Project 3: Statistical Pattern Recognition

Dr. Randy C. Hoover

April 1, 2020

Logistics

- Download the datasets at that contain images of 20 different objects being rotated about a single degree of freedom along with several "test" images of the same objects at selected orientations.
- D2L: Submit your Jupyter Notebook answering questions, illustrating methods, and examples of image classification (pose estimation and object recognition), along with the processes you used to generate them, table of classification and pose estimation results to the dropbox in D2L.
- Due: Wednesday April. 22, 2019

As for simply finding a solution to these problems on-line and copying someone else's source code, this is **STRICTLY FORBIDDEN**.

Project Goals: Develop your own *Python* functions to classify objects (recognize and estimate their pose) from appearance. The project will consist of three parts as described below.

Part 1: Computing and analyzing the eigenspace.

A set of training images of the objects depicted in Figure 1 (left) have been captured for you as depicted in Figure 1 (right). These images are provided to you on the course web page. The training images were captured at angles of $\alpha_p = \frac{2\pi p}{2B}$ for $p = 0, 1, \ldots, 2B-1$, the angle of co-latitude $\beta_p = 60^\circ$ and the optical orientation $\gamma_r = 0$. The bandwidth $B = 64$, i.e., we have 128 test images at different pose angles α_p for training. In addition to the training images, a set of test images are also provided along with a .txt file named *RandAng.txt* containing the orientation α_p each test image was captured at. Both the test images and the training images were captured in a similar fashion (i.e., they have the same statistics).

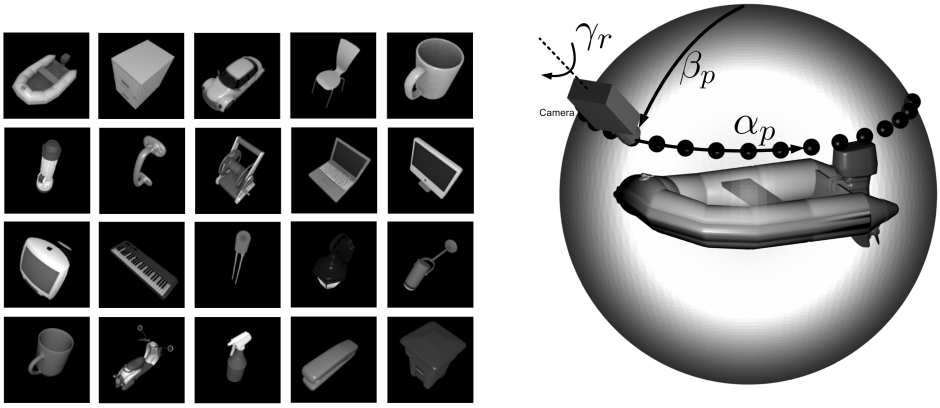

Figure 1: An illustration of images of objects used in this project for classification purposes [left]. Each object is acquired by placing them at the center of an imaging sphere and sampling lines of constant co-latitude [right].

In this part of the project you are required to compute the eigenspace and analyze the results. For each object across all poses, you will need to construct the "unbiased" image data matrix $\hat{X}$ and compute the basis vectors to provide the best image distinction between images in the set in the least-squares sense (i.e., the basis that maximized the column space of $\hat{X}$). As a first go, analyze the singular values and "estimate" the value k that might be sufficient for accurate classification (choose the eigenvalue index that captures most of the variance in the set). Using this k, keep the first k eigenvectors $\mathbf{u}_k$ of the co-variance matrix for each object. We denote each value of k by k_n where subscript n represents the object under consideration and in this case $n = 20$ different objects. In general, each object will require a different k value for accurate classification. Once we have each k_n-dimensional subspace, we need to project each training set $\hat{X}_n$ onto its corresponding eigenspace $\mathbf{u}_{k_n}$ to construct the local appearance manifold $\mathcal{M}_n$ for later classification.

Now that we have computed the eigenspace for each individual object, we need to compute what is referred to as the global eigenspace for the entire set of objects. This is accomplished via:

1. Concatenate all $2B$ images for each of the n objects into a matrix G and compute the eigenspace of this G.

You will again need to "analyze" the eigenspace to determine how many eigenvectors are required for accurate classification. Similar to how the local projections were computed to construct the individual appearance manifolds $\mathcal{M}_n$, we need to compute the global appearance manifold $\mathcal{M}_g$ for object recognition.

Once the local and global manifolds have been computed, as well as the local and global subspaces, you need to display your results for a few of the objects as well as the entire global

eigenspace (using a $k = 3$-dimensional subspace for visualization). I recommend having a look at: object 1 (boat), object 2 (cabinet), and another object of your choosing. Explain what you notice about object 2 and why it might be difficult to ascertain its orientation from a 3-dimensional subspace. In addition to examining the appearance manifolds, you need to display the first few (5-9) eigenimages for the same objects you analyzed above as well as the eigenimages of the global eigenspace. An example of what the eigenimages might look like is illustrated in Figure 2.

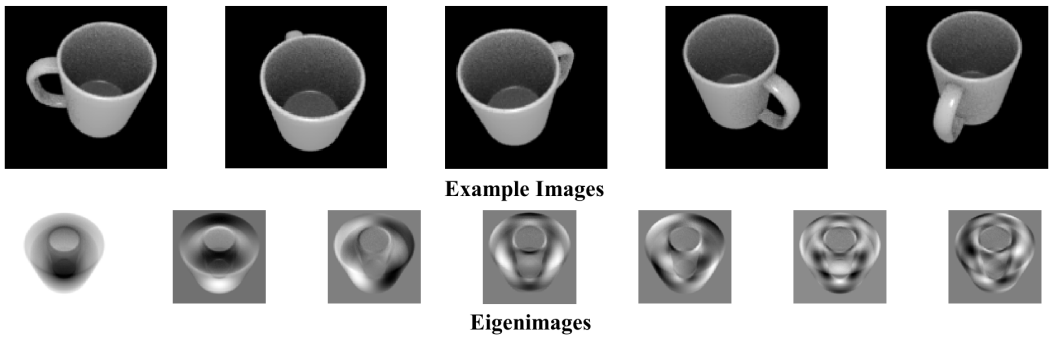

Figure 2: an illustration of what the eigenimages of a coffee cup might look like when sampled according to the right side of Figure 1.

Part 2: Computing the appropriate subspace dimension.

In Part 1 of this project, we chose the subspace dimension k and k_n by simply examining the eigenvalues. While this approach does work, it is somewhat ad-hoc because it's difficult to quantify the amount of variance captured. In this part of the project we will develop a technique to quantify how many eigenimages are required by examining how much energy they are capable of recovering from our original image set. Toward this end, we define the energy recovery ratio as a means by which we can estimate the amount of information recovered from a set of images for some (any) orthogonal k-dimensional subspace. The computation is as follows:

$$\rho(X, \mathbf{u}_k) = \frac{\sum_{i=1}^{k} \|\mathbf{u}_i^T X\|_F^2}{\|X\|_F^2} \tag{1}$$

where $\|\cdot\|_F$ is the Frobenious norm, X is the original image data matrix (pre-unbiasing), and $\mathbf{u}_i$ are the associated eigenvectors of $\frac{1}{2B} XX^T$. The energy recovery ratio achieves a maximum value of 1 when $k = 2B$.

For this part of the project, write a function `ComputeER` that takes as inputs the original image data matrix X, the original eigenimages $\mathbf{u}_i$, $i = 1 \ldots 2B$, and the desired energy to be recovered μ. The function should return the subspace dimension required to achieve this energy recovery as well as the energy recovered at each level so you can plot the energy

recovery ratio ρ as a function of subspace dimension k or k_n. In your results, create a table that has each object listed and the required subspace dimension for ρ to achieve 80, 90, and 95 % of the energy for each object. In addition to the table of results, **you need to display the energy recovery ratio as a function of subspace dimension** for a few objects of your choice.

Part 3: Object classification system.

Using both the local and global subspace computed above, develop a system to classify the objects from each of the test images. Your system should output a list of objects to the user and display the format you would like them to enter their choice on the command line. Once the user enters their choice, you should read this choice from the set of test images, classify the object, and return the results. At a minimum, your returning of the results should involve letting the user know what object they entered, what it's orientation is, and display (side-by-side) their test image along with the closest matching test image in the training set.

In addition to your user interactive system, you need to analyze the results of your system for **all** test images of each object. You might consider evaluating how accurate the classification results are using an ROC curve for the recognition step, and you can use simple statistics (ave error, etc.) for the pose of each object. **This should be a thorough analysis of your system and should be performed at different subspace dimensions to verify what energy recovery ratio might be sufficient for accurate classification.**

Deliverables:
You NEED to read very carefully what is required for each of the above steps and provide this!

Rubric

- +50 pts: Working implementation of image object classification (pose estimation and object recognition) with subsequent analysis for each object (i.e., how well it works $\rightarrow$ classification rate).
- +25 pts: Working implementation illustrating eigenimages for a few objects.
- +20 pts: Working demonstration illustrating the computation of energy recovered for a few objects.
- +05 pts: Writeup/Notebook.
- -05*n pts: Where n is the number of times that you do not follow the instructions.

Extra Credit: (10 pts. / item - max 20 pts.)
Here are some possible extra credit extensions (feel free to come up with your own variants to make your program more robust and efficient):

(a) Implement a version of LDA for object recognition and subsequent PCA for pose estimation.
(b) Develop some techniques to do a better analysis of the appearance manifold for each object (think differential geometric approaches here).
(c) Develop a method to map the appearance manifold to some low-dimensional manifold with desirable geometric characteristics (i.e., a circle in $\mathbb{R}^2$ for pose estimation).

Some suggestions:
This is a big project. Therefore, I suggest that you divide and conquer. You should write modular code so you can reuse it for future projects (i.e., functionalize everything if possible). I would also recommend you write some helper functions as needed.