

# Assignment 2: Image Morphology

Joe Smith - EE Affiliate

## I. INTRODUCTION

This assignment looks at image morphology in the context of character recognition. We start off with the basic erosion and dilation operations and build up an advanced detection algorithm using the principle of skeletonisation and pattern comparison. We shall look at different segmentation techniques. The additional complexity of finding rotated letters should make algorithms take long to process. Noise may affect pattern recognition outcomes, this will be looked into.

## II. MORPHOLOGICAL OPERATIONS IN MATLAB

The basic morphological operations of erosion and dilation are defined in MATLAB as *imerode* and *imdilate*. Fig 1 shows an image of spade that has dilated and eroded by the same 5 pixel radius ball structuring element. The functions work by running a defined structuring element over an image and comparing pixels in the structure to a defined base point (arbitrarily the centre pixel). Erosion grows the background and erodes the foreground. Dilation is the complement of this. The table shows how these operations work.

Center Pixel Value	Pixel in Structure Window	New Pixel Value
0	0	0
0	1	0
1	0	0
1	1	1

TABLE I: Erode

Center	Pixel in Struct Window	New Pixel
0	0	0
0	1	1
1	0	1
1	1	1

TABLE II: Dilate

Two other operations, open and close will be implemented by considering

$$A \circ B = (A \ominus B) \oplus B \quad (1)$$

$$A \bullet B = (A \oplus B) \ominus B \quad (2)$$

Where  $\ominus$  and  $\oplus$  denote erosion and dilation respectively. Opening removes thin bridges in symbols and closing removes small gaps.

There is an advanced morphological function called *bwmorph* from which various distinct operations can be performed. Fig 2 shows the skeletonisation of the spade. This advanced algorithm will be implemented in a custom function to understand its operation.

## III. FINDING THE LETTER E

The main objective of this report is to detect the letter *e* in two paragraphs of text.

The initial operation must be to make our image binary. A simple threshold operation is performed at 150 levels. A later section will look at alternative methods for binarisation for it best to minimise user decisions.

Next the image is skeletonised. Different *e* are not likely to have the same pixel footprint but more likely to have the same base skeleton hence skeletonisation increases the accuracy of detection.

Initially, the known *e* skeleton must be produced. A window is manually selected to crop *e* and skeletonisation performed. This is achieved through a series of morphological operations (see [1] for a good primer). As explained, only the *imerode* and *imdilate* functions will be used hence we rewrite this.

$$S(A) = \bigcup_{k=0}^K \{(A \ominus kB) - (A \ominus kB) \circ B\} \quad (3)$$

$$= \bigcup_{k=0}^K \{(A \ominus kB) - ((A \ominus kB) \ominus B) \oplus B\} \quad (4)$$

To test correct deconstruction, a reconstruction algorithm was also implemented. This reverses the procedure and obtains the initial symbol again.

In this equation, B is the structuring element or struct. A small symmetric struct is selected to maintain the centre. The effect of different sized structs is will documented in [1].

It can be seen that *elocator* employs a better skeletonisation implementation than the initial *findthee* function. The initial implementation required a user-defined number of iterations K. The second technique looks for the skeletonisation before no are pixels left and knows this is when to terminate.

All instances of *e* from the text are inputted into the algorithm to obtain different variants of the skeleton. Performing an OR on these obtains a master group skeleton.

After the skeleton form of the input image and the symbol to detect are ready, the second part of the algorithm performs matching. A symbol sized window is panned over the image and pixel values compared. The skeletonisation symbol was modified by adding ‘don’t care’ pixels with the value 2. The algorithm compares values for 0 and 1 and increments *pass* to each successful pixel match. Successful detection is calculated by multiplying the total 0 and 1 count in the symbol times some *tolerance* percentage. We must set the tolerance of the algorithm to less than 100% in order to recognise all of the *e* for there is a noticeable range in pixel footprints throughout the text. The point at which all *e* characters are found, yet no false positives produced, varies with the image input. Hence, this must be user-defined through trial and error. For a full application, a toggle slider could be used. The use of user interaction is a bridge to the optimal solution. A computer algorithm alone is unable to distinguish between false positives and true detection. An image with the extent and location of all *e* has been overlaid on the original image in cyan in Fig 4. Here, a tolerance of 85% was found optimal.

With the larger text, 45 *e* symbols were detected (see Fig 5). 2 of these were false positives - one *c* and one *a*. 4 true *e* were not found. With a larger paragraph, there is a greater variation in the pixel footprint of the letter *e* so a less accurate recognition is expected. This was found at a tolerance of 87%. Changing this value lead to either less successful detections or more false positives.

#### IV. MEAN SHIFT VS K-MEANS

Using a segmentation technique for binarisation is more effective than relying on a manual threshold. A comparison between the mean shift algorithm and k-means is performed. In means shift, a window radius is selected. As the algorithm progresses, windows overlap and duplicate windows removed. The algorithm is non-parametric. Nothing about the size or number of clusters is assumed. In contrast, with k-means, k number of seed centroids are chosen. Clusters are assumed to be shaped elliptically and the distance between points is computed to find the new centroid position. k-means is sensitive to initialisation i.e. initial seed position can change the output. Outliers can cause problems as centroid include their presence in calculating their position. Beneficially, computationally faster than mean shift.

A *meanshift* algorithm was coded. It works by setting a window at each pixel initially. First, duplicate windows are removed using the *unique* function. Then, the algorithm proceeds to assign number to each pixel location to denote cluster membership based on a defined window *radius*. The algorithm proceeds until the windows stop moving and cluster membership is static. The statistical-toolbox function *kmeans* needed to be aided by a wrapper function for it was not written for image data sets. Reshaping and scaling was necessary in *imkmean*.

It can be seen in Fig 7 that the k-means performs more accurate segmentation than the mean-shift. This may be because of the simplistic mean calculated. The algorithm could be improved by making use of a Gaussian kernel as seen in [2].

#### V. OTHER VOWELS

To test the validity and scope of the detection method, the other vowels were detected. These symbols were produced using the same method as the *e* was and are shown in Fig 6.

It seemed poor practise to refactor the symbol detection algorithm to just deal with vowels so the algorithm was kept as is, with an optional parameter for letter colour. Five colours were built by overlaying symbols on different combinations of R, G and B channels. This proved easier than defining a colormap. The code was run five times, with different input symbols, different colours, and different tolerances. The optimal output on the large more difficult paragraph is shown in Fig 8.

The main problem with the new symbol shapes was processing the letter *i*. This has a footprint of several joined vertical pixels, a pixel gap, and another pixel. Any character with a stick component would be picked up by the algorithm. Attempts to cut down the character footprint and put more weight on finding the pixel gap above the *i* had little impact. Also, the symbol for *a* is very similar to *n* resulting in many false recognitions.

An improvement to this algorithm would be to rank each pixel by characteristic importance. For instance, the dot on the *i* with a pixel gap beneath it is essential. The bottom of the *a* is more important than its tail. This would have to be done manually and a more complex pattern matching algorithm used.

#### VI. TIMING ANALYSIS ON BRUTE FORCE ATTACK ON ROTATED LETTERS

When the letters are at some arbitrary rotation, it is harder to detect them. This algorithm applies a rotation transformation to the input symbol at 5 degree increments.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos 5n & -\sin 5n \\ \sin 5n & \cos 5n \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (5)$$

where  $-90 \geq 5n \leq 90$  and  $n \in \mathbb{Z}$

To examine computation time, use of MATLAB functions *tic* and *toc* are implemented. The software outputs the time between calling these two functions. It was seen that the tolerance effects computational time significantly. A cycle with 90% tolerance takes 7 seconds whereas one with 85% tolerance takes 1 second.

It is seen in Fig 9 that the 5 degree increments fail to detect some of the letters yet produce some false negatives. However, more correct identifications are performed than false. 1 degree increments were experimented with. These operations were far less computationally efficient and to no obvious improvement. A key problem with this rotation procedure is that the initial symbols were small low resolution skeleton objects hence rotation produces high distortions as is seen in Fig 10.

Use of a rotation invariant basis such as the wavelet basis would improve this technique. It might also be beneficial to attempt detection without skeletonisation, to maintain higher resolution.

## VII. EFFECTS OF NOISE ON ALGORITHM PERFORMANCE

Use of Gaussian noise makes detection more difficult. One way to improve this algorithm is to detect noise and perform initial opening. Eroding the image by one pixel and then dilating will keep major features but remove stray pixels. Salt & Pepper noise with a density of 0.2 was added to the large text using *imnoise*. The *e* detection algorithm was run with the same tolerance of 0.87% and the output is shown in Fig 11. In contrast, barely any of the letters have been detected. Removing this noise proves difficult as the characters have key features on a pixel scale. Note this is done using the dual *closing* as the text is black; these operation were explained earlier. Although the noise has been removed, the characters are barely recognisable. Running the algorithm detected no instances of the character *e*.

## VIII. AN ALTERNATIVE DETECTION TECHNIQUE

There are various other methods of skeletonisation implemented in the *bwmorph* function. Character recognition using non-morphological techniques are also possible. It would be beneficial to see the accuracy of detection if the skeletonisation is simply bypassed by commenting out this part of the algorithm. The thresholding will

be bypassed too. New methods must be implemented as grayscale images are now being managed. This adds an extra layer of complexity. Instead of inverting binary images with  $I = I, imcomplement$  must be used. The written rotation function is defunct but Matlab has an in-built *imrotate*. Additionally, a further tolerance was implemented to determine the maximum intensity allowed between pixels. Through this technique, two of the *e* characters were correctly found (Fig 14). However, increasing the tolerance produced more false positives especially when looking at 1 degree increments (Fig 15). From Fig 13, it must be noted that the rotated grayscale symbols obtained are of a higher quality however there is a fluctuation in intensities across the text image resulting in a trade off between the two techniques. Another reason why there is less distortion is that the *imrotate* implements interpolation on rotation which my own algorithm did not.

## IX. CONCLUSIONS

In this assignment, basic morphological operations were explored and characterised successfully. Simple pattern detection on a skeletonised image was performed to detect multiple defined symbols on two base images. The skeletonisation algorithm was correctly implemented for this task. Multiple colours were used to show the outputs of different symbols. K-means and mean-shift detection were used to segment an image and comparisons drawn. The mean-shift algorithm was shown to segment less well than the built-in k-means although it is expected the addition of kernels would improve this result. Rotated letters were successfully detected using a custom function that rotated the symbol in degrees. The addition of noise to an image was shown to make detection more difficult. The use of morphological techniques to repair noise were explored. Finally, an alternative character detection algorithm was explored bypassing morphology and making use of grayscale image data. It can be seen from this report that all core tasks and additional tasks were attempted with good rates of success.

- 
- [1] P. Maragos and R. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 1228–1244, Oct. 1986.
- [2] D. Comaniciu, P. Meer, and S. Member, "Mean Shift : A Robust Approach Toward Feature Space Analysis," vol. 24, no. 5, pp. 603–619, 2002.



FIG. 1: Here is the spade suit dilated and then eroded by the same structural element

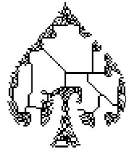


FIG. 2: Here is a skeletonisation carried out using the *bw-morph* command

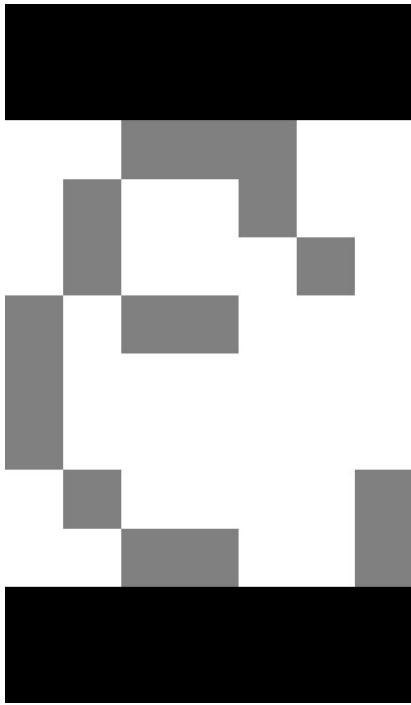


FIG. 3: Here is the super *e* symbol made from combining the skeletonisation of all instances. White pixels are don't care conditions

Let freedom ring from every hill and molehill of Mississippi and every mountainside.

FIG. 4: Here are all instances of the letter *e* correctly located on the small paragraph

And if America is to be a great nation, this must become true.  
So let freedom ring from the hilltops of New Hampshire. Let freedom ring from the mighty mountains of New York.

Let freedom ring from the heightening Alleghenies of Pennsylvania.

Let freedom ring from the snow-capped Rockies of Colorado.

Let freedom ring from the curvacious slopes of California.

But not only that, let freedom, ring from Stone Mountain of Georgia.

FIG. 5: Here most instances of letter *e* are correctly located on the big paragraph

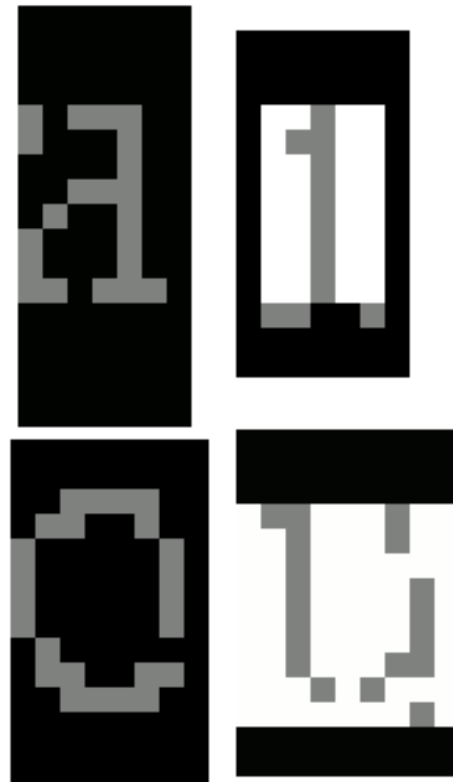


FIG. 6: Here are the symbols of the other vowels used

And if America is to be a great nation, this must become true.  
 So let freedom ring from the hilltops of New Hampshire. Let  
 freedom ring from the mighty mountains of New York.

Let freedom ring from the heightening Alleghenies of  
 Pennsylvania.

Let freedom ring from the snow-capped Rockies of Colorado.

Let freedom ring from the curvacious slopes of California.

But not only that, let freedom, ring from Stone Mountain of  
 Georgia.

And if America is to be a great nation, this must become true.  
 So let freedom ring from the hilltops of New Hampshire. Let  
 freedom ring from the mighty mountains of New York.

Let freedom ring from the heightening Alleghenies of  
 Pennsylvania.

Let freedom ring from the snow-capped Rockies of Colorado.

Let freedom ring from the curvacious slopes of California.

But not only that, let freedom, ring from Stone Mountain of  
 Georgia.

FIG. 7: Here is a comparison of segmentation algorithms showing *meanshift* above and *k-means* below

And if America is to be a great nation, this must become true.  
 So let freedom ring from the hilltops of New Hampshire. Let  
 freedom ring from the mighty mountains of New York.

Let freedom ring from the heightening Alleghenies of  
 Pennsylvania.

Let freedom ring from the snow-capped Rockies of Colorado.

Let freedom ring from the curvacious slopes of California.

But not only that, let freedom, ring from Stone Mountain of  
 Georgia.

FIG. 8: The algorithm has been applied five times to obtain some vowel recognition

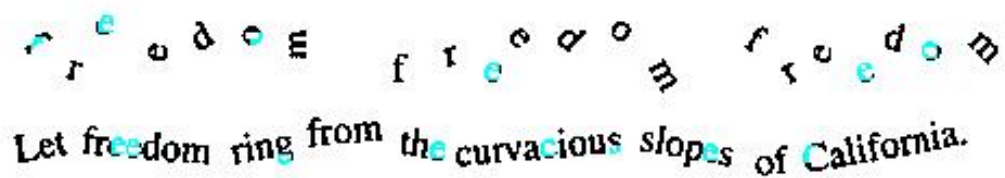
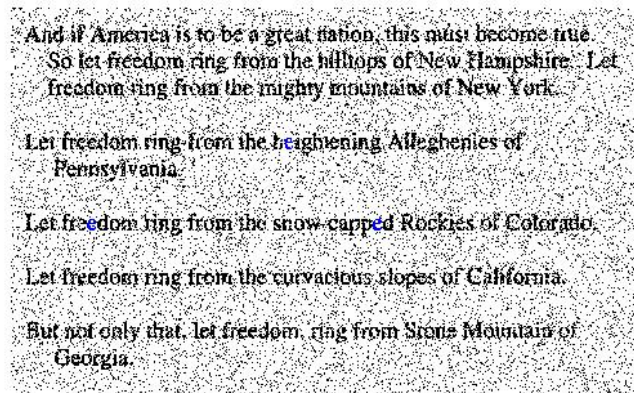


FIG. 9: The rotational algorithm has some success in 5 degree increments

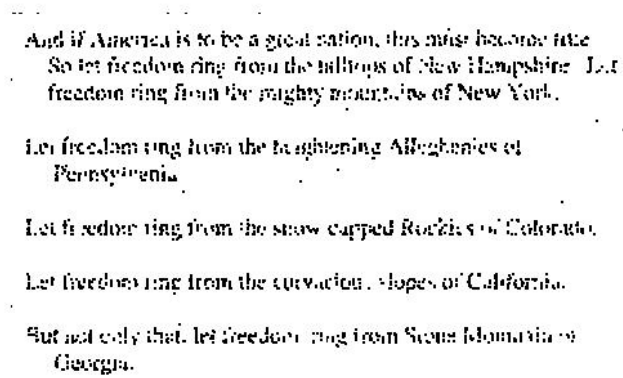


FIG. 10: Rotation introduces distortion in low quality symbols



And if America is to be a great nation, this must become true  
So let freedom ring from the hilltops of New Hampshire. Let  
freedom ring from the mighty mountains of New York.  
Let freedom ring from the hughening Alleghenies of  
Pennsylvania.  
Let freedom ring from the snow-capped Rockies of Colorado.  
Let freedom ring from the curvacious slopes of California.  
But not only that, let freedom ring from Stone Mountain of  
Georgia.

FIG. 11: The algorithm suffers heavily when the input is noise affected



And if America is to be a great nation, this must become true  
So let freedom ring from the hilltops of New Hampshire. Let  
freedom ring from the mighty mountains of New York.  
Let freedom ring from the hughening Alleghenies of  
Pennsylvania.  
Let freedom ring from the snow-capped Rockies of Colorado.  
Let freedom ring from the curvacious slopes of California.  
But not only that, let freedom ring from Stone Mountain of  
Georgia.

FIG. 12: The text is too low resolution to remove noise successfully





FIG. 13: Using grayscale symbols produced less distortion on rotation

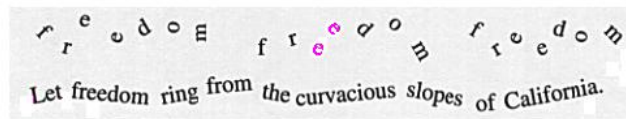


FIG. 14: Here is character recognition without the morphology stage

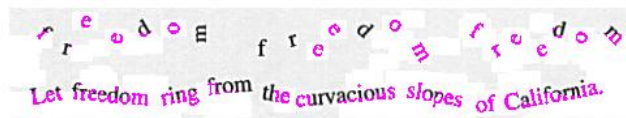


FIG. 15: Increasing the rotation angle produced many false positives, especially in the close together text