Homework #3: Shape symmetry detection [100 points]
Due Date: Wednesday, 3 June 2009

No late days are possible for this assignment.

- **The Programming Problem**

Problem 1. [100 points]

For this assignment, you are required to implement an algorithm for detecting reflection and rotational symmetries in 3d geometry. The details of the algorithm are summarized below. Your grade will depend on the quality of the results and on your understanding of your implementation.

Overview

The algorithm described here is based on the paper "Partial and Approximate Symmetry Detection for 3D Geometry" [6]. However, you are welcome to modify it and even choose your own algorithm if you are able to detect the correct symmetries.

Roughly speaking, the process of symmetry detection can be divided into three steps (See Fig.1). In the first step, a set of samples or feature points is selected from the input surface. In the second step, correspondences between these samples or feature points are established based on local geometry descriptors. Then these correspondences are grouped into clusters so that correspondences in each cluster are consistent with each other, in the sense that they can be mapped to each other via the same transform. Each cluster of correspondences represents a candidate symmetry, which is verified for validity in the third and final step.

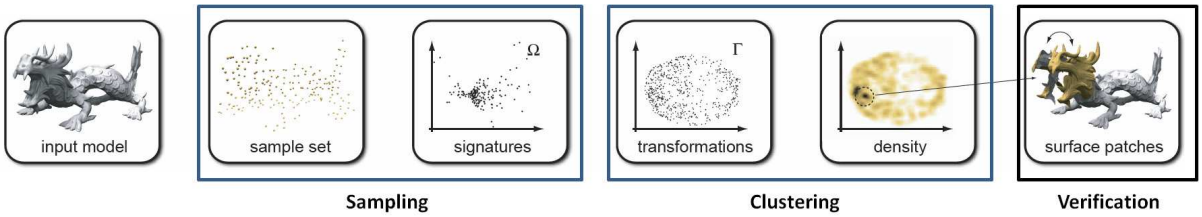
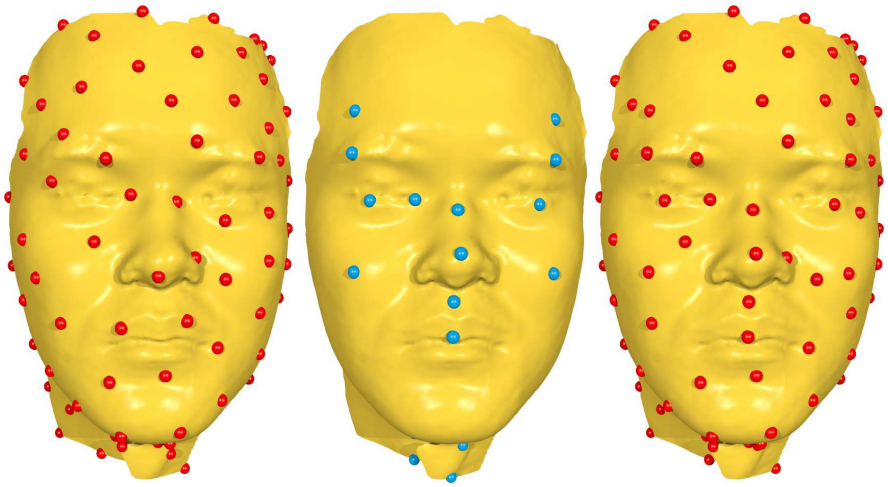


Figure 1: *The process of symmetry detection in [6].*



(a) (b) (c)

Figure 2: *Sampling and feature detection. (a) Uniform samples; (b) Extremal points of Gaussian curvature; (c) Snapped samples.*

Sampling

As shown in Fig.2(a), the samples can be taken uniformly at random. To generate uniformly distributed samples, you can overlay the input model with a grid and take a sample for each cell that intersects with the input model. For example, you can take the point on the input model that is closest to each cell center.

To build correspondences between samples, you need to compute signatures for them. One possibility is to use the curvature values estimated at each sample point. For curvature estimation, you can use the discrete curvature tensor method described in [2]. As an alternative, you can also employ the method introduced in [1] which is based on polynomial fitting. After curvature estimation, you should obtain the principal curvatures and the principal directions at each sample point.

As there is no guarantee that uniform samples are consistent with the underlying symmetries, we have to maintain a dense sampling such that there will be a sufficient number of correspondences for each of the underlying symmetries. To alleviate this issue, instead of extracting random samples you can extract the feature points on the input model. A simple but useful strategy is to extract points whose mean curvature value is either a minimum or a maximum in its neighboring points. An example is illustrated in Fig.2(b). Note that this criterion does not work for points on flat regions. Thus, you have to remove those features that have small variance of signatures in their neighbors.

You can also combine the strengths of these two methods by moving samples to their neighboring feature points (See Fig.2(c)). This is helpful in cases where there are not many feature points you can use. In this case, you still need to use the uniform samples.

Symmetry Detection by Clustering

In this step, you need to build correspondences between the sample points. As only reflection and rotational symmetries are considered in this assignment, corresponding points should have similar principal curvatures. Note that, in this process, you have to normalize the principal curvatures $k_i \rightarrow \frac{k_i}{\sqrt{1+k_i^2}}$ in order to set meaningful thresholds.

There could be many false positives in the set of correspondences obtained by matching principal curvatures at sample points. This is because we only consider the geometry in a small neighborhood around each sample point when building correspondences among them. Now we need to extract correct correspondences by studying their relations. As we have principal directions at each sample point. Each correspondence indicates a reflection/rigid transformation (See [6] for details). To group correspondences into clusters, we seek correspondences that indicate similar transformations. Each cluster represents a a candidate symmetry. The size of each cluster represents the strength of its corresponding symmetry.

For clustering, you can use the standard mean-shift clustering algorithm [3]. Here each transformation is represented as a data point which is a vector in R^{12}.

Note that you are not restricted to use mean-shift clustering for finding consistent correspondences. For example, you can also try alternative methods such as spectral matching method [5] or forward search method [4]

Symmetry Verification

After clustering, you should obtain sets of points that have counterparts under similar symmetries. Each cluster corresponds to a particular symmetry of the shape, but the transformation at the cluster's center does not necessarily provide the best possible transformation for matching the surface patches. To overcome this issue, you need to incrementally refine each transformation using the iterated closest points(ICP) algorithm. To speed-up the computation, it is better to use the efficient variants of ICP which are described in [7]. In particular, the point to plane distance metric is highly recommended.

On-Line Resources

The following is a list of on-line resources that could be helpful for you

- **PLY Viewer**: http://www.cs.princeton.edu/gfx/proj/trimesh2/ (By Szymon Rusinkiewicz).
- **Curvature Estimation**: http://www-sop.inria.fr/members/Pierre.Alliez/demos/curvature/ (By Pierre.Alliez).
- **Mean Shift Clustering(c++)**: http://www.caip.rutgers.edu/riul/research/code/AMS/index.html
- **Mean Shift Clustering(matlab)**: http://www.mathworks.nl/matlabcentral/fileexchange/10161

(a) Face (b) Bunny (c) Dragon

Figure 3: *Three input models. (a) The face model contains a global reflection symmetry. (b) The bunny model contains two partial reflection symmetries. (c) The dragon model contains two partial reflection symmetries and one rotational symmetry.*

What you will be given

You will be given three 3d triangular meshes in *.ply format (See Fig.3). The face model has a global reflection symmetry. The bunny model contains two partial reflection symmetries. The dragon model contains two partial reflection symmetries and one rotational symmetry

What to hand in

Remember that in programming assignments you are allowed to work in teams of up to three students, and that each team needs to hand in only a single write-up and a source-code package. On or before the due date, you must submit the following information:

- Final version of your source code: It is recommended to do the implementation in C++. You are free to use either Windows or Linux.
- The symmetries detected by your program: For each symmetry, you need to provide its type (reflection symmetry or rotational symmetry) and the corresponding symmetry transformation (which is an affine transformation).
- Paper write-up: Given an overview of your program, explain how your algorithm(s) work. Discuss how you chose the algorithms used in each step of the pipeline. Extra credit will be given for detailed comparisons between alternative methods in each step. You can imagine that you are writing a academic paper.

We will setup time periods during which you will have to demo your program to the instructor and the TA's.

References

[1] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Symposium on Geometry Processing*, pages 177–187, 2003.

[2] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321, New York, NY, USA, 2003. ACM.

[3] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[4] Qi-Xing Huang, Simon Flory, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006.

[5] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482 – 1489, October 2005.

[6] N. J. Mitra, L. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *TOG.*, 25(3):560–568, 2006.

[7] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.