Image and Video Processing Coursework

Stroke Analysis and Morphology Fused to Binarize Degraded Documents

Daniel Taylor

Introduction

Segmenting an image's foreground from its background allows objects of interest to be isolated, whilst the other objects in the image of no or little interest, can be discarded. Image segmentation is frequently used in image processing methods, such as image localization and object detection. However, it is also used in the healthcare industry and is capable of detecting cancerous cells or brain tumours. In this paper, segmentation is used to binarize documents, which have been degraded to the extent that they are illegible. Factors such as faded text, bleed through as well as shaded backgrounds, can all severely affect documents. Therefore, in order to improve the legibility of the text, the document must be converted into a digital format.

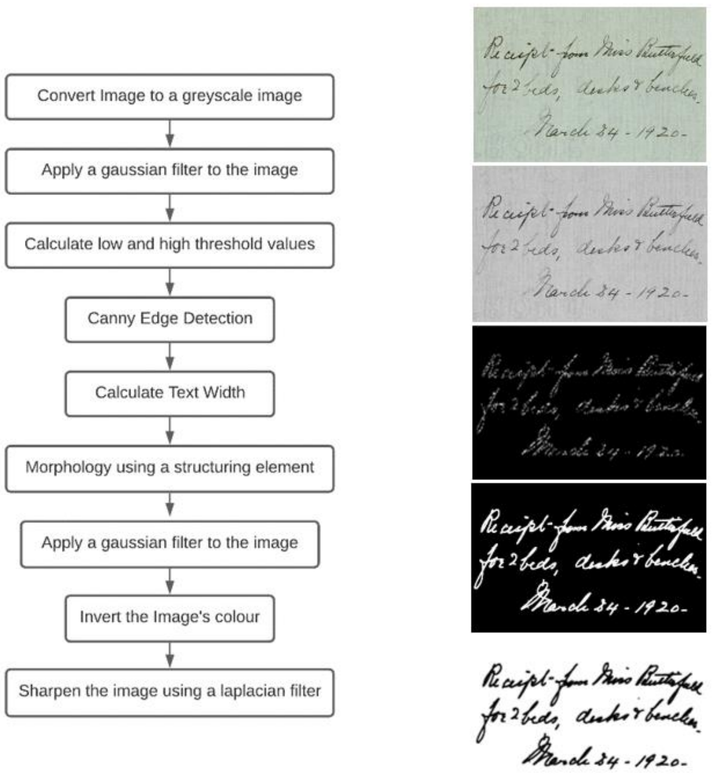
Thresholding is a commonly used method for document binarization, which is split into two categories, global thresholding and locally adaptive thresholding. A global thresholding method which produces highly accurate results, is Otsu's method ***[4]***, which utilizes the image's intensity histogram to determine a single threshold value. However, this approach has limitations, as images with substantially large regions of varying greyscale levels, can affect the valleys used to detect thresholds. Wu et al ***[9]*** Implemented a more robust method which firstly uses a low-pass gaussian filter to remove noise, and also smoothens the image's histogram to provide a clear valley to use for a threshold. However, this approach still has limitations as small print is lost due to scaling of images, leading to disjointed edges. Pixel values which are above or equal to thresholds, are segmented as part of the background, whilst the other pixels are segmented as the foreground ***[8]***.

Edges of objects can be detected in images, as the change of intensity between 2 adjacent pixels is large. However, their orientation is also to be considered, as edges can appear disjointed otherwise. The Sobel edge operator ***[6]***, can be optimized to detect edges of various orientations. The gradient magnitude and gradient direction are calculated at every point in the image, to analyse where there is a change in the intensity. The operator does however, produce results that are inaccurate as many edges are detected if noise is present ***[5]***.

Canny edge detection ***[1]*** extends the Sobel edge operator by using both the gradient magnitude and gradient direction of images to determine edges. Non maxima suppression is then utilized to thin the edges, in order to obtain one pixel for a single edge. Although canny edge detection is time consuming, as opposed to the Sobel edge operator, it can produce considerably more accurate results for complex images that contain multiple objects ***[3]***. Su et al ***[7]*** implemented a document binarization method which incorporates canny edge detection to obtain an outline of text, before calculating an estimate for the text stroke width, using the modal value of distances between edges, as the neighbourhood size of the adaptive threshold must be greater than the text width to produce accurate results. Cao et al ***[2]*** approach used an improved variation of canny edge detection with the intention to determine the region of the object in the foreground rather than binarize the image using it. A closed canny edge is used to eliminate the problem of weak edges being detected. The pixels with large intensities are used to calculate high and low threshold values, whilst those with low intensities are ignored, and therefore this produces a greater accuracy to detect absolute edges.

Methodology

In this paper, multiple techniques which were included in the literature review, such as Otsu's thresholding, canny's edge detection as well as an estimated stroke width similar to that used in ***[7]***, have been fused together to create a novel approach. The new algorithm can be used to binarize images of documents, and can be tested against each of the individual techniques involved in the composition of the new approach, using both subjective and objective procedures.



Convert Image to a greyscale image
Apply a gaussian filter to the image
Calculate low and high threshold values
Canny Edge Detection
Calculate Text Width
Morphology using a structuring element
Apply a gaussian filter to the image
Invert the Image's colour
Sharpen the image using a laplacian filter

Figure 1: A new approach for image document binarisation. The various steps being applied to an image in the order shown by the diagram on the left.

The pre-processing stage consists of converting the image into a greyscale format, before applying a gaussian filter to the image in order to eliminate noise. The equation used for applying the gaussian matrix to the image is illustrated in the following equation:

$$\mathbf{O}[\mathbf{x},\mathbf{y}] = \mathbf{I}\,[\mathbf{x},\mathbf{y}] * \mathbf{G}\,[\mathbf{x},\mathbf{y},\boldsymbol{\sigma}] \tag{1}$$

Where the values x and y correspond to the rows and columns of the matrix respectively, and σ is the value relating to the extent that the image is smoothened. G is the gaussian filter which is applied to the input image I, and O is the output image.

The mean value of the image's pixels is then calculated and adapted to establish both low and high threshold values, which are passed as a double threshold to the canny edge detection algorithm (closed canny edge). A double threshold is particularly important as the hysteresis method uses this to determine weak and strong edges. Therefore, if these values are poorly calculated, the edge map that is outputted will either contain additional edges that are irrelevant, or eradicate edges of characters which are crucial.

The closed canny edge map can be used to evaluate the width of the text in the foreground. The distance between two edges of a character, within the same row in a matrix, are calculated for each character. The modal value of the widths is used in preference over the mean, as it is a positive integer which will not be impacted by anomalies, such as instances in handwritten documents, where linking between letters will be calculated as a width value. The equation relating to this can be seen in equation 2.

$$\boldsymbol{\Theta} = \widehat{\mathbf{X}}\big(\forall(\boldsymbol{\alpha} - \boldsymbol{\gamma})\big) \tag{2}$$

Where the value α relates to the number of pixels between two edges of the same character within the same row of a matrix (including pixels used for edges). γ is the sum of the number of pixels which compose the characters width within the same row in a matrix, and excludes the number of edge pixels. ∀ symbolises a universal quantifier for all widths to be calculated and X̂ is the modal value of all widths. The value of Θ corresponds to the neighbourhood size of the structuring element used to dilate the canny edge map.

$$\boldsymbol{\Theta} = \widehat{\mathbf{X}}\big(\forall(\boldsymbol{\alpha} - \boldsymbol{\gamma})\big) + \mathbf{1} \tag{3}$$

Equation 3 is an adaptive method applied to handwritten documents which extends equation 2. The constant value of 1 is added to the result of the modal stroke width, in order to ensure a sufficient area of the character is filled. This is a necessity for handwritten documents. If the value of θ is too small, the large variations of text strokes are likely to cause the edges of the edge map to be thickened, but not fill in the character in the morphological step which follows. This would lead to OCR problems later. However, if the constant exceeds 1, the text will become illegible.

Morphological structuring is used to fill in the edge map, as it is a method which is capable of preserving both the shape and size of objects. A flat square structuring element, with a neighbourhood size of the text width, which was previously calculated, is used to dilate the image. This essentially fills in the majority of the edge map. Before the next morphological method is processed, a gaussian filter is applied to the image, as this encourages both the eyes and counters of characters to be retained. Otherwise, when the letters are dilated, these elements of characters are lost, having an impact on OCR detection – an example of this is letter ‘o’s being mistaken for ‘a’s and vice versa. Morphological closing can then be applied to the image, which uses the same structural element (a disk in the new approach), to dilate and then erode the image.

A gaussian filter, which is similar to that used in the pre-processing stage, is applied to the image, but with a smaller neighbourhood size, in order to smoothen the image within a smaller scale. The reasoning behind this, is that there may still be areas of characters, which have not been completely filled in after the morphological restructuring. Figure 2 exemplifies this. Implementing this into the approach helps to improve the probability that OCR will correctly detect characters, as there are fewer or no white pixels remaining in characters.

Penultimately, to prevent OCR technology not acknowledging white text on a black background, the image is inverted. To finish the binarization, the image must be sharpened to ensure that it is clear enough to be detected by both OCR technology and multiple subjects. The sharpening process incorporates the Laplacian filter as it is used to enhance discontinuities with greyscale pixels, whilst regions with varying greyscale levels are still enhanced but at a lower rate. Therefore, the outline of the letters becomes clearer whilst the shading inside the characters is affected to a smaller extent.

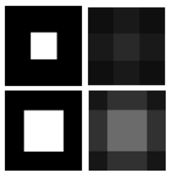


Figure 2: The result of applying a gaussian filter to fill in pixels that have not been filled after applying morphological techniques. The two images on the left show potential outcomes within characters after applying morphology. The images on the right are the results of applying the gaussian filter.

A drawback of this method includes the significant decline in accuracy when binarizing documents which contain varying sizes of text, as this will impact the modal value used as the text width, leading to an incorrect neighbourhood size for the square structuring element used to dilate the canny edge map. Another drawback of the approach is that the thresholding values will need adapting in order to improve the hysteresis method to detect strong and weak edges in a broader variety of images. For example, an image with extremely faint text will have different threshold values to images with large contrasts.

Results

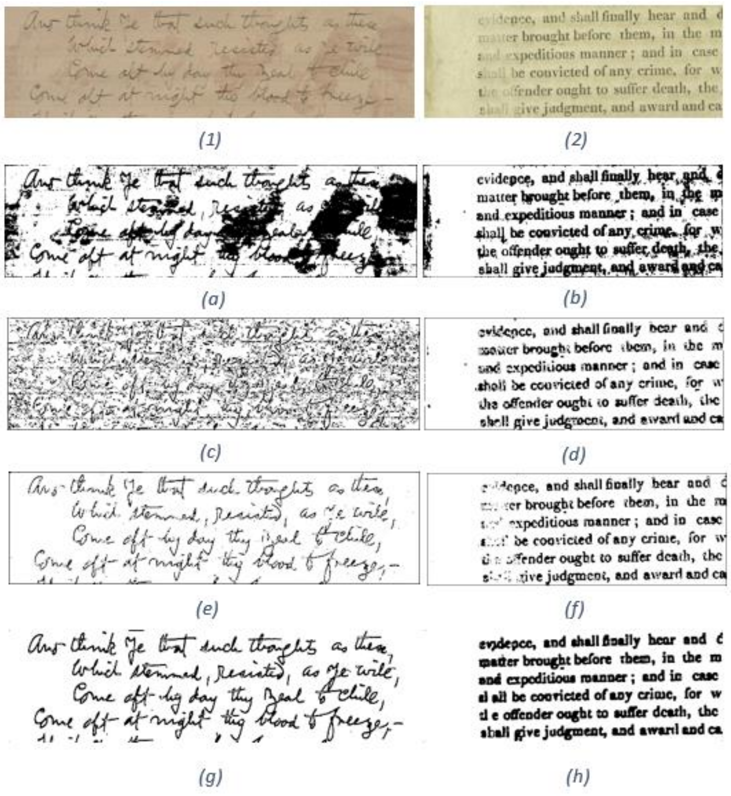


Figure 3: Results of Otsu's thresholding algorithm is shown in images (a) and (b). Images (c) and (d) are the results of Canny edge detection. Both (e) and (f) are results of sobel edge detection, whilst images (g) and (h) show the results of the new approach taken in this paper. The image (1) is the HW3 document in the DIBCO dataset and the image (2) is the PR8 document in the same dataset.

To ensure subjective results are collected fairly, images should be shown to subjects in an order of least legible documents to most legible. This will remove the possibility of them reciting words, but only to an extent. Using the results displayed in figure 3, image (c) should be presented first, followed by images (a),(e) and (g). However, for the printed document (2), the order should follow (f), (b), (d), (h).

	Handwritten Document		**Printed Document**	
	HW3		PR8	
	Characters	**Words**	**Characters**	**Words**
Original Image	116/116	30/30	180/180	38/38
Otsu Threshold	83/116	16/30	180/180	38/38
Canny Edge	79/116	15/30	179/180	37/38
Sobel Edge	105/116	23/30	163/180	34/38
New Approach	106/116	24/30	180/180	38/38

Table 1: Subjective results collected from 5 subjects and averaged. Images HW3 and PR8 of the DIBCO image dataset were used to test the various algorithms.

	Handwritten Document		**Printed Document**	
	HW3		PR8	
	Characters	**Words**	**Characters**	**Words**
Original Image	0/116	0/30	144/180	29/38
Otsu Threshold	0/116	0/30	55/180	7/38
Canny Edge	0/116	0/30	131/180	24/38
Sobel Edge	0/116	0/30	143/180	29/38
New Approach	2/116	0/30	169/180	32/38

Table 2: The OCR results collected using Kofax OmniPage Ultimate. Images HW3 and PR8 of the DIBCO image dataset were used to test the various algorithms.

It is proven that handwritten documents rely solely on subjective results to determine their accuracy, as it is currently not possible for handwriting to be effectively detected using OCR technology. 27% of the 116 characters were evaluated using the new approach and only a poor 6% of these characters were detected correctly. However, this still outperformed the other algorithms which failed to evaluate any characters at all.

However, 84% of words and 94% of characters in the printed document were detected using OCR, showing a high level of accuracy. The subjective results in table 1 also validate this as every character and word was correctly evaluated.

	HW3	**PR8**
PSNR	43.6930	39.4124
SSIM	0.0072	0.0079

Table 3: PSNR and SSIM value of the new approach in comparison to the "Ground Truth" image.

The Peak Signal-to-Noise Ratio values for the images shown in table 3, suggest that the image quality is similar to the ground truth, however, for document binarization, the legibility of documents is the area of focus rather than its similarity. Two images could have the same PSNR value and have a distinct difference in quality between the two, suggesting that this measure alone should not be considered to determine whether segmentation results are accurate. The structural similarity index measurement (SSIM) value is more reliable to determine levels of similarity to the ground truth, as it considers the luminance, contrast and structures of both images which PSNR does not. The SSIM results of both images were neither outstanding or poor, suggesting that this measure is also unreliable as images with mediocre SSIM results can still be interpreted accurately, generating a hypothesis that human perception is actually the best measurement for document binarization.

Conclusions

It can be concluded that although the subjective analysis of the printed document in table 1 illustrates a high level of accuracy, these results are arguably misleading as typoglycemia has a major influence on accuracy. Studies have shown that words can be read correctly as long as the first and last letters are in the correct positions, and therefore even if other letters in the word could potentially be perceived as other characters due to poor binarization, the correct word can still be recognized by subjects. Serial and parallel letter recognition will also most likely be unintentionally used by subjects, as it is a natural instinct to read a word, letter by letter, if parts of the word are illegible. Along with the context of previously read content, this also improves the subject's ability to predict what a word may be.

This therefore suggests that drawbacks may not necessarily be fixed to the algorithm but also to human perception. The order in which the images are shown to the subjects also potentially be seen as having bias as subjects have the ability to recite text, meaning that no matter what the approach outputs, the subject may still interpret highly accurate results.

It is clear that the new approach is capable of outperforming the other individual approaches, as long as the threshold values are correctly adapted to the image, and that the text size remains at a consistent value.

Appendix

generate_results.m

```
%This is the script which is to be run to display
%All the variations of techniques used on the images
%As well as displaying the psnr and ssim values

figure, subplot(3,2,1);
pic=imread("HW3.png");
imshow(pic), title("Original HW3");          %Displays the
original HW3 image

subplot(3,2,2);
pic = OtsuThresholding("HW3.png");
imshow(pic), title("HW3 Otsu Threshold");    %Displays
Otsus thresholded algorithm used on the HW3 image

subplot(3,2,3);
pic = CannyEd("HW3.png");
imshow(pic), title("HW3 Canny Edge");        %Displays the
Canny edge detection algorithm used on the HW3 image

subplot(3,2,4);
pic = SobelEd("HW3.png");
imshow(pic), title("HW3 Sobel Edge");        %Displays the
sobel edge detection algorithm used on the HW3 image

subplot(3,2,5);
finalhw3 = coursework_script("HW3.png");
imshow(finalhw3), title("HW3 New Approach");
imwrite(finalhw3, "HW3binarized.tiff");      %The new
approach applied to the image HW3 and written as a new
image

subplot(3,2,6);
[val,map] = ssimResult("HW3_GT.tiff",
"HW3binarized.tiff"); %Calculates the SSIM value of the
image HW3 and its ground truth
psnr = calculatePSNR("HW3_GT.tiff", finalhw3);
%Calculates the PSNR value of the image HW3 and its
ground truth
imshow(map,[]), title(['PSNR Value: ', num2str(psnr),
newline 'SSIM value: ',num2str(val)]); %Displays the PSNR
and SSIM values and the SSIM map
```

```
figure, subplot(3,2,1);
pic=imread("PR8.png");
imshow(pic), title("Original PR8");          %Displays the
original PR8 image

subplot(3,2,2);
pic = OtsuThresholding("PR8.png");
imshow(pic), title("PR8 Otsu Threshold");    %Displays
Otsus thresholded algorithm used on the PR8 image

subplot(3,2,3);
pic = CannyEd("PR8.png");
imshow(pic), title("PR8 Canny Edge");        %Displays the
Canny edge detection algorithm used on the PR8 image

subplot(3,2,4);
pic = SobelEd("PR8.png");
imshow(pic), title("PR8 Sobel Edge");        %Displays the
sobel edge detection algorithm used on the PR8 image

subplot(3,2,5);
finalpr8 = coursework_script("PR8.png");
imshow(finalpr8), title("PR8 New Approach");
imwrite(finalpr8, "PR8binarized.tiff");      %The new
approach applied to the image PR8 and written as a new
image

subplot(3,2,6);
[val,map] = ssimResult("PR8_GT.tiff",
"PR8binarized.tiff"); %Calculates the SSIM value of the
image PR8 and its ground truth
psnr = calculatePSNR("PR8_GT.tiff", finalpr8);
%Calculates the PSNR value of the image PR8 and its
ground truth
imshow(map,[]), title(['PSNR Value: ', num2str(psnr),
newline 'SSIM value: ',num2str(val)]); %Displays the PSNR
and SSIM values and the SSIM map
```

coursework_script.m

```
%This script contains the new approach to binarizing
%documents using canny edge detection and then applying
%morphology using different structering elements in order
%to fill in the letters before finaly smoothing the image
%to remove any noise

function edge_img = coursework_script(img)
%Used to check substrings in order to determine whether
the
%image is a printed document or handwritten document -
useful for
%the character width calculation
fileType=img;
typ ="PR";
typ2="H0";
typ3="H10";


%read the image in and convert it to grayscale
img = imread(img);
img = rgb2gray(img);


%Preprocessing stage to smoothen the image to reduce
noise - using a gaussian
%filter
img = filter2(fspecial('gaussian',5,1),img);


% %get the local threshold values of the specific image,
allowing all images to have different
% %thresholds rather than an absolute value for all
images.
high_thresh = hi_threshold(uint8(img));
low_thresh = lo_threshold(uint8(img));


%Canny edge detection used to segment the foreground from
the background
[grad_mag,grad_angle] = calculate_gradient(img);
grad_dir = discretize_gradient(grad_angle);
maxima_img = non_maxima_suppress(grad_mag,grad_dir);
strong_edges = threshold_matrix(maxima_img,high_thresh);
weak_edges = threshold_matrix(maxima_img,low_thresh);
edge_img = hysteresis(strong_edges,weak_edges);
```

```
%Checks what sort of document is being processed to
determine
%which calculation is to be carried out
if contains(fileType,typ) || contains(fileType,typ2) ||
contains(fileType,typ3)
    widthVal = calLength(edge_img); %calculates the width
of the printed text
else
    widthVal = calLength(edge_img) + 1; %calculates the
width of the handwritten text
end


%Morphology techniques applied to the edge map in order
to remove small
%areas of noise that may still occur whilst also filling
in the letters.
se = strel('square',widthVal);
edge_img = imdilate(edge_img,se);    %imdilate used to
fill in the majority of the letters
edge_img = filter2(fspecial('gaussian',3,1),edge_img);
%Retains areas within looped letters such as g or d
se = strel('disk',2);
edge_img = imclose(edge_img, se);


%filter is applied to the image to remove pixels that may
still not be filled
%in, which could be seen as similar to salt and pepper
noise
edge_img = filter2(fspecial('gaussian',3,1),edge_img);


%inverting the image before sharpening it using a
laplecian filter
edge_img = uint8(255*(1-edge_img));
edge_img = display_laplacian_result(edge_img);


%figure,imshow(edge_img),colormap(gray(256)),axis off
% imwrite(edge_img, "PR8binarized.png");
end


function [grad_mag,grad_dir] = calculate_gradient(img)
%CALCULATE_GRADIENT calculates the gradient magnitude and
gradient
%direction of an image.
```

```
     sobel_v = fspecial('sobel');
    sobel_h = rot90(sobel_v,3);
    grad_v = filter2(sobel_v,img);
    grad_h = filter2(sobel_h,img);
    grad_mag = sqrt(grad_v.^2 + grad_h.^2);
    grad_dir = atan(grad_v./grad_h);
end

function grad_dir = discretize_gradient(grad_angle)
%DISCRETIZE_GRADIENT takes in a matrix containg a set of
angles between
%-pi/2 and +pi/2 radians and discretizes them to an
integer indication the
%nearest direction in an image where the values
correspond to:
%   1 - Horizontal
%   2 - Diagonally upwards-and-right
%   3 - Vertical
%   4 - Diagonally upwards-and-left
    grad_angle = 8*grad_angle/pi;
    grad_dir = zeros(size(grad_angle));
    grad_dir((grad_angle>=-1) & (grad_angle<1)) = 1;
    grad_dir((grad_angle>=1) & (grad_angle<3)) = 2;
    grad_dir((grad_angle<-3) | (grad_angle>=3)) = 3;
    grad_dir((grad_angle>=-3) & (grad_angle<-1)) = 4;
end

function maxima_img =
non_maxima_suppress(grad_mag,grad_dir)
%Given a matrix of gradient magnitudes and a matrix of
gradient directions
%(as discretized by DISCRETIZE_GRADIENT),
NON_MAXIMA_SUPPRESS returns a
%matrix where all values which are non-maximal in their
local gradient
%directions are set to 0. Those which are maximal remain
unchanged.
    grad_mag_e = expand_matrix(grad_mag);
    dir_1_max = max(grad_mag_e(2:end-1,1:end-
2),grad_mag_e(2:end-1,3:end));
    dir_2_max = max(grad_mag_e(1:end-
2,3:end),grad_mag_e(3:end,1:end-2));
    dir_3_max = max(grad_mag_e(1:end-2,2:end-
1),grad_mag_e(3:end,2:end-1));
    dir_4_max = max(grad_mag_e(1:end-2,1:end-
2),grad_mag_e(3:end,3:end));
```

```
    correct_dir_max = zeros(size(grad_mag));
    correct_dir_max(grad_dir == 1) = dir_1_max(grad_dir
==1);
    correct_dir_max(grad_dir == 2) = dir_2_max(grad_dir
==2);
    correct_dir_max(grad_dir == 3) = dir_3_max(grad_dir
==3);
    correct_dir_max(grad_dir == 4) = dir_4_max(grad_dir
==4);

    maxima_img = grad_mag;
    maxima_img(grad_mag<=correct_dir_max) = 0;

    function new_mat = expand_matrix(old_mat)
    %EXPAND_MATRIX adds a border of zeros to the original
matrix passed to
    %it. It should only be used as a helper function for
    %NON_MAXIMA_SUPPRESS
        new_mat =
zeros(size(old_mat,1)+2,size(old_mat,2)+2);
        new_mat(2:end-1,2:end-1) = old_mat;
    end
end

function t_img = threshold_matrix(mat,thresh)
%THRESHOLD_IMAGE takes in a matrix and a threshold. It
returns a new matrix
%which is 0 where the original matrix was less than the
threshold and 1
%elsewhere.
    t_img = zeros(size(mat));
    t_img(abs(mat)>=thresh) = 1;
end

function joined_img = hysteresis(strong_edge,weak_edge)
%HYSTERESIS persorms image hysteresis, joining pixels in
strong_edge using
%connecting pixels in weak_edge.
    [r,c] = find(strong_edge);
    joined_img = bwselect(weak_edge,c,r,8);
end

%Calculates a local threshold using the mean of the
image's pixels
function hithresh = hi_threshold(img)
hithresh = mean(img(:));          %Changing 85 will have
an impact on which lines are picked up during the
```

```
hithresh = hithresh-85;           %canny edge detection
stage which follows. This value works for both images
end

function lothresh = lo_threshold(img)
lothresh = mean(img(:));          %Changing 8 will have
an impact on which lines are picked up during the
lothresh = lothresh/8;            %canny edge detection
stage which follows. This value works for both images
end


%Uses a laplacian filter to sharpen the image
function sharpened = display_laplacian_result(img)
    laplac = [0,1,0;1,-4,1;0,1,0];
%Laplacian filter matrix
    lap = uint8(filter2(laplac, img, 'same'));   %Ensures
the matrix size does not change when filters applied

    sharpened = 3*imsubtract(img, lap);          %Change
value of 3 to change level of sharpness (accuracy may
worsen)
    sharpened = imadd(img, sharpened);           %Adds
the sharpened map to the image for sharpening to take
place
end


%Calculates the width of the characters
function leng = calLength(img)
pic = logical(img);
[row,col] = size(img);
widths = [];
length = 0;

for rows=2:row-2
    for cols=2:col-2
        if  pic(rows,cols) == 1 && pic(rows,cols+1) == 0
&& length==0   %If the pixel is the first edge
            length = length + 1;
        elseif pic(rows,cols) == 0 && length > 0
%If the pixel is in between both edges
            length = length + 1;
        elseif pic(rows,cols) == 1 && pic(rows,cols+1) ==
0 && length > 0   %If the pixel is the second/last edge
            length = length + 1;
            length = length - 2;
            widths(end+1)=length;
            length = 0;
        end
    end
end

%Uses the mode over the mean as anomally results will not
impact the mode
%and are common with handwriting where letters may link
leng = mode(widths);
end
```

ssimResult.m

```
%Calculates the ssim value between two images. Takes into
%consideration the structural similarity of the two.
%Used in the coursework to compare the ground truth and
%the new approaches result
function [perc, ssimMap]= ssimResult(GrImg, img)
ground = uint8(imread(GrImg));
tester = uint8(imread(img));

[ssimVal,ssimMap] = ssim(ground,tester);
perc=num2str(ssimVal);
end
```

CannyEd.m

```
%This Script was used specifically to segment images
%using only the canny edge detection for testing
purposes.
%The novel approach in the paper conists of a more
complexed
%version with better accuracy.

function pic = Canny(img)
im = imread(img);
im = rgb2gray(im);
%Preprocessing stage of reading image, converting it to
img = filter2(fspecial('gaussian',5,1),im);
%greyscale and applying a gaussian filter to remove noise

pic = cannyEd(img);
%[Gx,Gy] = imgradientxy(pic);                        %The
magnitude and gradient is used in canny edge detection
%[Gmag,Gdir] = imgradient(Gx,Gy);                    %to pick
up the lines in order to get the edge map


se = strel('disk',2);
%morphologcial technique Used for filling letters
pic = imclose(pic, se);                              %as
imfill() method had severly poor results when testing
with OCR (<25%)
%imfill(pic, 'holes');
%Uncomment this line and comment out lines 16 and 17 for
the comparison

%Displays the magnitude of the edges
%figure, subplot(2,2,1), imshow(Gmag),
title("Magnitude");  %uncomment this line to view the
magnitude map

%Displays the direction of the edges
%subplot(2,2,2), imshow(Gdir), title("Direction");
%uncomment this line to view the direction map

pic = uint8(255*(1-pic));
%invertion of image colour
%Displays the final canny edged map
%subplot(2,2,3.5), imshow(pic), title("Canny Edge");
%uncomment this line to view the canny edge map
end
```

```
function pic = cannyEd(img)
pic = edge(img,'canny');       %Used the built in method
only for testing purposes of canny edge detection alone
end
```

SobelEd.m

```
%This Script was used specifically to segment images
%using only the sobel edge detection for testing and
comparison purposes.

function pic = Sobel(img)
im = imread(img);
im = rgb2gray(im);
%Preprocessing stage of reading image, converting it to
img = filter2(fspecial('gaussian',5,1),im);
%greyscale and applying a gaussian filter to remove noise

pic = sobelEd(img);
%Applies the sobel edge detection
pic=morph(pic);
%fills the letters

pic = uint8(255*(1-pic));
%Invert black and white

%Displays the final sobel edged map
%figure,imshow(pic), title("Sobel Edge");
end

%Used the built in method only for testing purposes of
sobel edge detection alone
function pic = sobelEd(img)
[~,thresh] = edge(img,'sobel');
fudFac = 0.5;
pic = edge(img,'sobel',thresh * fudFac);
end

function pic = morph(img)
se = strel('disk',2);           %morphologcial technique
Used for filling letters
pic = imclose(img, se);         %as imfill() method had
severly poor results when testing with OCR (<25%)
% imfill(img, 'holes');         %Uncomment this line and
comment out lines 26 and 27 for the comparison
end
```

calculatePSNR.m

```
%Calculates the psnr value between two images
%Used in the coursework to compare the ground truth and
%the new approaches result
function ret = calculatePSNR(img1,pic2)
pic = imread(img1);

diff = (uint8(pic) - uint8(pic2)).^2;
sumdiff = sum(diff, 'all');
[width,height] = size(pic);
pixels = width*height;
mse = sumdiff/pixels;
ret = 10.*log10(8*8/mse);
end
```

OtsuThresholding.m

```
%This Script was used specifically to segment images
%using only the Otsu Thresholding method for testing
%and comparison purposes.

function pic = Otsu(img)
im = imread(img);
im = rgb2gray(im);
%Preprocessing stage of reading image, convertint it to
img = filter2(fspecial('gaussian',5,1),im);
%greyscale and applying a gaussian filter to remove noise
pic = otsu_threshold(img);
%figure, imshow(pic);
%imwrite(pic, "PR8localThreshold.png");
end

function pic = otsu_threshold(img)
thresh = mean(img(:));                                  %The mean
of the image pixels is used as the threshold
img(img >= thresh)= 255;                                %to
convert the pixels to either black or white
img(img < thresh) = 0;
pic = img;
end
```

References

[1] Canny, J., 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-8(6), pp. 679-698.

[2] Cao, R., Tan, C. L., Wang, Q. & Shen , P., 2004. Segmentation and Analysis of Double-Sided Handwritten.

[3] Chandwadkar, R. et al., 2013. *Comparison of edge detection techniques.* Pune, India, s.n.

[4] Otsu, N., 1979. A Threshold Selection Method from Gray-level Histogram. *IEEE Transactions on System Man Cybernetics,* Vol. SMC-9(No. 1), pp. 62-66.

[5] Santhanaprabhu, G. et al., 2014. Text Extraction and Document Image Binarization Using Sobel. *Journal of Engineering Research and Applications,* 4(5), pp. 15-21.

[6] Sobel, I. & Feldman, G., 1968. *A 3 × 3 isotropic gradient operator for image processing..* s.l., s.n.

[7] Su, B., Lu, S. & Tan, C. L., 2013. *Robust Document Image Binarization Technique for Degraded Document Images,* s.l.: s.n.

[8] Trier, O. D. & Jain, A. K., 1995. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 17(12), pp. 1191-1201.

[9] Wu, Victor, Manmatha & Raghaven, 1998. *Document image clean-up and binarization,* San Jose: s.n.