

Introduction to Robotics

Lab 2a

Shaaf Farooque 08405, Mysha Zulfiqar 08443

Task 2.7 Find colored objects (20 points)

Develop a color segmentation stage for determining the pixels corresponding to all the cubes in the workspace of a single color. You are required to submit

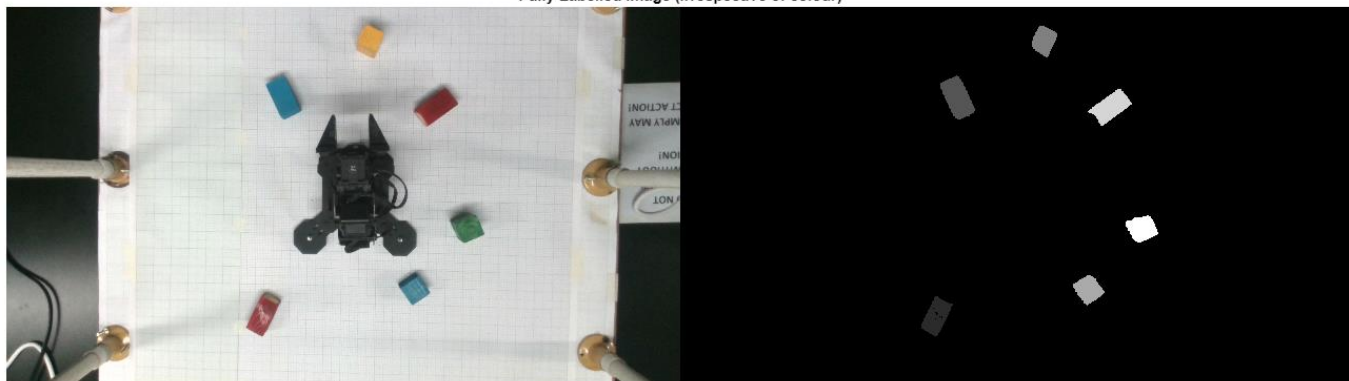
- (i) a note describing your choice of color segmentation strategy and the rationale behind it;
- (ii) aptly commented code for your algorithm;
- (iii) before and after images for two colors;
- (iv) comments on the success of the algorithm; Are areas of the object missed (zoom in)? Does the location of the object have an effect? Does ambient lighting have an effect?

- (i) We have used different color spaces for different colors, depending on which color space gave the best output. Some colors were best captured in LAB, some in HSV, and so on. After a lot of trial and error we have color segmentation functions that are very accurate.

```
function masks = getMasks(RGB)
    %This function uses colour segmentation functions created using
    %the colour thresholder app. The thresholds are designed
    %according to the lighting conditions that we have in the lab.
    %Different colour spaces were used to isolate each colour.
    %This function returns an array of masks which can later be
    %extracted used as necessary.
    %All the createMask functions were also updated by us to include the line
    %BW = bwpropfilt(BW, 'Area', [4000 15000]);
    %before returning the BW mask. This ensures only
    %objects that had an area between 4000 and 15000 were recognised.
    %This algorithm works very well in our setting.
    redMask = createRedMask(RGB); %function created using color thresholder app
    greenMask = createGreenMask(RGB); %function created using color thresholder app
    yellowMask = createYellowMask(RGB); %function created using color thresholder app
    blueMask = createBlueMask(RGB); %function created using color thresholder app
    combinedMasks = redMask | greenMask | yellowMask | blueMask; % Logical OR to combine all masks
    masks = {redMask, greenMask, yellowMask, blueMask, combinedMasks}; % returning all masks for ease
end
```

(ii)

Fully Labelled image (irrespective of colour)



(iii)

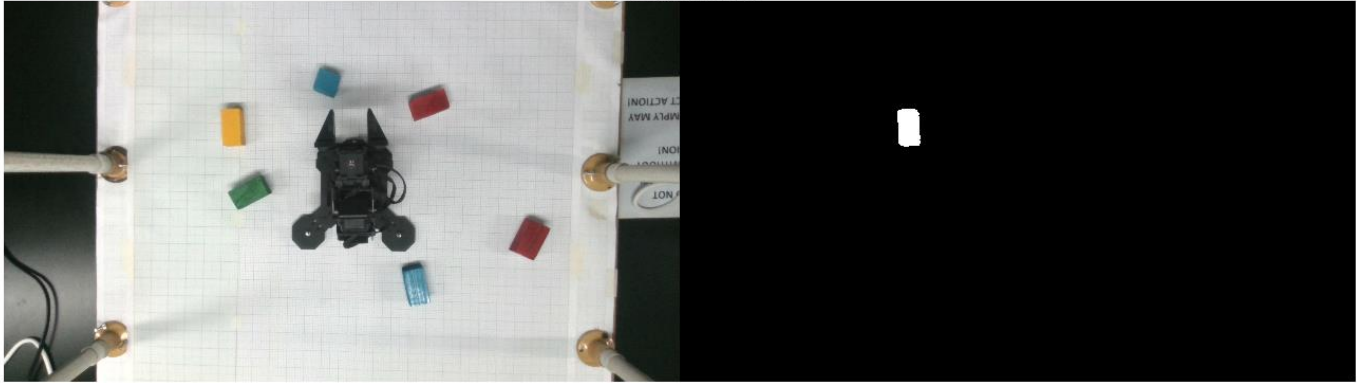
Red Labelled Image



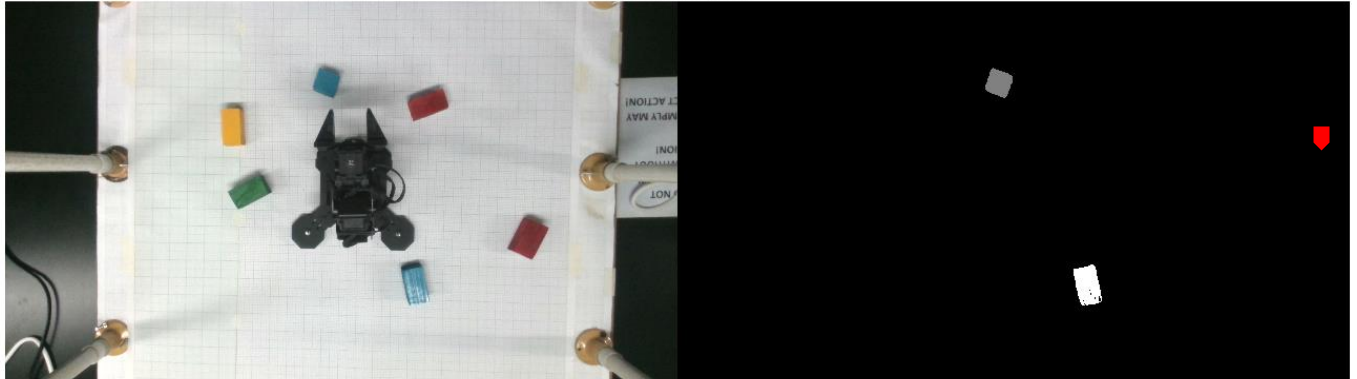
Green Labelled Image




Yellow Labelled Image



Blue Labelled Image



(iv) The algorithm is almost perfect, there are some rough edges still and sometime due to either reflection or poor lighting, a small amount of the cube is not highlighted. 

Task 2.9 Labeling cubes (10 points)

Develop a labeling stage that clusters the pixels corresponding to a cube. You are required to submit

- (i) aptly commented code for your algorithm;
- (ii) labeled image of the run;
- (iii) comments on the success of the algorithm.

(i)

```

function countAndShowCubes(RGB)
    %This function takes an image, generates masks for each colour,
    %converts them to labels and displays the image of the overlaid
    %label and prints the number of each coloured cube.
    masks = getMasks(RGB); % Function from the previous task to get masks
    fullyLabelledImage = bwlabel(masks{5});
    figure; %showing before and after image labelling. First irrespective of colour.
    imshowpair(RGB, fullyLabelledImage, 'montage')
    title('Fully Labelled image (irrespective of colour)');

    redLabels = bwlabel(masks{1});
    figure; %showing before and after image labelling.
    imshowpair(RGB, redLabels, 'montage');
    title("Red Labelled Image");
    redCubes = countCubes(redLabels);

    greenLabels = bwlabel(masks{2});
    figure; %showing before and after image labelling.
    imshowpair(RGB, greenLabels, 'montage');
    title("Green Labelled Image");
    greenCubes = countCubes(greenLabels);

    yellowLabels = bwlabel(masks{3});
    figure; %showing before and after image labelling.
    imshowpair(RGB, yellowLabels, 'montage');
    title("Yellow Labelled Image");
    yellowCubes = countCubes(yellowLabels);

    blueLabels = bwlabel(masks{4});
    figure; %showing before and after image labelling.
    imshowpair(RGB, blueLabels, 'montage');
    title("Blue Labelled Image");
    blueCubes = countCubes(blueLabels);

    fprintf('Red cubes: %d\n', redCubes);
    fprintf('Green cubes: %d\n', greenCubes);
    fprintf('Yellow cubes: %d\n', yellowCubes);
    fprintf('Blue cubes: %d\n', blueCubes);
end

```

(ii) Refer to outputs of previous task for labelled image. The outputs I have shown are the combination of this task and the previous.

```

Red cubes: 2
Green cubes: 1
Yellow cubes: 1
Blue cubes: 2

```

Task 2.11**Improved object detection pipeline (20 points)**

Based on the observed problems and knowledge acquired from the previous tasks, develop an improved pipeline from a raw RGB image to an image with labeled cubes of one color. You are required to submit

- (i) a note identifying each problem, your solution for it in the new pipeline, and your rationale for why it works;
- (ii) aptly commented code for your algorithm;
- (iii) raw image, labeled image from the previous pipeline, labeled image of the new run.

(i) Problem: Rough Edges in Detected Masks

- a. **Cause:** Noise in the raw image or imperfect segmentation due to variations in lighting or reflections.
- b. **Solution:** Use morphological operations such as `imopen()` to remove small artifacts and improve mask quality. Also, used `activecontour()` to refine mask boundaries by iterating over initial masks to capture accurate edges.
- c. **Rationale:** Morphological operations smoothen object boundaries and remove small, irrelevant noise. `activecontour()` improves edge refinement by iteratively adjusting the mask to fit object boundaries.

Problem: Incomplete Highlighting of Cubes

- d. **Cause:** Poor lighting or reflections causing some regions of the cubes to be below threshold levels for segmentation.
- e. **Solution:** Apply `imfill()` to fill gaps in detected regions and ensure a continuous mask for each cube. Additionally, `bwpropfilt()` filters out objects outside a specified size range to avoid irrelevant detections.
- f. **Rationale:** Filling gaps ensures that partially segmented regions are completed. Filtering by size removes noise and ensures only cube-sized regions are detected.

Problem: Inconsistent Color Segmentation

- g. **Cause:** Variations in color space thresholds due to changing lighting conditions.
- h. **Solution:** Tune the `createColorMask` functions for each color using color thresholding in different color spaces (e.g., HSV, LAB) and include filtering based on object properties.
- i. **Rationale:** Using multiple color spaces accounts for variability in lighting, while filtering ensures that only correctly sized and shaped objects are segmented.

(ii).



```
function masks = getMasks( RGB )  
%This function uses colour segmentation functions created using  
%the colour thresholder app. The thresholds are designed  
%according to the lighting conditions that we have in the lab.
```

```

%Different colour spaces were used to isolate each colour.
%This function returns an array of masks which can later be
%extracted used as necessary.
%All the createMask functions were also updated by us to include the line
%BW = bwpropfilt(BW, 'Area', [4000 15000]);
%before returning the BW mask. This ensures only
%objects that had an area between 4000 and 15000 were recognised.
%This algorithm works very well in our setting.
redMask = createRedMask(RGB); %function created using color thresholder app
greenMask = createGreenMask(RGB); %function created using color thresholder app
yellowMask = createYellowMask(RGB); %function created using color thresholder
app
blueMask = createBlueMask(RGB); %function created using color thresholder app
% Helper function to refine individual masks
% Step 1: Fill holes in the mask
% Step 2: Refine edges using active contour method
% Optional: Add morphological operations if needed (e.g., opening)

redMask = imfill(redMask,"holes");
redMask = activecontour(RGB, redMask,5,"Chan-vese", 'SmoothFactor',1.5);

greenMask = imfill(greenMask,"holes");
greenMask = activecontour(RGB, greenMask,5,"Chan-vese", 'SmoothFactor',1.5);

yellowMask = imfill(yellowMask,"holes");
yellowMask = activecontour(RGB, yellowMask,5,"Chan-vese", 'SmoothFactor',1.5);

blueMask = imfill(blueMask,"holes");
blueMask = activecontour(RGB, blueMask,10,"Chan-vese", 'SmoothFactor',1.5);

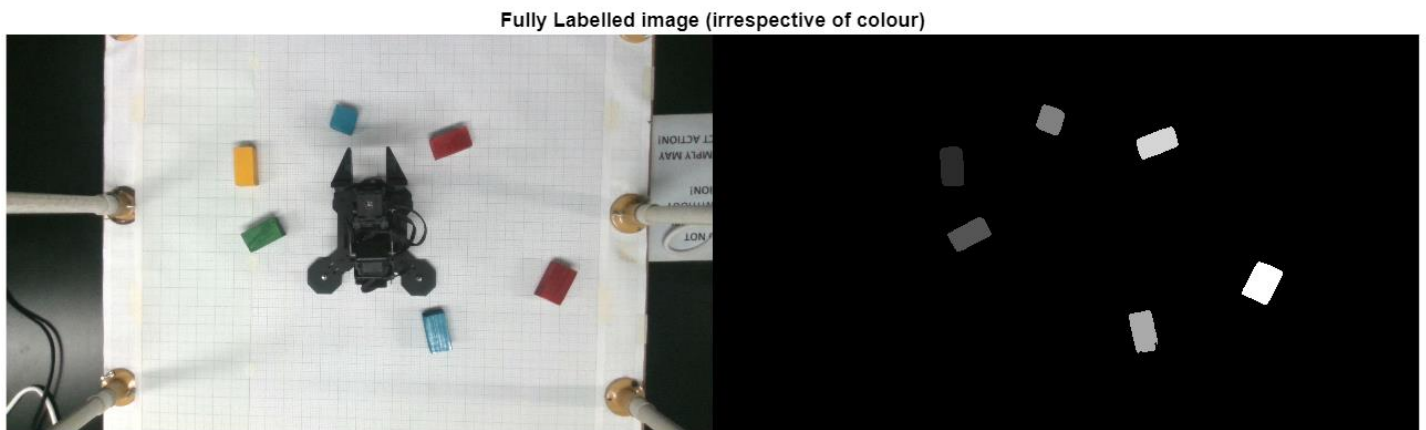
combinedMasks = redMask | greenMask | yellowMask | blueMask; % Logical OR to
combine all masks
masks = {redMask, greenMask, yellowMask, blueMask, combinedMasks}; % returning
all masks for ease
end

```


(iii). The raw image and image from previous pipeline without segmentation improvement:



The raw image and image from the new pipeline:



The new pipeline has significantly improved the rough edges of the cubes and is defined the area of the cube prominently. The new pipeline is ensuring that no area of the cube is left uncovered and that the next steps in the process i.e. finding the pose is much more accurate.

