Lab 2 - Optical Character Recognition

John Lawler

September 15, 2020

Introduction

The purpose of this lab is to teach students rudimentary character recognition when provided a template file and an image file containing text of some kind. This involves several mathematical operations performed on the given template and the image to provide a good estimate of where each true character location is in the given image of text. A ground truth file is used to verify the program's calculations. This file is also used to calculate the false and true positives that will later be plotted on an ROC curve.

Functionality

The process used to find the selected characters is matched spatial filtering (MSF). The template image itself must first be altered prior to convolving with the given image. The zero-mean-center (ZMC) of the template is found first by finding the mean of the template (summing all pixels and dividing by area), then subtracting the mean from each pixel in the original template. The code for that process is shown below in Figure 1.

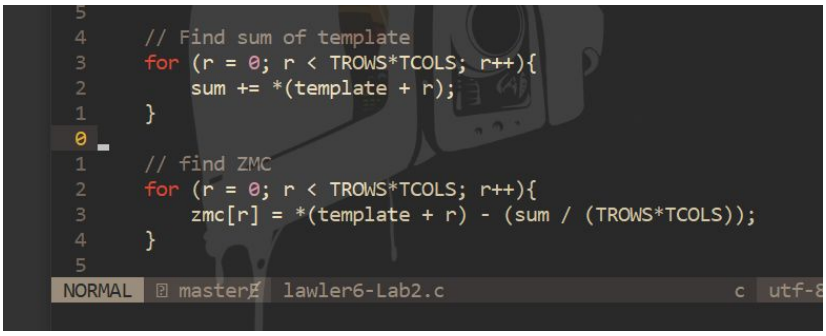


Figure 1: Zero-Mean-Center functionality

The ZMC template is now ready to convolve with the original input image. The original image is convolved with the ZMC template to produce the resultant MSF. However, since the ZMC template must be stored as an int-pointer to handle integer based calculations shown in Figure 1, the resultant MSF is also derived from integers. Prior to printing out the resultant MSF, it must be normalized to within 8 bits. The maximum and minimum pixels are found in the integer based MSF and are used for scaling every pixel in the MSF down to a range of 0 to 255.

The convolution is shown in code below. The maximum and minimum values of the MSF prior to normalization are found during the convolution of the two images to reduce unnecessary parsing later on to determine the values. Since every pixel value is calculated in the convolution and placed in the resultant MSF array, it's convenient to mark each new pixel value for the running maximum and minimum values during convolution.

```
24      for (r = 7; r < IROWS-7; r++) {
25          for (c = 4; c < ICOLS-4; c++) {
26              pixelVal = 0;
27              // Inside loop for handling the 7x7 around indexed pixel to convolve
28              for (tr = -7; tr <= 7; tr++) {
29                  for (tc = -4; tc <= 4; tc++)
30                      pixelVal += image[(r+tr)*ICOLS + (c+tc)]*zmc[TCOLS*(tr+7) + (tc+4)];
31              }
32              // Determmine min and max
33              msf[ICOLS*r + c] = pixelVal;        // appropriate msf pixel value
34              if (r==4 && c==7) max=min=pixelVal;
35              if (pixelVal < min) min = pixelVal; // Find min out of all msf values
36              if (pixelVal > max) max = pixelVal; // Find max out of all msf values
37          }
38      }
NORMAL  master  lawler6-Lab2.c                c  utf-8[unix]  36%  58/157  :13  [4]trai
```

Figure 2: 2D Convolution of the ZMC Template and Input Image

The resultant normalized MSF image is shown below in Figure 3 after the above process is completed. A binary file is also created to represent where each detected letter "e" is located according to the matched spatial filter after applying a user given threshold to the MSF. The binary file either shows a value of 255 (white) for each pixel that indicated an implied detection, and 0 (black) otherwise. The binary file is also shown below in Figure 4.

Figure 3: Normalized Matched Spatial Filtering of Input Image

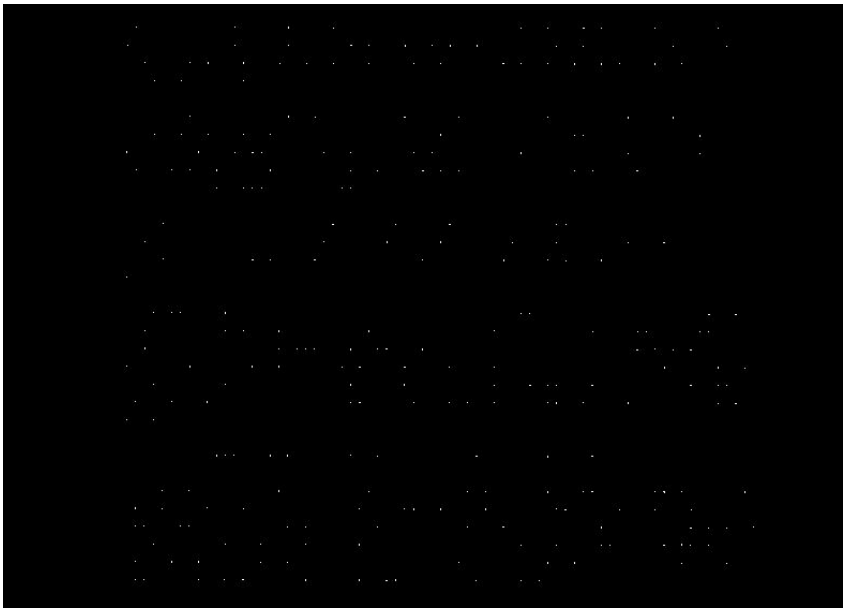


Figure 4: Binary File

The user provided threshold value sets a cutoff limit in the normalized MSF image and only allows pixel values greater than the given threshold to “leak” through. In Figure 5, a threshold of 200 was chosen as an example. Figure 5 shows the execution of this.

```
34
35      // Set threshold for new binary file
36      for (i = 0; i < IROWS*ICOLS; i++)
37          if (output[i] > threshold) binaryImage[i] = 255;
38          else binaryImage[i] = 0;
39
NORMAL  master  lawler6-Lab2.c                    c  utf-8[unix]
```

Figure 5: Output Threshold Cutoff

Once the threshold has been set for each pixel in the binary file, the binary file is then used to verify a set of ground truth values from a provided text document. Choosing one ground truth letter and location at a time, a 15 x 9 (row by column) rectangle is drawn around each ground truth location, and within each box around the true location each pixel value in the binary file is iterated over. If any of the 135 pixels surrounding the ground truth location are shown to be 255, then a detection occurs and the program determines whether the detected value was a true positive, true negative, false positive, or false negative. The program is tested for 30 intervals, and each data point is plotted in excel at each specified threshold from a range of 190 to 220. A discussion of the results follows.

Discussion of Results

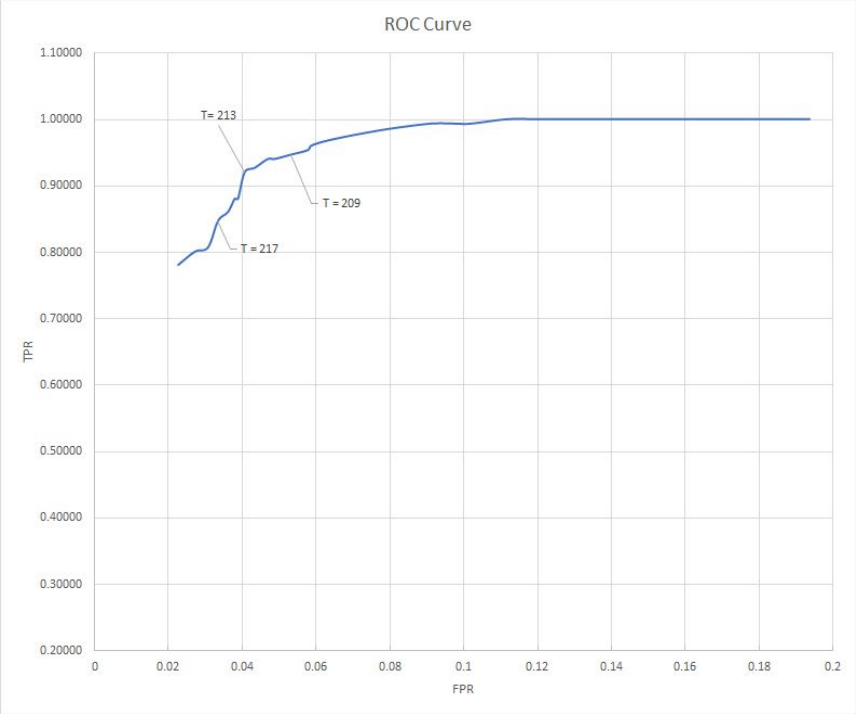
The Data and Tables section provides the graph for the receiver operating characteristic (ROC) curve and the data table used to plot the points. To achieve optimal results, where detection only detects true positives and no false positives, the key is to choose a threshold value that is closest to the point (0, 1). In most all cases this is almost impossible to achieve, so the next best choice is to choose a value on the curve that is closest to the upper left corner of the graph. Depending on what type of testing is involved, one may want to choose a threshold value that is less or greater than the optimal corner point. Choosing a value greater than the optimal corner point on a curve can ensure that every case is indeed caught and there are no false negatives; however, this also results in increasing false positives. Choosing a value further left on the graph, on the other hand, holds the opposite effect. By increasing the threshold value, one can soon reach values that provide no false positives, however not all true positives might be detected.

In the case of this lab, the optimal threshold value chosen is a threshold of 213. This value is decently close to the ideal theoretical point of (0, 1); it minimizes the false positives well enough and covers almost all of the true positives. From Table 1, the false positive value at a threshold of 213 is 45 and a true positive value of 139. That leaves 12 unaccounted for detections, i.e. false negatives.

Conclusion

Overall, this program provides a good foundation for basic character recognition when using a template file to detect with and a given input image file to search through. By convolving the two images together, information can be extracted from each rectangular window of pixels to find the right matches throughout. There are various ways to accomplish optical character recognition and template matching is one of many; the usage of the ROC curve is advantageous towards finding an optimal threshold value in any scenario to select whether the goal is to maximize true positives or minimize false positives.

Data Tables and Graphs



Graph 1: ROC Curve of TPR vs FPR

Threshold	FP	TP	FN	TN	TPR	FPR
190	215	151	0	896	1.00000	0.19352
191	200	151	0	911	1.00000	0.18002
192	187	151	0	924	1.00000	0.16832
193	180	151	0	931	1.00000	0.16202
194	169	151	0	942	1.00000	0.15212
195	162	151	0	949	1.00000	0.14581
196	148	151	0	963	1.00000	0.13321
197	141	151	0	963	1.00000	0.12772
198	133	151	0	978	1.00000	0.11971
199	124	151	0	987	1.00000	0.11161
200	113	150	1	998	0.99338	0.10171
201	108	150	1	1003	0.99338	0.09721
202	101	150	1	1010	0.99338	0.09091
203	90	149	2	1021	0.98675	0.08101
204	82	148	3	1029	0.98013	0.07381
205	75	147	4	1036	0.97351	0.06751
206	69	146	5	1042	0.96689	0.06211
207	65	145	6	1046	0.96026	0.05851
208	64	144	7	1047	0.95364	0.05761
209	59	143	8	1052	0.94702	0.05311
210	54	142	9	1057	0.94040	0.0486
211	52	142	9	1059	0.94040	0.0468
212	48	140	11	1063	0.92715	0.0432
213	45	139	12	1066	0.92053	0.0405
214	43	133	18	1068	0.88079	0.0387
215	42	133	18	1069	0.88079	0.0378
216	40	130	21	1071	0.86093	0.036
217	37	128	23	1074	0.84768	0.0333
218	34	122	29	1077	0.80795	0.0306
219	30	121	30	1081	0.80132	0.027
220	25	118	33	1086	0.78146	0.0225

Table 1: Data from 30 tests of different thresholds