318.

[3] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro : measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, June 1998.

[4] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH 97 Proceedings*, Aug. 1997, pp. 209–216.