Object Recognition and Pose Estimation

Dana Jensen

December 4, 2014

1 DESCRIPTION

Design a set of MATLAB functions that will receive multiple sets of 64 images that show a
3-D image at different orientations, rotated about a single axis but viewed at an off center
axis. Using these image sets "train" the system to recognize a different set of test images
of the same object at random orientations. To train the system the SVD is calculated for
the image sets, both individually and together, and used to calculate the Eigenspace of size
that gives sufficient energy recovery. The size sufficient for energy recovery is calculated by
using the image set and the Eigen vectors to compute the energy recovery ratio via equation
1 below. The size is k is increased until the energy recovery ratio is a least \( \mu \) where \( \mu \) is the
percentage of total energy. This is done for each object individually and then together to
create global and local Eigenspaces. After training the system the user is asked to input a
number between 1 and 192 which corresponds to a random object at a random orientation.
This object is then processed against the trained system which returns what object it is and
what angle it is at. The test object is displayed next to its match and the user is told the
object name and angle.

2 FUNCTIONS

This section outlines the method in which the system is trained and objects are matched
against it.
2.1 MATLAB Script: ObjectRec.m

This script first sets up which objects will be made into our training/testing set. This is done via line 5 of "ObjectRec.m", "objects=[1,7,17];". The user can change these three numbers to selected different objects for training and testing. The number can also be increased given a system with sufficient memory. The three objects selected for this demonstration and analysis were the boat, handle, and scooter. It then calls the function "compEigen.m" passing in the objects variable and receiving the local and global Eigenspaces and the local and global manifolds. The script then has two separate functions that can be chosen by the user by commenting out the sections defined in the code and un-commenting out the other section.

The first choice prompts the user for a number between 1 and 192. This will correspond to an object at a random orientation. The first 64 numbers correspond to the first object in the defined "objects" variable, the second 64 for the second object, and the third 64 for the third object. This automatically adapts if the user changes the number of training objects from three. This user input value is passed to the function "findObjectsAndAngles.m" along with the Eigenspaces and manifolds computed earlier. A cell array captures the name of the object and the estimated angle of its pose. The user is then prompted if they would like to test another image and will continue this process until the user selects to not continue. The second choice steps through all test images one at a time and uses "findObjectsAndAngles.m" to get all the object names and estimated angles. This function will automatically adapt for changes in the "objects" variable.

Figure 2.1: Sample Input Using Choice 1

2.2 MATLAB Function: compEigen.m

This function is called via the MATLAB script "ObjectRec.m" and functions to compute the local and global Eigenspaces as well as the local and global manifolds by receiving "objects" variable from "ObjectRec.m". For each object selected this function loads all 64 images from its training set and reshapes each one into a 16384x1 matrix, 16384 is the image size squared which is 128x128 in the case of these training images. Each image is then concatenated to two different matrices one for local calculations and one for global calculations.
The local matrix is replaced with each object but the global matrix continues to add each image on to the end.

The singular value decomposition is then calculated of the local matrix this will be used to calculate the Eigenspace and the manifold from the U. However the full U matrix is not needed to accurately describe the objects and computation time can be saved if it is only the size needed for enough information for proper recognition and pose estimation. To find the size need the function "computeER.m" is passed the local image matrix and a value $\mu$ which is a percentage of total information available. $\mu$ is hard coded but can be changed by this user to balance speed and resolution. The function returns a value, "k", that is the required size of the U matrix that will yield the desired $\mu$ of information. The local Eigenspace and local manifold are calculated via equation 2.1 and 2.2 below, X is the local image matrix. After these are calculated the process is repeated for each object selected. After all local Eigenspaces and manifolds are calculated the the process is repeated separately using the global X matrix.

$$\text{EigenSpace} = U(:, 1:k)$$ \hspace{1cm} (2.1)

$$\text{Manifold} = U(:, 1:k)^T \ast X$$ \hspace{1cm} (2.2)

### 2.3 MATLAB FUNCTION: computeER.m

This function calculates the required number of entries for the U matrix from the SVD that is required to provide proper identification of object and orientation. To find the proper k value $\rho$, the energy recovery ratio calculated via equation 2.3, is compared to the $\mu$ that can be changed in the function "compEigen.m". First the k value is selected to be 1 and $\rho$ is calculated. If $\rho$ is not over $\mu$ then k is increased until that happens. Each local image set is analyzed with this function as well as the global image set, this ensures that all objects can be properly identified.

$$\rho(X, U_k) = \frac{\sum_{i=1}^{k} ||U_i^T X||_F^2}{||X||_F^2} \hspace{1cm} (2.3)$$

### 2.4 MATLAB FUNCTION: findObjectsAndAngles.m

After calculating the local and global Eigenspaces and manifolds this function receives that data as well as the index of the test object and the object definitions. Using this data the test image is read in and is transformed into the global manifold space by using equation 2.2 where U is the global Eigenspace and X is the single point. This point is then compared to all the other points in the global manifold by using the norm between each global manifold.
point and the test point. The smallest value from this is the point of the training data that is most like the test point. The index of the minimum norm value is found and the test image is compared next to the training image corresponding to this index.

To tell what object these two points are the index value is divided by the number of images in each object set, 128. The floor of this value plus 1 results in which object is most like the test point. To calculate the angle of the test object the corresponding index in the training set is used. This can be done because the training images have been taken at evenly spaced angles around the full 360 degrees rotation. That means with 128 images a training image was taken at every 2.8125 degrees. Knowing this the angle is calculated via equation 2.4 below. The training image is printed next to the test image and the object name and estimated are returned to "ObjectRec.m".

\[
\Theta = \frac{2 \pi \times (I - 128 \times (S - 1) - 1)}{128}
\]

\[\text{(2.4)}\]

3 Approach Considerations and Analysis

3.1 Computing the Global Eigenspace

In computing the global Eigenspace there were two possible approaches; compute the local Eigenspaces, combine them, and compute a new Eigenspace or combine the training image sets and compute the Eigenspace. The two options however are not equivalent. The second approach was the one that was chosen and it turns out that approach one results in no useful information. To illustrate this the manifolds of both methods are compared below in Figure 3.1. It can be seen that the second approach has fluid manifolds for each object that have plenty of separation where as the first approach the manifold is randomly scattered. With the second approach object detection is simply not possible.

(a) Approach 1
(b) Approach 2

Figure 3.1: Computing the Global Eigenspace
3.2 Selection of Test Objects and the K-ρ Relation

In testing the approach the objects were chosen for specific reasons. The boat, handle, and scooter are all uniquely shaped, as in they feature lots of edges and curves and not many flat faces. With flat faces as seen with the cabinet and the LED the required dimensions to describe the majority of their information are small because there is so little variation through the image set. The difference can be seen in both the Eigen images and the manifolds. In Figure 3.2 the Eigen images of the Scooter and the LED are compared. Similarly in Figure 3.3 the manifolds are considered. In these Eigen images the more black and white there is indicates more variation in the information at those points where as gray indicates little change. Another way of thinking of this is to equate the black to -1, the white to +1, and the gray to 0. Doing this we see changes as a wave where gray can be seen as a constant. Inspecting the Eigen images reveals that the scooter has lots of data and variation where as the LED has much more gray area, lack of variation.

Inspecting the manifolds is potentially more valuable and descriptive of how hard an object and its orientation is to describe. The manifold of the scooter is symmetric and smooth but it does not intersect or overlap. The LED’s manifold however is smooth and symmetric but its points lie closely together. This shows that two different orientations have almost exactly the same values and are therefore hard to distinguish. That means that objects with low k values for high ρ are hard to distinguish. Therefore the hardest to distinguish objects are the; cabinet, mug, LED, and trash.

![Figure 3.2: Eigen Images](image)

![Figure 3.3: Object Manifolds](image)
To further strengthen this conclusion one can look at the data that comes from the function "computeER.m". Figure 3.4 shows the required value K such that the value $\rho$ meets a specific $\mu$. From the chart it is seen that to recover 99% of the data on the LED it only requires that K only be 7 where the keyboard requires K to be 112. From the graph it is visually simple to find which object are easiest to retrieve energy from and therefore the hardest to calculate orientation for, the quicker it reaches 100% the harder it is to correctly identify orientation.

(a) K vs. $\mu$ - Data   (b) K vs. $\mu$ - Graph

Figure 3.4: Illustrating the K-$\rho$ Relationship

4 ANALYSIS AND DATA

Using the implemented functions and data sets provided the following data was collected and analyzed, additional data can be collected by the user for other object selection. The first nine Eigen images for every object were recorded and can be found in the project attachments. Similarly the manifolds for each object can be found with the Eigen images. The Eigen images and manifolds for the selected objects are shown below in Figure 4.1 and Figure 4.2 respectively.

The energy recovery ratio for each object was calculated at different $\mu$ values to find the corresponding K value for this. The full data set and graph can be found above in Section 3 Figure 3.4. Below the graph for just the selected objects can be found in Figure 4.3. The most important part however was the results of the recognition and pose estimation. Detailed results can be found in the attached file "Tested.xlsx", which was produced using "ObjectRec.m" option 2. Summarized results can be found in Figure 4.4. From this data it can be seen that all objects in the test set were correctly identified as the correct object and the angle estimation of its pose had an average error of 0.646 degrees across all test images. The maximum error was 1.49 degrees while the minimum error was 0.018 degrees.
Figure 4.1: Selected Eigenimages

(a) Boat

(b) Handle

(c) Scooter

(d) Global

Figure 4.2: Selected Manifolds

(a) Boat

(b) Handle

(c) Scooter

(d) Global
Figure 4.3: K vs. $\rho$ - Selected Object Graph

Figure 4.4: Summarized Object Recognition and Pose Estimation Results

<table>
<thead>
<tr>
<th>Results</th>
<th>All Objects Correctly Identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Angle Error</td>
<td>0.011278 rads 0.646157 Degrees</td>
</tr>
<tr>
<td>Max Angle Error</td>
<td>0.025987 rads 1.488929 Degrees</td>
</tr>
<tr>
<td>Min Angle Error</td>
<td>0.00032 rads 0.018332 Degrees</td>
</tr>
</tbody>
</table>
5 Conclusion

The functions and scripts complete the required tasks and are adaptable to systems with more or less performance capabilities. The user can select what object and how many object sets to include in the training and testing bank by modifying the variable "objects" in the "ObjectRec.m" script. There also many commented out sections with different approaches or data printing functionality. It is currently setup to only give the test objects name, estimated angle, and to display the two matched images together. The performance of the system can be altered greatly by changing $\mu$ in "compEigen," function. A value between 1 and 100 can be placed here with lower values operating faster but potentially providing insufficient data to match well. A definitive number for accurate matching can not be given as it will depend on the individual object selected. Objects with lots of features will need a smaller energy ratio but a larger $K$ value to achieve that energy ratio. Objects with fewer features will require a much higher energy ratio with a smaller value of $K$ required to reach that value. Figure 5.1 below shows the working environment of the code.

![Figure 5.1: Object Recognition and Pose Estimation](image-url)