Moment-derived object models for vectorization

Mingyan Shao and Robert P. Futrelle
Biological Knowledge Lab
College of Computer & Information Science
Northeastern University, Boston, MA 02115
{myshao,futrelle}@ccs.neu.edu

Abstract

Various methods for the vectorization of line drawings have been developed in the past. The predominant ones are based on skeletonization/ thinning, edge detection and sparse pixel techniques. This paper describes a new approach to vectorization based on using spatial moment analysis of gray levels in **k** x **k** regions around each pixel to generate parameters for object models. The line model, in particular, consists of a core and two wing regions which are initially determined by a principal component analysis of the moments. The model is refined by fitting to the pixel gray levels, minimizing the sum of the core and wing errors. This study uses two evaluation procedures: In the first, vector parameters from a ground truth model are compared to the fitted models; in the second, pixel statistics for the difference between the images of the ground truth model and the fitted model are evaluated. The results show that our approach can produce excellent results for the vector-derived diagrams common in scientific papers, the class of diagrams our group works on.

1. Introduction

The problem of vectorization is the reduction of raster images to object representations, e.g., turning an image of a line segment into a few parameters -- the endpoints and width (Ablameyko & Pridmore, 2000). The raster images can arise from scanning hardcopy or from electronic files generated by various applications. The primary domains of application have been legacy engineering drawings and maps. Over the years, a variety of techniques for vectorization have been introduced, each attempting to extend the applicability of the methods and to increase the speed and accuracy of vectorization systems.

Much of the work on vectorization is focused on the development of efficient and accurate vectorization systems, with less attention paid to the *use* of the vectors produced. Our group has a different focus: Extracting knowledge from scientific diagrams, particularly those in the Biology research literature. The pipeline for such work is to first vectorize diagrams (this paper), then parse the resulting objects, producing a syntactic analysis (Futrelle, 1998; Futrelle & Nikolakis, 1995) and finally, produce a semantic analysis that will allow the diagram content to be stored and indexed for conceptual retrieval and related tasks.

This paper introduces a vectorization technique which goes beyond previous approaches. There are a number of factors that lead to our new approach, including: *Moore's Law* - Going beyond the 1 bit per pixel approaches that originated in a time when systems were much smaller and slower. *Open Source* - We plan to make our Java-based system available to all. *The Scientific Literature* - Unlike most vectorization research, our work focuses on the Biomedical literature which produces about 5 million distinct figures each year. *New Algorithms* - The desire to explore new algorithms and compare them with existing techniques.

We evaluate the vectorization (lines only) of a typical diagram from the Biology literature demonstrating that our method performs well, in some cases with sub-pixel accuracy. We discuss the challenges of Biology diagrams that lie beyond our successful approach to line vectorization, e.g., separating occluding elements. The system is built using Java, Java Swing, and Java 2D and runs without recompilation on Sun Solaris, Linux (Intel) and Mac OS X.

2. The Vectorization Process

Sections 2.1-2.4 first describe the class of images we analyze. Then the moment analysis of local pixel distributions and how it leads to line models are discussed. The line models are then fitted to the full gray-level image through optimization and extension procedures. The line models have natural ways of dealing with line intersections.

2.1 Images in scientific papers

When the author of a scientific paper creates a diagram, she/he creates it in a vector-based drawing application (Xfig, MATLAB, Adobe Illustrator). Then the figure is saved in a *vector-derived raster format* for transmission to the publisher, or the conversion is done by the publisher. Such raster

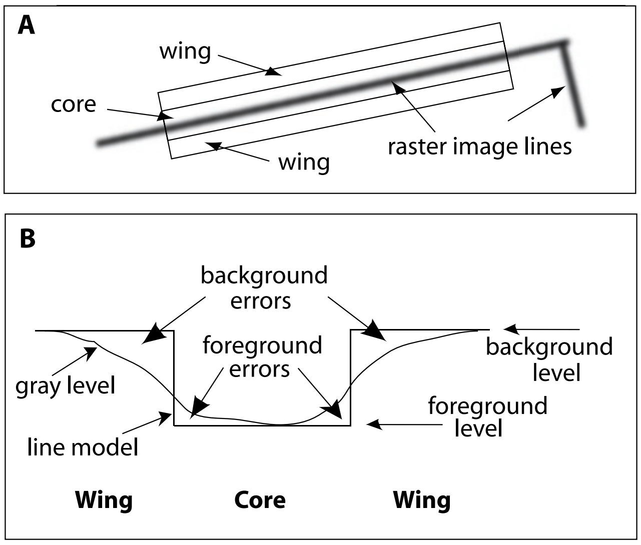

FIG. 1. The line model shown in **A** is initialized by the quadric fit (PCAs) of the gray level distribution. Then the positions of the two core/wing boundaries in the cross-section **B** are adjusted independently to minimize the sum of the foreground and background errors (differences between the gray levels and the model).

images typically contain precisely vertical and horizontal lines, lines whose width is precisely constant, etc., limited only by the raster conversion/compression algorithm. We mimic this protocol in our work, but maintain realism by "tracing" over published diagrams using Adobe Illustrator. The SVG vector version of the diagram is used to extract the vector coordinates. Our images are quite different from images generated from scanned hardcopies, the ones studied in much of the vectorization field.

2.2 Principal Component Analysis

Each pixel in the image is examined in turn. A **k** x **k** region, e.g., 16x16, centered on each pixel is used for principal component analysis (PCA) (Duda, Hart, & Stork, 2001) of the gray level distribution using the first three moments, M_0, M_1 and M_2 to produce a *quadric fit*. Quadrics that may correspond to a line are discovered using three tests in sequence:

1. Is the value of M_0 not too large (all background) and not too small (all foreground)?
2. Are the M_1 components, x_{av} and y_{av} approximately equal to the pixel position? (True if the pixel is near the center of the line or other symmetric object.)
3. Based on the M_2 components, is the aspect ratio of the quadric appropriate to a line, "long and narrow"?

The data for each acceptable fit is encapsulated in a *QuadricFit* object that contains the center point, orientation angle, height and width of the principal components. It also contains an integer *group_id*, used later when QuadricFit objects are clustered by similarity. A 2D *quadrics* array the size of the image is used to hold the QuadricFit objects. This is an example of using image-sized object arrays, something not normally done in earlier 1-bit or integer-based approaches, but easily accommodated in RAM today.

2.3 Clustering Quadric objects

The pixels making up a single line in the image can lead to the generation of many QuadricFit objects, all near the line's centerline. Clusters of these objects are built that correspond to uninterrupted straight segments, using the following agglomerative technique (Duda et al., 2001):

1. Initially, a small set of adjacent QuadricFit objects are forced into a cluster and the mean and variance of the width and orientation are computed for the cluster.
2. An iterative algorithm builds a queue of additional adjacent QuadricFit objects for consideration. Single objects are added to the cluster if their width and orientation are within a certain factor of the standard deviations of the width and orientation. This addition must not increase the standard deviations in the cluster beyond chosen limits.
3. When all acceptable QuadricFit objects have been added, the cluster is complete and additional unclustered QuadricFit objects are examined to create additional clusters.

2.4 Line model fitting and refinement

The collection of objects in each quadric cluster is used to define the termini and width of a *line model*. The line model is an idealized one that consists of a *core* with foreground gray level and two *wings* with background gray levels, Figure 1. It is important to note that our method uses a full and explicit model of the gray level distribution of an idealized line and compares this with the full gray levels of the image, without the information loss inherent in binarization (thresholding) of the image.

Quadric fits stop short of non-line regions, e.g., near line crossings, T intersections and line termini. The line models generated by quadrics are extended in such situations until they fail, e.g., when a core attempts to extend past the corner on the right in Figure 1A. The set of centerline endpoints of all of the maximally extended line models is clustered based on the distance between the endpoints, typically resulting in clusters containing one to four nearby endpoints. All pairs of endpoints in a cluster which terminate collinear lines are "sealed" by an additional core-only extension if the gap region has foreground values. This allows lines to be merged across intersections. Figure 2 shows the result of fitting, refinement and sealing for a typical data graph.

3. Evaluation

Quantitative evaluation of vectorization systems is important. The actual coordinates of the vectors that are being sought must be known -- the *ground truth*. The detected vectors are compared to the ground truth. There are useful descriptions of how to conduct evaluations; but our approach is somewhat different. For example, in line object evaluation in (Phillips, Liang, Chhabra, & Haralick,

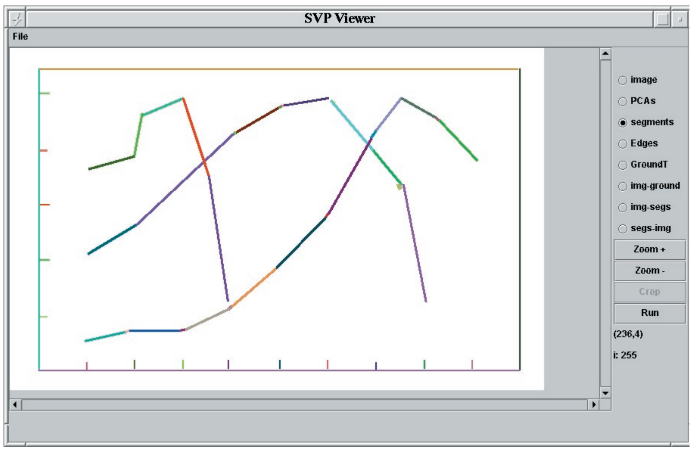

FIG. 2. The SVP Viewer, showing the segments the system discovered (in pseudocolor). The two major line crossing were successfully "sealed" by the algorithm. There are small artifacts at some of the polyline junctions which we are working to eliminate. The original black and white diagram was derived from a JPEG in a Biology paper, with the data point markers and text omitted in this study. This figure can be viewed in color at high resolution in Acrobat Reader.

1998), the relative positions of the endpoints of the ground truth and detected lines are not taken into account, a measure we use and feel is important. In (Liu & Dori, 1997), the wide variety of comparisons done leads to less than optimal specialized comparisons. In the evaluation here, besides comparing vectors, we conduct pixel-level comparisons by comparing anti-aliased rendering of the detected vectors with source pixel distributions to identify missed pixels and false alarms. In the discussion below, the ground truth vectors or raster image are labeled *source*, and the vectors generated by our system or the raster image rendered from them is labeled *detected*.

The first task in the vector comparisons is to match the detected lines with those in the ground truth. This is done by requiring that the endpoints of the two correspond within a few pixels and that the slopes of the lines are similar, e.g., within $\pi/125$ radians. Our current system produces small artifacts, short segments near the ends of some of the polylines, cf. Figure 2. These are called *minor* components, contrasting with the *major* ones include in the evaluations. There are 38 source vectors underlying Figure 2. Our system detects 39 major components, splitting one of the source lines into two. The Euclidean distance *separation* is computed between corresponding endpoints, taking into account the one split line. The *perpendicular distance* from the endpoint of the detected line to the centerline of the corresponding source is computed. These are averaged over the 76 endpoints. In addition, the difference in the lengths of the corresponding major source and detected components is computed. The results corresponding to Figure 1 are:

Average separation = 1.85 pixels
Average perpendicular distance = 0.07 pixels
Average length difference = 2.57 pixels

The average perpendicular distance, less than a tenth of a pixel, emphasizes how accurately our vectorization method can be when operating on vector-derived raster images.

Another evaluation metric we used is the comparison of the pixel distributions of the source and detected lines. This was done by thresholding both images at 128 (of 255 gray levels) to separate out black foreground pixels and counting *matched* pixels (in both), *missed* pixels (in source only) and *false alarms* (in detected only). The errors of interest are the missed and false alarms. To bound the error between 0 and 100 percent, the missed pixels were normalized to the source image foreground pixel count and the false alarms to the detected image foreground pixel count. The results are:

Missed pixel errors = 4.9%
False alarm pixel errors = 5.4%

The results of the evaluation show that our method is more than adequate at recovering source parameters from vector-derived images.

4. Previous Work

A common approach to vectorization is thinning to produce line skeletons (Lam, Lee, & Suen, 1992). This works well for portions of lines "in the clear". Skeletons also retain connectivity when passing through intersections. However, there are difficulties in reconstructing intersections correctly (Hilaire & Tombre, 2002). Our approach in which cores are projected into and through intersections is more straightforward than the various "correction" methods needed for the artifacts produced by thinning.

In the interest of efficiency, the Sparse Pixel Vectorization method has been developed (Dori & Liu, 1999). It examines fewer pixels than most methods. But the method requires a collection of Junction Recovery Procedures to deal with its artifacts. It is presumably more sensitive to noise because it ignores many pixels; our method uses the gray levels of every pixel to create and refine models.

The techniques closest to ours are described in an excellent paper (Song, Su, Tai, & Cai, 2002); a skeleton is created and then perpendicular runs are made to discover the line boundaries. This technique is "sparser" than ours and does not take into account full gray level distributions. The method relies on deleting objects once found, which destroys information from crossing lines, for example. Our method can find a pair of intersecting lines without doing any deletions, e.g., the two pairs of crossing data lines in the upper portion of Figure 2.

Many numerical parameters are used in evaluation systems. Often, the parameters that are optimal for one diagram or even for a portion of a single diagram are not optimal for another. In one paper (Chang, Lu, & Pavlidis, 1999), sweeps that cross contours can quickly find sets of parallel edges which could help in choosing parameters. The choice of parameters affects run times. Our prototype

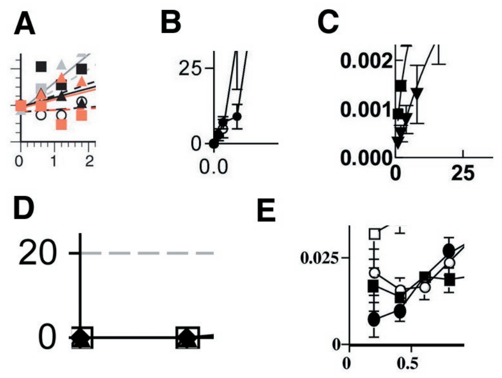

FIG. 3. One of the classes of challenges of scientific diagrams is occlusion, such as the data line and data point occlusion examples shown here. Data to be reported does not arrange itself neatly as a designer of an engineering drawing can do.

system has undergone no optimization for speed, because that would detract from our explorations of a variety of strategies. The prototype system uses about 1 minute of CPU time on a SunBlade 2000 to analyze a typical 1000x1000 pixel diagram.

5. Future Work

The next stage, following vectorization, is parsing. We have developed a successful diagram parsing system (Futrelle, 1998; Futrelle & Nikolakis, 1995) and are currently redeveloping it in Java. Parsing produces a syntactic analysis of a diagram that can be input to a knowledge-based interpretation system, ultimately resulting in a representation for a diagram that allows conceptual analysis, indexing and retrieval.

One need only glance through a recent issue of *Science* or *Nature* or an Open Access journal from BioMed Central to see the almost bewildering variety of diagrams being published. A large fraction of the diagrams in Biology present challenges for vectorization, e.g., Figure 3. Major additions are being made to our vectorization system to deal with the challenges. One such addition will deal with occlusion by "delayering" which can be understood by considering a square black data point icon lying on a data line, as in Figure 3. The unoccluded portions of the line are found first. What remains is the square, with the occluded line "within it". The pixels in the square are not deleted (set to background) but marked as being in the square. The line can then be extended across the gap, sharing pixels with the square. Strategies of this type are what led us to call our work the *Strategic Vectorization Project* or SVP.

6. Conclusion

We have demonstrated that the new vectorization technique using moment-derived object models is capable of producing excellent results when applied to lines in vector-derived raster images of Biology diagrams. There are millions of such diagrams published each year. Vectorized versions of them will allow parsing and semantic analysis which will make them available for conceptual analysis, indexing and retrieval. We are currently extending our vectorization techniques to model and analyze other object classes and multiple object occlusion.

Acknowledgement. We thank Dan Crispell for the development of the initial SVP. This material is based upon work supported by the National Science Foundation under Grant No. IIS-9978004 and the Northeastern University Institute for Complex Scientific Software, http://www.icss.neu.edu/.

References

Ablameyko, S., & Pridmore, T. (2000). *Machine Interpretation of Line Drawings: Technical Drawings, Maps and Diagrams*. London: Springer-Verlag.

Chang, F., Lu, Y. C., & Pavlidis, T. (1999). Feature analysis using line sweep thinning algorithm. *IEEE PAMI, 21*, 145-158.

Dori, D., & Liu, W. (1999). Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation. *IEEE PAMI, 21*(3), 202-215.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern Classification*. New York: Wiley.

Futrelle, R. P. (1998). *The Diagram Understanding System Demonstration Site*. *http://www.ccs.neu.edu/home/futrelle/diagrams/demo-10-98/*

Futrelle, R. P., & Nikolakis, N. (1995). Efficient Analysis of Complex Diagrams using Constraint-Based Parsing. In *ICDAR-95 (Intl. Conf. on Document Analysis & Recognition)* (pp. 782-790). Montreal, Canada.

Hilaire, X., & Tombre, K. (2002). Improving the Accuracy of Skeleton-Based Vectorization. In D. Blostein & Y.-B. Kwon (Eds.), *Graphics Recognition - Algorithms and Applications* (Vol. LNCS 2390, pp. 273-288): Springer Verlag.

Lam, L., Lee, S. W., & Suen, C. Y. (1992). Thinning Methodologies-A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 14*(9), 869-885.

Liu, W., & Dori, D. (1997). A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications, 9*(5-6), 240-250.

Phillips, I. T., Liang, J., Chhabra, A. K., & Haralick, R. M. (1998). A Performance Evaluation Protocol for Graphics Recognition Systems. In *GREC 98: Graphics Recognition, Algorithms and Systems* (pp. 372--389): Springer-Verlag.

Song, J., Su, F., Tai, C., & Cai, S. (2002). An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance. *IEEE PAMI, 24*(8), 1048-1060.