Homework #3: Shape correspondences, shape matching, multi-way alignments. [100 points]
Due Date: Wednesday, 20 May, 2026 (11:59 PM PT)

In this homework, you will experiment with two approaches for aligning the shapes in a given 3D model 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, you will build point-wise correspondences between pairs of 3D shapes based on feature matching in order to solve the alignment problem.

Python Environment Setup The python packages used in this homework can be installed by running the following.

```
pip install scipy scikit-image libigl pygeodesic
```

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 and computing statistics, 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 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, that is the rotation around the y-axis. For the coding parts of this problem, we have provided you with some skeleton code located at `hmwk03_code_data/code/hmwk3_p1.py`.

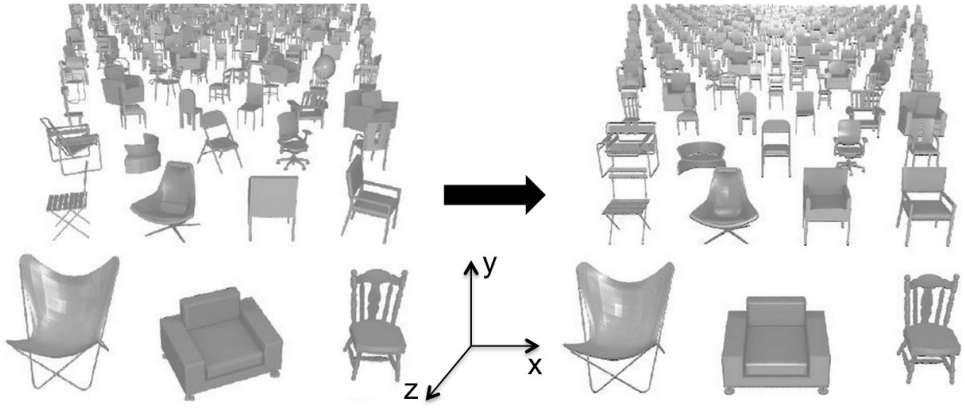


Figure 1: Joint shape alignment

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

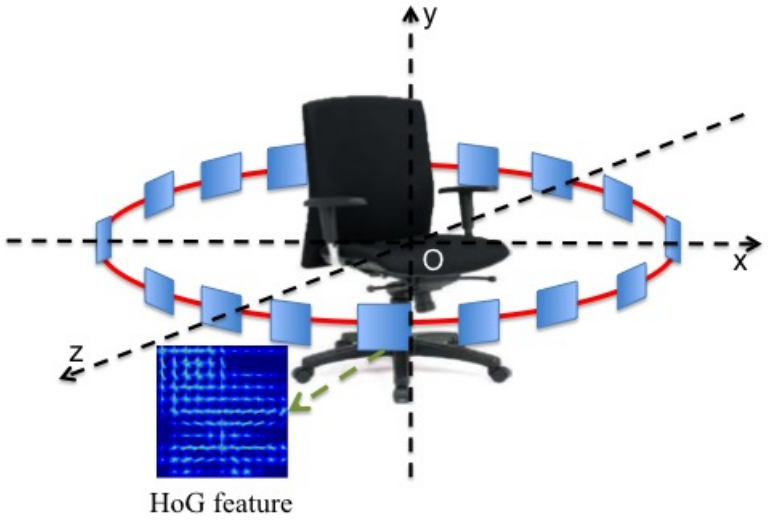


Figure 2: Multi-view shape representation

To extract the aforementioned feature representation you need to perform the following 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 `skimage.feature`[1]. This function takes an image as input and outputs its HoG feature. Please first resize all images to 120×120 pixels and then use the default parameters for the *hog* function.

[1] `https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html`

(v) Now we can represent each shape by the concatenation of the HoG features from all V views, with the pre-specified order. Namely, we represent each shape with a $(V \times H)$-dim 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 (`hmwk03_code_data/data_p1/100chairs.zip`). Because parts (i)-(iii) require a certain amount of familiarity with graphics, we have also provided you with the result of running these three steps. Specifically, within the `hmwk03_code_data/data_p1/100chairs_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.py` along with your output rendered images. Please complete the relevant part in `hmwk3_p1.py` for multi-view feature extraction. 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). You may use the `visualize` argument of `skimage.feature.hog` and the provided `save_hog_image` function for the hog feature visualization.

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

Assume we are given a pair of shapes S_i and S_j positioned in 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). $\theta_1, \theta_2 \in \{0, \ldots, \frac{2k\pi}{V}, \ldots, \frac{2(V-1)\pi}{V}\}$, i.e., we have discretized the rotation angles into V bins. This is consistent with the given rendering views: observing the shape in the next view is the same as rotating the shape by $\frac{2\pi}{V}$ in the current view.

Shape dissimilarity can be computed directly from the multi-view features of Part (a), which are sensitive to the orientation of the 3D shapes. We define $D_{ij}(\theta_1, \theta_2)$ as the L2 distance between the multi-view feature vectors of S_i rotated by θ_1 and S_j rotated by θ_2.

After computing $D_{ij}(\theta_1, \theta_2)$, 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 reduce the computation complexity from $O(V^2)$ (for every θ_1 and θ_2 combination) to $O(V)$.

Your task is to compute the pairwise dissimilarity $D_{ij}(\theta_1, \theta_2)$ for each pair of shapes. Complete the relevant part in `hmw3_p1.py` and further provide the visualization of D_{ij} matrix for the pair of `001.obj` and `002.obj` shapes (using provided `mat_show` function). For the same pair of shapes, plot the two pairs of views that correspond to the smallest and largest dissimilarities respectively.

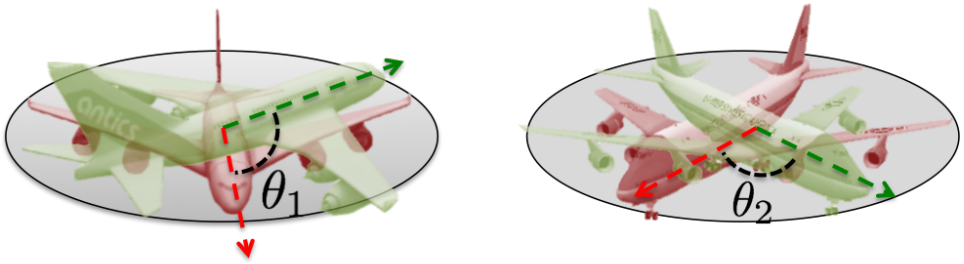


Figure 3: Computing pairwise dissimilarity scores. The red dashed line is the original heading orientation and the green dashed line is the rotated heading orientation.

(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 determine 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 $w_{ij}(k_1,k_2)$ based on 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.

Concretely, the joint shape alignment problem can be formulated as the following optimization problem:

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

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 aims 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 may use the provided implementation of the algorithm in [2] by Leordeanu and Hebert, which is a fast MRF solver. You may try different random initializations of the unary terms $\vec{U}_i$, e.g. by drawing 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.

Please complete the relevant parts in `hmwk3_p1.py` and submit a zip file `aligned.zip` that contains 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]

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 proximity 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).

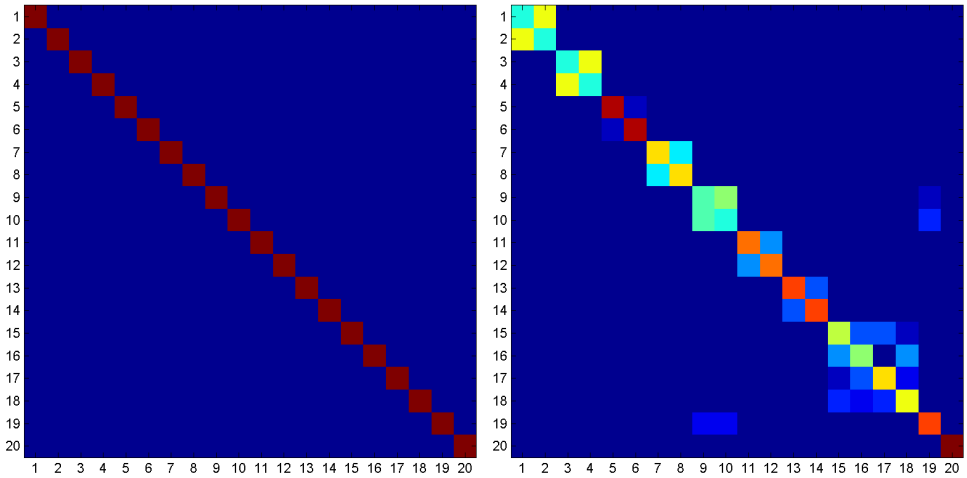
As a benchmark, we provide a template human body mesh and 20 other test meshes with the same human body shape but different poses, as shown Fig. 4 (See `hmwk03_code_data/data_p2/meshes` directory). Check out the SCAPE dataset for more information about the meshes (`http://robotics.stanford.edu/~drago/Projects/scape/scape.html`). 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 `hmwk03_code_data/data_p2/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.

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



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



(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 `hmwk03_code_data/run_desc_matching.py`. We recommend using 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 the 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, j_a) on the template and (i_b, j_b) on the test:

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

where $d(x,y)$ is the geodesic distance between x and y on the same mesh normalized by the longest distance on the surface. Implement the MRF problem in `hmwk03_code_data/run_pairwise_matching.py`. 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.)

What to hand in

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

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, 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*, pp. 545–552, 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 *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pp. 1626–1633, IEEE, 2011.