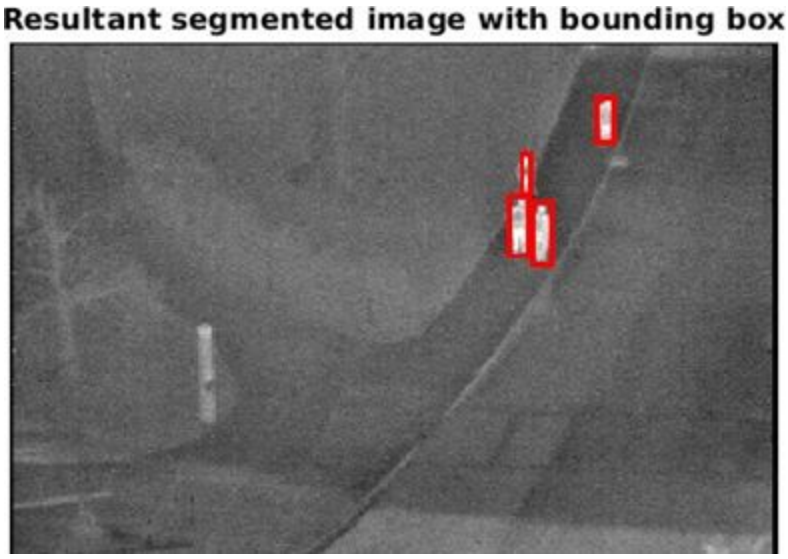
EE5532 MODULE 2 ASSIGNMENT

SEGMENTATION ALGORITHM FOR INFRARED IMAGES



Jaisil Rose Dennison

CONTENTS:

1) INTRODUCTION:

“One image is worth a thousand words". To make this understanding for a computer system, we need algorithms that can translate the 3D world data to computer languages. Computer vision is one such area where the computer is taught about understanding images. One of the crucial steps in the process of converting real world image to computer understanding representation is segmentation. According to wikipedia, segmentation is the processing of partitioning images into multiple segments. This helps the computer to understand the image better for further processes like classification or detection. The main goal of segmentation is extracting pixels that are in the region of interest in the image. There are more challenges during segmentation and it differs based on various factors like image modality, the end goal and the atmospheric conditions. The project deals with the Infrared images and the end goal is to segment the moving humans for manual control. The infrared images used for the project are taken from two directories of the OCTBVS infrared data set. One of the directories is considered for training the segmentation algorithm and the other set was taken for testing. These images are taken using an infrared camera having Raytheon 300D thermal sensor core and 70mm lens. The humans in the images are having the brightest pixels which seems easy to segment. But various climatic conditions lowers the quality of the image which makes it challenging for segmenting. Effective segmentation algorithm detects the object of interest, chucking out background and clutters.

2) Algorithm

The algorithm used for segmenting the humans out of the image(see figure 1) involves intensity based processing and morphological operations. The flow chart for the segmentation algorithm is given below:

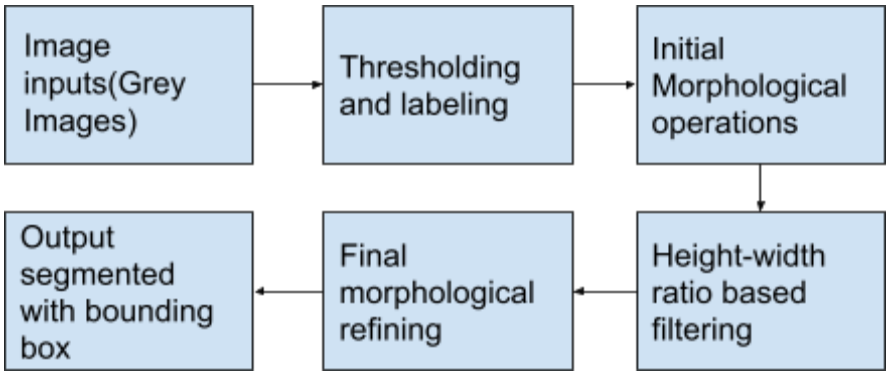


Fig 1: Flowchart of the segmentation algorithm

The input image is grey images of intensity range of [0 - 255]. Initially the histogram is generated by counting the number of pixels per intensity values and analyzed for choosing the best threshold. The objects are labeled from the resultant binary image on the basis of ones and zeros.The resultant images are area filtered to clutter out the small regions. Then morphological operations like closing and height width ratio based filtering are performed to retain the shape of the objects representing humans. Further there is a challenge which can be called a sticky object problem where two separate objects representing two humans are considered as one. This has been solved by performing morphological operations like opening and dilating based on certain conditions. Throughout the algorithm, various morphological operations have been used, among which area based filtering has been used mostly.

Fig 2: Train Input image

Fig 3: Test input image

3) Image thresholding and labeling

The input image is in the grayscale and it is converted to a binary image by thresholding. Thresholding is the process,where the pixel values are changed based on a reference intensity value. The pixel whose intensity values below the threshold are assigned zero value and other pixels are assigned value one. In addition,to this we consider to retain the pixels of interest and in this case are the brighter pixels. From the histogram, we choose the best value that retains the pixels of interest. The thresholded images are labeled based on white areas as individual objects

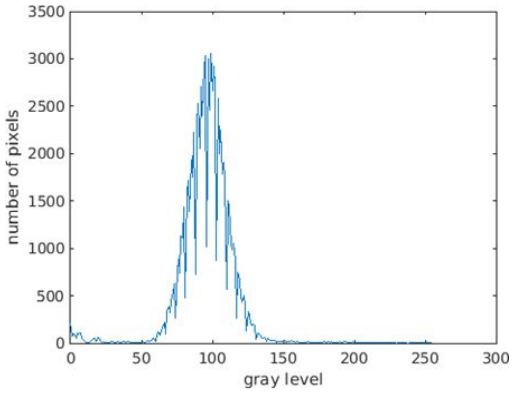


Fig 4: Histogram of training image

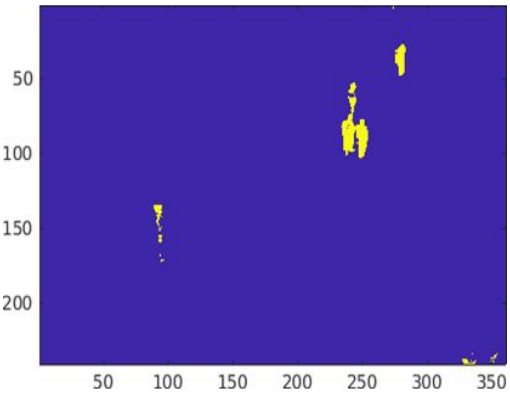


Fig 5:Thresholded image(Threshold: 156)

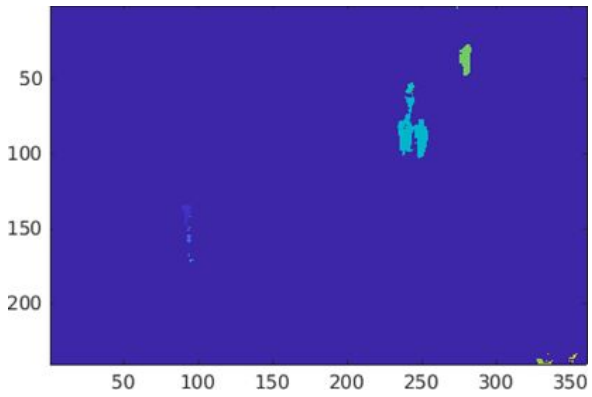


Fig 6: Labeled image

4) Length based filtering and Morphological based processing

The thresholded image retains some pixels that are not of current interest because they have the same intensity values as the pixels of interest. Such pixels can be alleviated using length based filtering. The length of each object is the number of rows within the object and for the current case the threshold length is 20. After all these processes, there are two challenges. One problem is that an object representing a human is cut into two if he or she wears a black dress. This is due to the fact that the black absorbs the light giving low intensity values. Another one is the sticky object problem discussed above. The first problem is solved by morphological closing which is dilation followed by erosion. The closing removes small dark spots and connects small bright cracks using a square structuring element. The structuring element is of various shapes formed using ones and it is chosen according to the requirement. For our case, the square structuring element of size 7 x 7 works better.

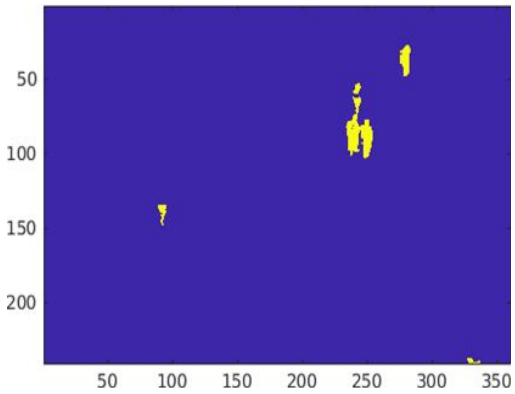


Fig 7: Length based filtered image

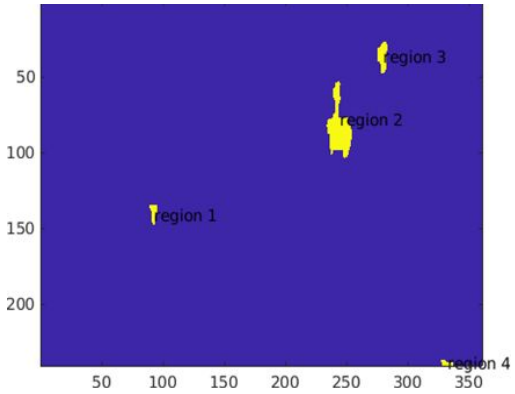


Fig 8: After morphological closing

5) Height_weight ratio filtering

Some objects are confused as representing humans even based on area. The solution for such a problem is height width ratio based filtering. The height width ratio is calculated by dividing the number of rows by the number of columns. The usual height to width ratio for a human ranges from above 1.This is an interesting feature that differentiates most of the human objects from other objects. After the height width ratio filtering, some objects who have the same height width ratio as human objects are removed using area based filtering. The area of an object is based on the number of the pixels within and for our purpose, the area range is above 60. All these processes end up in labeled images with human objects alone.

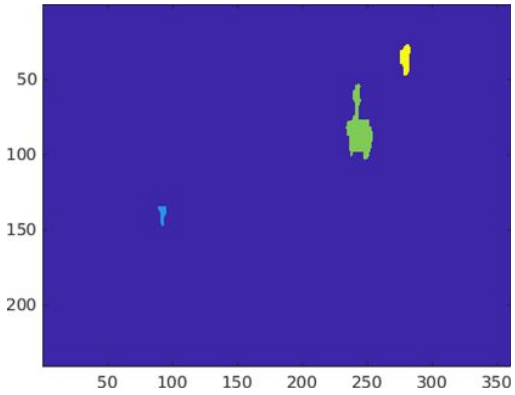


Fig 8: Height width ratio filtered image

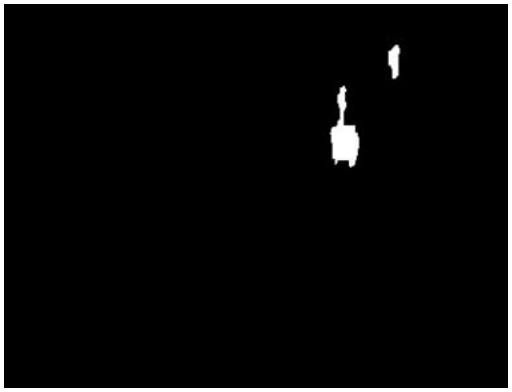


Fig 9: Length filtered image

6) Morphological refining

The step of morphology is included if there are any sticky objects. It is solved using morphological opening and dilation. This method works better compared to the other object separating algorithms like watershed segmentation, erosion for loop. After much testing, the combination of morphological operations works better. The sticky objects are extracted separately using area based filtering of area value ranging above 250. If there are no sticky objects then this filtering would yield an image full of zero values. In that case following steps are skipped and the current image would be the resultant segmented image. If there's any object left after filtering, the image is masked with the original image, followed by thresholding with the value of 145. Due to the morphological operations, the shape of the human object gets distorted, making it difficult to segregate the objects. The above two steps of masking and thresholding helps to alleviate the problem. Then the Image opening is performed (which is image erosion followed by dilating) using the rectangle of dimension 6 x3 as the structuring element. We dilate the objects a little bit using the rectangle of dimension 4 x 1 as the structuring element if there’s any over eroded object. The image with segregated sticky objects are combined with the original segmented image. Finally we get the human objects well segregated as a result of segmentation.

Finally the bounding box for each object is calculated based on the following step:

Step 1: Initially the rows and columns of each object are extracted.
Step 2: The maximum and the minimum values for both the rows and the columns are obtained. The upper left axis of the box is the tuple of the minimum values of the rows and the columns. The lower right axis is the tuple of the maximum values of the rows and the columns. The midpoint axis of the box would be the tuple including the midpoint of the row and the midpoint of the column.
Step 3: The number of rows and the columns of the box is calculated using the formulae [eq. 1] and [eq. 2].

Number of rows = Lower right row - upper left row +1 eq. 1

Number of columns = Lower upper column - upper left column + 1 eq. 2

Thus for each object the bounding box is drawn to show the segmented objects individually.

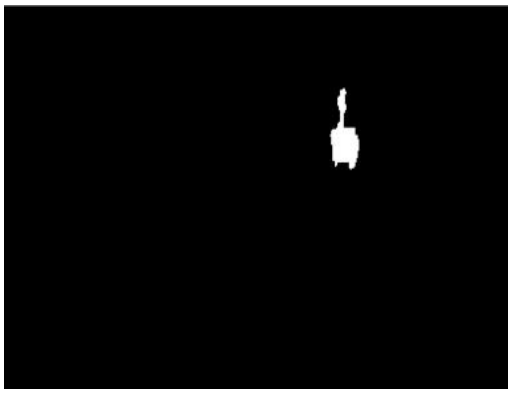


Fig10 : Sticky objects (Area based filtering)

Fig 11: Sticky images separated

So for the current case, we got four objects to segment(see figure 1). The segmented images can be visualized separately after bounding box:

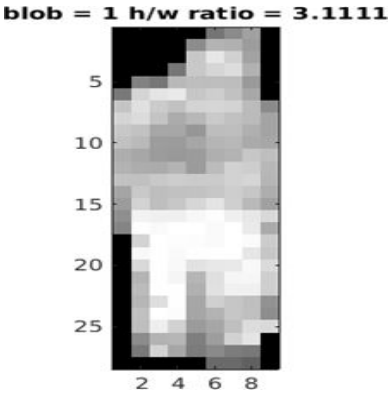


Fig 12: Object 1

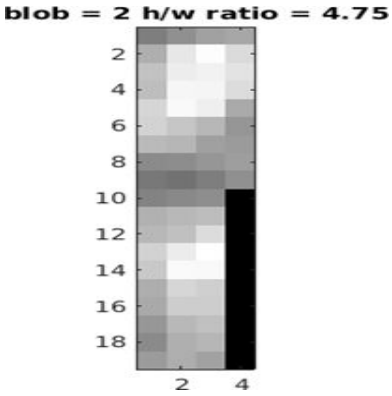


Fig 13: Object 2

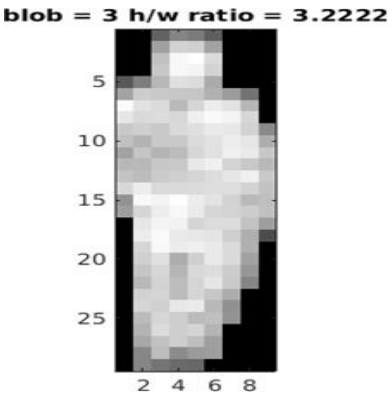


Fig 14: Object 3

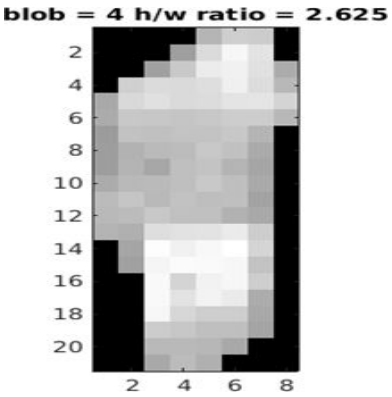


Fig 15: Object 4

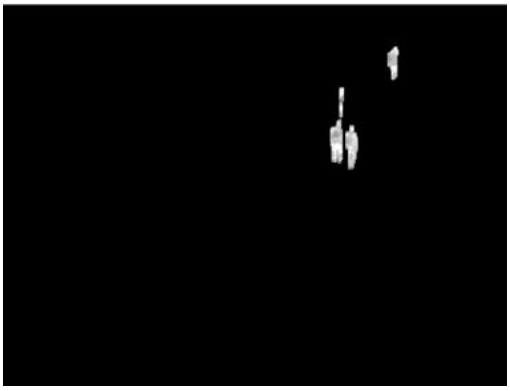


Fig 16: Final segmented image(masked)

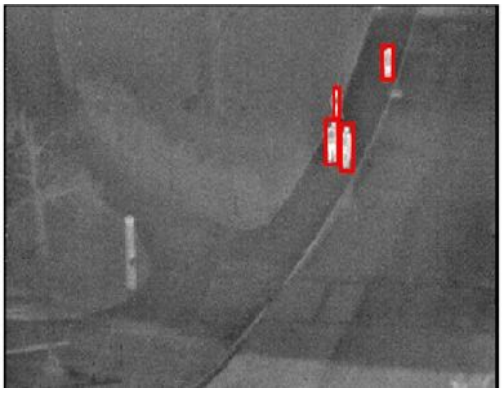


Fig 16: Final segmented image(masked)

Fig 17: Input test image

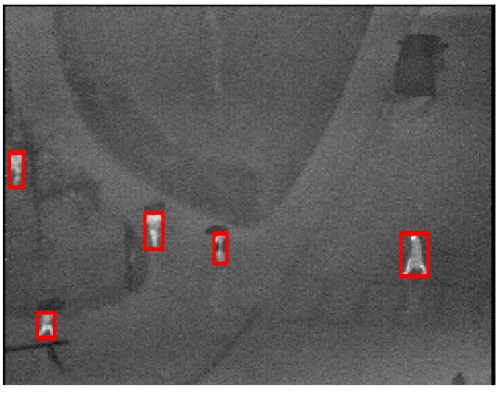


Fig 18: Final segmented image(test image)

7) Code snippet:

```
close all; clear all;
% Read and display image
img1 = double(imread('Train set/img_00001.bmp'));
img2 = mat2gray(img1);
figure
imagesc(img2)
axis('image')
title('Input image2')
colormap(gray(256))
% Generating histogram
[rlen,clen] = size(img1);
maximg1 = max(max(img1));
minimg1 = min(min(img1));
ngraylevels = maximg1 - minimg1 + 1;
graylevelvec = minimg1:maximg1;
[N,EDGES] = histcounts(img1,ngraylevels);
figure
plot(graylevelvec,N); xlabel('gray level'); ylabel('number of pixels');
%test threshold based on where the peak in the histogram settles down.
thresh = 156; %156 for the train images
[rlist,clist] = find(img1 >= thresh);
threshimg1 = zeros(rlen,clen);
for m=1:length(rlist)
threshimg1(rlist(m),clist(m)) = 1;
end
figure
imagesc(threshimg1)
axis('image')
title('threshold image')
% label the image to get the individual objects
[label_img1,num_regions] = bwlabel(threshimg1,4);
figure;
imagesc(label_img1)
axis('image')
title('labeled image showing differnt regions')
% removing objects of small area
pixelcnt_thresh = 20;
intermediate_img1 = zeros(rlen,clen);
 %array to hold the image with small regions removed
for m=1:num_regions
[rl,cl] = find(label_img1 == m);
%rl and cl are vectors holding locations of pixels with label m
if length(rl) >= pixelcnt_thresh
 %only turn on regions that are "big enough"
for n=1:length(rl)
intermediate_img1(rl(n),cl(n)) = 1;
end;
```

```
end;
end;
figure
imagesc(intermediate_img1)
axis('image')
title('image after removing small
regions')
% imclose
SE = strel('square',7);
intermediate_img2 = imclose(intermediate_img1,SE); figure;
imagesc(intermediate_img2);
title('Closed image');
% filter according to the area
[label_img2,num_regions] = bwlabel(intermediate_img2,4);
%find bounding box of each segmented region
% We will segment the image based on the height width ratio ulc_row =
zeros(num_regions,1);
ulc_col = zeros(num_regions,1);
lrc_row = zeros(num_regions,1);
lrc_col = zeros(num_regions,1);
middle_row = zeros(num_regions,1);
middle_col = zeros(num_regions,1);
nrow_blob = zeros(num_regions,1); ncol_blob
= zeros(num_regions,1); h_2_w_ratio =
zeros(num_regions,1);
bw2 = label_img2;
for m=1:num_regions
[rlist,clist] = find(label_img2 == m);
ulc_row(m) = min(rlist);
ulc_col(m) = min(clist);
lrc_row(m) = max(rlist);
lrc_col(m) = max(clist);
nrow_blob(m) = lrc_row(m) - ulc_row(m) + 1;
ncol_blob(m) = lrc_col(m) - ulc_col(m) + 1;
h_2_w_ratio(m) = nrow_blob(m)/ncol_blob(m);
if h_2_w_ratio(m)<1.1
bw2(label_img2 == m)=false; % this step is for filtering objects in an
image based on the height and width ratio
end
middle_row(m) = round((ulc_row(m) + lrc_row(m))/2);
middle_col(m) = round((ulc_col(m) + lrc_col(m))/2);
text(middle_col(m),middle_row(m),['region ',num2str(m)]);
end;
% again labeling the image after filtering
[label_img3,num_regions] = bwlabel(bw2,4); figure;
imagesc(label_img3);
title('Labeled filter filtering');
% some small structures escaped our
% again area filtering to eliminate smaller items
bw = imbinarize(label_img3);
area_filt = bwareafilt(bw, [60 1000]);
figure
imshow(area_filt)
title('Area filtered image')
```

```
% one last time for the area filtering to handle the sticky objects
newbw = bwareafilt(area_filt, [250 5000]);
% Tried watershed method, for loop erosion and dilation
% to seperate the sticky objetcs
if bwconncomp(newbw,4).NumObjects~=0
remaining =  area_filt-newbw;
figure;
imshow(newbw,[]);
title('Sticky objects to seperate');
% So we mask again
final = newbw.*img1;
g = final>145;
figure
imshow(g,[])
title('Thresholded sticky objects');
% Erode to seperate
SE = strel('rectangle',[6 3]);
g = imopen(g,SE);
if bwconncomp(g,4).NumObjects>2
% If they are oversegeregated
g = imdilate(g,strel('rectangle',[4 1]));
figure;
imshow(g)
title('Segregated image threshold');
else
figure
imshow(g,[])
title('Segregated image threshold');
end
% Combine all the objects
area_filt = g + remaining;
end
[final_label,n]=bwlabel(area_filt);
for m=1:n
[rlist,clist] = find(final_label == m);
ulc_row(m) = min(rlist);
ulc_col(m) = min(clist);
lrc_row(m) = max(rlist);
lrc_col(m) = max(clist);
nrow_blob(m) = lrc_row(m) - ulc_row(m) + 1;
ncol_blob(m) = lrc_col(m) - ulc_col(m) + 1;
middle_row(m) = round((ulc_row(m) + lrc_row(m))/2);
middle_col(m) = round((ulc_col(m) + lrc_col(m))/2);
blob = zeros(nrow_blob(m),ncol_blob(m));
for q=1:length(rlist)
rblob = rlist(q) - ulc_row(m) + 1;
cblob = clist(q) - ulc_col(m) + 1;
blob(rblob,cblob) = img1(rlist(q),clist(q));
end;
h_2_w_ratio1(m) = nrow_blob(m)/ncol_blob(m);
 %height and width ratio
figure
imagesc(blob)
axis('image')
colormap(gray(256))
```

```
title(['blob = ', num2str(m),' h/w ratio =
',num2str(h_2_w_ratio1(m))])
text(middle_col(m),middle_row(m),['region ',num2str(m)])
end;
%masking by multiiplying for visualizing the images
masked_img = img1.*area_filt;
figure;
imshow(masked_img,[]);
title('Masked image');
% images are shown with the bounding box figure;
imshow(img2,[]);
title('Resultant segmented image with bounding box');
measurements = regionprops(final_label, 'BoundingBox', 'Area');
 for k = 1 : length(measurements)
BB = measurements(k).BoundingBox;
rectangle('Position', [BB(1),BB(2),BB(3),BB(4)],...
'EdgeColor','r','LineWidth',2 )
end
```

8) Result and discussion

Thus based on training dataset, we train the segmentation algorithm. Of the two directories, the challenging dataset has been used as the training dataset such that it can be generalized for many cases. The images of the testing set are taken under the same condition though taken from different locations. The value for thresholding for the test images is 110 that is taken from its histogram. There are 28 images in the training set and 31 images in the testing set. To evaluate the performance of the segmentation algorithm, the segmentation is performed using the constructed segmentation algorithm and the given ground truth algorithm. The resultant confusion matrix is:

Total Images: 31

Total objects: 90

	OBJECTS	**CLUTTERS**
OBJECTS	**81**	**9**
CLUTTERS	**4**	**-**

From the above confusion matrix:

True positives: Number of correctly segmented objects = 81
Missed target error: Number of target segmented as clutters = 9
False alarm: Number of clutters segmented as target = 4

9) Conclusion and future applications

The segmentation algorithm is trained on a challenging dataset such that it can be generalized for other IR images in raining conditions. The segmentation algorithm uses series for morphological operations and this can be reduced by performing cascaded morphological operations in one step. The next step after segmentation algorithm would be classification. There are various other methods of segmentation based on edge detection and neural networks.

References:

1.http://vcipl-okstate.org/pbvs/bench/

2. https://en.wikipedia.org/wiki/Image_\egmentation

3.https://scikit-image.org/docs/dev/auto_examples/applications/plot_morphology.html#:~:text=Morphological%20closing%20on%20an%20image,and%20connect%20small%20bright%20cracks.&text=Since%20closing%20an%20image%20starts,the%20structuring%20element%20are%20removed**.**

4.https://en.wikipedia.org/wiki/Opening_(morphology)#:~:text=Opening%20removes%20small%20objects%20from,specific%20shapes%20in%20an%20image**.**

5.https://www.mathworks.com/help/images/structuring-elements.html#:~:text=A%20structuring%20element%20is%20a,process%20in%20the%20input%20image**.**