Barış Can 28/10/2019

21501886

HW1

1-)

Dilation:

Binary_image= dilation(source_image, struct_el) dilates the grayscale, binary, or packed binary image source_image, returning the dilated image, binary_image. struct_el is a structuring element object or array of structuring element objects.

I first created three arrays, one for struct elements, one for changing the imdata, struct, and the other for creating a dilated image, new_imdata. Then, I iterated all the rows and columns, and if the corresponding pixel is 1, I applied dilation. Below, I defined several variables such as a is the first-row element of the struct and b is the first column element of the struct. Also, I defined four if statement for the situation that 1x1 of imdata is also 1, and then I cut half of the struct. I finally obtained a struct element that is the cut version of imdata according to struct_el, then I checked if there is at least one 1 element in the array, if yes I made all the struct 1, else, if all the elements were 0 then I did not change the struct and new_imdata.

```
function binary_image = dilation ( source_image , struct_el )
    imdata = source_image;
    [row, column] = size(imdata);
    new_imdata = zeros(row, column);
    new_imdata = imdata;
    struct = zeros(struct_el, struct_el);
    new_struct = ones(struct_el);

    for k = 1:row
        for t = 1:column
            if (imdata(k, t) == 1)

                if (ceil(struct_el/2)-1) >= k
                    a = 1;
                elseif (ceil(struct_el/2)-1) < k
                    a = k - (ceil(struct_el/2)-1);
                end

                if (ceil(struct_el/2)-1) >= t
                    b = 1;
                elseif (ceil(struct_el/2)-1) < t
                    b = t - (ceil(struct_el/2)-1);
                end

                if (a+struct_el-1 > row && b+struct_el-1 > column)
                    struct(1:row-a+1, 1:column-b+1) = imdata(a:row,
b:column);
                elseif (a+struct_el-1 < row && b+struct_el-1 > column)
                    struct(1:struct_el, 1:column-b+1) =
imdata(a:a+struct_el-1, b:column);
                elseif (a+struct_el-1 > row && b+struct_el-1 < column)
```

```
                    struct(1:row-a+1, 1:struct_el) = imdata(a:row,
b:b+struct_el-1);
                elseif (a+struct_el-1 <= row && b+struct_el-1 <= column)
                    struct(1:struct_el, 1:struct_el) =
imdata(a:a+struct_el-1, b:b+struct_el-1);
                end

                [row1, column1] = size(struct);
                flag=0;
                for k1 = 1:row1
                    for t1 = 1:column1
                        if (struct(k1, t1) == 1)
                            flag = 1;
                        end
                    end
                end

                if flag == 1 && (k > ceil(struct_el/2) -1 ) && (k  < row -
ceil(struct_el/2)-1) && (t  > ceil(struct_el/2)-1) && (t  < column -
ceil(struct_el/2)-1)
                    new_imdata(a:a+struct_el-1, b:b+struct_el-1) =
new_struct;
                end
            end
        end
    end

    figure; imshow([source_image, new_imdata]);
    xlabel('Dilation Applied')
    imwrite(new_imdata,'hop.png');
end
```

Erosion:

Binary_image = erosion(source_image, struct_el) erodes the grayscale, binary, or packed binary image source_image, returning the eroded image, binary_image. Struct_el is a structuring element object or array of structuring element objects.

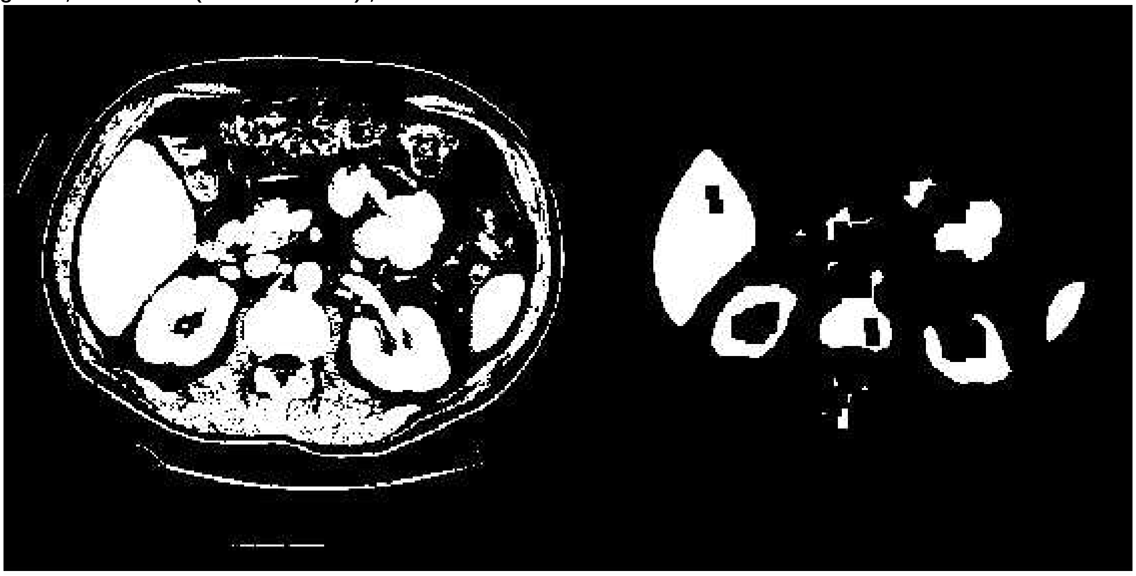
This implementation is very similar to dilation. I first created three arrays, one for struct elements, one for changing the imdata, struct, and the other for creating a dilated image, new_imdata. Then, I iterated all the rows and columns, and if the corresponding pixel is 1, I applied erosion. Below, I defined several variables such as a is the first-row element of the struct and b is the first column element of the struct. Also, I defined four if statement for the situation that 1x1 of imdata is also 1, and then I cut half of the struct. I finally obtained a struct element that is the cut version of imdata according to struct_el, then I checked if there is at least one 0 element in the array, if yes I made all the struct 0, else, if all the elements were 1 then I did not change the struct and new_imdata.

```
function binary_image = erosion ( source_image , struct_el )
    imdata = BW;
    [row, column] = size(imdata);
    new_imdata = zeros(row, column);
    new_imdata = imdata;
    struct = zeros(struct_el, struct_el);
    new_struct = zeros(struct_el, struct_el);

    for k = 1:row
        for t = 1:column
            if (imdata(k, t) == 1)
                if (ceil(struct_el/2)-1) >= k
                    a = 1;
                elseif (ceil(struct_el/2)-1) < k
                    a = k - (ceil(struct_el/2)-1);
                end

                if (ceil(struct_el/2)-1) >= t
                    b = 1;
                elseif (ceil(struct_el/2)-1) < t
                    b = t - (ceil(struct_el/2)-1);
                end

                if (a+struct_el-1 > row && b+struct_el-1 > column)
                    struct(1:row-a+1, 1:column-b+1) = imdata(a:row,
b:column);
                elseif (a+struct_el-1 < row && b+struct_el-1 > column)
                    struct(1:struct_el, 1:column-b+1) =
imdata(a:a+struct_el-1, b:column);
                elseif (a+struct_el-1 > row && b+struct_el-1 < column)
                    struct(1:row-a+1, 1:struct_el) = imdata(a:row,
b:b+struct_el-1);
                elseif (a+struct_el-1 <= row && b+struct_el-1 <= column)
                    struct(1:struct_el, 1:struct_el) =
imdata(a:a+struct_el-1, b:b+struct_el-1);
                end

                [row1, column1] = size(struct);
                flag=0;
                for k1 = 1:row1
                    for t1 = 1:column1
                        if (struct(k1, t1) == 0)
                            flag = 1;
                        end
                    end
                end

                if flag == 1 && (k > ceil(struct_el/2) -1 ) && (k  < row -
ceil(struct_el/2)-1) && (t  > ceil(struct_el/2)-1) && (t  < column -
ceil(struct_el/2)-1)
                    new_imdata(a:a+struct_el-1, b:b+struct_el-1) =
new_struct;
                end
            end
        end
    end
    figure; imshow([source_image, new_imdata]);
    xlabel(‘Erosion Applied’)
    imwrite(new_imdata,'hop.png');
end
```

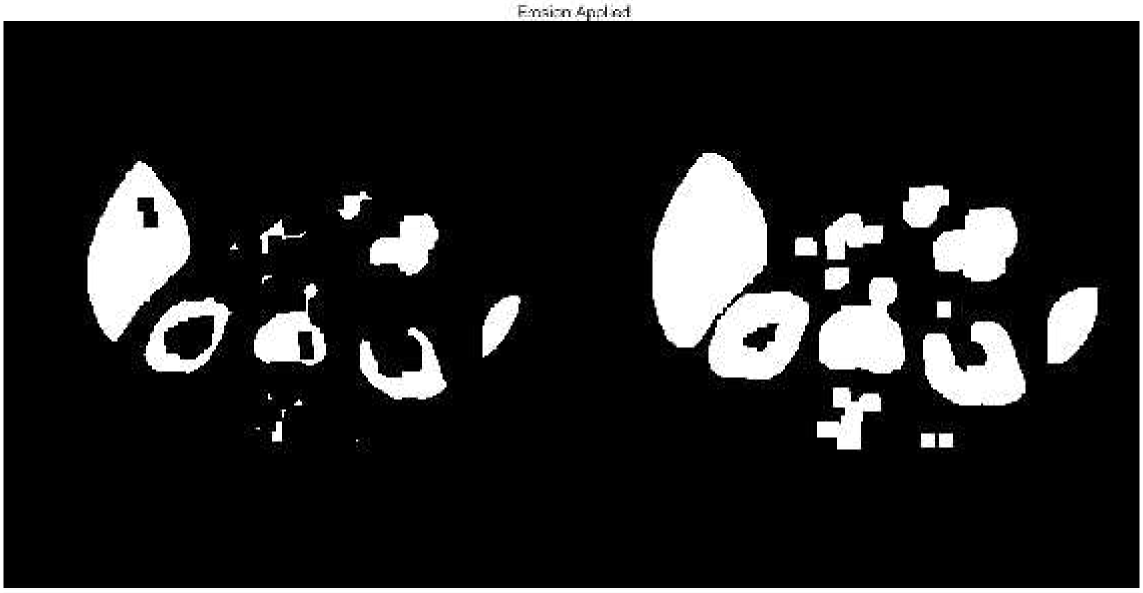
2-)

I first read the CT image. Then after some trials, I chose my threshold as 128 since 0.5 ratio is the default value for binary images and I could not find the better threshold. Then I applied erosion to increasing the black (0) areas to reduce the noise, I got rid of the noise, I increased white areas (1) by applying dilation. After many trials, I chose 7 as my struct element for erosion and 13 for dilation. Then I applied 8 connected component labelling and coloured the image. The results are at below.

```
hop = imread( 'ct.png');
BW = hop > 128;
erosion(BW, 7);
hop = imread( 'hop.png');
dilation(hop, 13);
hop = imread( 'hop.png');
cc = bwconncomp(hop, 8);
labeled = labelmatrix(cc);
RGB_label = label2rgb(labeled);
figure; imshow(RGB_label);
```



Erosion Applied



Dilation Applied

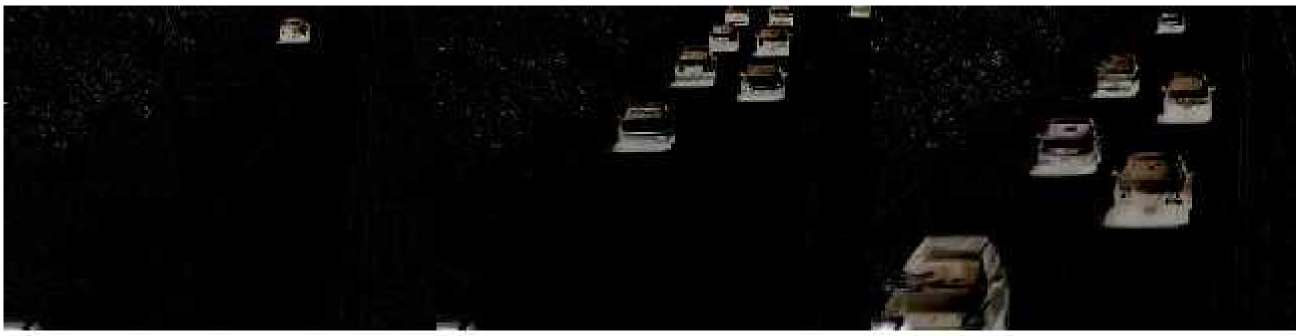
I think the most difficult part was to choose the morphological operators' struct elements. I had not many choices while selecting the threshold because after one pixel go down, there was a lot of noise and after one pixel going up, there was a lot of detail loss. Therefore, I stuck with the default one. But, selecting a struct element was difficult because not only you can increase the struct element in dilation, for example, but also you can decrease the dilation and decrease the erosion as well. The results were similar if you keep the dilation/erosion ratio, but there were minor changes so I tried to choose best of the combinations.

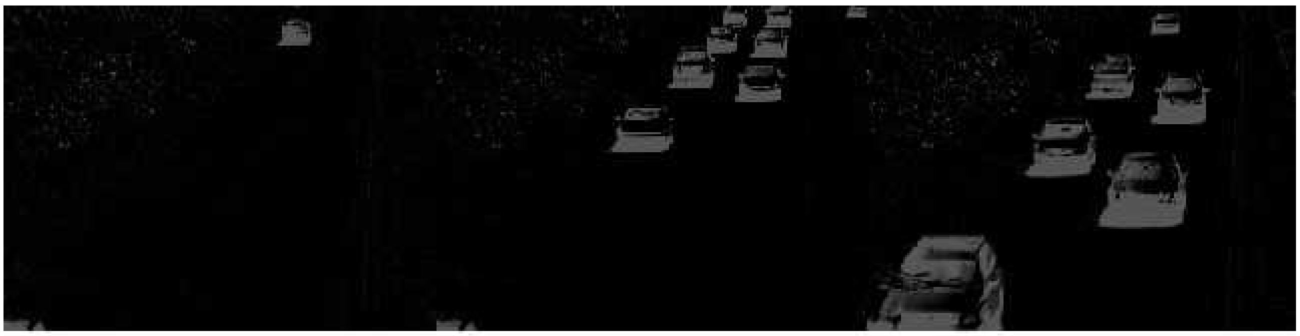
3-)

I first read all the images, then I subtract them from the background. After the subtraction, the cars looked like in brown colour. Then I converted them to a grayscale image by averaging the RGB values as told in the assignment. Finally, I converted all the images to binary format by thresholding them. After some trials, I realized that the best threshold was around %20 so I put the threshold 57. In 58, there was a little dot that gets bigger with dilation, so I did not want it and put the threshold as 57 and I applied morphological operators to all binary images. The results can be shown below.

```
hop1 = imread( 'in000470.jpg');
hop2 = imread( 'in000550.jpg');
hop3 = imread( 'in000750.jpg');
hop4 = imread( 'in000850.jpg');
difference1 = hop1 - hop2;
difference2 = hop1 - hop3;
difference3 = hop1 - hop4;

figure; imshow([difference1, difference2, difference3]);
xlabel('Difference of 1-2, 1-3, 1-4')

red = difference1( :, :, 1 );    % first (red) band
green = difference1( :, :, 2 );  % second (green) band
```

```
blue = difference1( :, :, 3 );   % third (blue) band
gray1 = ((red+green+blue)/3);

red = difference2( :, :, 1 );    % first (red) band
green = difference2( :, :, 2 );  % second (green) band
blue = difference2( :, :, 3 );   % third (blue) band
gray2 = ((red+green+blue)/3);

red = difference3( :, :, 1 );    % first (red) band
green = difference3( :, :, 2 );  % second (green) band
blue = difference3( :, :, 3 );   % third (blue) band
gray3 = ((red+green+blue)/3);
figure; imshow([gray1, gray2, gray3]);
xlabel('Grayscales of 1-2, 1-3, 1-4')

BW1 = gray1 > 57;
BW2 = gray2 > 57;
BW3 = gray3 > 57;
figure; imshow([BW1, BW2, BW3]);
xlabel('Binaries of 1-2, 1-3, 1-4')
erosion(BW1, 2);
hop = imread( 'hop.png');
dilation(hop, 5);

erosion(BW2, 2);
hop = imread( 'hop.png');
dilation(hop, 5);

erosion(BW3, 2);
hop = imread( 'hop.png');
dilation(hop, 5);
```

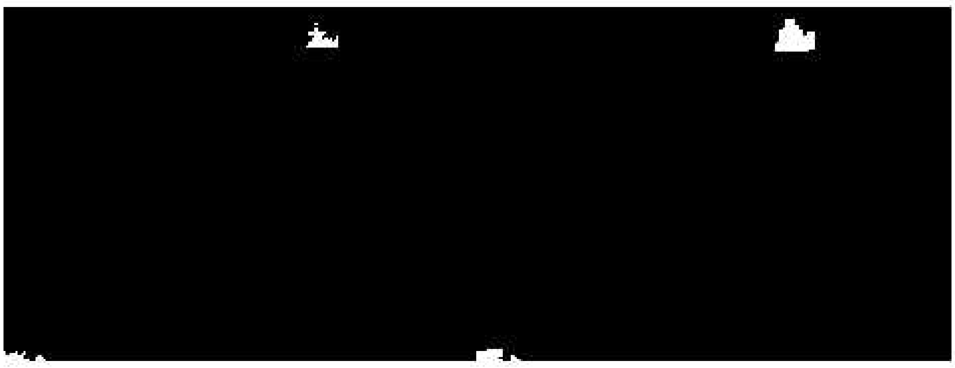

Difference of 1-2, 1-3, 1-4


Grayscales of 1-2, 1-3, 1-4

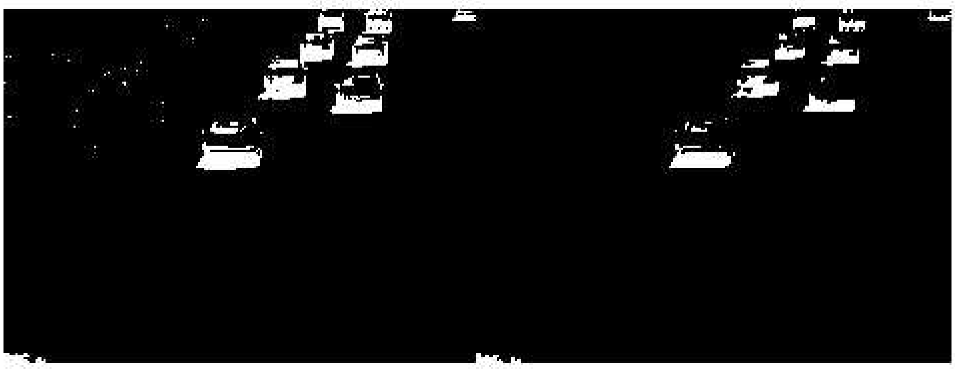
Binaries of 1-2, 1-3, 1-4

Results of binary 1-2:

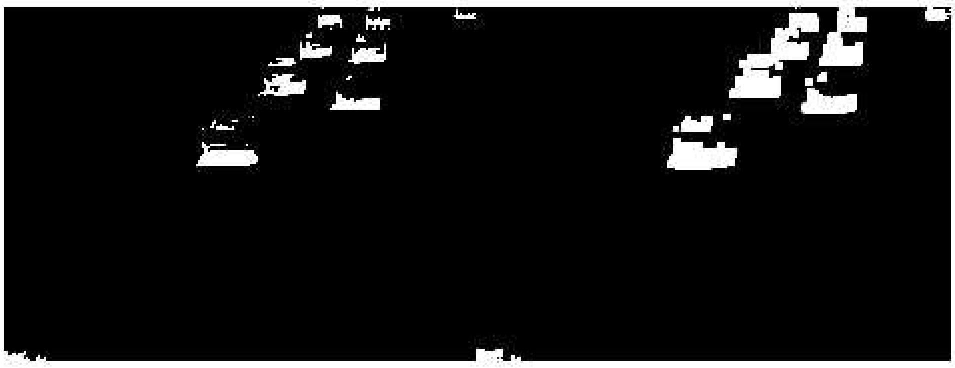
Erosion Applied


Dilation Applied

Results of binary 1-3:

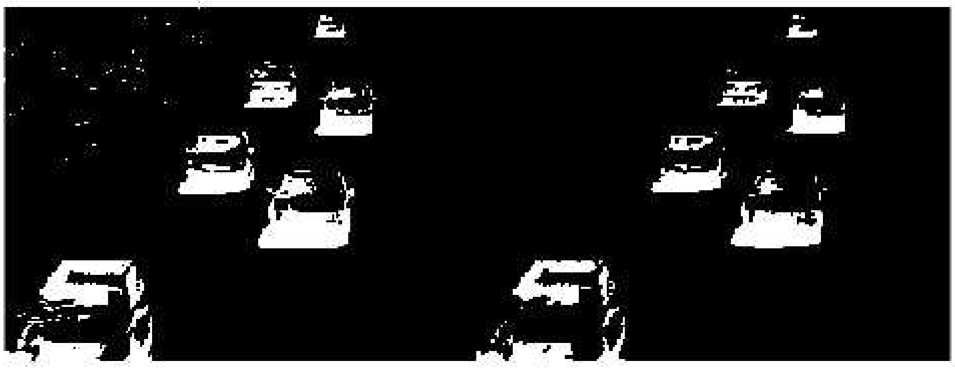


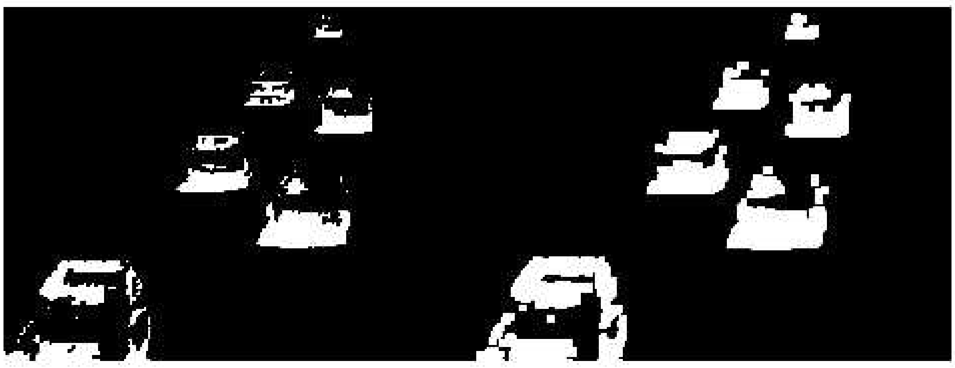
Erosion Applied



Dilation Applied

Results of binary 1-4:


Erosion Applied


Dilation Applied

After these operations, I think the most difficult part was to understand the logic of detecting an object through these operations. Since I already applied erosion, dilation, thresholding and converting grayscale on the previous question, they were not as hard as I first applied them. Therefore, although the question tells when to do what, I did not understand which operations I should do first. In my first several trials, I did not convert them to grayscale and the results were similar but worse. In addition, I did not understand when to convert grayscale, so converting the grayscale before subtracting the background for all images was a bad idea since all the white pixels in the background were gone so my morphological operators did not work. Only after I did many trials, I realized the correct sequence and applied them to images.