A VTK Algorithm for the Computation of the Hausdorff Distance

Release 1.0

Frédéric Commandeur, Jérôme Velut, Oscar Acosta

September 22, 2011

INSERM, U 642, Rennes, F-35000, France
Université de Rennes 1, LTSI, F-35000, France
frederic.commandeur@etudiant.univ-rennes1.fr
jerome.velut@univ-rennes1.fr
oscar.acosta@univ-rennes1.fr

Abstract

The Hausdorff distance is a measure of the distance between sets of points. There are many advantages to using this metric compared to other similarity measures. This document describes a VTK class for computing the Hausdorff Distance between two sets of points. The main contribution, compared to other implementations, lies in the definition of the distance not only to the closest point but to the closest point in the represented surface, which yields an accurate measure even between undersampling surfaces. This is achieved by implementing a point-to-cell distance instead of a point-to-point. Furthermore, a plugin for ParaView was implemented, which is also available with the code. After introducing the interest of this distance, the VTK code is explained and illustrated with some examples.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3251]
Distributed under Creative Commons Attribution License

Contents

1 Hausdorff distance: theoretical background

The Hausdorff distance [3] has been used in many applications to compute distance between sets of points. It can be used as a similarity measure for detection and tracking [2], registration [5] or validation [4]. As it is based on the Euclidean norm it is dimension-independant. It is always positive and it is zero only when the shapes to be compared are the same. Our implementation builds upon the definition proposed in [1] as follows.

1.1 Point to surface distance definition

Let p a point of $\mathbb{R}^3$ and $\mathcal{S}$ a 2-dimensional surface embedded in $\mathbb{R}^3$. The distance δ from p to $\mathcal{S}$ is defined as

$$\delta(p,\mathcal{S}) = \inf_{q\in\mathcal{S}} ||p-q|| \tag{1}$$

where $||.||$ corresponds to the Euclidean norm and q is a point of $\mathcal{S}$.

1.2 Relative distances between two surfaces

Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two 2-dimensional surfaces embedded in $\mathbb{R}^3$ and p_1 a point belonging to $\mathcal{S}_1$. The equation (1) allows to define a surface-to-surface relative distance $\Delta(\mathcal{S}_1,\mathcal{S}_2)$ as

$$\Delta(\mathcal{S}_1,\mathcal{S}_2) = \sup_{p1\in\mathcal{S}_1} \delta(p1,\mathcal{S}_2) \tag{2}$$

This distance is said relative as it is not symmetrical, *ie.* $\Delta(\mathcal{S}_1,\mathcal{S}_2) \neq \Delta(\mathcal{S}_2,\mathcal{S}_1)$.

1.3 Hausdorff distance

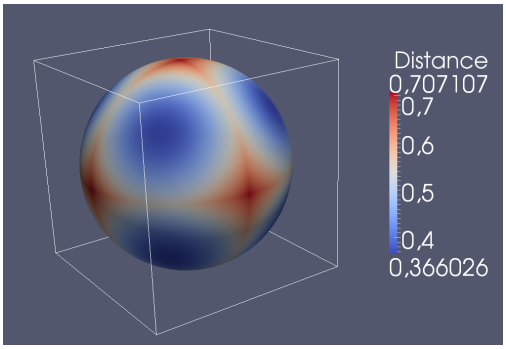
Finally, the Hausdorff distance d between two surfaces $\mathcal{S}_1,\mathcal{S}_2$ is defined as the maximum of the two relative distances:

$$d(\mathcal{S}_1,\mathcal{S}_2) = \max\{\Delta(\mathcal{S}_1,\mathcal{S}_2),\Delta(\mathcal{S}_2,\mathcal{S}_1)\} \tag{3}$$

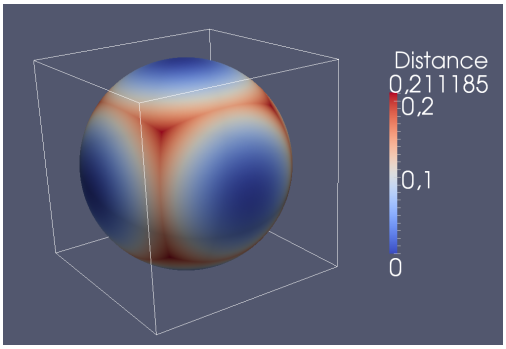
2 VTK Implementation

Our Hausdorff distance implementation is designed as a `vtkPointSetAlgorithm`-inherited class called `vtkHausdorffDistancePointSetFilter`. It takes two `vtkPointSets` as inputs: $input_A$ on input port 0

Distributed under Creative Commons Attribution License



(a) point-to-point computation method
$\Delta(\mathcal{S},\mathcal{C}) = 0.707$
$\Delta(\mathcal{C},\mathcal{S}) = 0.366$
$d(\mathcal{C},\mathcal{S}) = 0.707$



(b) point-to-cell computation method
$\Delta(\mathcal{S},\mathcal{C}) = 0.211$
$\Delta(\mathcal{C},\mathcal{S}) = 0.366$
$d(\mathcal{C},\mathcal{S}) = 0.366$

Figure 1: Point to surface distance δ computed between a cube and its inscribed sphere

and $input_B$ on input port 1.

Inputs are first copied on corresponding output ports. The algorithm computes then the Hausdorff distance $d(input_A, input_B)$ and the Relative distance $\Delta(input_A, input_B)$ and $\Delta(input_B, input_A)$. These distances are stored in the `FieldData` of both outputs. The `PointData` array of each output contain the points-to-surface distances $\delta(p_{input_A}, input_B)$ and $\delta(p_{input_B}, input_A)$.

2.1 Target distance

Although the mathematical definition of the Hausdorff distance given in section (1) is referred to $\mathbb{R}^3$ surfaces, the proposed implementation allows to perform comparisons between point sets. It is up to the user to choose which element from the compared object should be targeted: either points or cells. The choice can be made through the `TargetDistanceMethod` accessors:

```
int TargetDistanceMethod; //!< point-to-point if 0, point-to-cell if 1
vtkSetMacro( TargetDistanceMethod, int );
```

A point-to-cell distance implies the existence of cells in both inputs. This is typically the case with `vtkPolyData` representing polygonal surfaces (*eg.* triangulations). When such a method is used, the point-to-surface δ computation performs an geometric interpolation of the targeted cells, leading to a more precise distance calculation. Figure 1 illustrates the large differences between them (point-to-point and point-to-cell), considering the distance of a sphere $\mathcal{S}$ inscribed in a cube $\mathcal{C}$. A large difference appears between the two cases. In figure 1(a), the δ distances are computed between the sphere vertices and the cube vertices. It justifies the $\frac{\sqrt{2}}{2}$ minimum distance found in the cube face centers. In contrast, figure 1(b) shows that the cube face centers have a null δ thanks to the geometric interpolation.

2.2 Computational time optimization

In the Hausdorff distance computation, the bottle-neck in term of algorithmic complexity resides in the relative distances computation. Indeed, finding the minimum distance from each point p_1 in $\mathcal{S}_1$ to each point

Distributed under Creative Commons Attribution License

p_2 in $\mathcal{S}_2$ yield to an $O(m \cdot n)$ theoretical complexity where m (resp. n) is the number of points in $\mathcal{S}_1$ (resp. $\mathcal{S}_2$).

The proposed implementation harnesses the `vtkLocator` space-partitioning classes. The minimum point-to-point distance method uses `vtkKdTreePointLocator`, which decreases the complexity to $O(n\log(m))$. Similarly, the point-to-surface method involved an octree space partitioning prior to minimum distance computation as implemented in `vtkCellLocator`. For illustration, computing the Hausdorff distance between two point sets having each $1,000,000$ of points took $12s$ on a $2.27GHz$-cadenced processor.

2.3 ParaView plugin

The provided class can be build as a ParaView plugin by setting `BUILD_PARAVIEW_PLUGIN` option to ON during the CMake procedure.

A `vtkHausdorffDistancePointSetFilter` dynamic library will be created in the `bin` directory where the `ccmake` command was executed. This library is loadable from ParaView:

```
"Tools" -> "Manage Plugins" -> "Load New" -> vtkHausdorffDistancePointSetFilter.so/dll
```

It will create a `Hausdorff Distance` submenu in the `Filters` menu.

3 Result

In this section, we show a use case of our Hausdorff distance VTK algorithm. A common task in image processing and visualisation is to extract an isosurface from a binary volume. The quality of the extracted surface will strongly depend on the resolution of the volume. In figure 2, we show an extracted sphere at different coarsening levels. The Hausdorff distance from these surfaces to a ideal sphere (figure 3(a)) gives a quality measure that tends to zero when the sampling becomes finer. This is illustrated in the graph, figure 3(b).

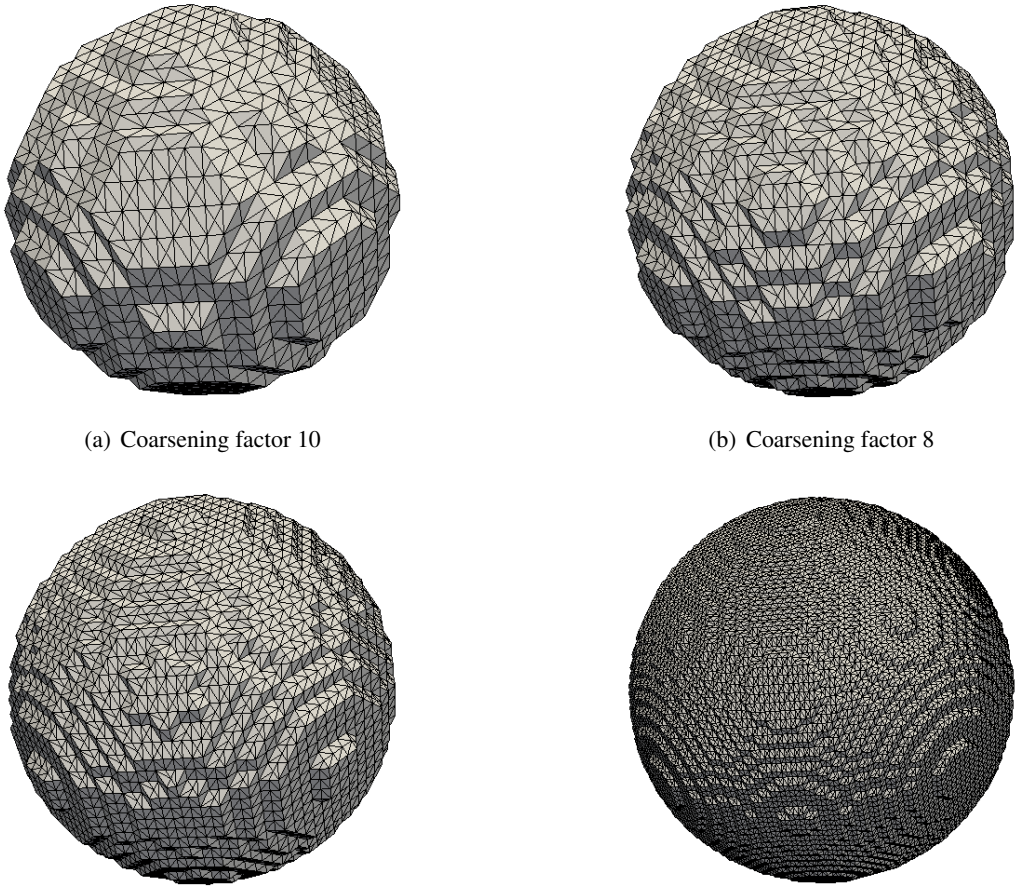
4 Software Requirements

This software was built on Ubuntu 11.04 and Fedora 14 against ParaView-3.10.1, git master at the time of submission and VTK-5.6.1.

5 Conclusion

We presented a class to compute the Hausdorff distance between two point sets. This distance is appropriate to quantify the similarity of surfaces. The provided class is inherited from `vtkPointSetAlgorithm` and can be executed in command line or as a ParaView plugin.

Distributed under Creative Commons Attribution License



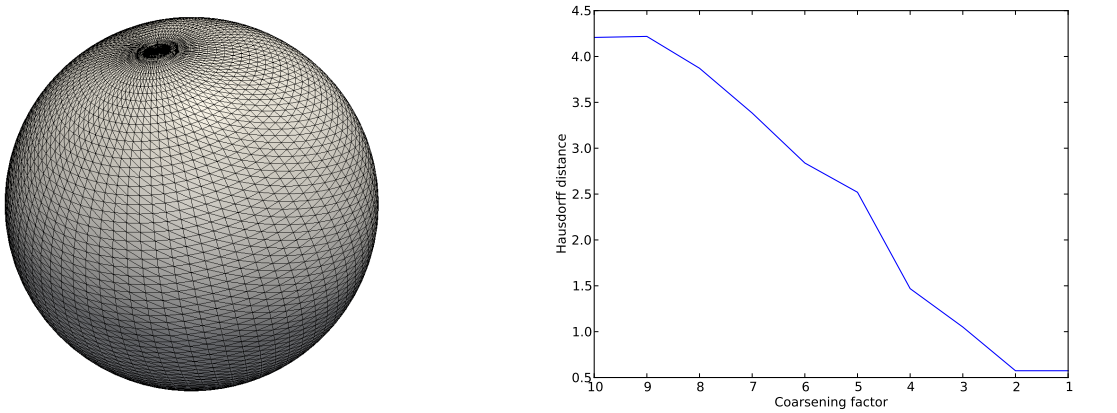
(a) Coarsening factor 10

(b) Coarsening factor 8

(c) Coarsening factor 6

(d) Coarsening factor 4

Figure 2: Isocontours of binary sphere with different resolutions. Initial volume (coarsening factor= 1) is $512 \times 512 \times 512$



(a) Ideal sphere

(b) Hausdorff distance between ideal sphere and isocontours

Figure 3: Ideal sphere and its hausdorff distances from isocontours of figure 2

References

[1] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. 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. http://mesh.epfl.ch. 1

[2] P. Gastaldo and R. Zunino. Hausdorff distance for target detection. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 5, pages V–661 – V–664 vol.5, 2002. 1

[3] Felix Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79:157–179, 1918. 10.1007/BF01457179. 1

[4] Arno Klein, Satrajit S Ghosh, Brian Avants, B. T T Yeo, Bruce Fischl, Babak Ardekani, James C Gee, J. J. Mann, and Ramin V Parsey. Evaluation of volume-based and surface-based brain image registration methods. *Neuroimage*, 51(1):214–220, May 2010. 1

[5] Jian-Wei Zhang, Guo-Qiang Han, and Yan Wo. Image registration based on generalized and mean hausdorff distances. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 8, pages 5117 –5121 Vol. 8, aug. 2005. 1

Distributed under Creative Commons Attribution License