# Lab 2: Optical Character Recognition

## Matched Spatial Filter

ECE 4310

Brian Guzman

September 18, 2019

**Introduction:**

The purpose of this lab was to write a program that could recognize letters in an image of text using a matched spatial filter. In this particular case, the letter to be recognized was the letter 'e'. To achieve letter recognition the program was to take in as inputs the image to be parsed, a template image, and a ground truth file. The template image was a 9 x 15 image containing an image of the letter 'e'. The ground truth file contained the pixel coordinates corresponding to the centers of all letters in the input image.

To implement this program, the following steps were necessary:

1. Read input image, template, and ground truth
2. Calculate matched spatial filter (MSF) using the equation below:
3. Normalize the MSF image to 8 bits
4. Loop through different thresholds to find best T
    a. Threshold at T the normalized MSF image to create a binary image.
    b. Loop through the ground truth letter locations
        i. Check a 9 x15 pixel area centered at the ground truth location. If any pixel in the MSF image is greater than the threshold, consider the letter "detected". If none of the pixels in the 9 x 15 area are greater than the threshold, consider the letter "not detected"
    c. Categorize and count the detected letters as FP ("detected" but the letter is not 'e') and TP ("detected" and the letter is 'e')
    d. Output the total FP and TP for each T
5. Plot ROC curve for different FPR and TPR

**Calculating MSF**

Once the files were read in, the next step was to calculate the matched spatial filter image. This was done using the following equation:

$$MSF[r,c] = \sum_{dr=-Wr/2}^{+Wr/2} \sum_{dc=-Wc/2}^{+Wc/2} [I[r+dr, c+dc] * T[dr+Wr/2, dc+Wc/2]$$

Equation 1: MSF equation

The result of the convolution was the following image:

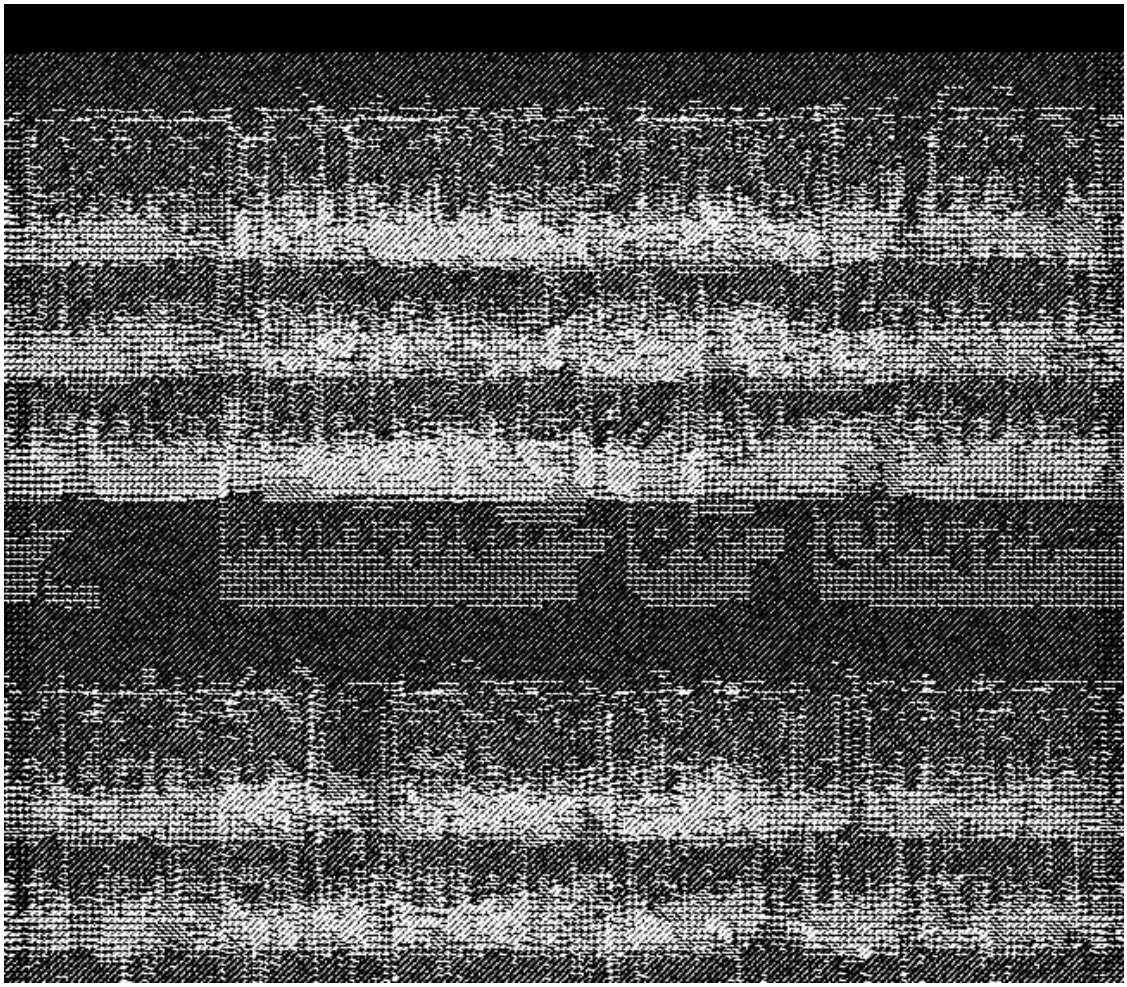

Figure 1: MSF output image

**Normalization**

The next step was to normalize the output MSF image since it was not 8-bits. To do so the maximum and minimum values of the MSF image were calculated. Once these two values were known, the following equation was used to calculate the new value of the pixels using NEWMAX = 255 and NEWMIN = 0:

$$I_N = (I - \text{Min})\frac{\text{newMax} - \text{newMin}}{\text{Max} - \text{Min}} + \text{newMin}$$

Equation 2: Normalization Equation

The output of this calculation was:

Figure 2: Normalized Image

**ROC Curve**

To obtain the ROC curve, the image was put through different thresholds (T). In this case the range of T was 0 – 255 incrementing by 5 at each iteration. Each time through the loop a new threshold output image was created where the pixel values were 0 or 255. If the value of the normalized image was equal to or greater than T the value of the threshold image was set to 255 otherwise it was set to 0. Once the threshold image was obtained another loop would check if the pixels whose value was set to 255 matched the coordinates of a known letter ‘e’ using the ground truth file. The values TP, TN, FP, and FN were calculated via the following criteria. If a pixel had a value of 255 and coordinates pointed to an ‘e’ in the ground truth file, TP was incremented. If the coordinates pointed to a different letter FP was incremented. If a pixel with value 0 pointed to a letter ‘e’, FN was incremented. If the coordinates pointed to a different letter, then TN was incremented. ROC’s are plots of false positive rate (FPR) vs true positive rate (TPR). To calculate these values, the following equations were used:

$$TPR = \frac{TP}{TP+FN} \quad FPR = \frac{FP}{FP+TN}$$

Equations 3 and 4: TPR and FPR

The resulting ROC curve was the following:

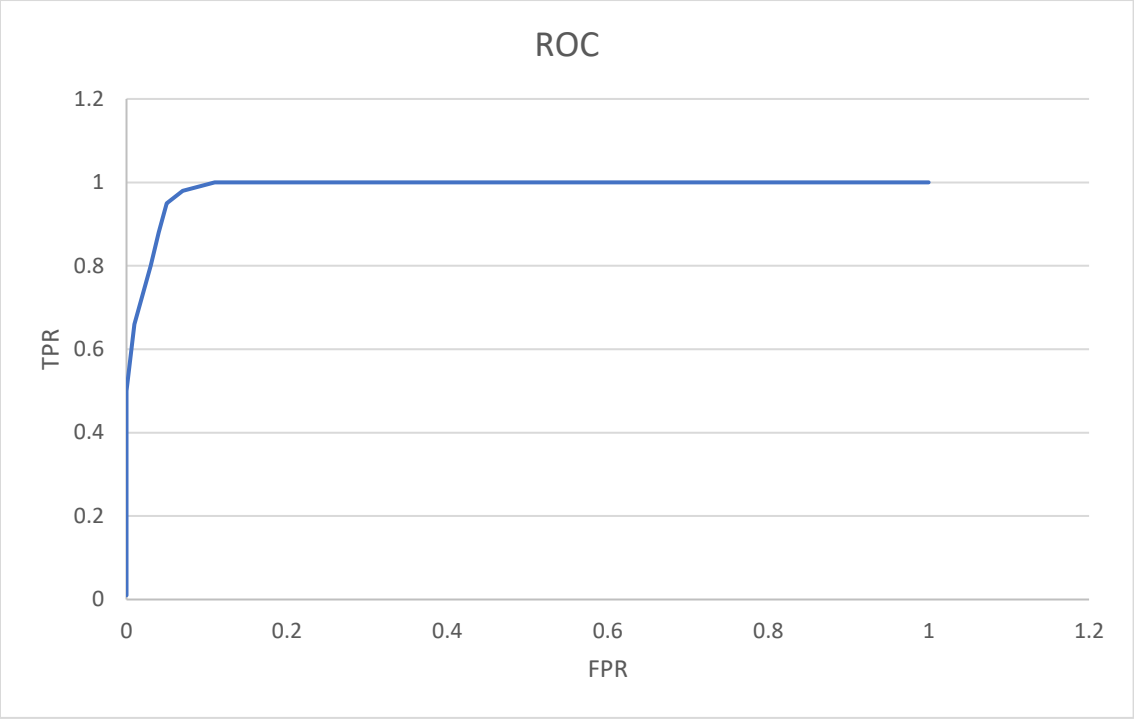


Figure 3: ROC Curve

**Results/Conclusion:**

The best value of T for an image is where the ROC curve is closest to a value of 1 on the TPR axis. In this case the best value of T fell between 203 and 207. The following output image is the character recognition at T = 205. At this value of the T, TP = 148 and FP = 82.

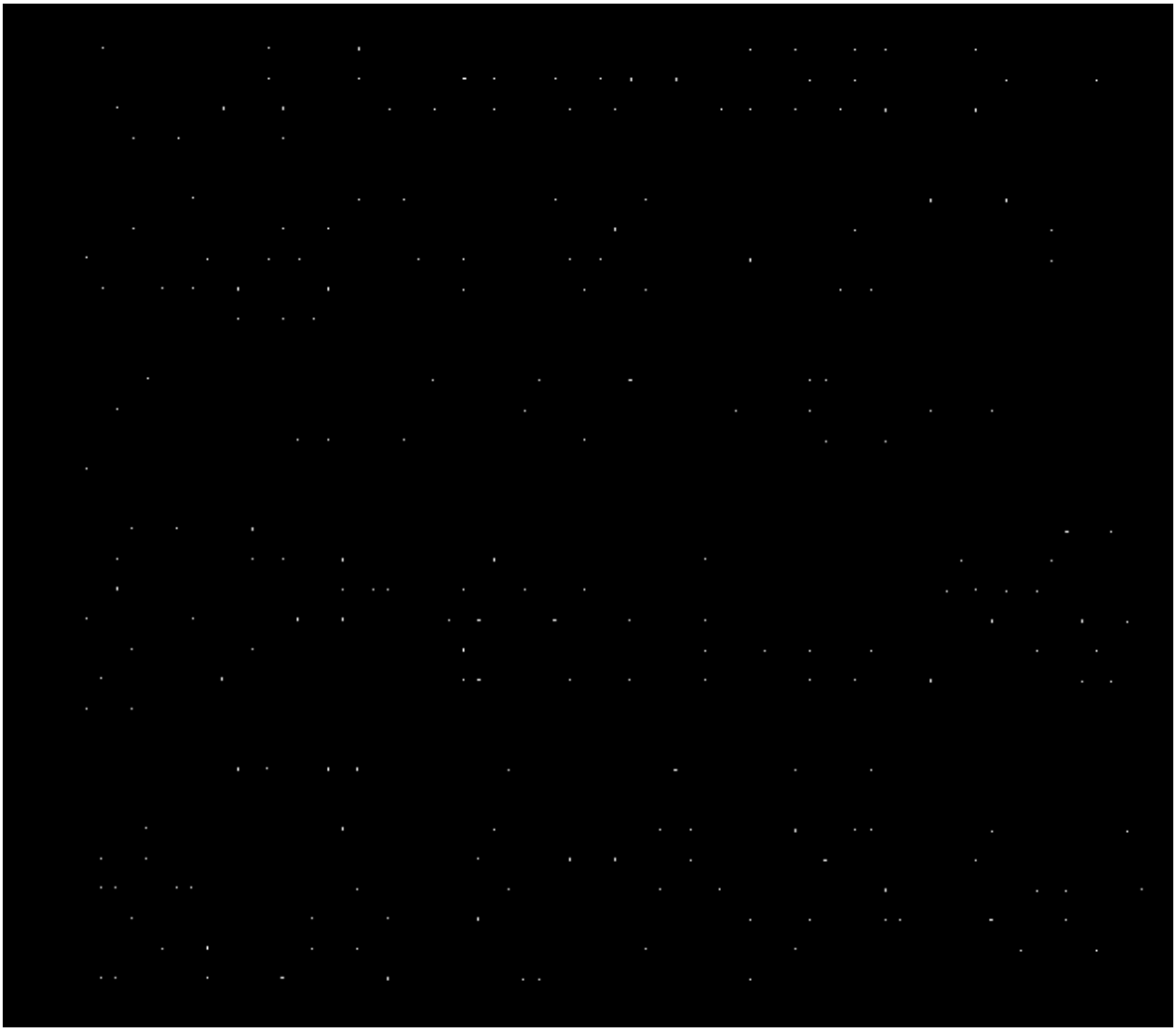

Figure 4: Character Recognition at T = 205

## Appendix A: ROC Table

| Threshold | TP | FP | TN | FN | TPR | FPR | PPV |
|---|---|---|---|---|---|---|---|
| 0 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 5 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 10 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 15 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 20 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 25 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 30 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 35 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 40 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 45 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 50 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 55 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 60 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 65 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 70 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 75 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 80 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 85 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 90 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 95 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 100 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 105 | 151 | 1111 | 0 | 0 | 1 | 1 | .0.88 |
| 110 | 151 | 1110 | 1 | 0 | 1 | 1 | .0.88 |
| 115 | 151 | 1107 | 4 | 0 | 1 | 1 | .0.88 |
| 120 | 151 | 1101 | 10 | 0 | 1 | 0.99 | .0.88 |
| 125 | 151 | 1094 | 17 | 0 | 1 | 0.98 | .0.88 |
| 130 | 151 | 1073 | 38 | 0 | 1 | 0.97 | .0.88 |
| 135 | 151 | 1047 | 64 | 0 | 1 | 0.94 | .0.87 |
| 140 | 151 | 1010 | 101 | 0 | 1 | 0.91 | .0.87 |
| 145 | 151 | 957 | 154 | 0 | 1 | 0.86 | .0.86 |
| 150 | 151 | 889 | 222 | 0 | 1 | 0.8 | .0.85 |
| 155 | 151 | 791 | 320 | 0 | 1 | 0.71 | .0.84 |
| 160 | 151 | 675 | 436 | 0 | 1 | 0.61 | .0.82 |
| 165 | 151 | 597 | 514 | 0 | 1 | 0.54 | .0.80 |
| 170 | 151 | 535 | 576 | 0 | 1 | 0.48 | .0.78 |
| 175 | 151 | 468 | 643 | 0 | 1 | 0.42 | .0.76 |
| 180 | 151 | 386 | 725 | 0 | 1 | 0.35 | .0.72 |
| 185 | 151 | 308 | 803 | 0 | 1 | 0.28 | .0.67 |
| 190 | 151 | 231 | 880 | 0 | 1 | 0.21 | .0.60 |
| 195 | 151 | 169 | 942 | 0 | 1 | 0.15 | .0.53 |
| 200 | 151 | 124 | 987 | 0 | 1 | 0.11 | .0.45 |
| 205 | 148 | 82 | 1029 | 3 | 0.98 | 0.07 | .0.36 |
| 210 | 143 | 59 | 1052 | 8 | 0.95 | 0.05 | .0.29 |
| 215 | 133 | 43 | 1068 | 18 | 0.88 | 0.04 | .0.24 |
| 220 | 121 | 30 | 1081 | 30 | 0.8 | 0.03 | .0.20 |
| 225 | 99 | 14 | 1097 | 52 | 0.66 | 0.01 | .0.12 |
| 230 | 75 | 5 | 1106 | 76 | 0.5 | 0 | .0.06 |
| 235 | 48 | 1 | 1110 | 103 | 0.32 | 0 | .0.02 |
| 240 | 31 | 0 | 1111 | 120 | 0.21 | 0 | .0.00 |
| 245 | 16 | 0 | 1111 | 135 | 0.11 | 0 | .0.00 |
| 250 | 5 | 0 | 1111 | 146 | 0.03 | 0 | .0.00 |
| 255 | 1 | 0 | 1111 | 150 | 0.01 | 0 | .0.00 |

## Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>


int main(int argc, char *argv[])
{
	//VARIABLES
	FILE *inputFile, *templateFile, *outputFile;
	char header[80], templateHeader[80];
	int inputRows, inputCols, inputMax;
	int templateRows, templateCols, templateMax;
	unsigned char *imageIn, *templateImage, *outputImage;

	//VERIFY COMMAND LINE ARGUMENTS ARE PRESENT
	if (argc != 4)
	{
		printf("Usage: ./convolution [inputImage] [templateImage]
[groundTruth]\n");
		exit(0);
	}

	//OPEN INPUT IMAGE
	inputFile = fopen(argv[1], "rb");
	templateFile = fopen(argv[2], "rb");
	if (inputFile == NULL)
	{
		printf("Unable to open file %s\n", argv[1]);
		exit(0);
	}
	if (templateFile == NULL)
	{
		printf("Unable to open file %s\n", argv[2]);
	}

	//PARSE HEADERS
	fscanf(inputFile, "%s %d %d %d", header, &inputCols, &inputRows, &inputMax);
	fread(&header[79], 1, 1, inputFile);

	fscanf(templateFile, "%s %d %d %d", templateHeader, &templateCols, &templateRows,
&templateMax);
	fread(&templateHeader[79], 1, 1, templateFile);

	//ALLOCATE MEMORY FOR INPUT IMAGES AND OUTPUT IMAGES
	imageIn = (unsigned char *)calloc(inputRows * inputCols, sizeof(unsigned char));
	templateImage = (unsigned char *)calloc(templateRows * templateCols,
sizeof(unsigned char));
	if (imageIn == NULL || templateImage == NULL)
	{
		printf("Unable to allocate sufficient memory\n");
		exit(0);
	}

	//READ IMAGE DATA
	fread(imageIn, 1, inputRows*inputCols, inputFile);
```

```
	fclose(inputFile);
	fread(templateImage, 1, templateRows * templateCols, templateFile);
	fclose(templateFile);

	//FIND ZERO MEAN TEMPLATE
	int sum = 0, count, mean;
	int *zeroMeanTemplate;

	zeroMeanTemplate = (int *)calloc(templateRows * templateCols, sizeof(int));

	for (count = 0; count < (templateRows * templateCols); count++)
	{
		sum += templateImage[count];
	}

	mean = sum / (templateRows * templateCols);

	for (count = 0; count < (templateCols * templateRows); count++)
	{
		zeroMeanTemplate[count] = templateImage[count] - mean;
	}

	//MSF ALGORITHM
	int r, c, r2, c2;
	int *msfImage;

	msfImage = (int *)calloc(inputCols * inputRows, sizeof(int));

	for (r = 7; r < inputRows - 7; r++)
	{
		for (c = 4; c < inputCols - 4; c++)
		{
			sum = 0;
			for (r2 = -7; r2 < templateRows - 7; r2++)
			{
				for (c2 = -4; c2 < templateCols - 4; c2++)
				{
					sum += zeroMeanTemplate[(templateCols * (r2 + 7)) + (c2
+ 4)] * imageIn[(inputCols * (r + r2)) + (c + c2)];
				}
			}
			msfImage[(inputCols * r) + c] = sum;
		}
	}

	//CALCULATE MAX AND MIN OF MSF OUTPUT
	int max;
	int min;

	max = min = msfImage[0];

	for (count = 1; count < inputRows * inputCols; count++)
	{
		if (msfImage[count] > max)
		{
			max = msfImage[count];
		}
		if (msfImage[count] < min)
```

```
			{
				min = msfImage[count];
			}
		}

		//NORMALIZE MSF IMAGE
		outputImage = (unsigned char *)calloc(inputRows * inputCols, sizeof(unsigned
char));

		for (count = 0; count < inputRows * inputCols; count++)
		{
			outputImage[count] = (msfImage[count] - min)*(255) / (max - min);
		}

		outputFile = fopen("normalized.ppm", "w");
		fprintf(outputFile, "P5 %d %d 255\n", inputCols, inputRows);
		fwrite(outputImage, inputCols * inputRows, 1, outputFile);
		fclose(outputFile);

		//CALCULATE ROC AND DETERMINE BEST THRESHOLD
		FILE *groundFile, *rocFile;
		int threshold, match, found;
		int TP, TN, FP, FN;
		char current[2];
		char templateChar[2];
		int groundRow, groundCol;
		unsigned char *thresholdOuput;

		thresholdOuput = (unsigned char *)calloc(inputCols * inputRows, sizeof(unsigned
char));
		strcpy(templateChar, "e");
		TP = TN = FP = FN = 0;

		rocFile = fopen("ROC.csv", "w");
		fprintf(rocFile, "Threshold,TP,FP,TN,FN,TPR,FPR,PPV\n");
		groundFile = fopen(argv[3], "r");
		if (groundFile == NULL)
		{
			printf("Error could not read file\n");
			exit(0);
		}


		for (count = 0; count < 256; count += 5)
		{
			TP = TN = FP = FN = 0;
			threshold = count;
			for (c2 = 0; c2 < inputCols * inputRows; c2++)
			{
				if (outputImage[c2] >= threshold)
				{
					thresholdOuput[c2] = 255;
				}
				else
				{
					thresholdOuput[c2] = 0;
				}
			}
```

```
		while (fscanf(groundFile, "%s %d %d\n", current, &groundCol, &groundRow) !=
EOF)
		{
			for (r = groundRow - 7; r <= groundRow + 7; r++)
			{
				for (c = groundCol - 4; c <= groundCol + 4; c++)
				{
					if (thresholdOuput[(r*inputCols) + c] == 255)
					{
						found = 1;
					}
				}
			}

			match = strcmp(current, templateChar);
			if (found == 1 && match == 0)
				TP++;
			if (found == 1 && match != 0)
				FP++;
			if (found == 0 && match == 0)
				FN++;
			if (found == 0 && match != 0)
				TN++;
			found = 0;
		}
		//WRITE OUTPUT DATA FILE
		fprintf(rocFile, "%d,%d,%d,%d,%d,%.2F,%.2F,.%.2F\n", threshold, TP, FP, TN,
FN,TP/(double)(TP+FN),FP/(double)(FP+TN),FP/(double)(TP+FP));
		rewind(groundFile);
	}
	fclose(groundFile);
	fclose(rocFile);
```