A spatio-temporal metric for dynamic mesh comparison

Libor Vasa, Vaclav Skala

University of West Bohemia, Department of Computer Science and Engineering
Univerzitni 22, Pilsen, Czech Republic
{lvasa, skala}@kiv.zcu.cz

Abstract. A new approach to comparison of dynamic meshes based on Hausdorff distance is presented along with examples of application of such metric. The technique presented is based on representation of a 3D dynamic mesh by a 4D static tetrahedral mesh. Issues concerning space-time relations, mesh consistency and distance computation are addressed, yielding a fully applicable algorithm. Necessary speedup techniques are also discussed in detail and many possible applications of the proposed metric are outlined.

1 Introduction

Dynamic mesh extraction from multicamera recordings of real scenes has become a common task of computer graphics of these days. Algorithms running in real time are being developed and used in common practice, producing high quality dynamic meshes that can be used for all kinds of purposes, from 3D television to elaborate experimental techniques requiring exact measurement.

However, today’s hardware is still far from being powerful enough to handle the produced data in the raw form. Limited bandwidth is usually the main bottleneck, but also processing power and memory requirements may become difficult to meet.

Various techniques of data rate reduction of dynamic meshes are already appearing, usually involving some kind of lossy value compression scheme combined with some elaborate prediction technique [3,7,5]. One can also expect that there will appear techniques of geometry decimation of the dynamic mesh, similar to algorithms used for static mesh simplification [8].

The purpose of this contribution is to provide an objective methodology of comparing dynamic meshes. Such technique will be needed in order to compare and evaluate the compression methods and we will show that it may be used for other purposes as well.

2 Problem definition

The problem we will solve is defined as follows: Let there be given a set $S = \{M_k\}_{k=1}^{N}$ of dynamic meshes. A dynamic mesh M is a sequence of triangle meshes of constant connectivity, which may be produced by some extraction technique [11,12]. We want to define a function $d(M_1, M_2)$ that will be a metric in the space of dynamic meshes. Namely, we expect the following properties:

$$\begin{aligned}
&d(M_1, M_2) = 0 \Leftrightarrow M_1=M_2 \\
&d(M_1, M_2) = d(M_2, M_1) \\
&d(M_1, M_2) < d(M_1, M_3) \Leftrightarrow \text{A human observer sees } M_2 \text{ as “more similar” to } M_1 \text{ than to } M_3
\end{aligned} \tag{1}$$

Of these conditions is of course the last one the hardest to achieve.

In the past, research was done in the field of comparing static triangle meshes [2,10], the basic idea is quite simple and is based on the definition of Hausdorff distance of two objects. The Hausdorff distance is defined as follows:

Let’s have two static triangle meshes, m_1 and m_2. Distance of a point to a mesh is defined as a minimum of Euclidean distances of the given point p and all points p_m of the mesh m:

$$d_{p,m} = \min_{\forall p_m \in m}\left(\left|p - p_m\right|\right) \tag{2}$$

From this one can define a one-way (non-symmetric) distance of a mesh m_1 to a mesh m_2:

$$d'_{m_1,m_2} = \max_{\forall p_m \in m_1}\left(d_{p_m,m_2}\right) \tag{3}$$

A symmetric Hausdorff distance is then defined as

$$d_{m_1,m_2} = \max\left(d'_{m_1,m_2}, d'_{m_2,m_1}\right) = d_{m_2,m_1} \tag{4}$$

In the implementations of the Hausdorff distance evaluators both meshes are usually sampled in order to gain distance of a point to a mesh (usually some elaborate point to triangle distance test is used) and various acceleration techniques (space subdivisions etc.) are exploited in order to reduce the computational complexity that is quadratic in the raw form of the definition.

Our approach is to adopt the Hausdorff distance and use it for comparison of dynamic meshes that will be represented by static objects on 4D. In order to do so, we will have to address several problems that arise with the higher dimension of the problem.

3 Human perception of time considerations

The Hausdorff distance measurement is based on the concept of the Euclidean distance. In 3D space there is no problem with units as long as the same units are used for all axes. However, in 4D we cannot use equal units, as one of the dimensions is time. Therefore we must answer the question which units should be used.

The key to the answer is the definition of the desired metric. It implies, that equal distance on each axis should cause equal disturbance in the mesh. It is important to realize that human perception of time is quite absolute and it is actually the spatial metric that causes problems. In computer graphics modeling it is quite usual to work with vaguely defined spatial units, while time is measured absolutely. Therefore, the question actually is "what spatial distance is equal to the given time span in the terms of human perception".

The problem is that distance of one unit may cause distance of half a screen in one model, as well as being barely distinguishable in some other model. One solution would be to consider the distance of point projections on human retina, but this distance also depends on the size of used screen.

Therefore we use a "relative distance", defined as distance in the model units divided by the size of the model's body diagonal. The task now is to find the coefficient alpha that will relate the relative distance to time units. In order to do so, we will have to perform subjective testing, but for the time of being we can do following considerations:

1. time span of 1/100s is almost unrecognizable for a human observer, while spatial shifts of 10% is on the limit of acceptability, therefore we expect alpha to be larger than 0.01/0.1 = 0.1
2. time spans of units of seconds are on the limit of acceptability, while spatial shift of 0.1% is almost unrecognizable, therefore we expect alpha to be smaller than 1/0.001 = 1000

Saying that, we can guess the value of the alpha coefficient to be about 10, i.e. time span of 100ms is equal to spatial shift of 1%.

4 Dynamic mesh as a static 4D object

We have mentioned that in order to use the Hausdorff distance concept we have to represent the dynamic mesh as a static object in 4D. As static mesh in 3D consists of triangles, which are elements one-dimension lower than the dimension of the 3D space, in 4D we will represent the dynamic mesh by a static tetrahedral mesh. Note that tetrahedron is not a simplex in 4D.

We can extract one frame from such mesh by cutting it by a plane t=const, because a tetrahedron cut by a plane gives one or two triangles. The procedure that converts a dynamic triangle mesh into a 4D tetrahedral mesh is based on the idea, that a triangle in two consequent frames forms a prism in 4D (see figure 1). The process of conversion is therefore simply a process of breaking such prisms into tetrahedra. Each prism can be divided into three tetrahedra.

However, we must be very careful about the breaking. One can see that the sides of the prisms are not planar, and therefore we must explicitly make sure that the mesh we are creating will be continuous. Namely, we must make sure that a side diagonal in neighboring prisms is always equal. In order to do so, we propose the following subdivision procedure:

1. find a vertex on the base of the prism with lowest index. Create a tetrahedron that is formed by the whole top of the prism and this vertex.
2. find a vertex on the base of the prism with the largest index. Create a tetrahedron that is formed by the whole base of the prism and the vertex above the vertex with largest index.
3. create a tetrahedron formed by remaining two vertices on the base and two vertices on the top of the prism.

Because the relations of largest/lowest index are kept on each face, one can see that the created tetrahedral mesh is consistent.

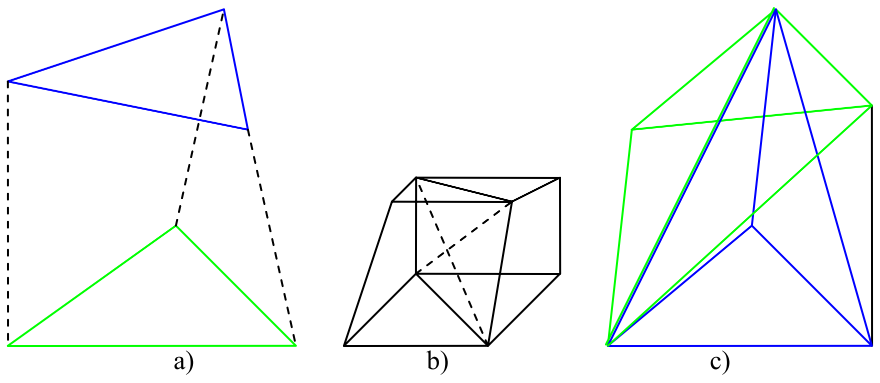

Fig. 1. Moving triangle as a 4D prism (green is the triangle in time t, blue is the triangle in time t+1), two possible diagonals on a common side, two tetrahedra used for consistent subdivision

5 Point to tetrahedron distance test

Our distance algorithm is based on the point to tetrahedron distance test. A distance to a tetrahedron may be in fact distance to one of following entities:

1. distance to the body of the tetrahedron. Is only possible when the orthogonal projection of the point lies within the tetrahedron
2. distance to a face of the tetrahedron. Is only possible when the orthogonal projection of the point to the plane of the face lies on the face
3. distance to an edge of the tetrahedron. Is only possible when the orthogonal projection of the point to the line of the face lies on the edge
4. distance to a vertex of the tetrahedron

Of these distances we must choose the lowest that meets its projection conditions.

5.1 Distance to the body of a tetrahedron

A tetrahedron is defined by three 4D vectors of Euclidean coordinates, defined as follows:

$$\begin{aligned} v_0 &= T_1 - T_0 \\ v_1 &= T_2 - T_0 \\ v_2 &= T_3 - T_0 \end{aligned} \tag{5}$$

Therefore a tetrahedron in 4D has a normal vector n:

$$n = v_0 \times v_1 \times v_2 \tag{6}$$

Note that we are using cross product that is a ternary operator in 4D.
Any point P can now be expressed as follows:

$$P - T_0 = a\vec{v}_1 + b\vec{v}_2 + c\vec{v}_3 + d\vec{n} \tag{7}$$

We can find the combination coefficients by solving a 4x4 set of linear equations, for example using Sarus rule. The projection to the tetrahedron space lies within the tetrahedron if following conditions hold:

$$a \ge 0, b \ge 0, c \ge 0, (a + b + c) \le 1 \tag{8}$$

In such case the distance can be expressed as $d^*|\mathbf{n}|$.

5.3 Distance to a face of a tetrahedron

The key feature one must consider is that a face in 4D (and a plane in general) has not a uniquely defined normal. Therefore, we must find a normal that is orthogonal to the face, and that passes through the evaluated point P. In order to do so, we can use the following procedure:

Let's have T_0, T_1 and T_2 vertices that define a face of the tetrahedron, and a point P.

$$\begin{gathered} v_0 = T_1 - T_0 \\ v_1 = T_2 - T_0 \\ p = P - T_0 \end{gathered} \tag{9}$$

we can now find a vector **b** that is orthogonal to all three vectors by using cross product

$$\mathbf{B} = \mathbf{v_0} \times \mathbf{v_1} \times \mathbf{p} \tag{10}$$

A normal vector n that can be used for the projection can be found as

$$\mathbf{N} = \mathbf{v_0} \times \mathbf{v_1} \times \mathbf{b} \tag{11}$$

Now we can use a similar procedure to find where the projection lies. We can write

$$\mathbf{p} = a*\mathbf{v_0} + b*\mathbf{v_1} + c*\mathbf{b} + d*\mathbf{n} \tag{12}$$

where we expect the c coefficient to be zero. If now $a \geq 0, b \geq 0$ and $(a+b) \leq 1$ then the projection lies on the face, and the distance is d*|n|.

5.3 Distance to an edge of a tetrahedron

For the distance of an edge one can use the properties of dot product that hold in 4D space. Let's define

$$\begin{gathered} v_1 = E_1 - E_0 \\ v_2 = P - E_0 \end{gathered} \tag{13}$$

It is well known that the line of the edge can be written as $E_0 + t * v_1$.

In order to determine the distance, we would like to find the t parameter of an orthogonal projection of P to the line. One can derive that t can be determined as follows:

$$t = (v_1.v_2)/(v_1.v_1) \quad (14)$$

From the known value of t we can easily determine whether the projection lies on the edge (0<=t<=1) and eventually express the distance as

$$d = \sqrt{v_1.v_1 - t^2 v_2.v_2} \quad (15)$$

6 Acceleration techniques

The distance tests shown above work for all kinds of tetrahedra (i.e. including obtuse tetrahedra), but may be very slow when each tested point is to be evaluated against each tetrahedron of the other mesh.

Our first acceleration technique is based on the following observation: A point can be projected to a face only if it is projected on at least two of the edges that define the face (for obtuse faces). Based on this idea we evaluate all edges before the faces. During the evaluation we increase a counter for each face if a point is projected to an edge that incides with the face (two counters representing two incident faces are increased whenever a point is found to be projected on an edge). A face is then only evaluated if its counter is larger or equal to two.

From the previous equations one can see, that evaluating an edge consists only of two dot products, one division and two comparisons, while evaluating the face includes solving a 4x4 set of linear equations (12). Moreover, the case when a point is projected to a face of a tetrahedron is a rare one. Therefore this simple technique provides a significant speedup of more than 50%. A further speedup can be achieved by postponing the square root operation that is part of each distance evaluation, to the latest possible moment, while keeping the square distances.

We are also utilizing spatial subdivision techniques in order to reduce the computational complexity. In a preprocessing stage we create a 4D grid of cells, where each cell holds a list of tetrahedra that intersect with the cell.

The usual approach determining which cells are intersected by some entity is to find a bounding box of the entity and mark all cells of the bounding box. Because the grid we are using is 4D, this would lead to unnecessary marking of many empty cells. Therefore, we have developed an improved technique based on the following observation: Each tetrahedron has its uniquely defined 3D space with a normal. This space is a hyperplane in 4D that divides the time-space into parts “above” and “below”. We can evaluate each corner of a cell according to whether it is above or below the tetrahedron (each cell has 16 corners, it can be imagined as a hypercube). Only cells that are neither completely below nor completely above the tetrahedron can be intersected by the tetrahedron. In our experiments, including all the cells of a bounding box of a tetrahedron, have lead to an average of approximately 35 cells per tetrahedron (for given tetrahedral mesh and grid density), while keeping only the cells that satisfy our condition has reduced this number to cca. 8 cells per tetrahedron and led to a speedup of about 30% (including the preprocessing stage).

In the evaluation stage, a cell that contains the evaluated point is found and searched for possible closest tetrahedron. Further cells are subsequently evaluated only if they can provide a tetrahedron that is closer than the already found one, i.e. only if the closest point of the cell is closer than the current distance. This technique vastly improves the performance, depending on the density of the grid.

We have also included precomputation of reused values and some further improvements (usage tables that show which faces and edges were already evaluated for a given point etc.). Our implementation is capable of evaluating about 180 V-M distances per second, where the mesh consists of about 120 000 tetrahedra. This allows us to compute distances of moderately complex animations within minutes, larger animations still must be evaluated offline (hours of processing time are needed).

7 Applications

We have already shown the main application of the proposed metric, it is comparison of dynamic meshes decimated by various methods, but it is not the only field where comparison of animations can be used.

Another natural area where this technique can be used is artificial intelligence, where the metric can be used to recognize various actions and to respond to them. In our experiments we have compared two recordings of a human jump [11,12], and we have found that the distance of one jump sequence to the other is significantly smaller than the distance of a jump to the sequence that represents the human walking. Each frame of the human jump sequence consists of about 30 000 triangles, and we have compared 50 frame subsets of the sequence.

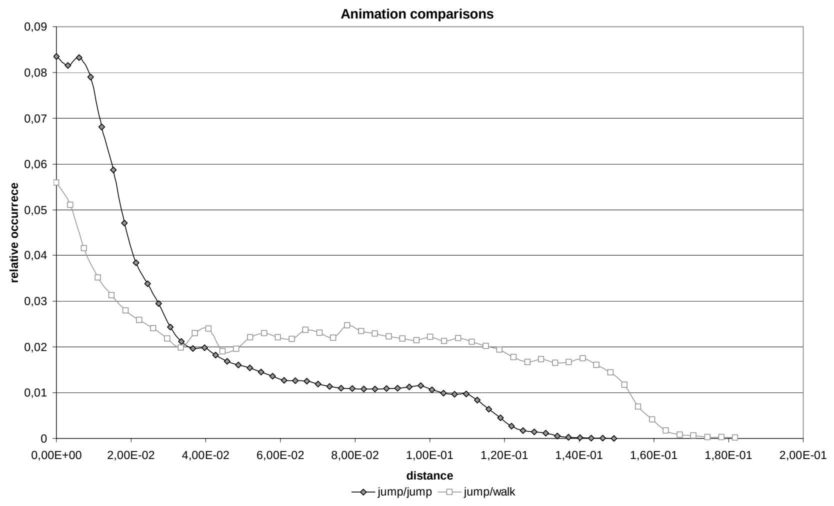

Fig. 2. Distance distribution experiments

The figure 2 shows the histogram of the measured distances for the above mentioned experiments. In order to compare longer time spans we have compared tetrahedral meshes that consist of only every other mesh of the animation, effectively reducing the frame-rate of the animation to one half of its original value.

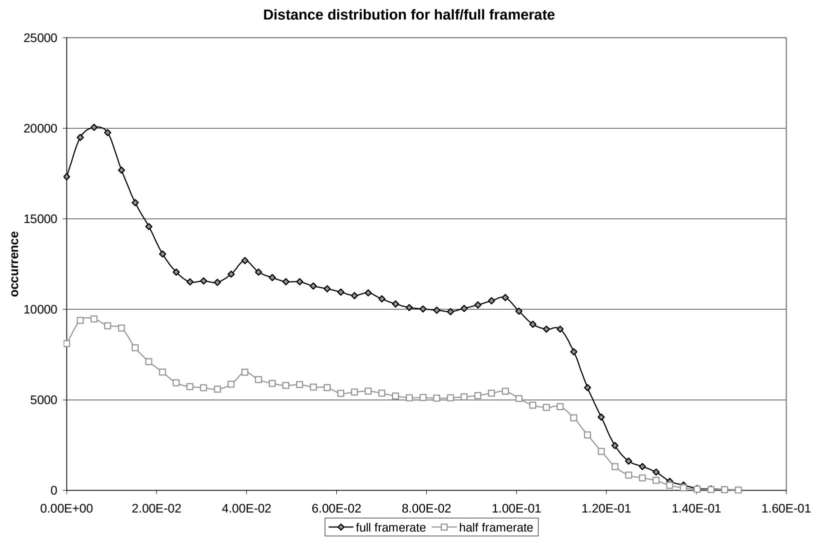

Fig. 3. Full/half frame-rate experiments

Our other experiment shows that such frame-rate reduction is possible, because it does not disturb the characteristics of the distribution of the error. Figure 3 shows the relative histogram of distance values for full frame-rate comparison and half frame-rate comparison of equal time span of an animation. The average difference is less than 4%, and the half-frame-rate curve keeps all the characteristics of the full frame rate. However, this is only possible when animation recognition is considered. The difference of 4% may be unacceptable when exact comparison for decimation evaluation is considered.

Another application is obvious from the previous one – the animation metric can be used to align animations in both time and space at the same time. This would require some slight changes in the software in order to look for average distance vector rather than maximum distance size but this can be done very easily.

One can also easily imagine applications like self-training, where the user would try to fit with her movements to some predefined pattern. Our method can then use rendering of the distance of the movements represented by surface colors that would tell the trainee where and when exactly she was following the pattern well or not. This technique can be used in wide range of areas from dance up to surgery training.

8 Future work

The proposed algorithm is still computationally expensive; therefore we will put effort into acceleration techniques that would make its use easier and more comfortable.

We would also like to further develop the idea of representing a dynamic mesh by a static mesh in 4D and propose a decimation method based on this representation and some tetrahedral mesh decimation algorithm provided with appropriate criteria.

Acknowledgements

The authors would like to thank Mark Dobbs for proof reading. This project was supported by the 6FP NoE project 3DTV – Integrated Three-Dimensional Television – Capture, Transmission and Display No. 511568 and MŠMT ČR project 1P04LA240.

References

1. Chopra, P., Meyer, J.: Tetfusion: An algorithm for rapid tetrahedral mesh simplification. In Proc. IEEE Visualization, pages 133--140, 2002.
2. Cignoni, P., Rochini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, Istituto I.E.I. - C.N.R., Pisa, Italy, January 1996.
3. Coors, V., Rossignac, J.: Delphi: Geometry-based Connectivity Prediction in Triange Mesh Compression. The Visual Computer 20(8-9): 507-520 May 2004.
4. Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes, IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1, January - March 1999.
5. Müller, K., Smolic, A., Kautzner, M., Eisert, P., Wiegand, T.: Predictive Compression of Dynamic 3D Meshes. Proc. International Conference on Image Processing (ICIP 2005), Genova, Italy, pp., September 2005.
6. Bayazit, U., Orcay, O., Gurgen, F.: Predictive Vector Quantization of 3D polygonal mesh geometry by representation of vertices in local coordinate system. Proc. of EUSIPCO 2005.
7. Ibarria, L., Rossignac, J.: Dynapack: Space-Time compression of the 3D animations of triangle meshes with fixed connectivity. Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2003.
8. Franc, M.: Methods for Polygonal Mesh Simplification. Internal technical report at University of West Bohemia, 2003.
9. Gumhold, S., Guthe, S., Straer, W.: Tetrahedral Mesh Compression with the CutBorder Machine. In Proceedings of the 10th Annual IEEE Visualization Conference, 1999.
10. Aspert, N., Santa-Cruz, D., Ebrahimi, T.: Mesh: Measuring errors between surfaces using the hausdorff distance. In Proceedings of the IEEE International Conference on Multimedia and Expo, volume I, pages 705--708, 2002.
11. Anuar, N., Guskov, I.: Extracting Animated Meshes with Adaptive Motion Estimation. Proc. of the 9th International Fall Worksop on Vision, Modeling, and Visualization, 2004.
12. Sand, P., McMillan, L., Popovic, J.: Continuous Capture of Skin Deformation. ACM Transactions on Graphics. 22(3), pp. 578-586, 2003.