MVA'94 IAPR Workshop on Machine Vision Applications Dec. 13-15, 1994, Kawasaki

THE EFFECTS OF DOCUMENT IMAGE DEFECTS ON LINE DRAWING ANALYSIS ALGORITHMS

Yuh-Lin Chang Daniel Lopresti
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, NJ 08540 USA
[yuhlin,dpl]@mitl.research.panasonic.com

ABSTRACT

Image defects and their effects on drawing analysis algorithms are investigated in this work. To study general drawing analysis systems, we use unconstrained, well-behaved random polygons as test inputs. We generate synthetic noisy samples through the use of image defect models. Image analysis algorithms are then applied to these samples, and the results are empirically evaluated by matching them against a ground truth. Discrepancies are regarded as image analysis errors and categorized. The relationship between image defects and an algorithm's error behavior can then be analyzed. Our study has applications in the development of robust and reliable document image understanding systems.

INTRODUCTION

Image defects and their effects on drawing analysis algorithms are investigated in this work. Line drawing understanding is an important application area in machine vision research [3]. Drawing analysis algorithms are being used to solve a wide range of practical problems, e.g., map understanding, CAD drawing conversion, scientific diagram analysis, etc. In this paper we are mainly concerned with the performance evaluation of line extraction algorithms. In the past, Ramesh and Haralick [8] have studied the problem of evaluating the performance of computer vision algorithms. However, to derive analytical formulae, they must make simplifying assumptions both for the noise models and for the image analysis algorithms. In our work, image analysis algorithms are empirically evaluated by matching their output with the ground truth.

Figure 1 depicts the overall structure of our work. The three chief building blocks – synthetic image generation, drawing analysis systems, and performance evaluation – are described in the following three sections. We then present experimental results and our conclusions in the last two sections.

SYNTHETIC IMAGE GENERATION

To study general drawing analysis systems, we propose to use unconstrained, well-behaved random polygons as test inputs. We have developed a program to automatically generate such polygons. Our criteria for "well-behaved" are

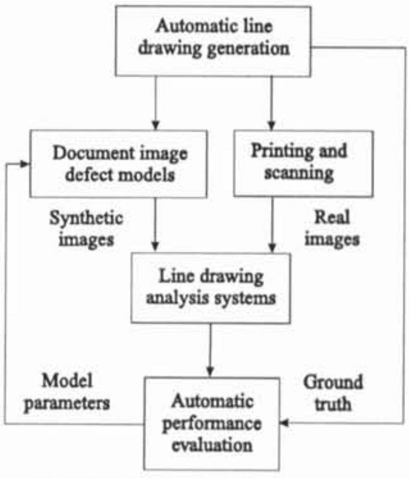
Automatic line drawing generation
Document image defect models
Printing and scanning
Synthetic images
Real images
Line drawing analysis systems
Model parameters
Automatic performance evaluation
Ground truth

Figure 1: Overview of the paper.

- each side has moderate length,
- no intersections except at end points, and
- no angles are too sharp or obtuse.

These constraints are designed to eliminate line extraction errors caused by inappropriate (i.e., unrealistic) inputs.

Document image defect models are then applied to generate synthetic images. Image defect models have been investigated previously [1, 5]. The goals are to build more robust OCR systems [1], and to study defects for a specific imaging system, i.e., photocopiers [5]. In this work we use a simplified version of Baird's model [1]. We consider effects such as ink spread, image blur, and speckle noise.

- The spreading of ink is modeled as degradation that changes a white pixel to black according to a probability P depending on the white pixel's distance d from the nearest black pixel.

$$P = P_i e^{-d^2} \tag{1}$$

  where P_i is a constant.

- Speckle noise randomly turns white pixels into black pixels with a uniform probability P_s (i.e., the distribution is Poisson).

- The defocusing effect induced by a scanner is modeled by a 3×3 averaging operation. Other more complicated alternatives include morphology operations and low-pass filters.

Hence, our image defect model is parameterized by two probabilities: P_i and P_s. A recent paper investigates the validation of image defect models for OCR systems [7]. The same concept can be applied to drawing analysis systems as well. In our work, we shall use real image samples to "calibrate" the defect model parameters, and then use the model to generate a large number of synthetic samples to investigate an image analysis system's behavior. We plan to address the validation problem in a future work.

DRAWING ANALYSIS SYSTEMS

To extract lines from a blurred and noisy image, two distinct procedures are involved: thinning and linking [9]. In the first step, the "skeletons" of image objects are extracted. We have tested the following two different approaches to image thinning:

- Medial axis transform of a binary image. To extract lines, an image is first binarized with a pre-defined threshold. The threshold value is set relatively low to avoid the fragmentation of lines. A skeleton image is then extracted from the bitmap by the medial axis transform [6].
- Watershed transform of a gray-level image. Skeletons can also be directly obtained from a gray-level image by the watershed transform. This procedure can be found in the *Khoros* package [6].

Two public domain image analysis packages, the Object Recognition Toolkit (*ORT*) [2] and *Khoros* [6], are used for edge linking. Both systems are capable of extracting polygons from bitmap images, although they employ slightly different approaches.

ORT extracts straight lines and curves from edge pixels by linking connected pixels and segmenting them into lines and curves. Four major steps are involved: chaining, segmentation, linking, and classification. In the beginning, edge pixels are first chained into linked lists. Chained sets are broken into subsets which are near-symmetric about the mid-point of the subset. Two adjacent segments are then linked together if they have similar curvature. The linked segments are finally classified as straight lines or circular arcs based on the distance from the mid-point of the subset to the straight line joining the two end points.

On the other hand, the *vpolygon* function in *Khoros* creates a vector image resulting from a polygonal approximation of an edge image. The least square method is used to fit straight lines to connected edge pixels. The fitting fails if the variance of the least square estimation is greater than a pre-defined threshold; a linked set is broken into subsets, which will then be fitted by shorter lines. The polygonal approximation process stops when all edge pixels are grouped into line segments. However, segments shorter than a predefined threshold are rejected as noise.

PERFORMANCE EVALUATION

The main contribution of this work is in the empirical evaluation of image analysis algorithms. The evaluation is performed by matching analysis results with a ground truth. This problem differs from traditional 2-D line matching because line splitting is an important concern for extraction algorithms and thus needs to be evaluated explicitly. We now describe our approach to the problem.

ERRORS AND COSTS

In the matching process, we consider the following four types of possible image analysis errors:

- Deletion: a feature is missed.
- Insertion: an extraneous feature is detected.
- Merge: two features are merged into one.
- Split: a single feature is split into two.

Costs are defined for each type of hypothesis, including a match hypothesis, as follows:

- Deletion cost $\equiv$ length of line deleted.
- Insertion cost $\equiv$ length of line inserted.
- Merge cost $\equiv$ midpoint-to-line distance + orientation difference, as Figure 2 illustrates. The line in the first image must be shorter than the line in the second image. To make the angular difference compatible with the point distance, the former is multiplied by a scale factor set between 100 and 200, as the range of point positions is 512 (the image dimension) and the range of orientations is π.
- Split cost $\equiv$ line-to-midpoint distance + orientation difference. In the current implementation, the split and merge costs are symmetric.
- Match cost $\equiv$ sum of distances between the two pairs of matched end points + orientation difference, as Figure 3 illustrates.

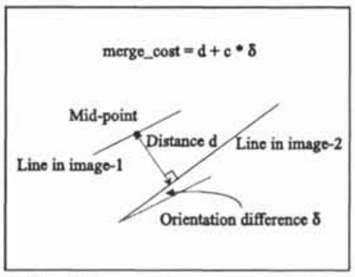

Figure 2: Cost for merging one line into another.

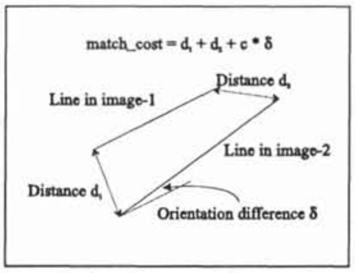

Figure 3: Cost for matching two lines.

HYPOTHESIS GENERATION

The matching process first finds all the neighbors for a line. It then generates matching, merging, and splitting hypotheses between all possible pairs of lines.

- Finding adjacencies: two lines are considered neighbors if the distance between their end points is below a threshold, as Figure 4 illustrates.
- Constructing hypotheses: a match, merge, or split hypothesis is generated for two lines in the two images if the cost is below a threshold value. The initial strength is then defined as $strength = 1.0/(1.0 + cost)$.

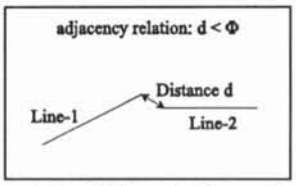

Figure 4: Definition of adjacent lines.

Note that the *strength* of a hypothesis can be interpreted as its *probability*. However, in our current implementation such a rigorous interpretation is not necessary.

Figure 5 serves as an example to illustrate this. The set of initial match hypotheses might look something like the following:

- line-a is deleted.
- line-a is matched with line-A.
- line-b is deleted.
- line-b is merged into line-B.
- line-c is deleted.
- line-c is matched with line-B.
- line-c is merged into line-B.

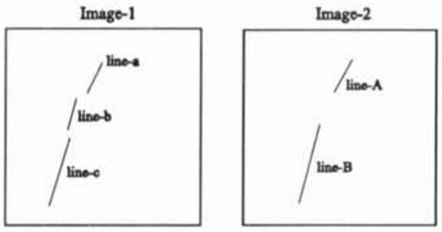

Figure 5: Example to illustrate hypothesis generation.

MATCHING BY PROBABILISTIC RELAXATION

The goal of the matching algorithm is to find the set of globally consistent hypotheses that has the minimal overall cost. Because of the combinatorial nature of the problem, exhaustive enumeration of all possible matching hypotheses would be much too inefficient. A popular approach for addressing such problems is to use a relaxation methodology. For our work, we implemented a modified version of the line drawing interpretation system proposed by Hori, *et al.* [4]. The main advantage of such an approach is its simplicity; only local information is used in the iterative optimization.

After hypothesis initialization, the strength of each match hypothesis is updated iteratively by the following process:

1. For each line, examine all of its hypotheses and collect *support* and *competition* evidence from its neighbors. The strength of a hypothesis is then updated based on the evidence, using the equation:

$$new\_strength = \frac{strength * (1.0 + support)}{(1.0 + competition)} \tag{2}$$

2. For each line, sort its hypotheses according to their strengths. Normalize the sum of their strengths to 1.0. Eliminate the weak hypotheses if their strengths drop below a pre-defined threshold value.

For our experiments, we iterate five times, but usually the winning hypotheses emerge after just two or three iterations. The success of the algorithm depends on how the support and competition functions are defined. We discuss these below.

- *Support* is used to increase the strengths of hypotheses that are consistent with their neighbors' hypotheses. The total support function is defined as the sum of the strengths of all *support* hypotheses:

$$total\_support = \sum strength(support\_hypothesis) \tag{3}$$

  1. Match. A match hypothesis is supported if it, together with a neighbor's match hypothesis, preserve the angle relation between the two adjacent lines. Two angles are considered the same if their difference is less than a threshold.
  2. Merge. A merge hypothesis is supported if a neighbor is merged or matched to the same line in the other image.
  3. Split. A split hypothesis is supported if a neighbor of the line in the other image is also split from the same line.

  The support function is summarized in Table 1. Entries in the table indicate conditions under which a hypothesis is supported.

Neighbor's	Current Hypothesis		
Hypothesis	*Match*	*Merge*	*Split*
Match	preserve angle	matched to the same line	
Merge		merged to the same line	
Split			split from the same line

Table 1: Neighborhood *support* function.

- *Competition* is used to avoid many-to-one matchings. When the *competition* evidence is collected from neighbors, only their strongest are considered. The total evidence is defined as the sum of the strengths of all *competition* hypotheses:

$$total\_compete = \sum strength(compete\_hypothesis) \tag{4}$$

1. Match. A match hypothesis competes with any other split or match hypothesis that uses the same line in the other image.
2. Merge. A merge hypothesis competes with any other split hypothesis that uses the same line in the other image.
3. Split. A split hypothesis competes with any other merge or match hypothesis that uses the same line in the other image.

The competition function is summarized in Table 2. A "$\surd$" mark in the table means that the two hypotheses are competing with each other if they use the same line in the other image.

Neighbor's Hypothesis	Current Hypothesis *Match*	*Merge*	*Split*
Match	✓		✓
Merge			✓
Split	✓	✓	

Table 2: Neighborhood *competition* function.

EXPERIMENTAL RESULTS

We have evaluated the two thinning algorithms and the two line extraction algorithms using 100 synthetic images under six different noise conditions. For medial axis transform thinning, Table 3 presents our results, while Table 4 presents our results for watershed transform thinning. The original number of lines in all the experiments was 4,396. In the tables, "I" stands for insertion, and "S" is for split. In our present set of experiments we did not encounter the other types of errors. The noise parameters were defined as $P_i = 0.01 * K$, and $P_s = 0.001 * K$. As noted earlier, 3×3 averaging was used to generate the defocusing effect in these experiments. For thresholds and other parameters, we used the default values provided by *ORT* and *Khoros*. Finally, Figure 6 gives a plot of the line detection accuracy versus noise. From these tables and figures we can make the following observations:

- Most errors are line fragmentation errors, which are caused by the combination of blurring and thinning. Most insertion errors are due to small fragments that are too hard to match to their original lines.
- When the watershed transform is used for thinning, *ORT* is much better than *Khoros*. If the medial transform is used, the situation is reversed.
- In terms of line detection accuracy, the watershed transform gives better results than the medial axis transform.
- For medial-axis transform thinning and *Khoros* polygon extraction, the line detection accuracy increases by 5% between $K = 1$ and $K = 2$. We suspect this is mainly due to the variations in thinning, and are currently investigating this anomaly.

System	Noise (K)	Detected Lines	Correct Lines	Errors I	S
Khoros	0	5,251	4,255	373	144
	1	5,345	3,777	257	619
	2	4,876	3,995	73	401
	3	5,071	3,791	64	605
	4	5,471	3,484	149	912
	5	5,946	3,156	262	1,240
ORT	0	5,459	3,679	734	573
	1	6,335	3,037	663	1,337
	2	6,472	2,943	650	1,449
	3	7,081	2,640	940	1,756
	4	7,821	2,260	1,292	2,134
	5	8,946	2,001	1,569	2,392

Table 3: Results for medial axis transform thinning.

System	Noise (K)	Detected Lines	Correct Lines	Errors I	S
Khoros	0	5,251	4,255	373	144
	1	5,345	3,777	257	619
	2	4,876	3,995	73	401
	3	5,071	3,791	64	605
	4	5,471	3,484	149	912
	5	5,946	3,156	262	1,240
ORT	0	5,459	3,679	734	573
	1	6,335	3,037	663	1,337
	2	6,472	2,943	650	1,449
	3	7,081	2,640	940	1,756
	4	7,821	2,260	1,292	2,134
	5	8,946	2,001	1,569	2,392

Table 4: Results for watershed thinning.

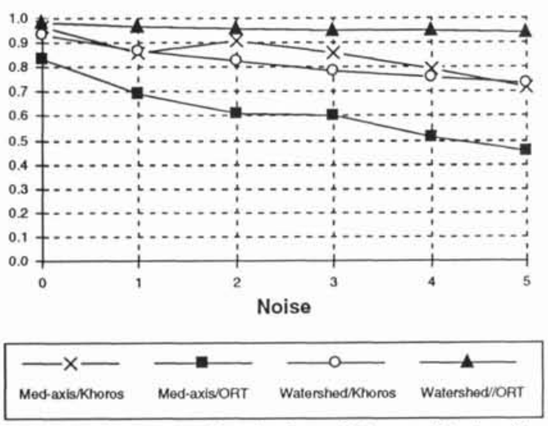

Figure 6: Accuracies for the four different thinning/line extraction algorithms.

We also present several examples from our experiments. The first is for high noise ($K = 5$). Figure 7 shows the original random polygons, and the image after ink spreading and speckle noise has been added. Figure 8 displays the images after blurring, cleaning, and thresholding, and after thinning. Thinning is performed by the medial axis transform [6]. Finally, Figure 9 presents the image analysis results from both *ORT* and *Khoros*. The extracted lines are then matched to the ground truth data, and the discrepancies are regarded as image analysis errors. For comparison, Figure 10 shows the analysis results for lower noise ($K = 1$). Figure 11 gives a magnified view of the image thinning results for the watershed transform. Figure 12 shows the lines extracted by *ORT* and *Khoros* under high noise conditions, and Figures 13 shows the extraction results under low noise.

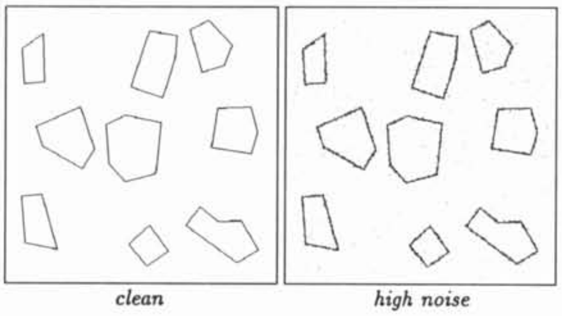

Figure 7: Original and high noise images.

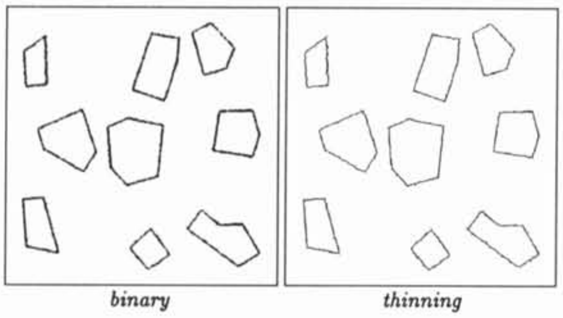

Figure 8: Binarized and thinned images.

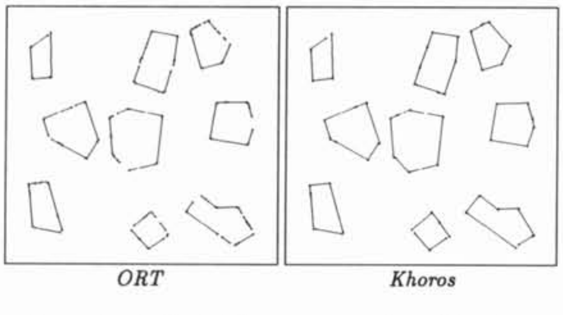

Figure 9: *ORT* and *Khoros* results under high noise.

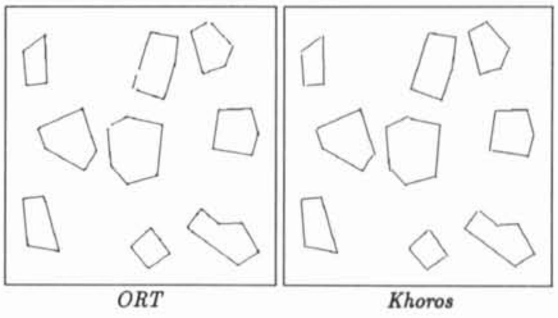

Figure 10: *ORT* and *Khoros* results under low noise.

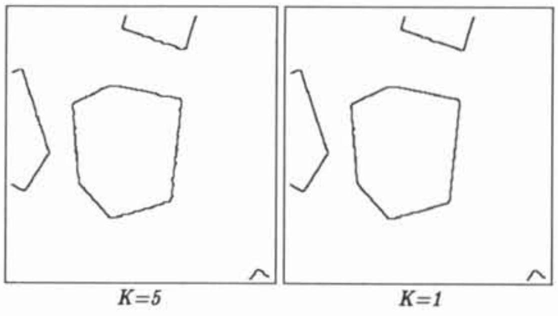

Figure 11: Watershed transform thinning.

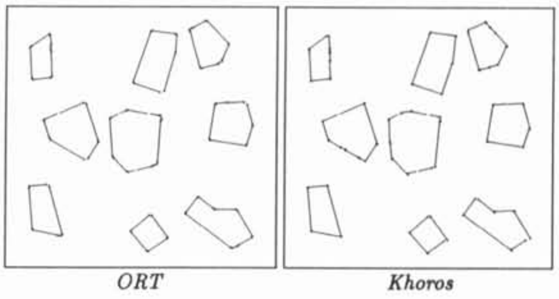

Figure 12: *ORT* and *Khoros* results under high noise.

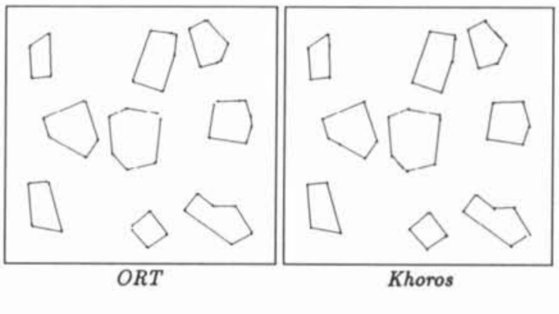

Figure 13: *ORT* and *Khoros* results under low noise.

CONCLUSIONS

In this paper, we have investigated image defects and their effects on line drawing analysis algorithms. To study general drawing analysis systems, we used unconstrained, well-behaved random polygons as test inputs. We generated synthetic image samples through the use of image defect models. Two algorithms, the watershed transform and the medial axis transform, were used for image thinning. Two line extraction algorithms, *ORT* and *Khoros*, were then applied to the synthetic samples, and the results were empirically evaluated by matching them against a ground truth. We regard discrepancies as image analysis errors, and classify them as deletions, insertions, merges, and splits. Each error is assessed a cost, and a matching is determined using probabilistic relaxation.

To evaluate line extraction results, we generated 100 synthetic images totaling more than 4,000 lines. We concluded that the watershed transform is better than the medial axis transform, and that *ORT* generally performs better than *Khoros*, but is sensitive to thinning errors. It is also clear from our experiments that image defects can have a significant effect on image analysis accuracy. A quantitative study of the relationship between image defects and image analysis results is therefore crucial to the development of robust and reliable document image understanding systems.

References

[1] H. S. Baird, "Document image defect models and their use," in *The Proceedings of the Second International Conference on Document Anaysis and Recognition,* Tsukuba, Japan, October, 1993, pp. 62-67.

[2] A. Etemadi, "Robust segmentation of edge data," Technical Report, University of Surrey, UK, 1991.

[3] L. O'Gorman and R. Kasturi, guest eds., *Special Issue on Document Image Analysis.*, IEEE Computer, vol. 25, no. 7, July, 1992.

[4] O. Hori *et al.*, "Probabilistic relaxation method for line-drawing interpretation," in *The Proceedings of International Conference on Pattern Recognition,* June, 1992, pp. 158-161.

[5] T. Kanungo, R. M. Haralick and I. Phillips, "Global and local document degradation models," in *The Proceedings of the Second International Conference on Document Anaysis and Recognition,* Tsukuba, Japan, October, 1993, pp. 730-734.

[6] The Khoros Group, *The Khoros User's Manual.* University of New Mexico, 1992.

[7] Y. Li, D. Lopresti and A. Tomkins, "Validation of document image defect models for optical character recognition," in *The Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval,* April 1994, pp. 137-150.

[8] V. Ramesh and R. M. Haralick, "Random perturbation models and performance characterization in computer vision," in *The Proceedings of Computer Vision and Pattern Recognition,* Champaign, IL, June 15-18, 1992, pp. 521-527.

[9] R. W. Smith, "Computer processing of line images: a survey," *Pattern Recognition*, vol. 20, no. 1, pp. 7-15, 1987.