Questions may continue on the back. Please write clearly. What I cannot read, I will not grade.

1. This exercise will guide you through the construction of software for retrieving images of pressed leaves from a database. The database was created by crawling the site http://www.missouriplants.com for images whose name ends with the string `pressed_leaves.jpg`, carving out individual leaves, and pasting them onto a perfectly black background. The resulting images are available on the homework web page.

Since pressed leaves are unlikely to appear distorted by viewpoint changes other than scale and rotation changes, regular SIFT descriptors, without the affine invariant property, will suffice. To compute these, you will need Andrea Vedaldi’s SIFT code, which is available through the class syllabus web page.

Only shape is relevant to this exercise, so it’s best to first transform the database images to binary format. Vedaldi’s code assumes images in double format, so if `img` is a (color) database image, you can do

```matlab
binary = double(mean(img, 3) > 0);
```

(a) Compute SIFT frames and descriptors for all the images, and save each `(frame, descriptor)` pair into a file whose name is related to the image file name. As you go along, count how many descriptors you have in total, so you can later allocate matrices of appropriate sizes. Hand in your code for this part.

(b) Compute two matrices `F` and `S`, with 5 and 128 rows, respectively. The first four rows of `F` are a concatenation of the `frame` outputs from Vedaldi’s SIFT code `(x, y, σ and θ for each descriptor)`. The fifth row stores the number of the image that the descriptor comes from. The matrix `S` concatenates the `descriptor` outputs from the SIFT code. Hand in your code for this part.

(c) The file `PCA.m` available with this assignment computes the principal components of a matrix. Specifically, the call

```matlab
[S P m sigma] = PCA(S, c);
```

subtracts the mean `m` from the columns of `S` to obtain a centered matrix `Sc`, computes a matrix `P` whose rows are an orthonormal basis for the `c`-dimensional space that captures most of the variance of the columns of `S`, and replaces `S` with `P*Sc`. The vector `sigma` has as many entries as the input matrix `S` has rows. These nonnegative entries are the standard deviations of the data distribution in the principal directions.

Think of this operation as a “cleanup” and compression of the original data, with the new columns of `S` having now only `c` entries rather than 128. A new SIFT descriptor `s` can be projected into this new basis by the assignment

\[
s = P * (s - m);
\]  

First run `PCA` with an arbitrary (positive integer) value for `c`, and plot `sigma` as dots (that is, with the ‘.’ line style). Then decide a value for `c`, and run `PCA` again with that value. Hand in your plot and your value of `c`. A good value corresponds to the last, clean gap in the descending values in `sigma`. If no such gap exists, or if it comes too early, pick a reasonable value.

(d) Run the `k-means` algorithm (help `kmeans` in Matlab’s Statistics Toolbox) on the reduced matrix `S` obtained in the previous question. Use the squared Euclidean distance (default), and compute `K = 20` clusters. Read the tutorial help files for `kmeans` to understand how to use the ‘replicate’ option to reduce the problem of local minima. Show your code.

(e) Read the tutorial help files for the Matlab Statistical Toolbox `silhouette` function to understand what a silhouette diagram is, and why this gives an idea of the quality of a clustering solution. Using the Euclidean distance in `silhouette` (rather than the squared Euclidean distance), show the silhouette diagram for your solution.

(f) This question invites you to think about the meaning of a silhouette diagram. Using again the Euclidean distance, describe the simplest situation you can think of (that is, a very small number of clusters for a very small number of data points in a very small number of dimensions) in which (i) the mean vectors `m_k` are plausible outputs from the k-means algorithm and (ii) the silhouette value for at least one point `s_i` is negative. Explain why the situation you describe satisfies these two properties.

As discussed in the Matlab help files, the silhouette value for point `s_i` is

\[
v_i = \frac{(\min_{k \neq k_i} b_{ik}) - a_i}{\max(a_i, \min_{k \neq k_i} b_{ik})}
\]

where

- `k_i` is the index of the cluster that point `s_i` belongs to;

Due on October 9, 2008
• $a_i$ is the average distance between $s_i$ and the other points in cluster $k_i$;
• $b_{ik}$ is the average distance between $s_i$ and points not in cluster $k_i$.

Also, a set of means $m_1, \ldots, m_K$ is a plausible output from the k-means algorithm if it is a local minimum for the quantization error

$$e(m_1, \ldots, m_K) = \sum_{i=1}^{N} \min_{k=1,\ldots,K} \|s_i - m_k\|^2.$$

(g) Build a matrix of the term-frequency, inverse-document-frequency (tf-idf) scores for your database, using the means $m_k$ from the previous problem as your “words.” The word identifiers for each descriptor are already contained in the first output argument from kmeans. The tf-idf score is defined in the paper by Sivic and Zisserman (accessible through the class syllabus web page). A “document” in our context is of course one of the database images. Hand in your code.

(h) For each of the leaf images $I_d$, use the tf-idf matrix computed earlier to rank all images in order of decreasing similarity $\lambda(I, I_d)$ to $I_d$, where

$$\lambda(I, J) = \frac{\text{score}(;I)^T \text{score}(;J)}{\|\text{score}(;I)\| \|\text{score}(;J)\|}$$

is the standard “cosine metric” used in document retrieval, and score is the tf-idf score computed earlier. The result of this step is a square matrix $R$ with as many rows and columns as there are images in the leaf database. Entry $R(i,j) = r$ means that image number $r$ is the $j$-th most similar image to image number $i$. Check that $R(i,1) = i$, that is, that each image is most similar to itself. Hand in your code and an image $I$ in the style of the one in Figure 1. The thumbnails in that image are 32 by 32 pixels in size. You can make thumbnails with the Matlab imresize function.

(i) Comment on your results. What works, what doesn’t, why, and how could results be improved?
Figure 1 The first thumbnail in each row of five represents a “query” image; the other four are the four top ranked matches (without displaying the query image a second time). There are 163 rows of five thumbnails each, one row for each image in the leaf database. Results may vary. The ones in this Figure are neither great nor terrible.