Shade Alabsa

CS 7455

HW 01

Mignon Kang

GH link: https://github.com/shade34321/CS\_7455

1. The first part was quite easy to solve once I saw the hint in the homework assignment about C = ~A. Ultimately what I did was use bwlable to define each connected component. From there I followed the code in the slide from the elementary operations by creating a component image. Then I obtained the largest component id and returned that image. In my initial work I had to take the inverse of the image coming in and inverse the image I was returning which baffled me. I couldn’t understand why taking out the inverses didn’t work but in testing today I found the inverse weren’t necessary. It’s possible something I tried was causing a problem before and the effects lingered. Below are the pictures and code I used. 

\

% remove\_holes Removes holes from a black and white image rendering objects

% in the foreground one color with the background a different

% color

% test\_zebra = remove\_holes(zebra)

% Where zebra is a binary image and the test\_zebra is the image with the holes

% removed.

%

% Author Shade Alabsa

% Version 1.0

% Date 17 February 2017

function holeless\_image = remove\_holes(input\_image)

% Use the inverse of the image to get a lable matrix and number of

% connected objects.

[labels, number] = bwlabel(input\_image, 4);

% Allows us to store how many pixels per component.

counters = zeros(1,number);

for i = 1:number

% for each i, we count the number of pixels equal to i in the labels

% matrix

% first, we create a component image, that is 1 for pixels belonging to

% the i-th connected component, and 0 everywhere else.

component\_image = (labels == i);

% now, we count the non-zero pixels in the component image.

counters(i) = sum(component\_image(:));

end

% find the id of the largest component

% We don't care about the area so we use ~ to ignore it

[~, id] = max(counters);

holeless\_image = (labels == id);

end

1. So this problem caused me quite a few problems. Only the players portion, the field was quite easy. For the field I filtered out each color into its own matrix of R, G, B. Removed some noise with pulling out the green and using imdilate. Then used bwlabel to get the components and remove\_holes to fill it in.

For the players my initial thought on how to solve this problem was to take the colors of the picture like you showed for the tennis courts. From there combine the colors to only show the specific colors, i.e. only show green for the field, for the red players show red but subtract out the field, etc. All of my initial tests didn’t quite work and actually my best images were just filtering based on color but it was quite noisy with equations like the one listed below.

red = ((r - g > 10) & (r - b > 10));

green = ((g - r > 10) & (g - b > 0));

blue = ((b - r > 10) & (b - g > 10));

I couldn’t them cleaned up very well, and showing the results of (red & green), (blue & green), etc didn’t seem to help much. Unfortunately, I didn’t keep the exact code I used to paste here but if needed I could probably recreated it. At this point I was left with images that were close but not exact so I started playing the formulas to identify each area and it helped but not much. From there I started looking for ways to combine the red, blue, and green in some fashion where it would add the colors I didn’t need and somehow combine it with the color I needed in a way which left the players alone, so some form of (~red | ( green & blue)). This didn’t get me very far. I tried to use imdilate to help reduce the noise a bit but it seemed to return a blocky result and thus it didn’t look very good. Finally, I remembered another class mate saying something about imclose and imerode so I looked those up, one was in the lecture notes. Ultimately it looked like a good way to filter out some noise from the picture. Below are the pictures I came up with and the code.

% The next two lines were for me in testing. Just put the filename into the

% variable filename and it should work.

%addpath E:\Dropbox\School\SPSU\_KSU\CS\_7455\CS\_7455\hw01;

%cd E:\Dropbox\School\SPSU\_KSU\CS\_7455\CS\_7455\hw01;

filename = 'E:\Dropbox\School\SPSU\_KSU\CS\_7455\CS\_7455\hw01\data\soccer\_field4.jpg';

soccer = imread(filename); %read in the file.

% Extract all three color bands into their own respective spots.

color = double(soccer);

r = color(:,:, 1);

g = color(:,:, 2);

b = color(:,:, 3);

% Identify all the green areas. Playing with this gives various results.

green = ((g - r > 50) & (g - b > 50));

green = imdilate(green, ones(7,7)); %dilate the image a bit. Based off code from the slides.

[labels, ~] = bwlabel(green, 4);

field = (labels == 2);

field = remove\_holes(field);

% Here we filter out some more noise and then identify all the red and blue

% areas

green = imclose(green, ones(7,7));

green = imerode(green, ones(7,7));

red = ((r - g > 100) & (r - b > 100));

blue = ((b - r > 50) & (b - g > 50));

% Since we dilated the green the green is also included with the red and

% blue section.

% I didn't dilate the red and blue since it seemed to make blocks and

% return an ugly result.

red\_players = (red & green);

blue\_players = (blue & green);

% Display the field

figure(1);

imshow(field);

% Display the red players

figure(2);

imshow(red\_players);

% Display the blue players

figure(3);

imshow(blue\_players);

