CS233: The Shape of Data Handout # 4
Geometric Geometric and Topological Data Analysis
Stanford University Wednesday, 11 May 2016

Homework #3 **v4**: Shape correspondences, shape matching, multi-way alignments. [100 points, *Extra credit*: 25 points]
Due Date: Thursday, 19 May 2016

In this homework, you will experiment with two approaches aiming to align the 3D shapes in a given collection. In the first problem, you will study how to jointly align the collection of 3D shapes by formulating and solving for an appropriate Markov Random Field (MRF). In the second problem, in order to solve the *alignment problem*, you will build point-wise correspondences between pairs of 3D shapes based on feature matching.

Problem 1. Joint Alignment of 3D Shapes [70 points]

Aligning 3D shapes according to their orientation is an important preprocessing step for many shape analysis tasks, including finding point-to-point correspondences, performing shape comparisons, doing shape retrieval, etc. However, manually aligning large sets of 3D shapes is time consuming and error prone. In this problem, we study how to automatically align a set of 3D shapes in a consistent orientation (Fig. 1). To simplify the problem we make the assumption that that all input models have been pre-aligned in terms of the their "up" direction — their y-axis in our terminology, e.g., seats of chairs face upwards. This assumptions allows us to parameterize the orientation to be estimated by only the azimuth angle, i.e., the rotation around the y-axis. For the coding parts of this problem we have provided you with some skeleton code located at `code_and_data/problem1/code`.

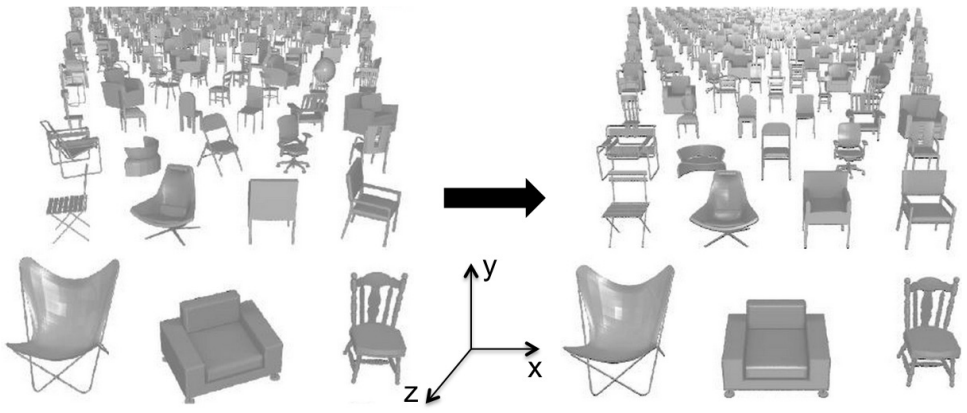


Figure 1: Joint shape alignment

(a) [20 points] **Multi-view shape feature extraction**

For each 3D shape we build a feature that is sensitive to its orientation. For this, we will use a multi-view 2D-representation, i.e., we will represent a shape by the collection of its

2D-renderings coming from predefined viewpoints. Concretely, we represent each shape as N regularly sampled 2D-views which we further summarize by a discriminative image feature such as the Histogram of Gradient (HoG) [1]. This is illustrated by Figure 2.

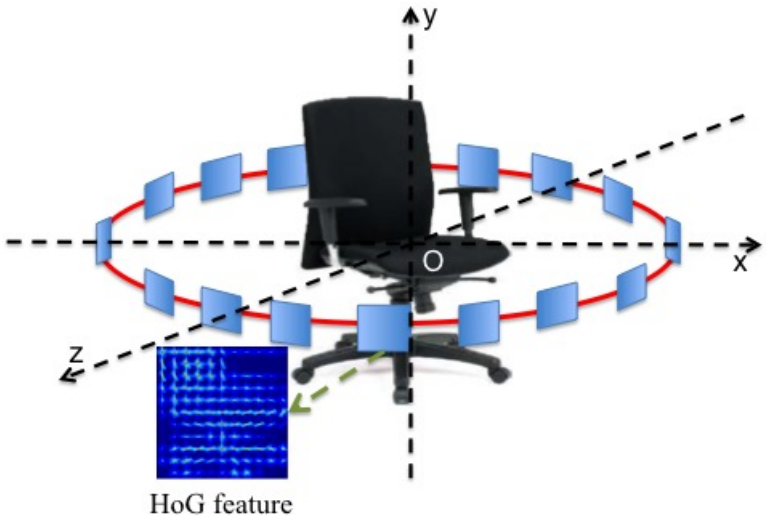


Figure 2: Multi-view shape representation

To extract the aforementioned feature you will follow these steps:

(i) Sample $V = 16$ views evenly along the equator of xOz plane (views on the red circle in Fig. 2). Specify an order of views in the world frame.

(ii) Normalize each shape to have unit radius and place the shape at the origin of the world frame.

(iii) Render an image at each view for every shape.

(iv) Compute the HoG feature of each view, which produces an H-dimensional vector. A HoG feature extractor is provided via the *hog* function in Piotr's toolbox (`https://github.com/pdollar/toolbox`). This function takes an image as input and outputs a 3-D tensor. Vectorize the output to obtain the HoG feature. Please first resize all images to 120×120 pixels and then use the default parameters for the *hog* function.

(v) Represent each shape by the concatenation of the HoG features with the pre-specified order. This way a shape will correspond to a $(V \times H)$-dimensional feature.

Note that this shape representation is orientation sensitive, since the order of views is defined in a world frame.

You are asked to follow the previous steps and derive the multi-view feature for each shape in the provided dataset of the 100 3D chairs (`data/100chairs.zip`). Because parts (i)-(iii) require a certain amount of familiarity with graphics, we have also

provided you with the end result of running these three steps. Concretely, at `data/100chairs_rendering.zip` you can find rendered images from 16 views for every chair. In other words parts (i)-(iii) are *optional* and you may start your implementation directly at part (iv) and exploit the provided images. If you decide to implement (i)-(iii), submit your shape normalization code in a file named `shape_normalization.m` along with your output rendered images. For the multi-view feature extraction code submit a file named `hog_extraction.m`. Also submit a visualization of the computed HoG features for the shapes: `001.obj`, `002.obj`, `003.obj`, (select 3 views that you like for each shape). For the visualization of the HoG features you can use the *hogDraw* function in Piotr's toolbox.

(b) [20 points] **Pairwise orientation dissimilarity computation.**

Assume a pair of shapes S_i and S_j that lies on the world frame (as illustrated by the red planes in Figure 3). In this part we will compute a view-sensitive shape dissimilarity $D_{ij}(\theta_1, \theta_2)$, which captures the shape differences when S_i is rotated by θ_1 and S_j by θ_2 (see Fig. 3). Concretely, $\theta_1, \theta_2 \in \{0, \ldots, \frac{2k\pi}{V}, \ldots, \frac{2(V-1)\pi}{V}\}$, i.e., we have discretized the rotation angles into V bins. This shape dissimilarity can be computed directly from the multi-view features of Part (a), which are sensitive to the orientation of the 3D shapes. With $D_{ij}(\theta_1, \theta_2)$ fixed, we can choose the optimal pair of angles (θ_1, θ_2), i.e., those that correspond to the smallest dissimilarity. Notice, how this dissimilarity score is the same for the set of view pairs if their relative views are the same, i.e., $D_{ij}(\theta_1, \theta_2) = C_{ij}(\theta)$ for $\theta \equiv (\theta_2 - \theta_1) \bmod 2\pi$. This property can help us to reduce the computation from $O(V^2)$ (for every θ_1 and θ_2 combination) to $O(V)$.

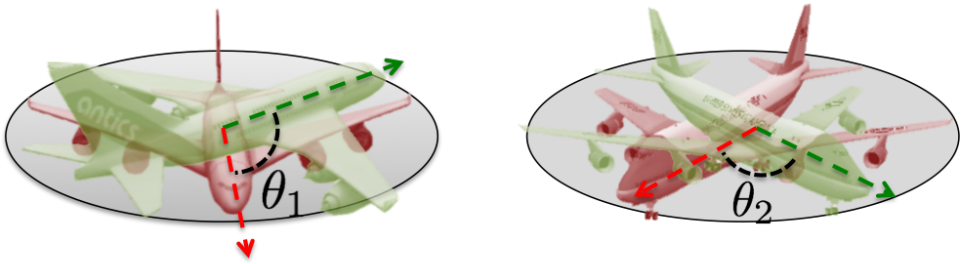


Figure 3: Computing pairwise dissimilarity scores. Red dashed line is the original heading orientation and green dashed line is the rotated heading orientation.

Your task is to compute the pairwise dissimilarity $D_{ij}(\theta_1, \theta_2)$ for each pair of shapes. Use the L_2 distance when comparing the multi-view features for D. In your submission, upload the source code as `pairwise_dissimilarity.m` and further provide the visualization of D_{ij} matrix for the pair of `001.obj` and `002.obj` shapes (using *imagesc* function from Matlab). For the same pair of shapes, plot the two pairs of views that correspond to the smallest and largest dissimilarities respectively.

(c) [30 points] **Joint shape alignment by MRF.**

In Part (c), we will try to jointly align all shapes of the collection. To do so, we will utilize a probabilistic graphical model known as Markov Random Field (MRF). Given the groundwork done in parts (a) and (b), we are now ready to build a "second-order" MRF.

To judge the likelihood for shapes S_i and S_j, to be aligned with orientations $\theta_1 = \frac{2(k_1-1)\pi}{V}$ and $\theta_2 = \frac{2(k_2-1)\pi}{V}$, respectively; we define the affinity $w_{ij}(k_1,k_2)$ by modulating the dissimilarity of Part (b):

$$w_{ij}(k_1,k_2) = \exp\{-D_{ij}(\theta_1,\theta_2)/\sigma\}, \tag{1}$$

where σ is a bandwidth parameter.

Conceptually, we build a complete graph for our shape collection where each node corresponds to a shape and each edge is decorated with a matrix W_{ij} which captures the aforementioned pairwise affinities. Our end goal is to assign a V-dimensional indicator binary vector $\vec{X}_i \in \{0,1\}^V$ to every node that indicates its optimal rotation. We set $X_{i,v}$ to be 1 if a shape is rotated by $2(v-1)\pi/V$ and 0 otherwise.

Now, one way of solving the problem of estimating jointly shape orientations, can be given as the solution to the following optimization problem:

$$\begin{array}{lllll} \text{maximize}_{\{\vec{X}_i\}} & \sum_{(i,j)\in\mathscr{E}} \vec{X}_i^T \mathbf{W}_{ij} \vec{X}_j + \sum_{i\in\{1\ldots,N\}} \vec{U}_i^T \vec{X}_i & & & \\ \text{subject to} & X_{i,v} \in \{0,1\} & \text{for } i = 1,\ldots,N & v = 1,\ldots,V \\ & \sum_v X_{i,v} = 1 & \text{for } i = 1,\ldots,N & \end{array} \tag{2}$$

Where, $\vec{U}_i \in [0,1]^V$ is a *random* unary term associated with every vertex and N is the total number of vertices.

[3 points] If we ignore the unary terms, can you explain how the above formulation is aiming to align the shapes consistently?

[3 points] Why do we need to add the random unary terms? [Hint: given a solution a, how different is another solution b where all shapes in b are rotated as in a plus some constant angle?]

[24 points] Your task is to build the $\mathbf{W}_{ij}$ matrices for this model and solve an approximation of Problem (2). To solve the optimization problem, you need to implement the algorithm in [2] by Leordeanu, M. and Hebert, M, which is a fast MRF solver. Initialize the unary terms $\vec{U}_i$ by drawing for each dimension i.i.d. samples from the uniform distribution in $[0,1]$. You may need to tune the parameter σ to get the best results. Analyze your results and describe your discoveries.

Save the aligned 3D shapes in OBJ format using the provided code (`write_wobj.m`). Make sure that your models are normalized before saving them. In this case, you might also find useful the MATLAB function `trisurf` for 3D model visualization.

In your submission, please include your MRF solver code as `mrf.m`, alignment pipeline code as `P1c.m` and a zipped package `aligned.zip` of all aligned 3D shapes. If you use the pre-computed views, `aligned.zip` should include the optimally-aligned view of each object (i.e., one of the 16 images provided per shape).

Finally, please make a single plot that visualizes for 9 shapes (`001.obj` to `009.obj`) the output of your alignment. Include this plot in your write-up.

Problem 2. Point Correspondences for Non-Rigid 3D Shapes
[30 points, *Extra credit* 25 points]

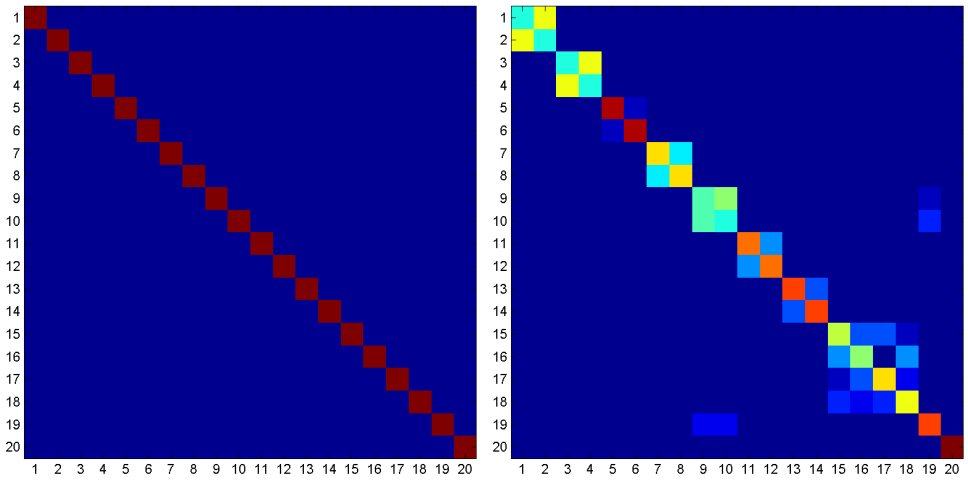
Finding point-to-point correspondences is an essential task for many graphics applications, including shape morphing, deformation transfer, statistical modeling and animation. The previous problem's “”view alignment" can provide a rough point-to-point mapping among shapes based on the distance in Euclidean space after the global alignment. Nevertheless, it cannot provide good dense correspondences, especially when the shapes are non-rigid and deformable. In this problem, we will find correspondences between landmark points by comparing intrinsic properties of the shapes (properties invariant to isometric deformations).

As a benchmark, we provide a template human body mesh and 20 other test meshes with the same human body shape but different poses, as shown Fig. 4 (See `data/meshes` directory). Check out the SCAPE dataset for more information about the meshes (`http://robotics.stanford.edu/~drago/Projects/scape/scape.html`). The goal of this problem is to find correspondences of landmark points from the template mesh to all test meshes. The landmark points are given with vertex IDs and names (See `data/landmark_vids.txt` file). The meshes in our benchmark are already perfectly aligned, so all meshes have the same number of vertices and faces with the same connectivity, plus all corresponding vertices have the same vertex IDs. Thus the point correspondences can be easily evaluated by comparing the vertex IDs.

Figure 4: Template (left upper corner) and test meshes with landmark points.

We provide a simple code to visualize the point correspondences using a binary matrix. A perfect matching will be shown as a diagonal matrix (Fig. 5a). Also, an overall frequency matrix will be generated (Fig. 5b) by accumulating the binary matrices across all test cases. From that matrix you can see which point is mostly confused with which other point (red color indicates higher frequency).

(a) [10 points] We first try to find landmark correspondences using point descriptors. We



(a) Perfect matching (b) Example of overall frequency matrix

Figure 5: Matrix visualization of point correspondences.

provide a code for computing *Heat Kernel Signatures* (HKS) [3] and *Wave Kernel Signatures* (WKS) [4] at the vertices. Find the best set of landmark point pairs by comparing these descriptors in `src/run_desc_matching.m` (Find 'FILL HERE' and fill the blank in code). We recommend to implement the Hungarian algorithm (`https://en.wikipedia.org/wiki/Hungarian_algorithm`), but you can also implement any reasonable heuristic method (then, briefly explain your method). Report the overall confusion matrix and accuracy. Can you see any pattern in the failure cases? Discuss why the failure pattern happens when using point descriptors. (Hint: See landmark names and which point is confused with which point from the frequency matrix.)

(b) [20 points] We can assume that all different poses of human bodies are near-isometric deformations of the template mesh, which means that distance between any two points over the surface is nearly preserved. Let us compare pairwise geodesic distances using the MRF solver you implemented in Problem 1. You can consider the landmark points on the template mesh as *nodes* of the MRF and the same ones on the test mesh as *labels* of MRF. Similarly with Eq.1 in Problem 1, we define pairwise potential for point pairs (i_a, i_b) on template and (j_a, j_b) on test:

$$w((i_a,i_b),(j_a,j_b)) = \exp\left(-\frac{\|d(i_a,j_a) - d(i_b,j_b)\|^2}{2\sigma}\right), \tag{3}$$

where $d(x,y)$ is geodesic distance between x and y on the same mesh normalized by the longest distance on the surface. Implement the MRF problem in `src/run_pairwise_matching.m`. Set σ to 0.05 (or tune it yourself). We provide code for computing the geodesic distances. Report the overall confusion matrix and accuracy. Can you see any pattern in the failure cases? Is there any difference in the failure pattern compared to the

previous case? Explain briefly. (Do not worry even if the result is not better than the previous one.)

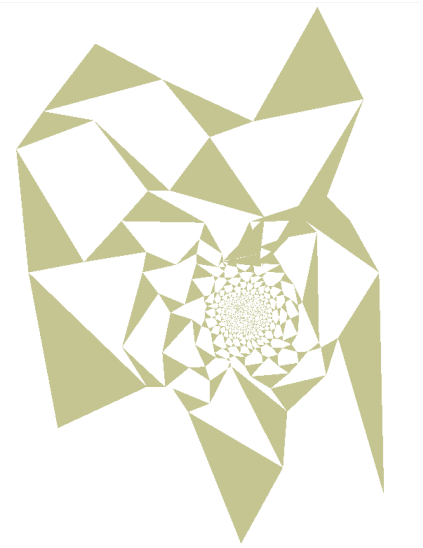


Figure 6: Mid-edge-flattened template mesh.

(c) [**Extra credit** 20 points] In `src/run_mobius_matching.m`, now we implement Möbius voting method introduced in Lipman et. al. [5].

The basic idea of Möbius voting is integrating information from multiple conformal maps. Conformal means angle-preserving, so it is a weaker condition than isometric (distance-preserving). But for shapes with the topology of the sphere, a conformal map of two entire surfaces can be simply defined by three point correspondences (check out the uniformization theorem for details). The idea in the paper is to randomly generate a large number of conformal maps, and find some good mappings by measuring deformation errors. For accumulating them, they use a Hough transform style approach, but in this problem we will try a RANSAC approach for reducing the computation time.

Please read section 6 and 7 of the paper to see how to define a conformal map using a triplet of point correspondences, and how to measure a deformation error on the canonical 2D extended complex plane. We provide mid-edge-flattened meshes in `data/flattened_mesh` as shown in Fig. 6 (Section 5), and uniformly sample points over the surface (Section 3, `data/uniform_sample_eids.txt`).

The original Möbius voting algorithm takes long time to compute particularly when implemented with MATLAB (without code optimization). Thus we slightly **modify** the algorithm in the following ways:

- We provide 5 **sparse** landmark points (`data/sparse_landmark_eids.txt`). Use these sparse landmark points for sampling triplet points (z_1, z_2, z_3, and w_1, w_2, w_3

in the paper Algorithm 2), but use all uniformly sampled points for finding mutually closest point pairs ($\bar{z_k}$ and $\bar{w_k}$ in the paper Algorithm 2) and computing deformation error. See the code for these two point sets. You can just try all triplet pairs instead of sampling since now the number of landmark points is small.

- The original idea of Möbius voting is voting for mutually closest point pairs ($\bar{z_k}$ and $\bar{w_k}$) instead of a transformation. But since now we use different sets of points for transformation sampling and deformation measure, just vote for triplet point pairs ($z_1 - w_1$, $z_2 - w_2$, and $z_3 - w_3$) that define the transformation.
- Use $K = 0.3$ (or tune yourself).

Report the overall confusion matrix and accuracy. You can also compare the results with previous ones by turning on the `use_sparse_landmarks` flag to true in the previous codes and running them with sparse landmark points. Does the Möbius voting solve the problem in the previous cases? Why or why not?

(d) [**Extra credit** 5 points] Can you see any failure case in Möbius voting results? It is fine (actually very nice) if you get perfect matching results! You can find an example of failure cases in the Fig. 9 in [5]. Why does this problem (still) happen? (Hint: Why is it difficult to map human body shapes?) Briefly discuss how the problem is handled in the Blended Intrinsic Maps (BIM) method introduced in Kim et. al. [6].

What is the time complexity of comparing point descriptors in (a)? What is the time complexity of comparing geodesic distances for point pairs in (b)? What is the time complexity of trying all possible conformal maps in (c)? Why is the time complexity of BIM $O(N^{12})$? Briefly explain (See section 6 in the paper).

What to hand in

Submit source code files you edited (`run_desc_matching.m`, `run_pairwise_matching.m`, `run_mobius_matching.m`, and other files you added). In your document, attach the overall frequency matrices for all cases, and report accuracy values. Provide answers for the all discussion questions in a few sentences.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the international conference on Computer Vision and Pattern Recognition (CVPR), 2005.*, vol. 1, pp. 886–893, IEEE, 2005.

[2] M. Leordeanu and M. Hebert, "Efficient map approximation for dense energy functions," in *Proceedings of the 23rd International Conference on Machine learning (ICML)*, pp. 545–552, ACM, 2006.

[3] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Computer Graphics Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[4] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 1626–1633, Nov 2011.

[5] Y. Lipman and T. Funkhouser, "Möbius voting for surface correspondence," in *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, (New York, NY, USA), pp. 72:1–72:12, ACM, 2009.

[6] V. G. Kim, Y. Lipman, and T. Funkhouser, "Blended intrinsic maps," in *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, (New York, NY, USA), pp. 79:1–79:12, ACM, 2011.