6.801/6.866 Machine Vision

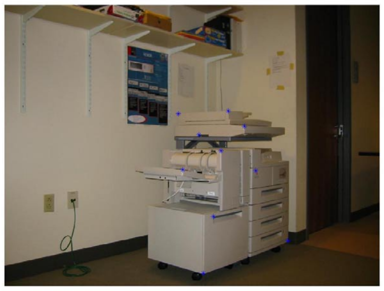
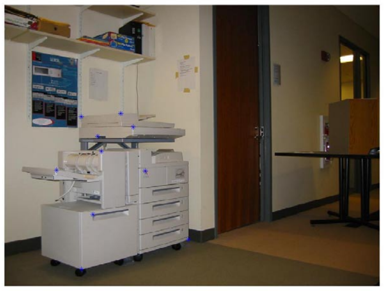
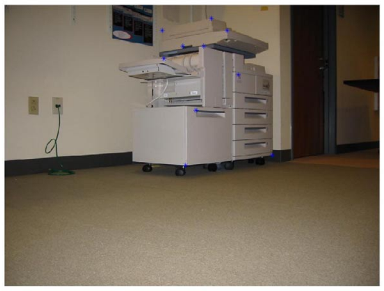
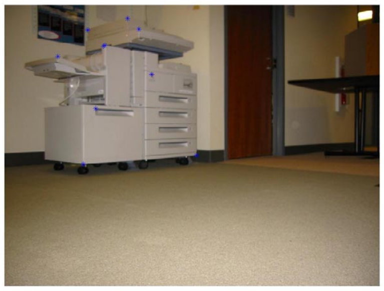
Problem set 4

Assigned: 11/09/02
Due: 11/21/02 at 2:30pm

Problem 1 Structure from Motion (Matlab)

The goal of this problem is to recover the shape of an object (Xerox photocopier) and the movement between each frame. To achieve this goal we will implement the original structure-from-motion algorithm of Tomasi and Kanade.

Here are the four images used in this problem to recover shape and motion:

In the first picture, features have been detected using a sophisticated algorithm. Using a similar algorithm, we tracked these features between images. The Matlab file `features.mat` contains the results of this tracking.

a) For each frame, recenter the camera referential at the object centroid by constraining the feature locations to have zero mean.

b) Write a Matlab function based on the factorization approach proposed by Tomasi and Kanade to compute the affine motion and the feature shapes given a set of tracked features. Your function should have the following syntax:

```
function [A, P] = TomasiKanade(D)
```

Test your function using the recentered features. Reproject the features shapes `P` onto the orthographic planes xy, yz and zx using the Matlab function `plot3`.

c) As described in section 12.4.2 (F&P), some affine transformation Q can be applied to A and P to upgrade from affine to Euclidean.
Find the matrix `C` by imposing the orthographic constraints on the affine motion matrix `A`. First create an objective function that returns a residual vector `F` given

an estimate of `C`. Then using the `lsqnonlin` Matlab function, minimize the orthographic constraints.
In our case, what displacement `d` should we apply?
Apply the *Euclidian upgrade* Q to the affine motion `A` and the features shapes `P`.
Reproject the new features shapes `P` onto the orthographic planes xy, yz and zx.

d) After applying the Euclidian upgrade, the rows of matrix `A` should represent the orientations of the horizontal and vertical camera reference axes. How can you align the first camera reference system with the world reference system? Reproject the aligned features shapes `P`.

Hint: To debug your Matlab implementation, create a synthetic data set using the first feature set (x_1,y_1) and some approximated depth values z_1. Apply arbitrary rotations on this 3D point cloud (x_1,y_1,z_1) to create 3 new feature sets. You will be able to compare your results with your original data set.

Problem 2 Eigenfaces for Recognition (Matlab)

In this problem we explore the use of eigenfaces for person recognition. Download the Matlab data file `faces.mat`, which contains cropped, normalized 51x43 images of 34 individuals. Each individual has been photographed with two facial expressions: “neutral” and “smiling”. The pixels of each person’s image have been stored as columns of the `neutralFaces` and `smileFaces` matrices. To convert back and forth between image and vector representations, use the `reshape` command:

```
faceImage = reshape(smileFaces(:,index), 51, 43);
faceVector = reshape(faceImage, 51*43, 1);
```

a) To (approximately) account for illumination variations, normalize each of the face vectors so that it has zero mean and unit variance.

b) Assume that we have a database of neutral face images, and want to determine the identity of the smiling individuals. For each of the 34 smiling faces, measure the norm of the difference between their normalized appearance vector and each of the 34 neutral faces. Classify each smiling face according to its nearest neighbor. What percentage of the smiling faces are correctly recognized?

c) Using the Matlab `svd` command, determine the principal components of the set of normalized neutral faces (do *not* include the smiling faces). Be sure to subtract the mean face before performing your PCA. Plot the mean face and the first three principal components (the “eigenfaces”).
Hint: To avoid excessive memory usage when calculating the SVD, use Matlab’s “economy size” option.

d) Determine the number of principal components required to model 90% of the total variance in the neutral face set. Project each of the neutral and smiling faces onto the corresponding eigenfaces. Use the coefficients of these projections to classify each smiling face. Compute the percentage of correctly recognized faces, and compare to part (b).

e) Repeat part (d) using the numbers of principal components required to model 80%, 70%, 60%, and 50% of the total neutral face variance. Plot the corresponding recognition rates.

Problem 3 Eigenfaces for Detection (Matlab)

a) Construct the principal components (eigenfaces) which best model the combined variations of the neutral and smiling faces from Problem 2. Be sure to normalize each face vector as in 2(a). Plot the mean face and first three eigenfaces, and compare to 2(c).

b) Load the test image `mad.png`. Use Matlab's `ginput` command to (approximately) locate the centers of this image's two (human) faces.

c) Repeat this part for each of the faces from part (b). For each pixel in an 80x80 pixel square centered around the face, determine the best reconstruction of the 51x43 patch centered at that point using the first 10 principal components from part (a). Each image patch should be normalized as in 2(a) prior to this reconstruction procedure. Plot the distance between the input and reconstructed (normalized) patches as a function of patch location. For the smallest error shift, plot the input and reconstructed image patches.

d) Repeat part (c), but this time search over the entire image. Make the same pair of plots as before.

e) Are eigenfaces suitable for face detection in general scenes? Why or why not? If not, are there controlled situations where they would prove more useful?

Problem 4 Affine cameras

Show that affine cameras (and the corresponding epipolar geometry) can be viewed as the limit of a sequence of perspective images with increasing focal length receding away from the scene. (F&P 12.8)

Problem 5

Exercise 22.2 from *Computer Vision: A Modern Approach* (Forsyth and Ponce)

Problem 6

Exercise 22.5 from *Computer Vision: A Modern Approach* (Forsyth and Ponce)