CS 378/395T Computer Vision
Problem Set 1

Due Tuesday September 25

For this assignment, please submit **hardcopy for all parts** (including code printouts and figures), and then **also submit your code electronically** using the turnin system. Tar all the files together, and use your first initial and last name to name the tar file. Then submit with:

```
turnin --submit svnaras pset1 <yourfilename>
```

Let Sudheendra know if you experience any problems submitting the code via turnin.

Part I.

1. **Color matching.** In color matching experiments with cone vision, at least how many primary lights are necessary to assure good perceptual matches? How many are needed for rod vision? Why? When multiple primary lights are employed for a color matching experiment, what property should their spectral compositions have in order to offer the most representational power?

2. **Moments and invariance.** The set S is a collection of (x, y) image coordinates that define a connected component centered around the origin. The (j, k)-th central moment of S is defined as

$$\mu_{jk} = \sum_{(x,y)\in S} (x - \bar{x})^j (y - \bar{y})^k, \tag{1}$$

where $(\bar{x}, \bar{y})$ is the centroid for S. The central moment μ_{jk} is invariant to translation, meaning that a new shape defined by a given shift of all points in S will produce the same value for μ_{jk} as S does. Define the *normalized* (j, k)-th central moment $\mu^{(n)}_{jk}$ as:

$$\mu^{(n)}_{jk} = \frac{1}{|S|} \sum_{(x,y)\in S} \Big(\frac{x - \bar{x}}{\sigma_x}\Big)^j \Big(\frac{y - \bar{y}}{\sigma_y}\Big)^k, \tag{2}$$

where $\sigma_x = \sqrt{\frac{\mu_{20}}{|S|}}$ and similarly $\sigma_y = \sqrt{\frac{\mu_{02}}{|S|}}$. Show that the normalized central moments are invariant to scale and translation. Note that for every point (x, y) in S the scaled and translated point will have coordinates $(ax + b, ay + c)$.

Part II.

1. **Chamfer-based shape matching.** The shape of a human silhouette gives some cues about the pose the person is in. For this exercise, you will write a program for nearest-neighbor Chamfer matching and apply it to some silhouette shapes of people in different poses. Download the .pgm images provided on the class webpage. There are 56 database examples and five query examples. For each query image, the goal will be to retrieve from the database those images that have the most similar contour shape as measured by the Chamfer distance.

   Write a program that reads in all the images and uses distance transforms to compute the symmetric Chamfer distance between every query and every database example. To make the distances symmetric, add the Chamfer distances between the query and database image in both directions ($D = chamfer(A, B) + chamfer(B, A)$). Then for each query, (1) sort the database images according to this distance, and (2) display the query image and its top k nearest-neighbor images from the database.

   Analyze the results: how well does this work for comparing 2D silhouettes? For finding similar examples in terms of the underlying 3d pose? What makes the different results better or worse? Your submission should include the figures displaying each query and its nearest $k = 3$ neighbors.

   (Useful Matlab function: ‘bwdist’.)

2. **Edge detection.** The Canny edge detector uses hysteresis to attempt to get rid of local maxima due to noise: a higher threshold is first used to start an edge chain, and then a lower threshold is used when following it. The higher threshold determines which next unvisited edge pixel is to be processed, while the lower threshold determines how strong a connected local maxima must be in order to be included in the current chain.

   Use the built-in Matlab function ‘edge’ to experiment with three parameters to the Canny edge detector: the Gaussian filter’s standard deviation (σ), the ‘low’ threshold, and the ‘high’ threshold. Perform edge detection with the supplied image named ‘canny_test_image.pgm’. Empirically find a few (3) settings for each parameter that yield noticeable changes in the resulting edge image. Then form three figures: for each of the σ settings, display in one figure the edge images resulting from each pair of low/high thresholds. Label the images in the figure according to the parameter settings so you can tell which is which.

   Analyze the impact of each parameter on the algorithm and describe why the images you have created look the way they do. Briefly summarize your findings, and also submit the saved figure plots for reference.

   (Useful Matlab function: ‘edge’.)

3. **Background subtraction and blob tracking.** For many video-based vision systems, the initial processing requires identifying any foreground objects, and tracking them over time. For example, to analyze data coming from a traffic

surveillance camera, we'd first want to isolate the regions of interest corresponding to cars or pedestrians, and to maintain those foreground "blobs" for as long as they are present in the scene. When knowledge about the appearance of the background is available—and particularly when that appearance is fairly static—*background subtraction* methods may be used to attempt to extract and segment the foreground objects.

Write a program to do automatic background subtraction and blob tracking. Use the images in the provided 'bg_sub_data/background' directory to model the background as described below, and then mask out (subtract) that background from every frame of the provided test sequence in 'bg_sub_data/test_sequence'. There are a total of 11 background images and 201 test sequence frames. For each component below, submit a single representative frame that displays the foreground output.[1]

(a) The simplest way we can describe the background is to represent it with a single instance of the empty scene, in which no objects present are considered foreground. Take one random image from the 11 background images provided. For every test image, compute the squared difference between the current frame and the selected background image. Choose an appropriate threshold on the squared difference values in order to generate a binary foreground mask, in which only those pixels with intensities differing greatly from the known background are marked as 1. Apply that mask to the current frame to display the result. Play with the threshold as needed to get the best foreground detection.

(b) Now enhance the background model by using multiple background frames. Estimate the mean and variance of the intensity at every pixel using all the background images (this should be reminiscent of problem set 0). We want to use the background's intensity distributions to take into account the relative range of fluctuations that occur in each pixel, and penalize deviation from the mean background value accordingly. For every frame in the test sequence, compute the Mahalanobis distance between the mean background image and the current frame. For a background pixel with mean value x and variance σ^2, given a new pixel intensity y, the squared Mahalanobis distance will be $d(x,y) = \frac{1}{\sigma^2}(x-y)^2$. Watch out for any 0-variance pixels.

    Choose an appropriate threshold on these distances to form a foreground mask. View the results for the entire test sequence. Though a static scene, the background frames differ. Where is the most variance? Why? Compare the results to the single-image model used above.

(c) Use morphological operators to improve the foreground masks. Experiment a bit with structuring elements and operators, but then apply the same dilations and/or erosions to every test sequence frame. Try to make it so

[1] Single frame displays are all you need for the assignment, but FYI - Matlab has functions to compile movies from image frames (see 'movie', 'immovie', etc.).

that the largest foreground blob will often correspond to the object of interest.

(d) Label the connected components in the binary foreground images. Have the program identify the largest connected component in each test frame and treat it as the object of interest. Plot a marker on the centroid of this blob on top of each test frame, and as it displays the processed sequence, print to the screen the series of (x, y) locations that are tracked.

Your submission should include an analysis of the output, as well as figures showing a single processed frame that results when you use (a) the single-frame background model with squared differences (b) multiple-frame background model with the Mahalanobis distance, (c) add morphological operators, and (d) plot a point denoting the centroid of the tracked object.

(Useful Matlab functions: ‘var’, ‘mean’, ‘bwlabel’, ‘regionprops’, ‘hold on’, ‘plot’, ‘pause’, ‘imdilate’, ‘imerode’.)