Vector Templates for Handprinted Symbol Recognition

J.R. Parker
Laboratory for Computer Vision
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

The variation found in handprinted symbols makes it difficult to use template matching methods. One approach is to use an adaptable template, one that can be scaled and rotated easily. Here we explore the use of vector templates, and apply the solution to handprinted digits, map symbols, and music notation.

Template matching techniques in many forms have been applied to the problem of recognizing handprinted digits using a computer[BROW88, CAI91, GADE91,MORI84]. The basic idea is that each digit has a particular shape that can be captured in a small set of models, usually stored as raster images. An incoming (unknown) digit, also in raster form, is compared against each template, and the one that matches most closely is selected as belonging to the same digit class as the unknown. The system that performs template matching can be 'taught' new forms by simply adding new templates to its set. This is often done when an incoming digit can't be identified well enough; a human classifies it, and the unknown image can be added as a new template if desired.

Once the learning phase is complete, template based recognition methods work quite well for machine printed characters[SRIH93]. These are uniform in size, shape, and orientation, and pre-processing methods can be devised that produce quite recognizable characters for any particular document or set of documents that were created in the same way. On the other hand, characters printed by a human show a large degree of variation in shape, size, orientation, and grey level intensity, even in sets of characters printed by the same person. This variation mitigates against the use of templates.

One solution is to represent the template digits as vectors. This is commonly done in computer typography systems, where the fonts are stored in vector form [KNUT86]. This permits easy scaling and rotation, allowing one set of characters to be used for all sizes, plus bold and italic forms. The fonts were originally produced, painstakingly and with human assistance, so as to be of high visual quality when scaled to large sizes and thick line widths. Although fonts are often stored as outlines, it seems that the vector form generally has the properties needed of a good template.

For applications in digit recognition vectors that form the skeleton of the characters are used rather than the outline. This yields a good abstraction of the shape, and permits the lines to be thickened in an arbitrary way. The templates are stored as sets of four integers: the starting and ending row and column on a standard grid. All templates have the same size: 10 by 10; this means that all coordinates in any template have an integer value between 0 and 9 inclusive. Given a scale and rotation, then, all templates in the collection would be modified in a consistent way.

There have been some efforts to normalize handprinted characters[CASE82], but these are only successful for certain types of variation. Orientation, slant, and scale can be accounted for to some extent, but other aspects, such as line thickness, have not been. Thus, either an enormous number of raster templates are needed to account for all possible variations, or standard template matching techniques fail. There is enough difference in shape between different digits to permit human recognition at high rates of success; perhaps the templates should abstract the shape more accurately than possible using a raster model, which depends on individual pixel to pixel correspondence and not more global shape properties.

1 Template Creation

A vector template can be produced using only a pencil, and perhaps some graph paper, and indeed, the first set of templates were generated in just this way. An example appears in Figure 1, which shows a template for the digit ‘3’. Figure 1a shows the vector coordinates, which were obtained manually from a line drawing of a ‘2’ on a 20x20 grid. This is drawn as lines (Figure 1b) using the original scale, and also using a new scale: 20x10 (Figure 1c). This particular template is, by itself, able to match 96% of the sample ‘3’ images encountered in our data set, when the matching method described below was used.

It is also possible to create a template from a data image, and may be desirable when starting to process data from a new source. The first step in this process is to threshold and then thin thee input image. Thinning can be done using any competent algorithm: we have used both Holt’s variation on Zhang-Suen [ZHAN84, HOLT87] and a force based [PARK94] method to yield acceptable sets of skeletal pixels. The result is a binary image in which only skeletal pixels have a value of 0; all others are 1.

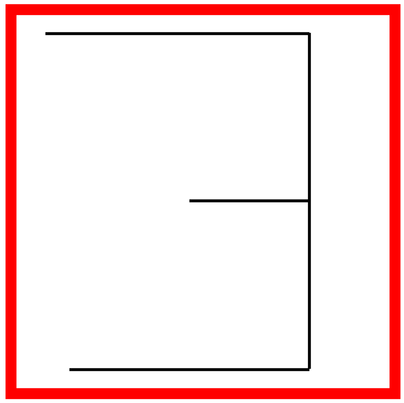
At this point the pixels are collected into sets, each representing a curve. An end pixel is found (either a pixel connected only to 1 other or, or an arbitrary starting point if no such pixel exists) and the set of pixels connected to it area saved, taking care to trace only one curve. The method described in Lam and Suen works very well here [LAM88]. Finally, a set of vectors is extracted from each curve using a recursive splitting technique, a relatively simple and common method for vectorizing small, simple images. Briefly, the endpoints of the curve are presumed initially to be the start and end points of a line, and the distances between all pixels in the curve and the mathematical line are computed. If the maximum distance for this set of pixels exceeds a pre-determined threshold then the curve is broken into two curves at the pixel having that maximum distance, and the same procedure is applied again (recursively) to each of the two curve sections. Alternatively, if the maximum distance is less than the threshold then the curve is presumed to be an approximation to a line, and the endpoint coordinates are saved as one of the vectors in the template. The coordinates are scaled down to the standard 10x10 grid after the extent of all of the vectors has been determined. Figure 2 shows this vectorization process applied to a symbol from printed music, a quarter rest to be precise, and shows both the extracted curves and the vectors produced by those curves. This particular template matched the majority of the rest images in the test data set.

The program that creates templates from images actually emits C code for the initialization of the template data array in the matching program. Source code is freely available in C.

0, 0 - 0, 9
0, 9 - 9, 9
9, 9 - 9, 0
4, 5 - 4, 9

(a)

(b)



(c)

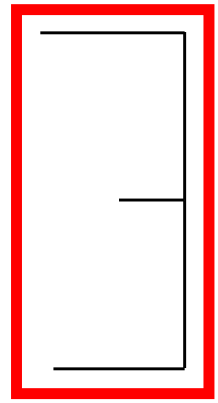


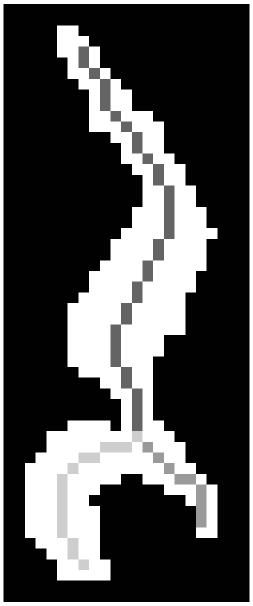
Figure 1 - An example vector template. (a) The coordinates of the vector endpoints for '3' template. (b) Vectors drawn on a 20x20 grid. (c) Vectors drawn on 20x10 grid.

2 Template Matching

Once all of the templates have been generated, and there are multiple templates for each digit, the system is ready to recognize digits. An incoming image is first pre-processed in any desired fashion and is then thresholded. The width of the lines in the image is then estimated using horizontal and vertical scans. A histogram containing the widths of the black portions of the image on all slices is produced, and the mode of this histogram has been found to be a close enough approximation to the actual line width. A better approximation can be had by computing the gradient at each pixel on the outline of the digit and finding the width of a slice though the digit in a direction perpendicular to the outline at that point, but this rarely gives a result sufficiently better to be worth the extra computation time. While the line width is being computed, the actual extent of the digit image is also found so that the templates can be scaled. This is saved as the coordinates of the upper left and the lower right pixels.

At this point the scaling factors for the templates can be computed. The templates will be scaled in the X and the Y directions independently, and the same scale factors can be applied to all templates. The factors include an adjustment that results in a correct scaling accounting for the thickness of the line. Now the template vectors are drawn into an otherwise clear image the same size as the input image, producing an initial raster template that represents the scaled skeleton. Finally, each pixel is 'grown' equally on all sides to give a line width comparable to that found in the input image. The result is a raster template with some similar properties to those found in the input image. Figure 3 illustrates the process of generating a raster template from a vector one.

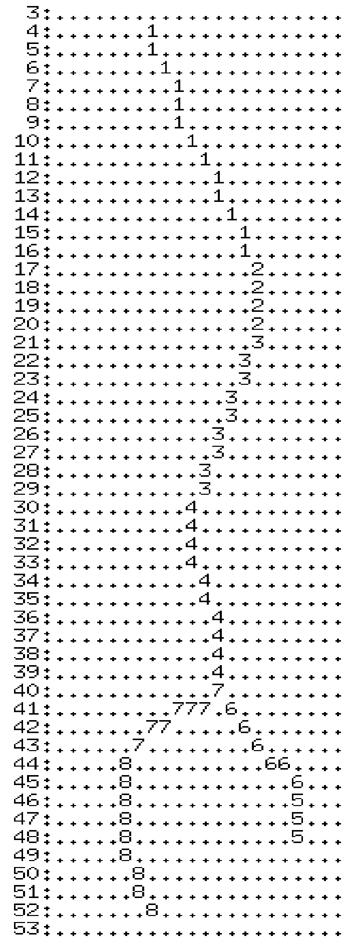
The matching process is somewhat different from that used in other template matching systems, but the goal is still to produce a measure of distance between the template and the image. The first step is to locate those pixels that are black in both images. These have a distance between them of zero, and are ignored in future processing. Next, each black pixel in the image has its nearest corresponding pixel in the template located and marked. The 8-distance between these pixels is noted, and a sum of these distances is computed. After all image pixels have been assigned corresponding pixels in the template the total distance is an initial measure of similarity. Efforts have been made to reduce the distance total by looking at pairs of corresponding pix-



(a)

1: (0,1) (2,6)
2: (2,6) (3,6)
3: (3,6) (4,3)
4: (4,3) (6,4)
5: (7,0) (9,1)
6: (7,9) (6,5)
7: (6,4) (7,0)

(b)



(c)

Figure 2 - Finding vectors in the thinned image. (a) The curves encountered (3 of them) in an image of a 1/4 rest. (b) Extracted vector coordinates. (c) Vectors (linear features) marked in the thinned image.

els and swapping those having a smaller distance after being swapped. This is a very time consuming process, and does not greatly improve the distance.

At this point a numeric value that can be used as a goodness of match metric has been found. It is normalize to a per-pixel distance and stored as a measure for the digit having the same class as the template. The class having the smallest such measure over all templates is chosen as the class of the input digit image. Figure 4 shows the overlapping pixels and a distance map for the example begun in Figure 3.

There is a problem with using this method to match a '1', or any other object that has an extreme ratio of height to width. The problem is that the vectors for almost any template will match, after having been scaled to fit. For example, the sides of a '0' will be brought together and the central hole will be filled in due to the width of the two lines. When recognizing '1' digits we used one template for those cases where the image is sufficiently wide (2% of the sample) and a combination of aspect ratio, line width VS image width, and eigenvalues of the moment matrix for the rest of the cases. Once the '1' digits were recognized, templates were used in all of the remaining cases.

3 Implementation and Results

The software implementing this method is written in C for both a Sun SPARC II workstation and an IBM PC; both versions make use of the Alpha Vision System as a library. The time needed for pre-processing varies depending on the kind of image acquired, but the template scaling and drawing takes 0.019 seconds per template. Since there are 28 different templates for digits, the total time needed to create the raster templates in this case is 0.55 seconds. This could easily be done in parallel which would reduce the real time needed. Pixel pairing and matching takes and average of 0.015 seconds per template, or 0.84 seconds per input image. The average time to recognize a digit without using parallel processing is 4.4 seconds on a Sun SPARCstation 2, and about twice as long on the PC. Recognition rates are relatively good, although at this time there is only a little data available for testing (2000 digits). The results are:

	0	1	2	3	4	5	6	7	8	9
Correct	99%	94%	98%	96%	94%	92%	90%	93%	95%	92%

Data was obtained from various sources, including ICDAR [JTCO93] and scanned locally obtained documents. There are multiple templates for each digit, but not necessarily the same number. The current digit system uses:

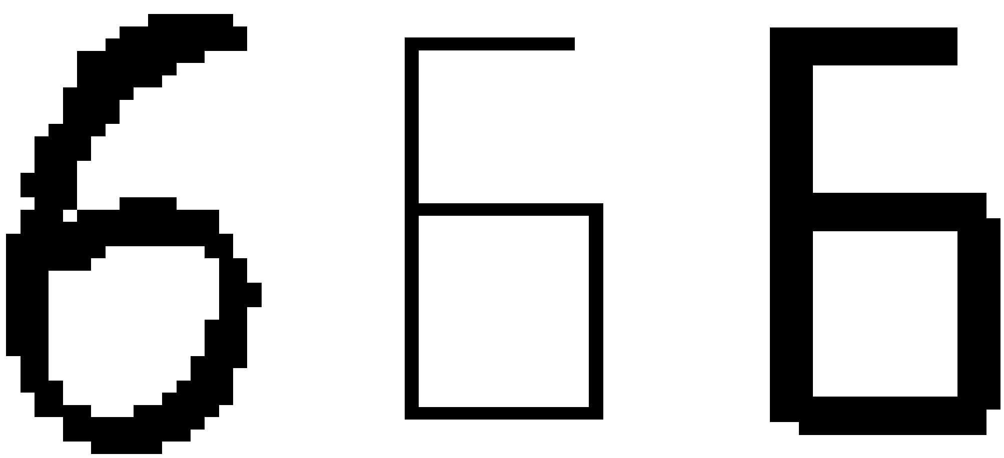
Table 1: Number of templates per digit

0	1	2	3	4	5	6	7	8	9
1	1	3	1	5	3	2	4	4	4

As an ultimate component of a larger parallel system, which naturally needs a high recognition rate, this method shows great potential. In fact, this digit classification method was originally designed to be a part of a parallel recognizer using many diverse algorithms. The vector template technique has been used to classify characters, map symbols, and symbols extracted from printed music.

4 References

[BROW88] Brown, R.M., Fay, T.H., and Walker, C.L., **Handprinted Symbol Recognition System**, *Pattern Recognition*, Vol. 21 No. 2, 1988. pp 91-118.



(a) (b) (c)

Figure 3 - Matching using a vector template. (a) Input digit image (to be matched). (b) Scaled vector template. (c) Thickened vector template (not a good match here).

[CAI91] Cai, Z., **A Handwritten Numeral Recognition System Using a Multi-microprocessor**, *Pattern Recognition Letters*, 12, 1991. pp 503-509.

[CASE70] Casey, R.G., **Moment Normalization of Handprinted Characters**, *IBM Journal of Research and Development*, Sept. 1970. pp 548-557.

[GADE91] Gader, P. et al, **Recognition of Handwritten Digits Using Template and Model Matching**, *Pattern Recognition*, Vol. 24 No. 5, 1991. pp 421-431.

[HOLT87] Holt, C.M. et al, **An Improved Parallel Thinning Algorithm**, *CACM* Vol 30 No. 2, 1987. pp 156-160.

[JTCO93] Japanese Technical Committee for Optical Character Recognition, **ETL Character Database**, *ICDAR '93*, Tsukuba, Japan, October, 1993.

[KNUT86] Knuth, D.E., **Computer Modern Typefaces**, Volume E of *Computers & Typesetting*, Addison-Wesley, Reading, MA. 1986.

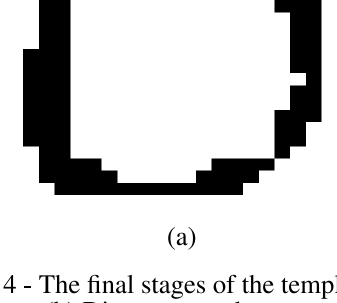
[LAM88] Lam, L. and Suen, C.Y., **Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Zip-Code Numbers**, *Pattern Recognition*, Vol. 21 No. 1. pp. 19-31, 1988.

[PARK94a] Parker, J.R., **Vector Templates and Handprinted Character Recognition**, *Proc. 12th IAPR Conference on Pattern Recognition,* Jerusalem, Israel. Oct 9-13, 1994.

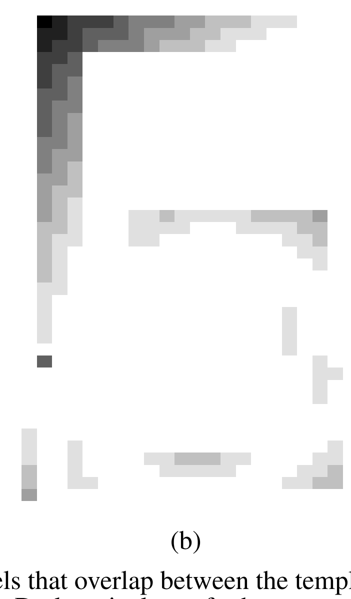
[PARK94b] Parker, J.R., Jennings, C. and Molaro, D., **A Force Based Thinning Strategy with Sub-Pixel Precision**, *Vision Interface '94,* Banff, Canada. May 18-20, 1994. Pp. 82-87.

[SRIH93] Srihari, S.N., **Recognition of Handwritten and Machine-printed Text for Postal Address Interpretation**, *Pattern Recognition Letters*, 14 (1993). pp 291-302.

[ZHAN84] Zhang, Y.Y. and Suen, C.Y., **A Fast Parallel Algorithm for Thinning Digital Patterns**, *CACM* Vol. 27 No. 3, 1984. pp 236-239.



(a)



(b)

Figure 4 - The final stages of the template match. (a) Pixels that overlap between the template and the image. (b) Distance map between template and image. Darker pixels are farther away.