Optimal Line and Arc Detection on Run-Length Representations

Daniel Keysers and Thomas M. Breuel

Image Understanding and Pattern Recognition Research Group
German Research Center for Artificial Intelligence (DFKI GmbH)
and University of Kaiserslautern
D-67663 Kaiserslautern, Germany
E-mail:{keysers, tmb}@iupr.net

Abstract

The robust detection of lines and arcs in scanned documents or technical drawings is an important problem in document image understanding. We present a new solution to this problem that works directly on run-length encoded data. The method finds globally optimal solutions to parameterized thick line and arc models. Line thickness is part of the model and used during the matching process. Unlike previous approaches, it does not require any thinning or other preprocessing steps, no computation of the line adjacency graphs, and no heuristics. Furthermore, the only search-related parameter that needs to be specified is the desired numerical accuracy of the solution. The method is based on a branch-and-bound approach for the globally optimal detection of these geometric primitives using runs of black pixels in a bi-level image. We present qualitative results of the algorithm on images used in the 2003 GREC arc segmentation contest.

Keywords: Graphics Recognition, Line Drawings, Technical Drawings, Branch-and-Bound algorithms.

1 Introduction

Large amounts of technical and engineering information is still available only in paper form. The goal of graphics recognition is to make that information available in electronic format. After initial scanning, a key problem in the conversion of drawings into electronic form is the reliable detection of the geometric primitives from which those drawings are constructed [8].

This problem has been the subject of intensive research, and it has even been the subject of several graphics recognition contests (held in conjunction with the GREC workshops; please refer to [18, 20] and the references therein for an overview).

A variety of approaches to line detection have been described in the literature, many of them based on the Hough transform [7]. A survey of various methods is given in [19]. Many approaches also first perform a vectorization, i.e. they determine a representation by poly-lines and then use this intermediate step for line or arc detection, e.g. [5, 6].

While these existing approaches have found numerous practical applications, they have some difficulties and limitations. Hough transforms, for example, require careful choice of parameters like bucket size in order to reduce the risk of losing solutions. Preprocessing steps like vectorization and polygonalization, are themselves complex processing steps, and, since they generally only represent approximations to the original bitmap, also potentially reduce the accuracy and reliability of the overall system.

This paper introduces an algorithm for line and circle detection that operates directly on a pixel-accurate representation of the original binary image. This avoids the complexity and potential for errors introduced by separate preprocessing steps. Furthermore, it yields optimal solutions for a broad class of well-defined geometric and statistical models of thick lines.

The basis of the algorithm is the RAST algorithm [1], which is guaranteed to find globally optimal solutions under given error models and requires no parameters other than desired numerical accuracy to be specified by the user. RAST algorithms are closely related to (hierarchical) Hough transforms but have more desirable

combinatorial properties for object recognition. Recently, effective and simple to implement variants of RAST algorithms that are based on interval arithmetic have been introduced [4].

The work described in this paper is directly based on the principles presented in [4, 2], where these principles have been applied to settings in which not pixels but the outlines of connected components were used. The application of branch-and-bound algorithms to matching problems has been described in the computer vision literature before [1, 10, 12, 14, 15, 16].

In our approach, we use run length coding as the representation of the image content. Run-length coding for images is commonly used for in-memory representation of large binary images, thereby making it easy to apply the algorithm to potentially large images in memory. Run-length coding is also familiar as an important intermediate step in some binary image compression methods.

Run-length coding is not only a convenient representation in terms of memory, it also serves a semantic function: in a binary image, each line becomes a collection of adjacent runs in the runlength encoded representation. Previous approaches to line detection in images have attempted to take advantage of this representation by considering the line adjacency graph and, for example, applying Hough transform methods to the midpoints of each run [17]. We use run-length coding in a slightly different way. Rather than using the midpoints of each run, we match the complete run against a thick line model.

In the rest of the paper, we first describe the framework for optimal matching and its use of interval arithmetic, followed by a description of the specific quality functions used for the detection of the thick lines and arcs. We then present results on example images of scanned technical drawings and end with concluding remarks.

2 Detection Framework

Assume we are given a set $R = r_1, \ldots, r_N$ of pixel runs in the image plane, where each run $r = (x_0, x_1, y)$ is defined by its starting point (x_0, y) and its end point (x_1, y). The computation of this set from a given (document) image is straight-forward. Note that the decision to choose runs in horizontal direction is arbitrary and only based on the most common pixel order found in the representation of digital images. The algorithms presented here would work equally well with runs of pixels in a diagonal direction.

Now we are interested in finding the best matching geometric primitives with respect to the set R. The geometric primitives are characterized by a set of parameters $\vartheta \in T$, e.g. center, radius, and thickness for a circle. We perform the detection by finding the maximizing set of parameters

$$\hat{\vartheta}(R) := \arg\max_{\vartheta \in T} Q(\vartheta, R) \tag{1}$$

where the total quality $Q(\vartheta, R)$ of a parameter set is defined as the sum of local qualities

$$Q(\vartheta, R) := \sum_{n=1}^{N} q(\vartheta, r_n) \tag{2}$$

where $q(\vartheta, r)$ is a local quality function that evaluates the goodness of fit for a given run of pixels r and a set of parameters ϑ.

This maximization will be a complex task for most functional forms of Q. In many applications, such fits of parameters are carried out iteratively and heuristically, which involves the risk that the results found are only locally optimal solutions. Other methods include randomized approaches like e.g. random sample consensus [9].

We propose to employ a branch-and-bound technique [1, 3, 16] to perform this task that guarantees to find the globally optimal parameter set by recursively subdividing the parameter space and processing the resulting parameter hyper-rectangles in the order given by the upper bound on the total quality. Moreover, the algorithm allows us to efficiently determine the k best matches, not only the best match.

Given a computation of an upper bound on the quality of any geometric primitive with parameters in a hyper-rectangular region T (as detailed below), we can organize the search as follows (for details see [1, 3]):

1. Pick an initial region of parameter values T containing all the parameters that we are interested in.
2. Maintain a priority queue of regions T_i, where we use as the priority the upper bound on the possible values of the global quality function Q for parameters $\vartheta \in T_i$.
3. Remove a region T_i from the priority queue; if the upper bound of the quality function associated with the region is too small to be of interest, terminate.
4. If the region is small enough to satisfy our accuracy requirements, accept it as a solution; remove all runs involved in that solution from further consideration and continue at Step 3.
5. Otherwise, split the region T_i along the dimension furthest from satisfying our accuracy constraints and insert the subregions into the priority queue; continue at Step 3.

This algorithm will return all maximum quality matches greedily in decreasing order of total quality. In order to make this approach practical and avoid duplicate computations, we use a matchlist representation [1]. That is, with each region kept in the priority queue in the algorithm, we maintain a list (the matchlist) of all and only those runs that have the possibility to contribute with a positive local quality to the global quality. These matchlists will be large for the initially large regions of the parameter space but shrink with decreasing size of the regions T_i. The justification for the use of matchlists is that runs for which the distance from the geometric primitive is surely greater than half the maximum line thickness d that is contained within T_i do not contribute to the computation of the quality function at all. Furthermore, it is easy to see that the upper bound of a parameter space region T_i is also an upper bound for all subsets of T_i. When we split a region in Step 5, we therefore never have to reconsider runs in the children that have already failed to contribute to the quality computation in the parent.

The use of matchlists enables us to add another feature to the algorithm: Before the execution of Step 5 above, we introduce another conditional processing of the region T_i. If the geometric primitive described by T_i is associated with a matchlist of runs that can be split into two parts that are sufficiently separated (say, by 10 pixels), split the matchlist into these two parts and re-insert the region T_i into the queue twice, once with each part of the matchlist associated. Doing so avoids the unification of primitives that belong to separated regions in the image. It furthermore separates noise components from the matchlists, which then can be pruned in the further search.

After running the algorithm, the bounding boxes for the lines are determined by considering all associated runs in the matchlists and for circular arcs the smallest arc that encompasses all runs in the matchlist.

3 Use of Interval Arithmetic

One of the key points in the RAST algorithm is the calculation of the upper bound of the quality function for a subset of the parameter set. This upper bound can be determined using interval arithmetic both efficiently from a computational point of view and from the point of view of coding the actual function.

Interval arithmetic was originally proposed as a means of bounding the error of numerical computations. For detailed information about interval arithmetic and its implementation, the reader is referred to the references [11, 13].

The extension of the basic numerical operators to intervals is straight-forward. Instead of a rigorous definition, let us give two examples of such extensions to illustrate the principle. The operators $+$ and $\times$ are defined on intervals in the following way:

$$\begin{aligned} [x_0, x_1] + [y_0, y_1] &:= [x_0 + y_0, x_1 + y_1] \\ [x_0, x_1] \times [y_0, y_1] &:= [\min(x_0y_0, x_0y_1, x_1y_0, x_1y_1), \max(x_0y_0, x_0y_1, x_1y_0, x_1y_1)] \end{aligned}$$

Thus, any addition of numbers from $[x_0, x_1]$ and $[y_0, y_1]$ will result in a number from $[x_0, x_1] + [y_0, y_1]$, and analogously for the multiplication operation. Due to monotonicity the extensions for $\log()$ and $\exp()$ are simple and for other functions like e.g. $\sin()$ a slightly more elaborate analysis is necessary.

If the quality function used in the RAST algorithm is a composition of simpler functions like multiplication, addition, sine, minimum, square, etc., the function can be directly implemented using the basic interval arithmetic functions. In other cases, it might be more straight-forward to base parts of the determination of the upper bound on the interval arithmetic, like e.g. the calculation of the distance of a pixel from a line, and separately code other parts.

The use of interval arithmetic instead of an explicit derivation of bounding functions, as used in prior work on branch-and-bound matching algorithms, has several advantages [4]: In many geometric matching problems of practical interest, deriving bounding functions and implementing them correctly can be difficult and error prone. This problem is substantially alleviated by using interval arithmetic, which allows us to simply replace calculations on scalars by the corresponding calculations on intervals in many cases. Furthermore, the use of interval arithmetic also makes geometric matching numerically reliable, i.e. we can guarantee that no solutions are lost due to roundoff errors. This last statement is based on the use of interval functions that guarantee to include the real-valued result within an interval bounded by finite-precision floating-point numbers as representable on a computer [11].

4 Quality Functions for the Detection of Lines and Circular Arcs

For the detection of the geometric primitives considered in this work we use the following parameterizations. The line model is parameterized by its angle φ, its distance s from the origin, and its thickness d. The circle (and circular arc) model is parameterized by the coordinate of its center point (x, y), its radius s and its thickness d.

During the search algorithm, each partition of the search space is described by a Cartesian product of intervals for the parameters, i.e. for a line a set of the form $T = [\varphi_0, \varphi_1] \times [s_0, s_1] \times [d_0, d_1]$ and for a circle a set of the form $T = [x_0, x_1] \times [y_0, y_1] \times [s_0, s_1] \times [d_0, d_1]$.

Given a run $r = (x_0, x_1, y)$ we then determine for each pixel $p = (x, y)$ with $x_0 \leq x < x_1$ an interval $d_p = [d_{p0}, d_{p1}]$ of possible distances of p from the center of the line or circle, respectively. We then compute an upper and lower bound for q in the straight-forward way. The total upper bound of the quality Q is then used in the branch-and-bound algorithm.

The local quality function for a run and a geometric primitive (line or circle) is then defined to be

$$q(\vartheta, (x_0, x_1, y)) \;=\; \max\left(0,\;\; d^{-\frac{1}{2}} \sum_{x=x_0}^{x_1-1} \operatorname{sgn}\left(\frac{d}{2} - d_\vartheta(x, y)\right)\right) \tag{3}$$

where $d_\vartheta(x, y)$ denotes the (absolute) distance of the point (x, y) from the geometric primitive defined by the parameters ϑ. This quality function is greater than zero, iff at least half of the pixels in the run lie on the line and then increases with the number of pixels that are covered by the line. The factor $d^{-\frac{1}{2}}$ counteracts the increasing quality of a line with growing thickness due to more pixels being covered by a thicker line.

It is of course possible that we may find quality functions that are more suitable for the task at hand and one can think of a variety of other functions measuring the goodness of match between a run and a line with finite thickness. We experimented with a few other possibilities (distance of run end-points from expected y-run of the line, relative overlap of run and expected y-run of the line). Among these, the informal experiments suggested that the local quality function (3) produces the best results. Nevertheless, in current work we are experimenting with other local quality functions.

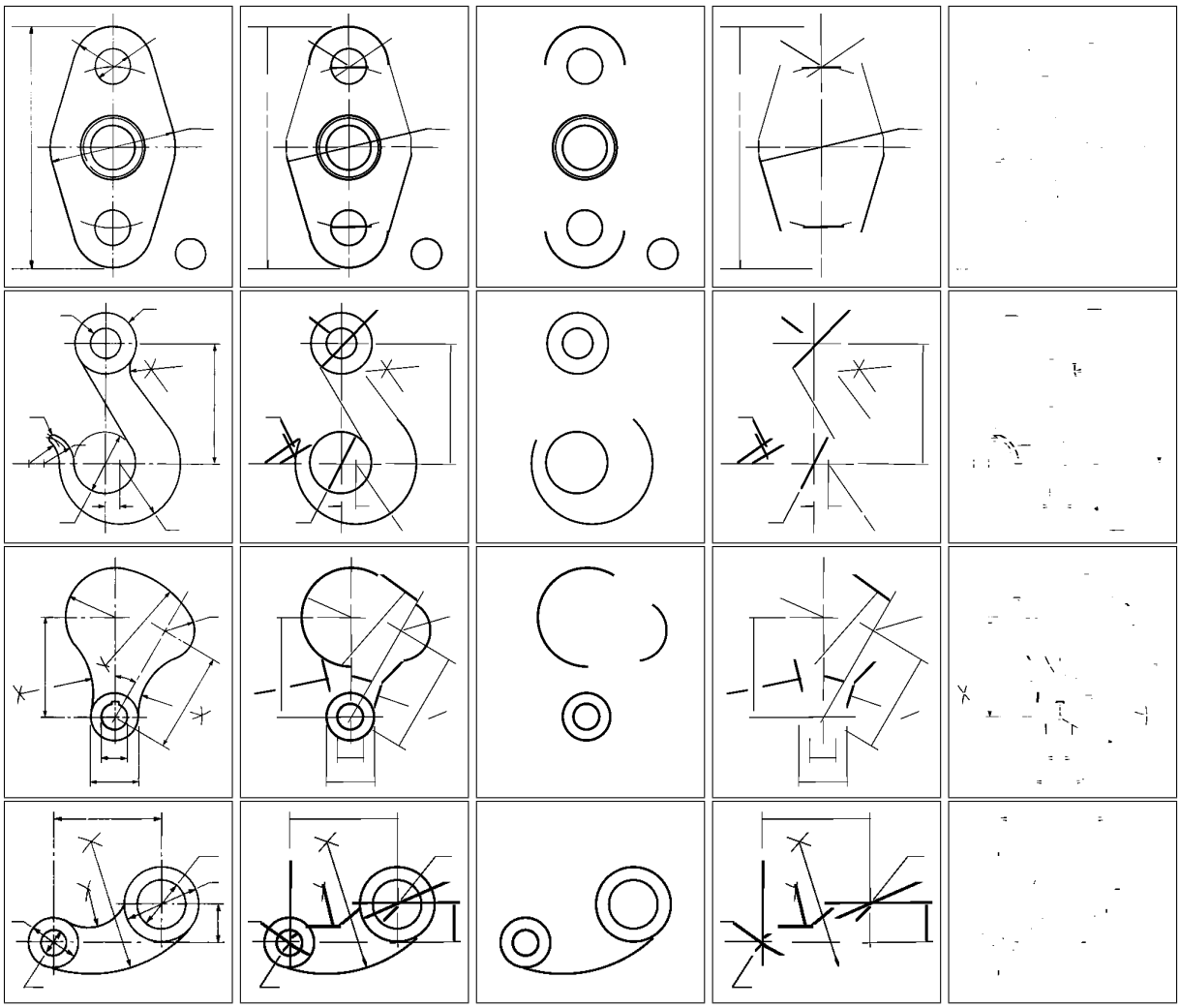

Figure 1: Results for the detection of arcs and lines using the proposed method. Each row shows (left to right) input, detected arcs and lines overlaid, detected arcs, detected lines, remaining runs not covered.

5 Results

We applied the algorithm to unprocessed images that were used in the fifth contest on arc segmentation at GREC 2003 to show the applicability and the quality of the results. These images show scanned technical drawings. A more thorough evaluation of the performance including a comparison to other state-of-the-art methods will be performed in future work.

Each of the rows in Figure 1 shows (in that order) the original image, the union of all detected primitives, the detected circular arcs, the detected lines on the remaining runs not covered by arcs, and the remaining runs in the image that were not covered by any arc or line.

Note that in some cases errors occur where geometric primitives extend over a larger range than visually appropriate. In that case e.g. arcs extend too far and thus prevent the detection of complete lines.

6 Conclusion

We have described a simple and effective method for locating geometric primitives like lines and circular arcs in bi-level images of scanned documents. Except for a library for interval arithmetic and image file I/O, the

algorithm itself is easily implemented in a few hundred lines of C++ code. Another property of the algorithm that simplifies its implementation and use is that the only parameter that the user needs to specify are the maximum thickness of the lines to be found. First experimental results show a promising behavior of the algorithm on images of technical drawings.

Note that the development of the described algorithm is work in progress, although it relies on two well-understood and tested methods, which are the RAST framework of branch-and-bound search and the use of interval arithmetic for the computation of bounds of functions defined on regions of parameter space.

To our knowledge, the algorithm presented in this paper is the first practical, globally optimal algorithm for the detection of thick lines and thick circles under well defined error models in technical drawings; other methods described in the literature generally rely on suboptimal matching methods (like the Hough transform) and approximations during preprocessing (like polygonal approximations).

We will be evaluating and comparing the performance of our algorithm with the simple geometric and statistical models described above against methods described in the literature. While the performance of our algorithm looks good in the cases that we have tested, it is possible that previous methods perform better in some cases. The reason is that existing systems often include complex, hand-tuned rules related to grouping, continuity, and plausible parameter ranges. If we identify any such cases or constraints, we can incorporate them into the quality function used by our algorithm. The optimality guarantees of the algorithm still hold (i.e., the algorithm will find the optimal results under the improved quality function), and the main advantages of the algorithm (few parameters, strict separation of what is being optimized from the search algorithm itself) still hold.

The most important next step in the development of this algorithm will therefore be the formal evaluation on a larger set of test cases, and a comparison with other systems.

Acknowledgments

This work was partially funded by the BMBF (German Federal Ministry of Education and Research), project IPeT (01 IW D03).

References

[1] T.M. Breuel: Fast Recognition using Adaptive Subdivisions of Transformation Space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 445–451, 1992.

[2] T.M. Breuel: Robust Least Square Baseline Finding using a Branch and Bound Algorithm. In *Proceedings of the SPIE - The International Society for Optical Engineering*, pp. 20–27, 2002.

[3] T.M. Breuel: Finding Lines under Bounded Error. *Pattern Recognition*, 29(1):167–178, 1996.

[4] T.M. Breuel: On the Use of Interval Arithmetic in Geometric Branch-and-Bound Algorithms. *Pattern Recognition Letters*, 24(9–10):1375–1384, 2003.

[5] D. Dori, L. Wenyin: Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(3):202–215, March 1999.

[6] P. Dosch, G. Masini, K. Tombre: Improving Arc Detection in Graphics Reognition. In *International Conference on Pattern Recongition*, volume 2, Barcelona, Spain, pp. 2243–2246, September 2000.

[7] R.O. Duda, P.E. Hart: Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[8] D. Elliman: TIF2VEC, An Algorithm for Arc Segmentation in Engineering Drawings. In *Proc. 4th IAPR Workshop on Graphics Recognition, Springer LNCS 2390*, pp. 359–358, 2001.

[9] D.A. Forsyth, J. Ponce: *Computer Vision: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2003.

[10] M. Hagedoorn, R.C. Veltkamp: Reliable and Efficient Pattern Matching Using an Affine Invariant Metric. *International Journal of Computer Vision*, 31(2/3):203–225, 1999.

[11] T.J. Hickey, Q. Ju, M.H. van Emden: Interval arithmetic: From principles to implementation. *JACM*, 48(5):1038–1068, September 2001.

[12] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge: Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–63, 1993.

[13] L. Jaulin, M. Kieffer, O. Didrit, E. Walter: *Applied Interval Analysis*. Springer Verlag, Berlin, 2001.

[14] F. Jurie: Solution of the Simultaneous Pose and Correspondence Problem Using Gaussian Error Model. *Computer Vision and Image Understanding*, 73(3):357–373, 1999.

[15] D.M. Mount, N.S. Netanyahu, J.L. Moigne: Efficient algorithms for robust feature matching. *Pattern Recognition*, 32(1):17–38, 1999.

[16] C.F. Olson: Locating Geometric Primitives by Pruning the Parameter Space. *Pattern Recognition*, 34(6):1247–1256, June 2001.

[17] T. Pavlidis: *Algorithms for Graphics and Image Processing*. W. H. Freeman & Co., New York, NY, 1983.

[18] L. Wenyin: Report of the Arc Segmentation Contest. In *Proc. 5th IAPR Workshop on Graphics Recognition, Springer LNCS 3088*, Barcelona, Spain, pp. 364–367, 2003.

[19] L. Wenyin, D. Dori: From Raster to Vectors: Extracting Visual Information from Line Drawings. *Pattern Analysis and Applications*, 2(2):10–21, 1999.

[20] L. Wenyin, J. Zhai, D. Dori: Extended Summary of the Arc Segmentation Contest. In *Proc. 4th IAPR Workshop on Graphics Recognition, Springer LNCS 2390*, pp. 343–349, 2001.