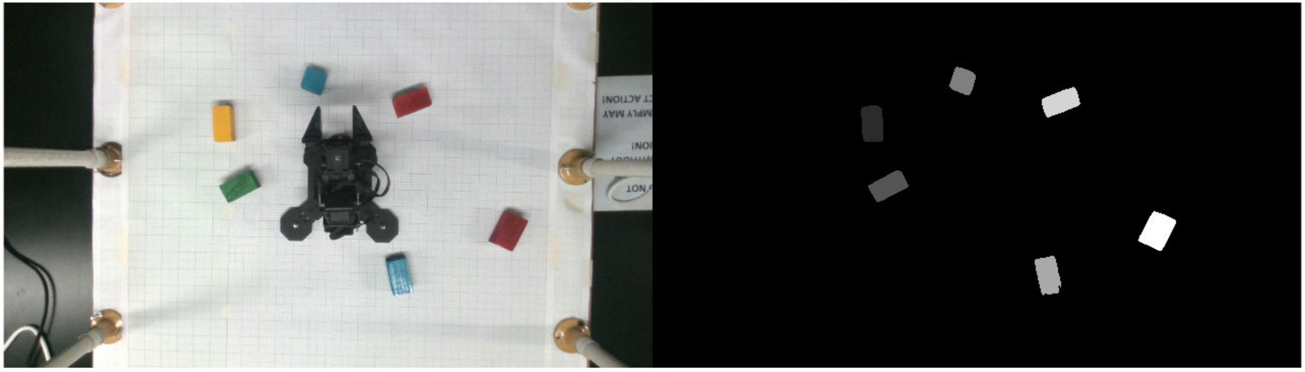
Introduction to Robotics

Lab 2a

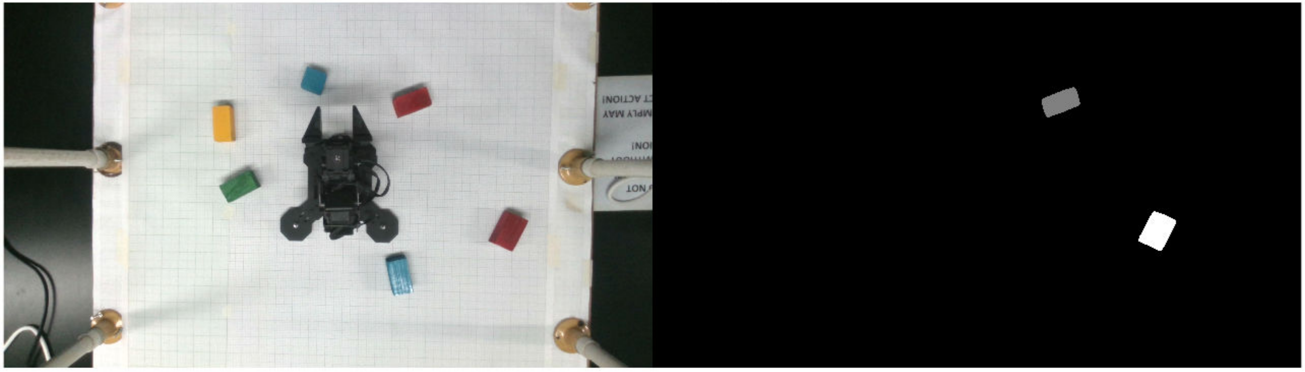
Shaaf Farooque 08405, Mysha Zulfiqar 08443

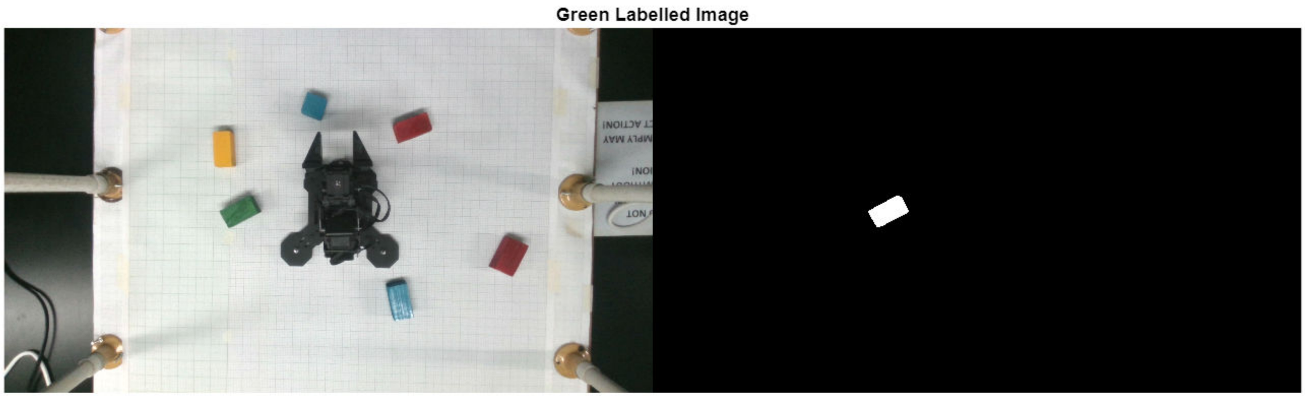
```
clc; clear; close all;
cubes = imread("manycubes.png");
cubes2 = imread("manycubes2.png");
%imfill, bwareaopen, strel, imclearborder activecontour(bison,
%bisonBW,'Chan-vese','ContractionBias',-0.2);

countAndShowCubes(cubes);
```

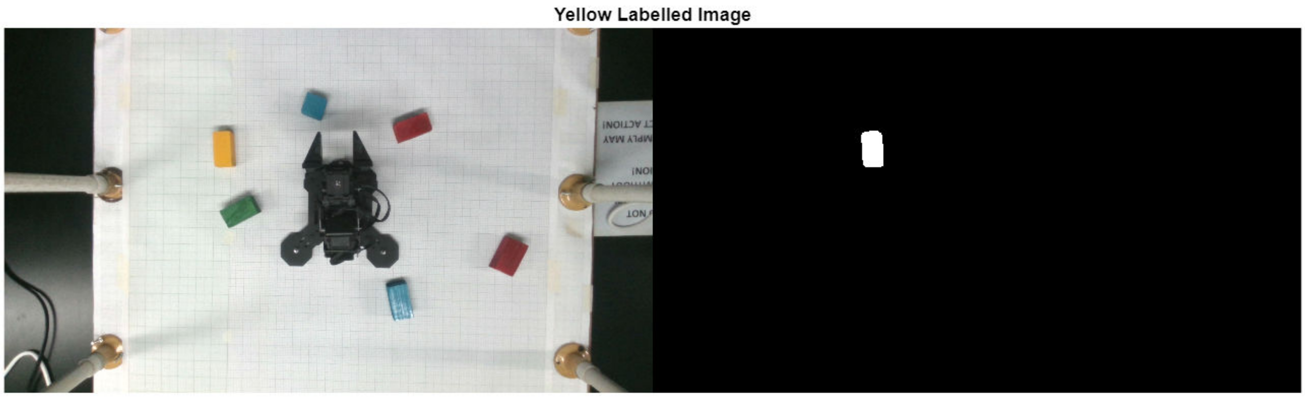
Fully Labelled image (irrespective of colour)

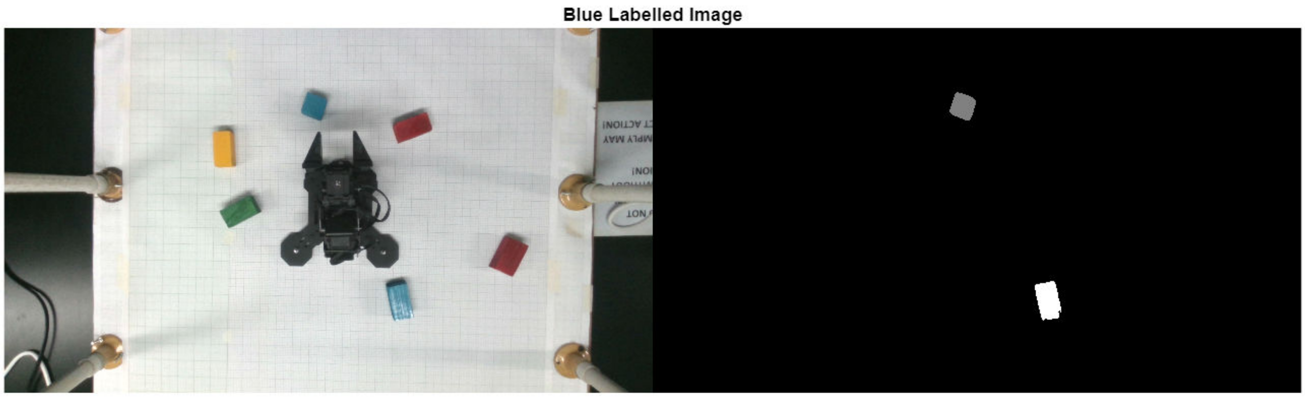


Red Labelled Image









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

```
% countAndShowCubes(cubes2);
```

```
function countAndShowCubes(RGB)
    %This function takes an image, generates masks for each colour,
    %converts them to labels and displays the image of the overlayed
    %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

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}; % returing
all masks for ease
end

function count = countCubes(colMask)
    %Takes a BW mask as an input, uses bwconncomp to extract
    %information and returns the number of objects. Cubes in this case.
    CC = bwconncomp(colMask);
    count = CC.NumObjects;
end
```