Homework #2: Joint shape alignment using MRF; Shape segmentation. [100 points]
Due Date: Friday, May 5

Homework policies

CS468 is a highly technical course, so doing the homework is the only way to acquire a working knowledge of the material presented. We encourage you strongly to start working on the homework problems right away—the problems below, as well as those to follow, have considerable technical depth and you are unlikely to be able to solve them if you wait until the evening before the due date.

Collaboration in solving the problems is encouraged in this class—you have a lot to learn from your fellow students. However, in order to make grading the homeworks a meaningful way to measure your effort and your understanding of the material, we must ask that:

- *On programming problems [such as in this homework], groups of up to three students can work together as a team, handing in a single body of code and documentation for their joint effort.*

It is very important in this course that every homework be turned in on time. We recognize that occasionally there are circumstances beyond your control that prevent an assignment from being completed on time. You will be allowed two classes of grace during the quarter. This means that you can either hand in two assignments each late by one class, or one assignment late by two classes. Any further assignments handed in late will be penalized by 20% for each class that they are late, unless special arrangements have been made previously with the instructor or the CA(s).

Software

We ask that you use `MATLAB` for this assignment. You can use functions like `trimesh`, `triplot`, `trisurf` to visualize the triangulated shapes.

Goals of this assignment

In the first problem of this homework, you will implement an approach aiming to align the 3D shapes in a given collection. 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 implement a learning shape labeling algorithm shown in the lecture.

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`.

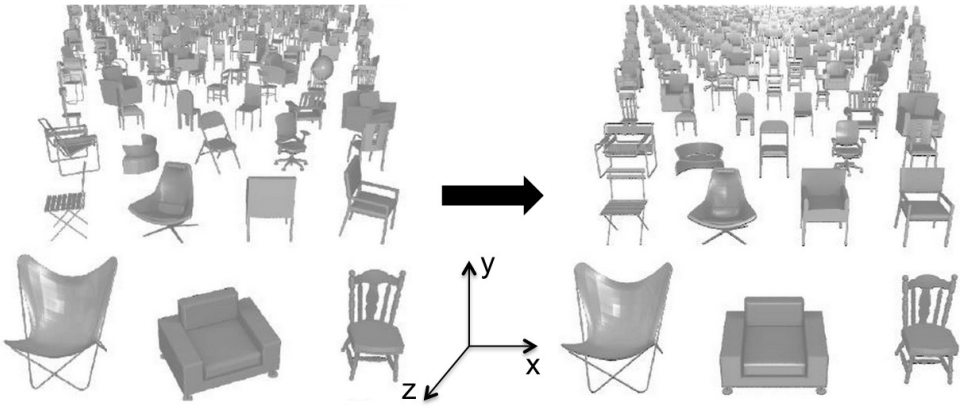

Figure 1: Joint shape alignment

(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.

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.

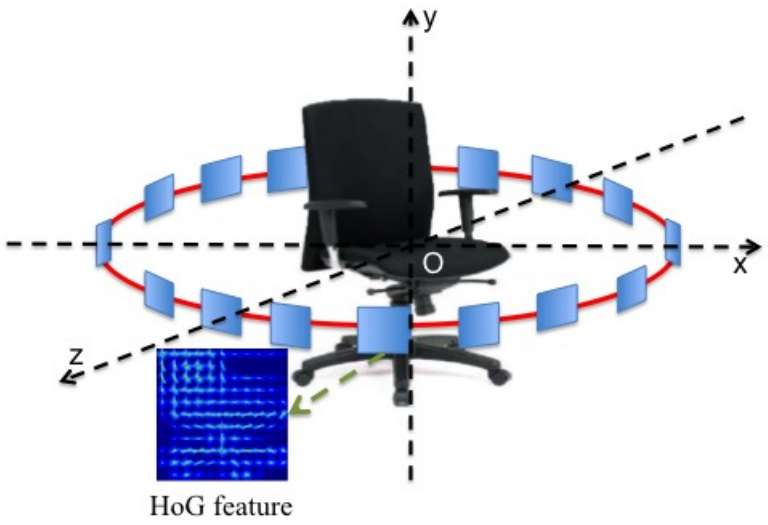

Figure 2: Multi-view shape representation

(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 $O(V)$.

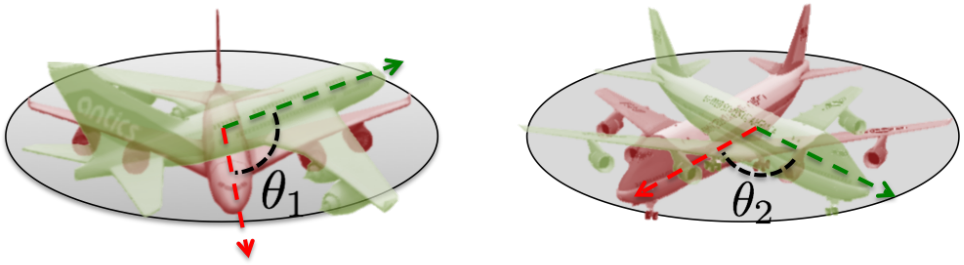

Figure 3: Computing pairwise dissimilarity scores. Red dashed line is the original heading orientation and green dashed line is the rotated heading orientation.

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, use D_{ij} to find and plot the two pairs of multi-view repesentations 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:

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

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 may use an implementation of the algorithm in [2] by Leordeanu, M. and Hebert, M, which is a fast MRF solver[1][2]. Alternatively, you can implement it yourself - in this case, use the code template provided in `mrf.m`. 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 the MRF solver code as `mrf.m`, alignment pipeline code as a script `P1c.m` and a zipped package `aligned.zip` of all aligned 3D shapes.

[1] `https://sites.google.com/site/graphmatchingmethods/`
[2] `https://goo.gl/6wgPmD`

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. Learning to segment and label 3D shapes
[30 points, *Extra credit* 25 points]

We saw several algorithms for shape segmentation and labeling. In this part of the assignment, you will implement a supervised method described in [3]. The algorithm pipeline you will implement is as follows: the algorithm is given a set of labeled shapes as input. First, it computes a set of per-face shape descriptors. These shape descriptors and ground truth face labels are then used to train a classifier, which, given a new shape, produces per-face label probabilities. These probabilities can be used either by themselves, to obtain a maximally probable shape face classification, or as a unary term in a pairwise MRF. The MRF is solved using graph-cuts to obtain smooth boundaries between differently labeled regions. Here, you will use the *labeled PSB* dataset, available for download at [3]'s project webpage[3], as `labeledDb.7z`. The dataset consists of 19 classes of objects, as shown at the webpage. Since the training process is quite time consuming, for the exercise, choose three classes of shapes, with varying number of labels (e.g., humans, horses and chairs), and perform all the experiment below for each of these three classes separately. In you report, clearly state which shape classes you chose.

(a) [30 points + 10 EXTRA] **Training per-face shape classifier**

To simplify the problem, you will use pre-computed shape features, made available for download by the authors of [3]. You can find them at the project webpage[3], as `features.7z`. Each shape has a corresponding text file, containing F feature vectors, where F is the number of mesh faces. Note the difference with the paper: here, each feature is a 628-dimensional vector, and only its first 593 entries are non-zero, for all shapes and all feature vectors. In your experiments, truncate feature vectors accordingly.

[5 points] Split shapes in each class into training and test sets. The training set should contain $10-15$ shapes. Load training shapes' features and concatenate them into a single matrix, with rows corresponding to different features. Concatenate their corresponding label vectors into a single column vector. Make sure the shape are consistently labeled - that is, if label 1 corresponds to a right foot of the first human shape in the training set, it should correspond to a right foot in the rest of human shapes. Visualize the shapes color-coded according to face labels, to make sure you did this correctly, and include the image in your report. See Figure 4 for visualization example.

[3] `http://people.cs.umass.edu/~kalo/papers/LabelMeshes/index.html`

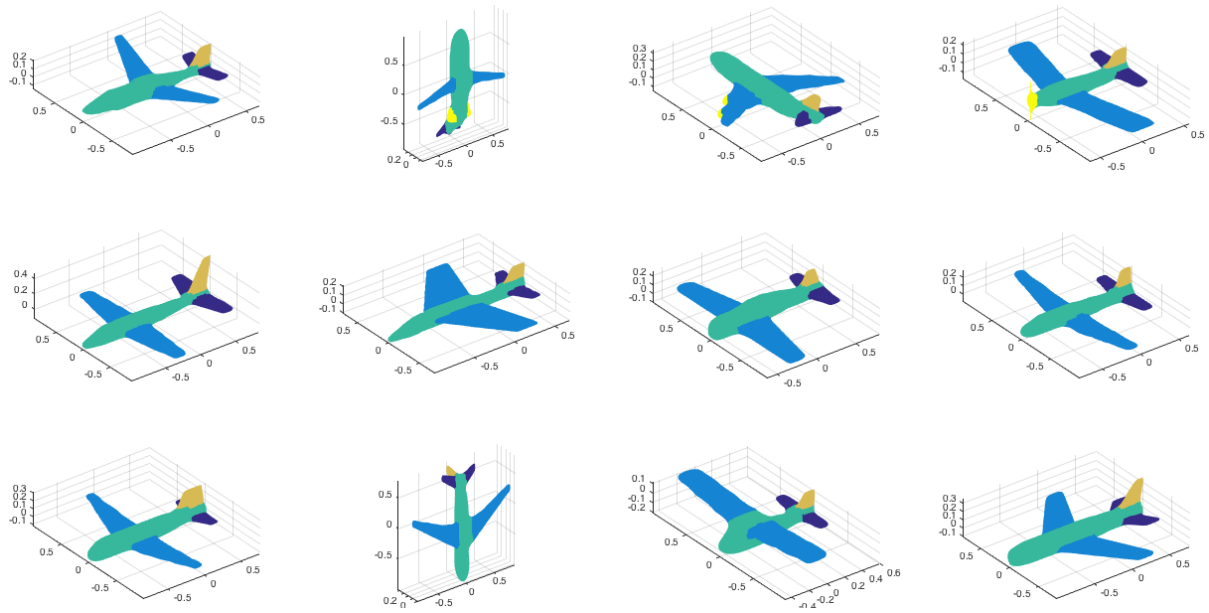

Figure 4: Training shapes visualization.

To load the mesh and the labels from `labeledDb`, you can use the provided functions `loadMesh.m` and `loadLabels.m`. To load features, you may find Matlab functions `textscan` and `reshape` to be useful.

Note: read the next clause now. If you choose to work with a classifier other than the suggested one, in this clause you should organize data according to the requirements of the classifier. Add a short explanation about data structures you used in the report.

[18 points] *Learn a classifier.* Following the paper, you will use JointBoost classifier [4] to perform classification. You can find a Matlab implementation at Antonio Torralba's webpage[4]. Use features and labels from the previous clause to train the classifier. In the report, state the algorithm parameters you used, and why you chose them (cross-validation, algorithm description given in the paper, visual inspection, etc.). Compute average training accuracy as

$$ave\_acc = \frac{1}{\#(\text{training shapes})} \sum_{\text{training shapes}} \frac{\#(\text{correctly classified faces})}{\#(\text{faces})},$$

and include it in your report.

Alternatively, you can use any other classifier of your choice (e.g., a neural network) for this task. In this case, include a sufficiently detailed description of the classifier in your report, together with the training procedure (e.g., algorithm parameter choice) description. See also the note in the previous clause.

[4]Here is a link to the implementation: `http://people.csail.mit.edu/torralba/code/sharing/sharing.zip`.

[7 points] *Test the classifier.* Test the classifier on the test set. If you used JointBoost, compute label probabilities using soft-max applied to the output `Fx` of the classifier, as follows

```
exp_Fx = exp(Fx);
prob = bsxfun(rdivide, exp_Fx, sum(exp_Fx,2));
```

Use the label with the maximal probability as your label prediction. Visualize the classification results using color-coded test shapes, and include them in your report. Compute and add to the report the average test set classification accuracy, similar to previous clause. Compare with train set classification accuracy.

[10 points *EXTRA*] Train both JointBoost classier, and an additional classifier, and compare their performance on the test set. Illustrate classification results of the two classifiers on the test set side-by-side. Add a short analysis of advantages / disadvantages of each classifier to the report.

Relevant `MATLAB` *functions:* `axis image`, `cameratoolbar`, `bsxfun`, `caxis`, `trisurf(...,'EdgeColor','none',...)`.

(b) [15 points *EXTRA*] **Classification using graph cuts**

As you may have observed, face feature-based classification produces fragmented region boundaries, and sometimes results in mis-classification. We may try to correct these problems by solving a full pairwise MRF problem, as suggested in [3]. The unary term of the MRF is defined by

$$E_1 = \sum_{\text{mesh faces } f_i} -log(prob(label_i)).$$

To simplify the problem, the smoothness term will be defined as

$$E_2 = \mu \cdot \sum_{\text{neighboring faces } f_i, f_j} -log(\theta_{ij}/\pi) l_{ij},$$

where θ_{ij} are the dihedral angles measured between the normals of the neighboring faces, and l_{ij} are the distances between face centers. Note that in the above, the summation is only over neighboring vertices. μ is a relative weight of the smoothness term in the total optimization problem. Hence, optimal labels are obtained by minimizing the sum of the above two terms. This optimization problem can be efficiently solved using graph cuts [5]. The following webpage by Olga Veksler lists various graph cut algorithm implementations: `http://vision.csd.uwo.ca/code/`, e.g., the implementation of the Multi-label optimization algorithm, in `gco-v3.0.zip`.

To solve this clause, implement a solution of the above labeling problem using a graph cut algorithm. Report on the algorithm parameters used, and the obtained labeling accuracy. Visualize labeling results, side-by-side with the results obtained in the previous clause, and include this visualization in your report. See Figure 5 for visualization example. Add a short discussion of the results.

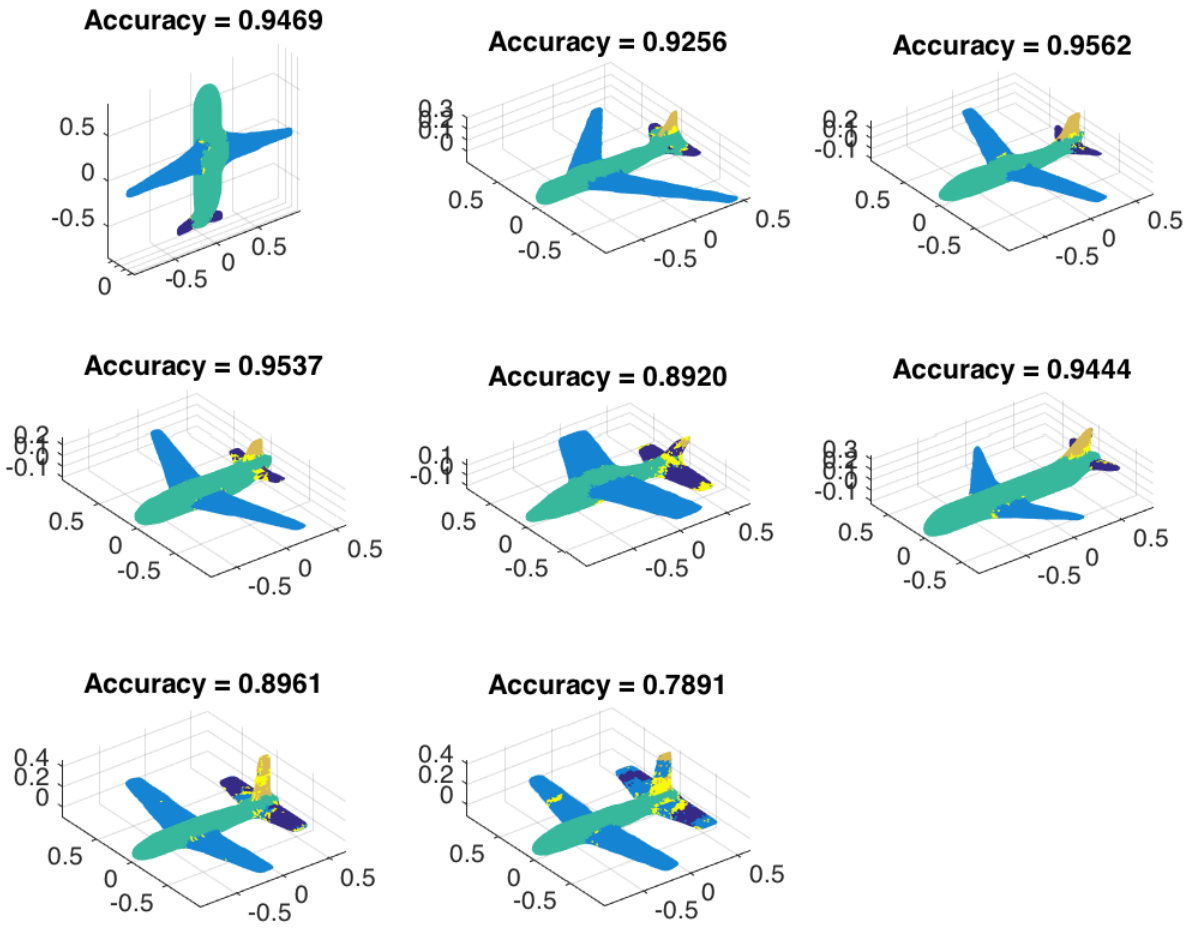

Figure 5: Test shapes classification visualization.

In your submission, please include the labeling pipeline code as a script `P2.m`, as well as any additional functions you wrote or libraries you used. If these libraries require compiling `mex` files, include the compilation commands in your script. DON'T submit a directory with feature vectors.

What to hand in

Submit all source code files. In your document, attach the required visualizations, and report accuracy values. Provide answers for all discussion questions in 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] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3d mesh segmentation and labeling," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 102, 2010.

[4] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, 2007.

[5] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.