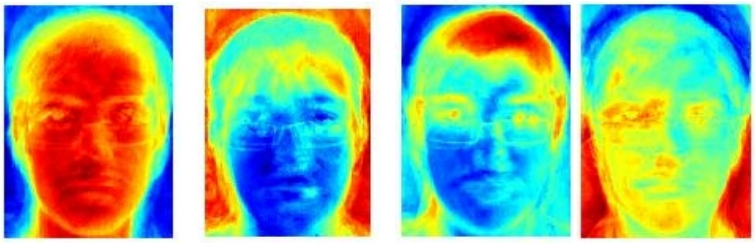
CS 378 / CS 395T Computer Vision
Fall 2007
Problem set 4
Due Tuesday Dec 4

Submit both as hardcopy and electronically through the turnin program. If you have multiple files, tar them together, and use your first initial and last name to name the tar file. Then submit with:

turnin --submit svnaras pset4 <yourfilename>

In your hardcopy, include the code as well as any relevant figures.

Part I. Eigenface recognition (50 points)



For this problem, the goal is to perform face recognition using the Eigenface approach by Turk and Pentland. We have provided code to compute a face subspace and project face images onto that subspace, as well as two datasets of face images, one from the Caltech database, and one from photos contributed by students in the class. The code provided is in eigenfaces_code.m, and the data are in caltech_face_data.mat and class_face_data.mat.

Be sure to read all the comments in eigenfaces_code.m before starting this problem, and also run it. This code does not yet perform the final recognition; you will fill this and other additional steps in below. Note that you will need to choose appropriate values of 'numTrainPerClass' and 'k' for each dataset, as they contain different numbers of images.

Complete and respond to each of the following. Unless otherwise noted, apply each step to both of the datasets provided.

a) Use the distances in face space between the novel test inputs and the labeled training examples to classify each face in the test set according to the person's identity. Assume that every test input is indeed a face, and you need to say which one. What is the accuracy, in terms of the percentage of test images correctly classified? What would the accuracy of a random guess (chance) be for each dataset?

b) Compute the reconstructions for each the training and testing example according to their projections onto face space. Given a mean face vector $\boldsymbol{\mu}$, the reconstruction $\hat{\mathbf{x}}$ for an input $\mathbf{x}$ that has the face space projection $\mathbf{w} = [w_1, \ldots, w_k]$ is:

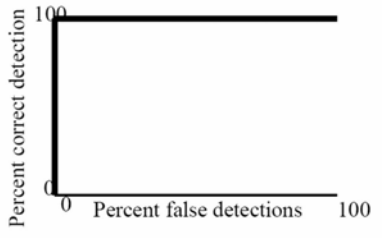
$$\hat{\mathbf{x}} = \left(\sum_{i=1}^{k} w_i \mathbf{u}_i \right) + \boldsymbol{\mu}$$

Then display and view the results as images (see the use of '`reshape`' in the given code to turn a vector representation back into an image). First describe qualitatively how the reconstructed images of the training vs. testing images compare for a fixed value of *k*.

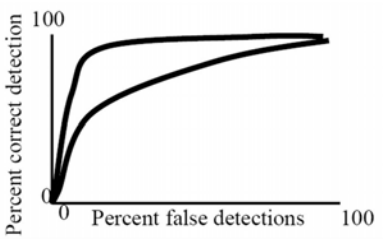
Then try a low and high value of *k* and compare the reconstructed images. Explain why the results look the way they do and include figures showing some of the reconstructed images for each case.

c) Now include the face detection task to filter out non-faces. For the test examples in the Caltech data, there are also some images that are not faces. For each test example $\mathbf{x}$, check the distance between it and its reconstruction: $d = \left\| \mathbf{x} - \hat{\mathbf{x}} \right\|^2$. If this value is above a threshold ($d \geq \theta$), then the input is too far from face space to be a face. Otherwise, say it's a face and label its identity as above. Perform this step only for caltech_face_data.mat file, as there are no non-faces in the class_face_data.mat file.

d) An ROC (receiver operating characteristic) curve allows us to see the tradeoff between false positives and true detections as a threshold in the decision function is changed.



Ideal ROC



Two typical ROCs

Each setting of the threshold will result in particular false positive and true positive rates on the test data, and thus will generate one point to plot on the curve. In this problem, the false detection rate is the percentage of non-faces that are labeled as faces, and the correct detection rate is the percentage of true faces that are labeled as faces.

Generate an ROC curve for the test data in caltech_face_data.mat that shows the tradeoff as the threshold for face detection θ is varied. Be sure to choose a reasonable range of thresholds so that you observe the tradeoff occurring. Submit a figure showing the curve, and explain the result.

e) **(Optional, extra credit worth up to 25 points)** How can we attempt to visualize what variation is being captured by the eigenfaces? Design a way to explore the modes of variation captured in these face subspaces. Explain your approach and interpret the results. Show figures to illustrate as necessary.

Part II. Image metamorphosis (50 points)

First image | Morph (α=0.5) | Second image

To morph between two images is to generate intermediate interpolated images in which the object in the first image slowly changes into the object in the second. Computing a morph can be useful for both vision and graphics applications. In general, we have to consider both the shape and the appearance (texture) of the object we are morphing. Basic image morphing relies on two primary steps:

1) Warp both images in order to align their shapes.
2) Blend the appearance of the two aligned images.

For this problem, you will use the observed motion between two sequential frames (the optical flow) to estimate the aligning warp. Then you will blend the warped images by taking a weighted average of their intensities at each position.

Consider two image frames from some video sequence, I_1 and I_2. First, we need to compute the optical flow from I_1 to I_2; then we get the optical flow from I_2 to I_1. (Note: both of these flow fields are already computed for you. They are the matrices 'A' and 'B', respectively.) Given these flow fields, we want to compute two reverse-warped images W_1 and W_2 as follows:

$$W_1(x,y) = I_1(x + \alpha dx^{(2\to 1)}, y + \alpha dy^{(2\to 1)}), \text{ and}$$
$$W_2(x,y) = I_2(x + (1-\alpha) dx^{(1\to 2)}, y + (1-\alpha) dy^{(1\to 2)}),$$

where α is a weight specifying how much of the flow to add, and the values $dx^{(i\to j)}$ and $dy^{(i\to j)}$ correspond to the flow amounts *from* image i *to* image j. (So, the displacements come from 'B' in the first equation, and from 'A' in the second equation. Both 'A' and 'B' are n x m x 2 matrices, which give the x and y flow displacements at each of the pixels for the n x m images. A(:,:,1) and B(:,:,1) are the flow values in the x-dimension, while A(:,:,2) and B(:,:,2) are the flow values in the y-dimension.)

It may help to first imagine computing these warped images with $\alpha = 0.5$, which would generate two warped images that both estimate the midpoint shape between the two input frames. Note that if $\alpha = 0$, W_1 will simply yield the first image; if $\alpha = 1$, W_1 will yield the second image (assuming perfect flow).

The optical flow fields are at sub-pixel precision, so the positions required for the warped images may be non-integer as well. In order to compute the intensity of an image at a sub-pixel coordinate, use the `'interp2'` function. The parameters to `interp2` are the 2d image data, and the list of *x* and *y* positions you want to sample it at. For example, say `samplex` and `sampley` are *h* x *w* matrices listing the positions we want to sample from for each (*x,y*) in an *h* x *w* image `im1`. Then to get the values of `im1` at the positions listed in `samplex` and `sampley`:

```
sampled = interp2(double(im1), samplex, sampley);
```

The result is another *h* x *w* matrix. Watch out for any `NaN`'s generated by `'interp2'`; you can find them with the function `'isnan'`.

Having computed these two warped images for some value of $\alpha \in [0,1]$, we are now ready to blend their appearance. Compute the blended image B as the weighted average of the shape-aligned images W_1 and W_2:

$$B(x, y) = (1-\alpha)W_1(x, y) + \alpha W_2(x, y).$$

Note that the weights are opposite what they were above for the warp, relative to the first and second image. This is the final morph for that particular value of α.

The data for this problem are in the files flow_example_1_data.mat and flow_example_2_data.mat. Use 'load' to load them in. Each file contains the variables 'A', 'B', 'im1', and 'im2'.

For each of the two data files provided, use the given images and optical flow fields to compute a smooth morphing between the first and second frame, for a slowly increasing blending parameter, $\alpha = [0:0.1:1]$. Display the morphs for every step together in one figure, in order, labeled by the α value. You can also use the provided function `'writeMovieFromImageStack'` to write an avi movie file for each example that shows this transition. The movie you produce should look like a slow motion interpolation of the motions of the actors in these frames.

Submit two figures showing the smooth morphs (10 images each) for the two examples, as well as your analysis of the following:

a) Where are their apparent flaws in the optical flow estimates, and why?

b) Why is it important to correct for shape alignment before blending the intensities?