CSE 360/460-010
Introduction to Mobile Robotics

Laboratory Assignment 3: Real-time Color Segmentation

Lab Report Due Date: 3 April 2019 @ 5PM

A. Objectives: The objective of this lab is to implement two different versions of color segmentation. We will compare them in terms of both segmentation performance as well as computational requirements.

B. Lab Procedure:

1. Acquiring Images: You will be provided a skeleton Matlab function `lab3` which streams images to a Matlab figure, and returns the latest to the workspace on a keyboard hit. You will modify this file as necessary to complete this lab. However, the final file you submit MUST have the same name.

2. Making the Color Model: The first step in color segmentation is constructing the color model. While these can be adaptive, we will look at 2 fixed color models.

    a. Save several images of the target at different ranges to the workspace.
    b. From each image, you need to calculate 1st and 2nd order statistics (in other words, the mean μ and covariance matrix *C*) of pixel colors in the regions of interest. To do this:
        i. Use `roipoly` to create a mask for each image
        ii. For each channel (i.e., R-G-B), use this mask to isolate the appropriate pixels in that image channel.
        iii. For the pixels from each image, calculate μ and *C*. The Matlab `mean` and `cov` functions will be useful here.
        iv. While not completely correct, you can take the average values of your different means and covariance matrices and use this for construction your color models.
    c. You will make 2 color models as outlined in more detail in the lecture notes (see appended slides also):
        i. Model A will be based upon the "box method," taking ± *n* standard deviations (σ) from the mean of each color, and thresholding the images using these values as upper and lower bounds. Note you can infer σ from the diagonal terms of *C*.
        ii. Model B will use the squared Mahalanobis distance as normalized by the covariance matrix. You should use the squared distance (n^2) as the threshold value rather than taking the square root for all pixels.

3. Target Tracker: You will implement a simple tracker that assumes that the object of interest is the largest feature of that color in the image. To accomplish this, you will be provided a skeleton code file that you must fill in as follows:

    a. Perform the color segmentation IAW the thresholding approaches described above. This will leave you with a single channel binary (black and white) image.
    b. Obtain the bounding box of the largest connected component using the `regionprops` function. You should use the default connectivity (8).
    c. Plot a rectangle around this connected component in real-time to ensure that it is being properly tracked. You can do this by passing the bounding box to the `rectangle` function.

C. Testing:

1. Segmentation Performance – Maximum Range: To assess the tracking performance, we will see for each group at what range the target remains reliably visible (for nearly every frame). To test this:
    a. Run your color segmentation code using Model A. You should tune the value of *n* so that the target can be tracked while admitting a minimal number of noise pixels.
    b. Once you have decided upon your optimal value of *n*, move the target away from the camera until it can no longer be tracked. This establishes the limit for Model A. Do NOT move the computer or the target at this point.
    c. Start your code for Model B and tune the value of *n* similarly. Note that values of *n* for the two models may be different. Note what range you were able to reliably track the target.
    d. Repeat a-c for several different computer positions, and note the effective range of each of the approaches.

2. Segmentation Performance - Signal-to-Noise: We are also interested in the presence of noise in the images. Choose a range where the target is readily visible for both trackers. Take several images **with and without the target present.** For the images with the target present, determine the area of the largest connected component (`regionprops` will again be useful here). You can use the average of several images for this. For images without the target, calculate the total number of "valid" color pixels for each of the two models. You can do this using `sum( sum( im ) )`. As these are "false positives" they will give us insight into how well the different models suppress noise in the image. You should report the details for both models.
3. False Positives: For both models, look around the room with the camera for instances where your models pick up any other objects that have colors similar

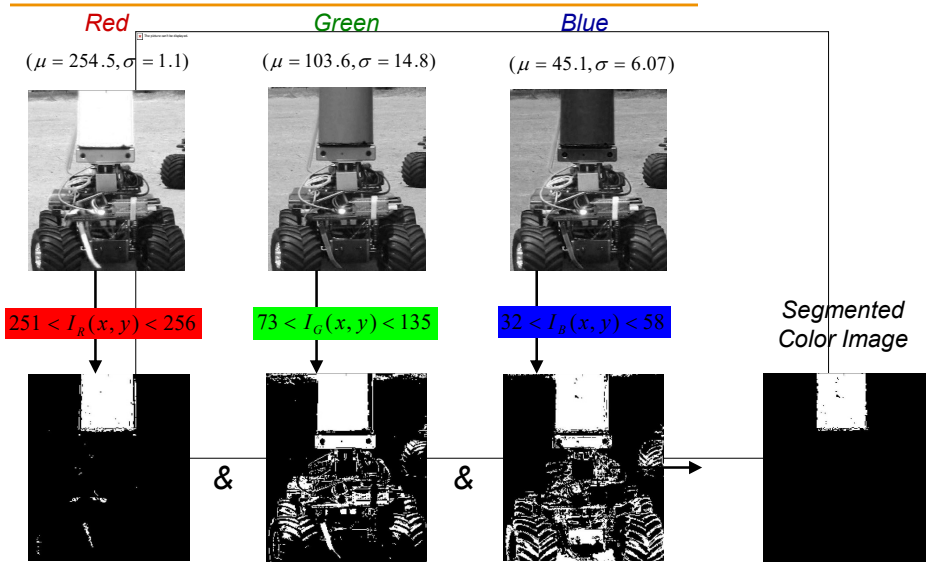
to the target. Save images for any of these occurrences and discuss the relative performance of the two models in your report.

4. Color Model Volume – At Home Exercise: For your tuned values of *n* for each model as obtained in C.1, determine the volume of the box / ellipsoid that you are using. This will also give us some insights into the potential for false positives if these are not observed during the lab session. If *n* was the same for both models, what would the ratio of the two volumes be?

5. CPU Performance: To assess the CPU performance, run a loop where you acquire and segment at least 100 images. Use the built in `tic/toc` functions to estimate the average time for running each of the different models. Note, you should time only the color segmentation aspects of your code (not image acquisition, displaying the images, etc.).

D. Turn In: You are to turn in your code, along with **a short report** outlining the results of the experiment outlined in Section C above. This should include an assessment of the strength and weaknesses of each model, and whether the computational costs of Model B are warranted by any observed improvements in performance. This is to be submitted via Course Site a single .zip file. Only 1 report submission is required per group. Please name your submission `lastnamestudent1_lastnamestudent2.zip`.

Simple RGB Color Segmentation



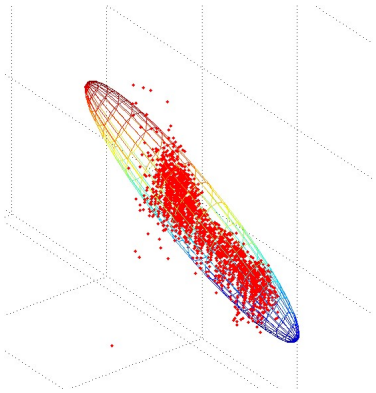
Color Segmentation ++

$C =$

$$\begin{matrix} 3.5207 & 4.7485 & 1.6528 \\ 4.7485 & 205.5231 & 69.6209 \\ 1.6528 & 69.6209 & 36.4481 \end{matrix}$$

$mu\_color =$

$$\begin{matrix} 253.9224 & 103.9645 & 45.3376 \end{matrix}$$



- **So, our color segmentation reduces to**

$$im_{bin}(i,j) = \left[pix(i,j)^T C^{-1} pix(i,j) \le 16 \right]$$

where

$$pix(i,j) = im_{color}(i,j) - \mu_{color}$$