Computer Science 570/670 Computer Vision

Assignment 5: Due on last day of class

The goal of this problem set is to build a backgrounding system and use it to do simple object tracking in a video stream.

Part 1. Color histograms and "soft histogramming"

Write a function in matlab which forms a 3-D histogram from a set of rgb pixel values. In MATLAB, you could represent a set of rgb pixel values as an Nx3 matrix, where each row represents one rgb value. You should normalize each histogram by dividing by the number of elements in it, which will convert it into a probability distribution.

Once you have this working, alter your code so that instead of adding a point to a single histogram bin that it falls in, it adds points to the eight nearest bins weighted by the distance from the bin center. This is called "soft histogramming". If you have trouble visualizing this for a 3-D histogram, you might want to start with a 1-D or 2-D histogram. For example, in 1-dimension, if a point falls on the boundary between two bins, you want to add .5 to each of the two bins. If it's 70% of the way between the center of the first bin and the center of the second bin, then add 0.3 to the first bin and 0.7 to the second bin. You should generalize this procedure to 3-D. Finally, if a point falls on the boundary of the color space, you may only want to distribute its weight among 4, 2, or 1 bins, depending upon where it falls.

Part 2. On the course web site, you will find files called training_10.mat, training_100.mat, testing_10.mat, and testing_100.mat. You will use the training files to build your background model and the testing files to detect moving objects. The numbers 10 and 100 refer to the number of frames in each file. Start by using the smaller versions (with 10 frames), and once you get your code working you can try it on the larger files.

Each file is in matlab format. Just load each one in by typing "load filename" without any parentheses or anything. Each file contains a matlab "movie". You can use the movie command to play the movie. You won't see much in the training movies, since there is little movement, but you should see people walking around in the test movies.

It is easier to deal with the movies as collections of images rather than as movies. A movie is an array of "frames". You can use the command frame2im to convert each frame to an image before processing it. It will turn into an rgb image.

The next step is to turn the set of training images into a distribution field using the color histogram tools you developed in part 1. Write a matlab function which takes the whole movie and returns a distribution field consisting of normalized color histograms at each pixel.

Part 3. Use the distribution field you built from the training data to assess the likelihood that each pixel of a test image is in the background. There are many ways to do this including:

1. Set a threshold on the probability of a pixel color. If the value is above the threshold, it's determined to be background. If the probability is below the threshold, it's determined to be foreground.

2. Compute the entropy of the normalized histogram at a pixel. This is done by computing $-p(x)\log p(x)$ for every bin in the histogram (except for the ones with value equal to zero) and adding them together. Then check if a pixel has negative log likelihood less than some fixed percentage of the entropy. For example, if $-\log p(color) < .1H$, where H is the entropy of the distribution at a particular pixel, then consider it to be background.

3. There are many other ways to do this. You can use one of the methods above, or invent your own.

Write a function which classifies each pixel in a text image as foreground or background by comparing it to your background model.

Part 4. Your results from part 3 may end up being scattered pixels, rather than coherent "blobs" representing people or cars. To try to find blobs, write a function to dilate the binary image produced from part 3. The function dilate should take a binary image and color each pixel white if either it or its neighbors is white. Then write a function dilateN which dilates an image multiple times according to an argument N. Experiment with dilateN to see what value connects the pixels in a tracked object with expanding beyond the object too much.

Part 5. Once you have dilated an image to get some "blobs", write a function which finds the set of all blobs (connected sets of white pixels) in the image. The easiest way to do this is to use a "flood fill" on the binary images. You can look up flood fill on Wikipedia if you don't know how it works. Perform flood fills on the binary image until all of the blobs are found. You may want to ignore blobs with fewer than some number of pixels.

Part 6. Finally, for each blob found in part 5, draw a box around the blob and in the same position in the original image. Make a movie out of the images, showing the boxes around the original tracked objects.

Turn in all of your code, and a few example images from each stage (if there are any images to show).

Part 7. Required for graduate students, extra credit for undergrads. Augment the above model to use a foreground model and a prior model of foreground and background. You can create these models however you want, but one suggestion would be to create a foreground model from the colors obtained in the first version of the program running on the test_1000 video. Use Bayes rule to classify pixels as foreground or background. Is there any significant difference between the Bayesian backgrounding and maximum likelihood backgrounding?