CS 663 : Digital Image Processing : Assignment 3

Instructor : Suyash P. Awate

Note: The input data / image(s) for a question is / are present in the corresponding data/ subfolder.

5 points are reserved for submission in the described format.

1. (30 points) Harris Corner Detection.

   Input image: `1/data/boat.mat`

   Assume the pixel dimensions to be equal along both axes, i.e., assume an aspect ratio of 1:1 for the axes.

   Shift and rescale the intensities in the image to lie within the range $[0, 1]$.

   Implement the Harris corner detector algorithm. The parameters underlying this algorithm are: two Gaussian smoothing levels involved in computing the structure tensor (the first Gaussian to smooth the image before computing the gradient, the second Gaussian for the weighted averaging to compute the structure tensor), the constant k in the corner-ness measure. Tune these three parameters to get the best results.

   • (16 points) Write a function `myHarrisCornerDetector.m` to implement this.

   • (4 points) Display the derivative images, corresponding to the partial derivatives along the X and Y axes.

   • (4 points) Display an image (along with a colormap) of the principal eigenvalue of the structure tensor evaluated at each pixel. Display another image (along with a colormap) of the other eigenvalue of the structure tensor evaluated at each pixel.

   • (6 points) Display the image (along with a colormap) of the Harris corner-ness measure. Tune the free parameters such that positive values in this image should correspond to "corner" structures in the image. Report all three parameter values used.

2. (45 points) Image Segmentation using Mean Shift.

   Input images: `2/data/baboonColor.png, 2/data/bird.jpg, 2/data/flower.jpg`

   Take each image, smooth it using Gaussian convolution with a standard deviation of 1 pixel-width unit, and subsample the smoothed image by a factor of 2 in each spatial dimension to produce a smaller image. Use these smaller-sized images for the following experiment. If these images still lead to a computational cost that is beyond your computer's capabilities to allow for a sufficiently comprehensive experimentation, then you may resize further.

   • (24 points) Write a function `myMeanShiftSegmentation.m` to implement the algorithm for mean-shift image segmentation using both color (RGB) and spatial-coordinate (XY) features. Tune parameters suitably to get a segmented image with at least 5 segments and no more than 50 segments. To improve code efficiency, you may use Matlab functions like knnsearch(), bsxfun(), etc. For these images, about 20–40 iterations should be sufficient for reaching close to convergence, but feel free to tune this stopping criterion as suitable. You may select a random subset of nearest neighbors, in feature space, for the mean-shift updates to reduce running time. Each iteration can run in about 10-40 seconds on a typical personal computer.

• (12 points) For each input image, display the (i) original image along with (ii) the segmented image that shows color-coded pixels (and, thus, segments) using the color component of the converged feature vectors.

• (9 points) For each input image, report the following parameter values: (i) Gaussian kernel bandwidth for the color feature, (ii) Gaussian kernel bandwidth for the spatial feature, and (iii) number of iterations.

3. (50 points) Spatially Varying Blurring (to mimic the background-blur effect in video chats)

   Input images: `3/data/bird.jpg, 3/data/flower.jpg`

   If these images still lead to a computational cost that is beyond your computer's capabilities to allow for a sufficiently comprehensive experimentation, then you may downsample the images.

   Each input image has a visually distinct foreground and a background. In case of *flower.jpg*, the flower in focus at the center is the foreground and the rest is background. Similarly, in case of *bird.jpg*, the bird is the foreground and the rest is background. Your task is to keep the foreground intact and blur the background according to a spatially varying scheme as follows.

   • (10 points) Generate a mask image M with values 1 for the foreground and 0 for the background. Design a partially/fully automatic scheme to generate this mask. Fully manual solutions will not be considered at all. Explain your algorithm in detail, and motivate and justify it.

   Display the (i) the original image, (ii) the mask image M, (iii) the original image with the pixels values in the foreground (as determined by M) set to black, (iv) the original image with the pixels values in the background (as determined by M) set to black.

   • (20 points) Once the mask image M is generated, your task is to blur background using a spatially varying kernel. The kernel will be a circular disc function, with weights summing to 1. However, the disc radius in the kernel at every pixel location in the background will vary as a function of the pixel location as follows. Let P be any pixel location in the background, and let d_P be the shortest distance from P to the foreground region in the mask image M. Set the radius r of the disc kernel to be spatially dependent as follows: $r = d_P$ if $d_P \leq \alpha$, and $r = \alpha$ if $d_P > \alpha$, where $\alpha := 20$ for *flower.jpg* (at the original image size) and $\alpha := 40$ for *bird.jpg* (at the original image size).

   Write a function *mySpatiallyVaryingKernel.m* implementing this scheme. Note that you SHOULD NOT blur the foreground objects (i.e., the flower, the bird) at all.

   • (10 points) For the image of spatially-varying r values, display a contour plot or a jet-colormapped image to clearly demonstrating the variation of r with respect to the distance from the foreground region in the mask M.

   • (4 points) Display, as images, the blurring kernels (disc-shaped) computed at distances d_P as 0.2α, 0.4α, 0.6α, 0.8α, and α.

- (6 points) Display the output images, with a clearly-visible sharp foreground and blurred background.