PROBLEM SET 1

In this problem set we will explore some of the topics discussed in class. We will give you fairly detailed instructions about what to do, just to make sure that your Matlab® skills are current. Get in touch with us if you have problems programming. Do it early - we’ll help. You can work together, but try to write your answers independently.

A couple of useful hints
(1) use TrueColor images, no colormaps;
(2) use double(...) to convert the images (typically uint8) to double type;
(3) use uint8(...) to convert your processed images to uint8 for display;
(4) use colormap(gray) to display grayscale images;

1. WHAT’S DUE

Submit the images, 1-2 sentences for questions in each problem and the code for the problem one. If you can’t get something to work - submit what you have.

2. COLOR SPACES

This problem illustrates the differences between YUV and RGB color spaces. Use PS1Image1.jpg.

(1) Open a color .jpeg image
You will get a $Y \times X \times 3$ matrix, where each $Y \times X$ plane corresponds to R, G and B channels of the color image;

(2) Convert it to YUV
Basically, this is just a matrix multiplication if you convert the image to the right format. Write a function that takes a 3-D $Y \times X \times 3$ matrix and reshapes it to a 2-D one, which is $3 \times XY$. After multiplication you might want to reshape it back to the original form.

(3) Then for each channel:
(a) Blur it
You need to convolve the channel with a blurring filter. "Design" the filter with fspecial Matlab function. Filter the channel using conv2.
(b) Add some noise
   Use \texttt{randn} or \texttt{rand} to generate a $Y \times X$ noise image. Make sure the values are large enough to be noticeable. Add it to the values in the channel.
(c) Convert it back to RGB

(d) Display it
(4) Do the same for the original RGB image
What do you see?

3. Histograms

Use histogram back-projection to segment an image. Use \texttt{PS1Image2.jpg} and \texttt{PS1Model.jpg}.

(1) Load a color image
(2) Load the object color model
(3) Convert both to HSV color space
   You can use the \texttt{rgb2hsv} function
(4) Build a histogram of the hue channel of the model
   You can use something like \texttt{hist(H(:, :), ...}, where H is the first channel of your HSV image. Try different number of bins. Careful with the ranges.
(5) Back-Project the histogram into the image
   Create an empty grayscale image (\texttt{zeros(szy, szx)}). Iterate through the pixels in the image’s H channel. Find the bin in the histogram that corresponds to the pixel value. Put the value from the histogram bin into the grayscale image.
(6) Threshold the image
   Find a good value of the threshold that separates the object from the background reasonably well
(7) Segment the object
   If the result of your thresholding is a binary image with just zeros and ones, then you can just multiply the source with the result of thresholding pixel by pixel. You can use \texttt{Mask3 = repmat(Mask, [1 1 3])} to create a 3-channel mask from the single channel binary image and then just dot-multiply.

When would this approach fail?

4. Edges

Experiment with different edge detectors. You can use Matlab\textsuperscript{\textregistered} \texttt{edge} function. Use \texttt{PS1Image3.jpg}.

(1) Load a grayscale image
(2) Add some noise to it
(3) Run edge detectors
Use 'sobel', 'log' and 'canny'. You should start your experiments with parameters for each filter set to the values where the output is similar for all of them for the noiseless image.

(4) Display
Basically, show the table of images, each row corresponding to a different edge detector and each column - to the noise variance for 3-4 different values

How would you compare the quality of the detectors?