Constantin Gemmingen, Luke Peppard, Fabian Neumüller - Group 4.3

Prelab 05

Prelab Q1 - Color Tracking with Matlab

Print and upload your matlab code along with the figures

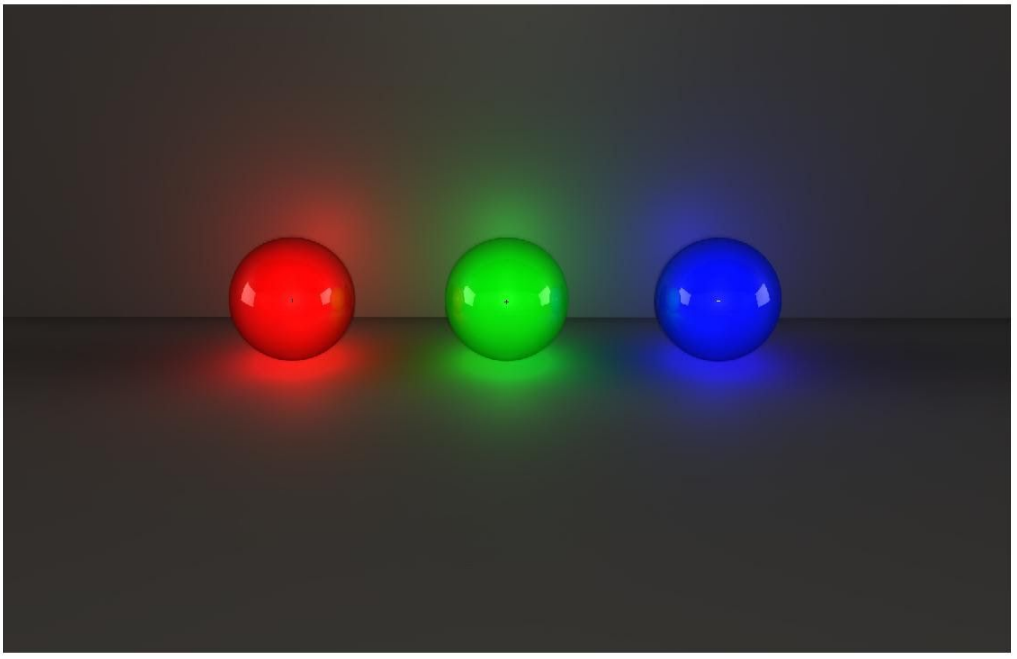
```
% tabula rasa:
% clc;
% clear all;
% close all;

% This is to solve Prelab Q1:
% Write a MATLAB script that reads the image, plots the three hsv channels
% seperately, uses threshhold on hue and saturation to find the red, green
and blue shapes independently and determines the centroid of the object

% Read the image given to you (use function imread())
image_original = imread('image1.jpg');

% plot the original image (use imshow())
imshow(image_original)
% convert your image into hsv color space (use function rgb2hsv())
image_hsv = rgb2hsv(image_original);
figure()
imshow(image_hsv)

%% plot the grayscale images of hue, saturation and value of your image
seperately (use imshow() again)

% image_grey = rgb2gray(image_original);

image_hue = image_hsv(:, :, 1);
figure()
imshow(image_hue)
title('greyscale of hue')

image_saturation = image_hsv(:, :, 2);
figure()
imshow(image_saturation)
title('greyscale of saturation')

image_value = image_hsv(:, :, 3);
figure()
imshow(image_value)
title('greyscale of value')
```

```
%% use the hue image you just plotted to find the hue lower and upper
bounds for each color

% const
const_lower_red = 0.8334;
const_upper_red = 0.1666;
const_lower_green = 0.1667;
const_upper_green = 0.5;
const_lower_blue = 0.5001;
const_upper_blue = 0.8333;

% set all upper_bounds to lowest possible and vice versa
upper_bound_red = 0;
lower_bound_red = 1;
upper_bound_green = const_lower_green;
lower_bound_green = const_upper_green;
upper_bound_blue = const_lower_blue;
lower_bound_blue = const_upper_blue;



for j = 1:1080
    for i = 1:1680
        temp_hue = image_hue(j, i);
        % upper bound
        if temp_hue < const_upper_red
            if temp_hue > upper_bound_red
                upper_bound_red = temp_hue;
            end
        end
        if temp_hue < const_upper_green && temp_hue > const_lower_green
             if temp_hue > upper_bound_green
                upper_bound_green = temp_hue;
            end
        end
        if temp_hue < const_upper_blue && temp_hue > const_lower_blue
             if temp_hue > upper_bound_blue
                upper_bound_blue = temp_hue;
            end
        end

        % lower bound
        if temp_hue > const_lower_red
            if temp_hue < lower_bound_red
                lower_bound_red = temp_hue;
            end
```

```
        end
        if temp_hue < const_upper_green && temp_hue > const_lower_green
             if temp_hue < lower_bound_green
                lower_bound_green = temp_hue;
            end
        end
        if temp_hue < const_upper_blue && temp_hue > const_lower_blue
             if temp_hue < lower_bound_blue
                lower_bound_blue = temp_hue;
            end
        end

    end
end
clear i j temp


%% use the saturation image you just plotted and find one single lower and
upper bound for all your colors
lower_bound_sat = min(min(image_saturation));
upper_bound_sat = max(max(image_saturation));


%% use these thresholds to create a mask for each color, plot your three
masks separately (for each
% color you should have a black-white image showing only the blob of that
color)
red_mask([1:1080], [1:1680]) = zeros;
green_mask([1:1080], [1:1680]) = zeros;
blue_mask([1:1080], [1:1680]) = zeros;

sat_factor = 0.92;

for j = 1:1080
    for i = 1:1680
        temp_hue = image_hue(j, i);
        temp_sat = image_saturation(j, i);
        if (temp_hue < const_upper_red || image_hue(j, i) >
            const_lower_red) && temp_sat > sat_factor * upper_bound_sat
            red_mask(j, i) = 1;
        end
        if temp_hue < const_upper_green && temp_hue > const_lower_green &&
                                temp_sat > sat_factor * upper_bound_sat
            green_mask(j, i) = 1;
        end
        if temp_hue < const_upper_blue && temp_hue > const_lower_blue &&
```

```
                                temp_sat > sat_factor * upper_bound_sat
            blue_mask(j, i) = 1;
        end
    end
end


figure()
imshow(red_mask)
title('red mask')
figure()
imshow(green_mask)
title('green mask')
figure()
imshow(blue_mask)
title('blue mask')

clear j i temp sat_factor

%% rgb mask
rgb_mask(:, :, 1) = red_mask;
rgb_mask(:, :, 2) = green_mask;
rgb_mask(:, :, 3) = blue_mask;
figure()
imshow(rgb_mask)
title('rgb mask')


%% find the centroid of the three colors using their respective masks (
use function regionprops();  be aware that it can return more than one
centroid)

centre_red = regionprops(red_mask, 'centroid');
xy_centroid_red = cat(1, centre_red.Centroid);

centre_green = regionprops(green_mask, 'centroid');
xy_centroid_green = cat(1, centre_green.Centroid);

centre_blue = regionprops(blue_mask, 'centroid');
xy_centroid_blue = cat(1, centre_blue.Centroid);

%% plot the original image with the center of the centroid (use function
insertMarker())

marker_position = [xy_centroid_red; xy_centroid_green; xy_centroid_blue];
```

```
marker_colour = {'black', 'black', 'white'};
image_original_mk = insertMarker(image_original, marker_position, 'color',
marker_colour);

figure()
imshow(image_original_mk)
```



Prelab Q2 - Pixy2 Camera

What is the primary advantage of using a Pixy2 camera for your robot, as compared to the majority of other cameras compatible with Arduino boards?

You can teach it to detect only the objects you want it to detect. In addition, the Pixy2 has new algorithms that detect and track lines for use with line-following robots.

Prelab Q3 - Pixy2 Camera

You have set up the Pixy2 camera to detect green coloured balls in its field of view (see 1, 2). The camera detects these green balls and outputs the position of the balls.
Connecting Pixy2 to the microcontroller:

1. **How can you connect the Pixy2 camera to the microcontroller on the UDOO to read out the position information?**
   Using the ISP interface you need to connect the pins 1, 3, 4, 7 on the Pixy to the pins 50, 51, 52, 31 on the UDOO (GPIO pins 55, 56, 57, 137).
2. **What communication protocol can be used to talk between the camera and microcontroller?**
   A serial protocol can be used for communicating between the camera and the microcontroller. ISP is typically a faster interface.
3. **Draw a schematic of the connections and corresponding pins used on both devices (hint)**