Locating Corners in Noisy Curves by Delineating Imperfect Sequences

Z. Aviad

6 December 1988

CMU-CS-88-199 z

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA. 15213

This research was primarily sponsored by the U.S. Army Engineering Topographic Laboratories under Contract DACA72-87-C-0001 and partialy supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, Amendment 20, under contract number F33615-87-C-1499, monitored by the:

Avionics Laboratory
Air Force Wright Aeronautical Laboratories
Aeronautical Systems Division (AFSC)
United States Air Force
Wright-Patterson AFB, Ohio 45433-6543

Table of Contents

List of Figures

Locating Corners in Noisy Curves by Delineating Imperfect Sequences

Z. Aviad

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

Line-based image analysis and area-based image analysis are rarely combined, although it is quite clear that these two approaches complement each other. One reason for the infrequency of the combination is that automatic segmentation procedures usually produce lines of poor quality, and such lines are hard to analyze. This paper addresses the problem of locating right angle corners in unsmooth digital lines, employing the concept of *imperfect sequences*, that has proved useful in various applications, including road finding and RNA homology detection. A detector of imperfect sequences is used to create a line analysis method that is fast and simple, yet robust enough to produce reasonable corner localization even in noisy lines. We have found that using this algorithm gave us a powerful tool for combining the line-based and area-based approaches.

Subject Terms: Digital Corners, Imperfect Sequences.

1. Introduction

Region-based image segmentations frequently display very jagged lines in the boundaries of the segments as result of the fact that machine segmentation algorithms are designed to optimize image intensity criteria and not line smoothness. Edge finders, on the other hand, are based on optimizing curvilinear features, but provide less two dimensional information. In order to apply line reasoning to machine segmentation results it is necessary to simplify the segment boundaries, a complex process in general. Consequently, the combination of line-based analysis and area-based analysis is relatively rare. Figure 1-1, a result of a shadow finding program [6], is a typical example of the quality of machine segmentations.

Locating significant points on curves is a difficult problem to which much research effort has been devoted. Scale-space analysis seem to be the current method of choice when one is dealing with the general curve partitioning problem [1, 11], but scale-space filtering has its shortcomings [2, 7], and in

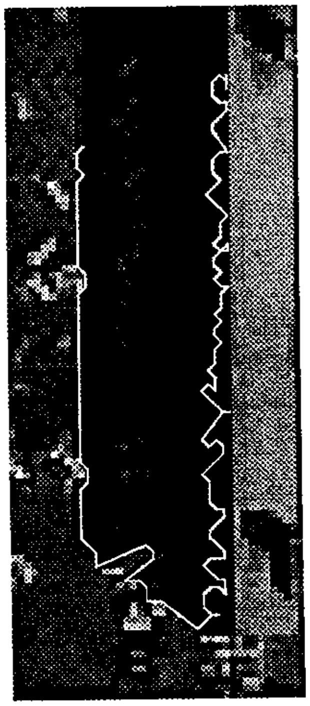


Figure 1-1: Example of a machine segmentation

some applications more specific requirements make it possible to use simpler and faster algorithms. This paper is concerned with locating right angles in noisy digital curves, a requirement motivated by experiments in building detection.

Our approach is to test local properties of the line, and base the recognition of features on consecutive sequences of points where the local properties hold. In the example above (Figure 1-1) the relevant local property of the line is "going down", "going left" or "going up". Since the lines we are dealing with are noisy, the local property sequences that correspond to various features are imperfect, and the definition of consecutivity must be relaxed to the effect that some sequence breaks are ignored. Section 2 defines the local property tests, Section 3 describes an algorithm that delineates imperfect sequences, and Section 4 presents the results of running the algorithm on various curves from an edge detector, a shadow finding program, and an area based segmentation program. An experiment of comparison with Fourier approximation is also reported. Finally, Section 5 is a brief conclusion.

2. Local Property Testing

In this section we describe four functions, referred to as *classifiers*, that are used for detecting right angle corners. A classifier examines the segment between two consecutive points on a digitized curve, and returns a 1 or a 0. Our four classifiers were selected so that when examining a noise-free curve, for every right corner there is at least one classifier that has a different response on the two sides of the corner.

The requirement of detecting right angles lets us classify lines into eight directions being sure that no right angle has both edges in the same class. Each of the four classifiers distinguishes between two groups of four directions, and is therefore able to detect any corner composed of a member of each group. For each classifier we also require that it be able to distinguish between lines that are anti-parallel, so that very sharp angles do not go unnoticed. Given (x_i,y_i) and (x_{i+1},y_{i+1}), two consecutive points on the curve, the four classifiers inspect $dx=x_{i+1}-x_i$ and $dy=y_{i+1}-y_i$, responding to the following conditions respectively:

1. **If** $dx \neq dy$ **then** $dx > dy$ **else** $-dx > dy$
2. **If** $-dx \neq dy$ **then** $-dx > dy$ **else** $dx < dy$
3. **If** $dx \neq 0$ **then** $dx > 0$ **else** $dy < 0$
4. **If** $dy \neq 0$ **then** $dy > 0$ **else** $dx > 0$

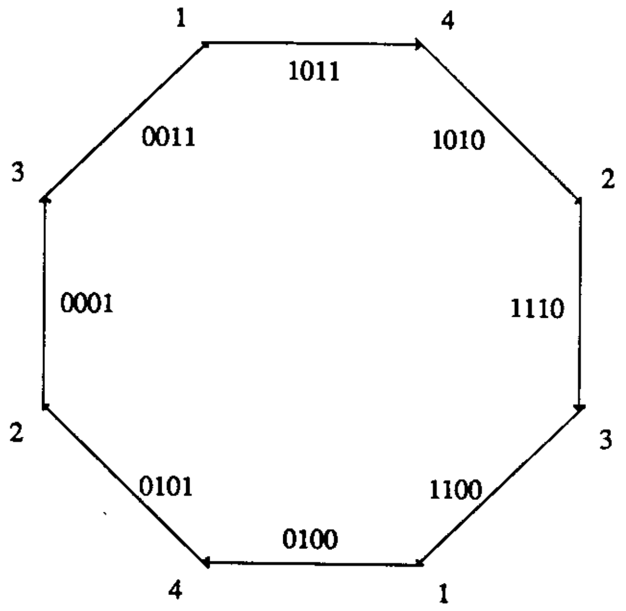


Figure 2-1: Line directions and corners. The numbers outside associate corners to the classifiers that detect them. Inside are the classifiers' responses to each line.

Figure 2-1 shows the eight line directions, the responses of each classifier to each line (inside), and which corner is detected by which classifier (outside). It should be noted that sometimes the data is ambiguous, and if one wants to be able to detect multiple right angle hypotheses there must be a capability to detect 135° angles. An example of such a case is given in Figure 2-2.

To detect corners, first a dense curve representation is created by applying a DDA to the input. Then four sequences of ones and zeroes are generated by applying the four classifiers to the dense curve. Next, each of the four sequences is partitioned by delineating *Imperfect Sequences*, a process that is discussed in the next section. The partition points are collected as corner candidates. Due to the nature of the classifiers, some of these candidates do not correspond to significant corners, so after all the partition points are collected from all the classifiers, those that form angles of at most 135° are selected

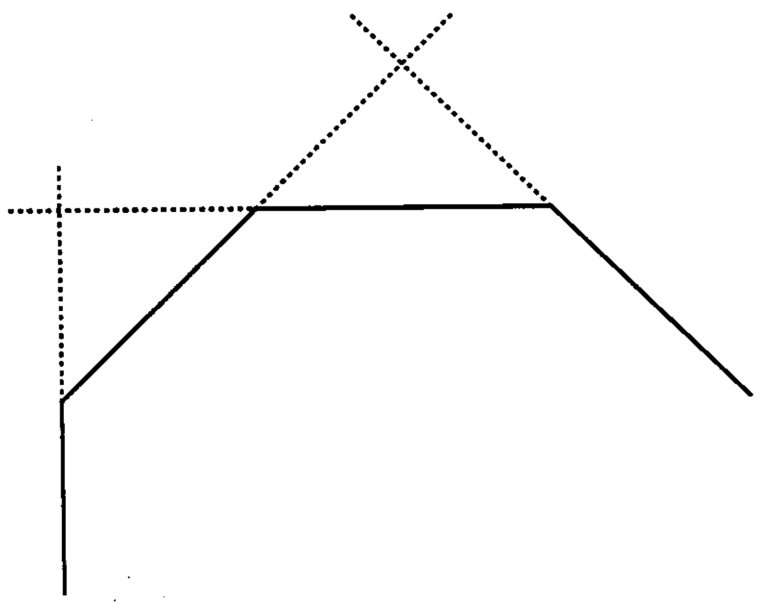


Figure 2-2: Example of multiple corner hypotheses

as the final break points of the curve. The results are presented in detail in section 4.

3. Delineation of Imperfect Sequences

Many recognition tasks share the general structure of testing for a local property and requiring a sequence of successful tests for confirming the recognition. In matching bit strings, for instance, the local property is an occurrence of the same bit in corresponding places in the matched strings, and a match is declared when such an occurrence happens over the full length of a string. However, it is often the case that the desired local property cannot be detected continuously over the whole tested entity, and recognition must be based on the delineation of *imperfect sequences* of successful tests. Furthermore, it is frequently extremely difficult to characterize the properties of the errors in such sequences, due to the lack of an applicable statistic model [8].

The problem can be stated as follows: Let P be a process that changes back and forth from state *On* to state *Off*, and let $S=s_0, \ldots, s_n$, be a sequence of noisy samples of the state of P, where $s_i=1$ if the i-th sample indicates that P is *On*, and $s_i=0$ otherwise. The object of an Imperfect Sequence Detector (ISD) is to guess when the process P really changed state. The guessed state change points delineate imperfect sequences of samples of the same kind.

In order for the ISD's guesses to have any significance, one must have some knowledge about the nature of the changes in P's states. Otherwise there would be no reason not to take S itself as the best guess of state changes. We concern ourselves with the case that: (1) sampling is known to be faster than the state changes, and (2) there are two known numbers k_{on} and k_{off} such that a sequence of k_{on} *On* samples indicates beyond reasonable doubt that P is *On*, and a sequence of k_{off} *Off* samples indicates beyond reasonable doubt that P is *Off*.

The ISD algorithm described in this section has been found useful in detecting imperfect sequences under the conditions discussed above. It was successfully used in many applications, among them finding homologies in RNA molecules [3], and road finding [4].

In the ISD algorithm two hypotheses, whether P is currently *On* or *Off*, compete for being accepted. As samples are processed these hypotheses gain support, and when that support exceeds a supplied threshold (*onthresh, offthresh*) the corresponding hypothesis is accepted. The sequence detector's performance is a result from the way support is evaluated with respect to the input samples. Two kinds of evaluation rules are used:

1. *Updating support in accordance with confirming or contradicting samples:* Each consecutive confirmation has a greater contribution to the confidence in the supported hypothesis. In practice, a pair of variables are associated with each hypothesis: a score variable records the current support and an increment variable contains the increment to the support if the next sample confirms the hypothesis. With each consecutive confirmation, the increment is added to the score and then the increment variable is itself incremented. With each contradicting sample the increment is set back to one and not added to the score. (This is equivalent to computing kinetic energy, where the length of a consecutive sequence of one value is treated like velocity, so the score variable actually measures the total energy of the imperfect sequence.)

2. *Evaluating interrelations between the supports of the contradicting hypotheses:* There are two cases:

   a. *Accepting one hypothesis implies rejecting the other:* When a hypothesis' score exceeds the corresponding threshold that hypothesis is accepted and the score-increment pairs of both the hypotheses are reset to zero-one values.

   b. *Overriding confidence buildup in an unaccepted hypothesis:* When the score that corresponds to the currently accepted hypothesis exceeds the score that corresponds to the competing hypothesis, then *both* score-increment pairs are reset. This rule serves to make sure that a new hypothesis will not be accepted as long as the current samples have a locally *better* fit to the currently accepted one.

To trace the beginnings and ends of the sequences the algorithm maintains two place variables, $enter$ and $exit$, that hold possible beginnings of the guessed *On* and *Off* states respectively. The $enter$ variable is set whenever the following three conditions hold: (1) the current state of P is assumed to be *Off*; (2) the current sample is 1; and (3) the previous $enter$ has no support. The $exit$ variable is treated similarly with respect to the other hypothesis.

Figure 3-1 is an example of an input sample sequence. Figure 3-2 shows the execution of the algorithm on the first twenty nine samples of that input for $k_{on}=k_{off}=4$, implying $onthresh=offthresh=10$. The scores in the table are given before processing of the current input. Finally, Figure 3-3 shows the complete results for this example.

```
1 1 1 1 0 1 1 0 1 1   0 0 1 0 0 1 0 0 0 1   0 0 0 0 1 1 1 1 0 1

1 1 0 1 1 1 1 0 0 0   0 0 0 1 1 0 0 0 1 1   0 0 0 1 1 0 0 0 1 1

0 1 0 1 0 1 1 1 1 1   1 1 1 0 0 0 0 0 0 0   0 0 0
```

Figure 3-1: Example of a sample sequence.

		ON		OFF		
PLACE	SAMPLE	score	inc	score	inc	COMMENTS
0	1	0	1	0	1	enter := 0
1	1	1	2	0	1	
2	1	3	3	0	1	
3	1	6	4	0	1	
4	0	10	5	0	1	
5	1	10	1	1	2	ON decided.
6	1	0	1	0	1	
7	0	0	1	0	1	exit := 7
8	1	0	1	1	2	
9	1	1	2	1	1	
10	0	0	1	0	1	exit := 10
11	0	0	1	1	2	
12	1	0	1	3	3	
13	0	1	2	3	1	
14	0	1	1	4	2	
15	1	1	1	6	3	
16	0	2	2	6	1	
17	0	2	1	7	2	
18	0	2	1	9	3	OFF decided.
19	1	0	1	0	1	enter := 19
20	0	1	2	0	1	
21	0	1	1	1	2	
22	0	0	1	0	1	Rule 2.b
23	0	0	1	0	1	
24	1	0	1	0	1	enter := 24
25	1	1	2	0	1	
26	1	3	3	0	1	
27	1	6	4	0	1	
28	0	10	5	0	1	
29	1	10	1	1	2	ON decided.

Figure 3-2: A trace of the sequence detector's execution.

A detailed discussion of the delineation of Imperfect Sequences may be found in [5].

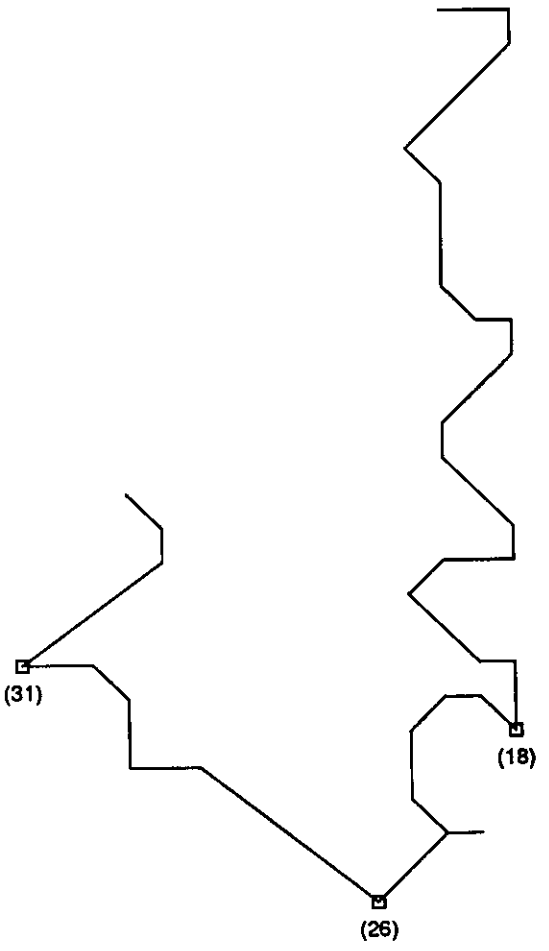
```
ON  ( 0,10): 1 1 1 1 0 1 1 0 1 1
OFF (10,24): 0 0 1 0 0 1 0 0 0 1 0 0 0 0
ON  (24,37): 1 1 1 1 0 1 1 1 0 1 1 1 1
OFF (37,58): 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0
ON  (58,73): 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1
OFF (74,84): 0 0 0 0 0 0 0 0 0 0
```

Figure 3-3: Final results for the example in Figure 3-1.

4. Experimental Results

The corner detector described in the previous sections was tested on many curves generated by a Nevatia-Babu edge finder [10], a region merging segmentation program [9], and a shadow finding program [6]. In many cases the corners detected were based on perfect sequences of classifier tests, but in many others the ISD played a significant role. Figure 4-1 shows the partition of a the lower right corner of the shadow in Figure 1-1. The considered curve is on the left, and an ISD trace on the right. The "+" signs mark the suggested break points, and the comments column shows when each break point was decided. The break point at point 18 was rejected because the angle it forms is not sharp enough. The thresholds *onthresh, offthresh* in this case were both set to 150.

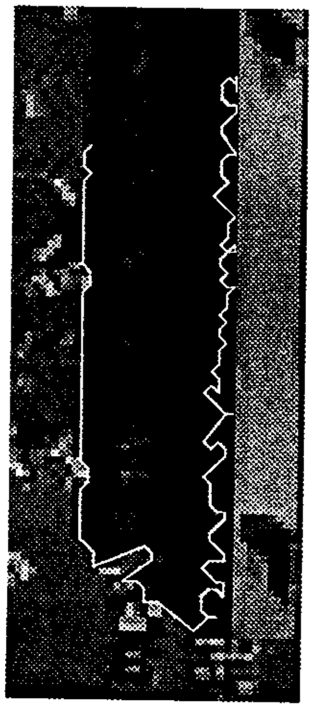
For Figure 4-2 a line fitting algorithm was applied to each of the sequences delineated for the example in Figure 1-1, also repeated in the left of the Figure. Corner points were selected so as to minimize the sum of squared distances between the corner and the intersecting lines. The delineation of the shadow is almost as good as could be expected from an edge-finding program.



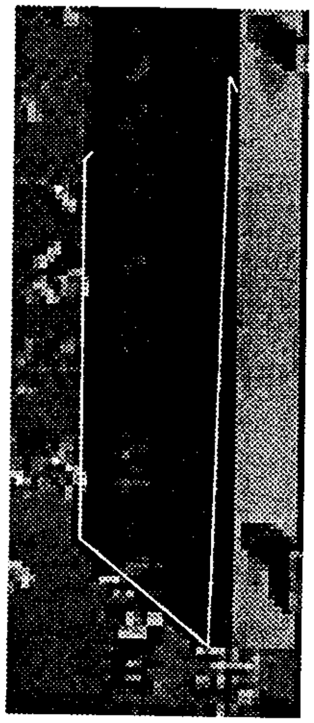
```
PNT  DX DY  t0 t1 t2 t3 COMMENTS
---  -- --  -- -- -- -- --------
10   -1  0   0  1  0  0
11   -2 -2   1  1  0  0
11   -2 -2   1  1  0  0
11   -2 -2   1  1  0  0
12   -1  0   0  1  0  0
13    0  2   0  0  0  1
13    0  2   0  0  0  1
14   -1  1   0  1  0  1
14   -1  1   0  1  0  1
15   -2 -2   1  1  0  0
15   -2 -2   1  1  0  0
15   -2 -2   1  1  0  0
16    0 -1   1  1  1  0
17   -2  0   0  1  0  0
17   -2  0   0  1  0  0
                      +
18    1  1   0  0  1  1
18    1  1   0  0  1  1
19    0  1   0  0  0  1
20   -1  1   0  1  0  1
20   -1  1   0  1  0  1
21   -2  0   0  1  0  0
21   -2  0   0  1  0  0
22   -1 -1   1  1  0  0
22   -1 -1   1  1  0  0
23    0 -1   1  1  1  0
24    0  1   0  0  0  1
25   -2  2   0  1  0  1
25   -2  2   0  1  0  1
25   -2  2   0  1  0  1
                +  +
26    4  5   0  0  1  1
26    4  5   0  0  1  1
26    4  5   0  0  1  1
26    4  5   0  0  1  1  t[3] in
26    4  5   0  0  1  1
26    4  5   0  0  1  1
26    4  5   0  0  1  1
27    0  2   0  0  0  1
27    0  2   0  0  0  1
28    2  0   1  0  1  1  t[1] out
28    2  0   1  0  1  1
29    1  1   0  0  1  1
29    1  1   0  0  1  1
30    0  2   0  0  0  1
30    0  2   0  0  0  1
             +  +     +
31    3 -4   1  1  1  0
31    3 -4   1  1  1  0
31    3 -4   1  1  1  0
31    3 -4   1  1  1  0  t[2] in
31    3 -4   1  1  1  0
32    1  0   1  0  1  1
33    1  1   0  0  1  1
33    1  1   0  0  1  1
END                      t[0,1] in,
                         t[3] out
```

Figure 4-1: Example of a partitioning of a machine-segmented region

To demonstrate that the corners found by our algorithm are significant not only in selected cases, in Figure 4-3 the original curve between corners is ignored, and the corners are connected by straight lines, without even trying to fit the line to the original curve. (This processing will henceforth be referred to as *beautification*). The minimal significant feature (corresponding to k_{on} and k_{off} of Section 3) is 3 in this run.



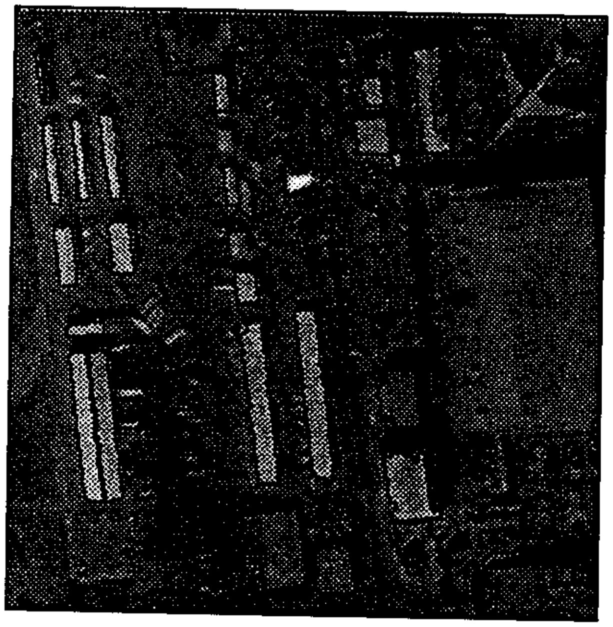
Original Data



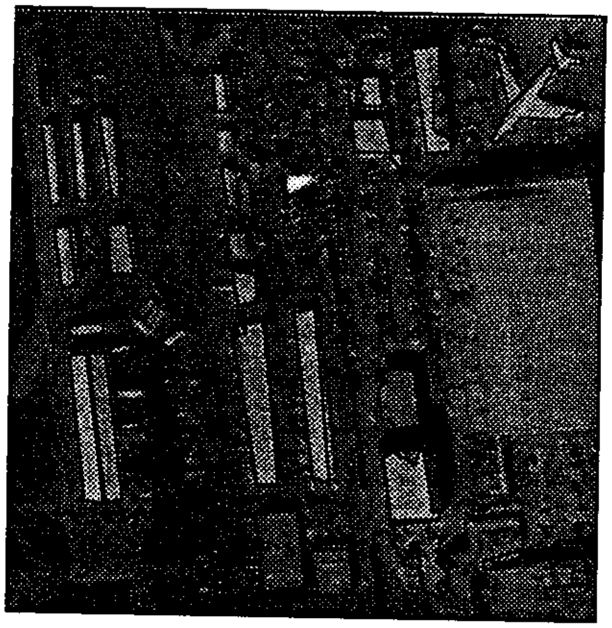
With ISD and Line Fitting

Figure 4-2: Fitting lines to a partitioned machine-segmented region

As can be seen, in spite of the lines' being jagged as is characteristic in machine segmentations, the beautified image is suitable for line based reasoning.



Original Data



Beautified

Figure 4-3: Beautification of a machine segmentation

Figure 4-4 shows corners detected in an edge-finder output. The typical difficulty in edge-detector outputs is that the corners are smoothly rounded. Nevertheless, our corner detector does a good job of breaking the curves into their roughly straight components. The left side of the figure shows the original curve, with the break points marked, and the right side shows an idealized version, like the one in Figure 4-3.

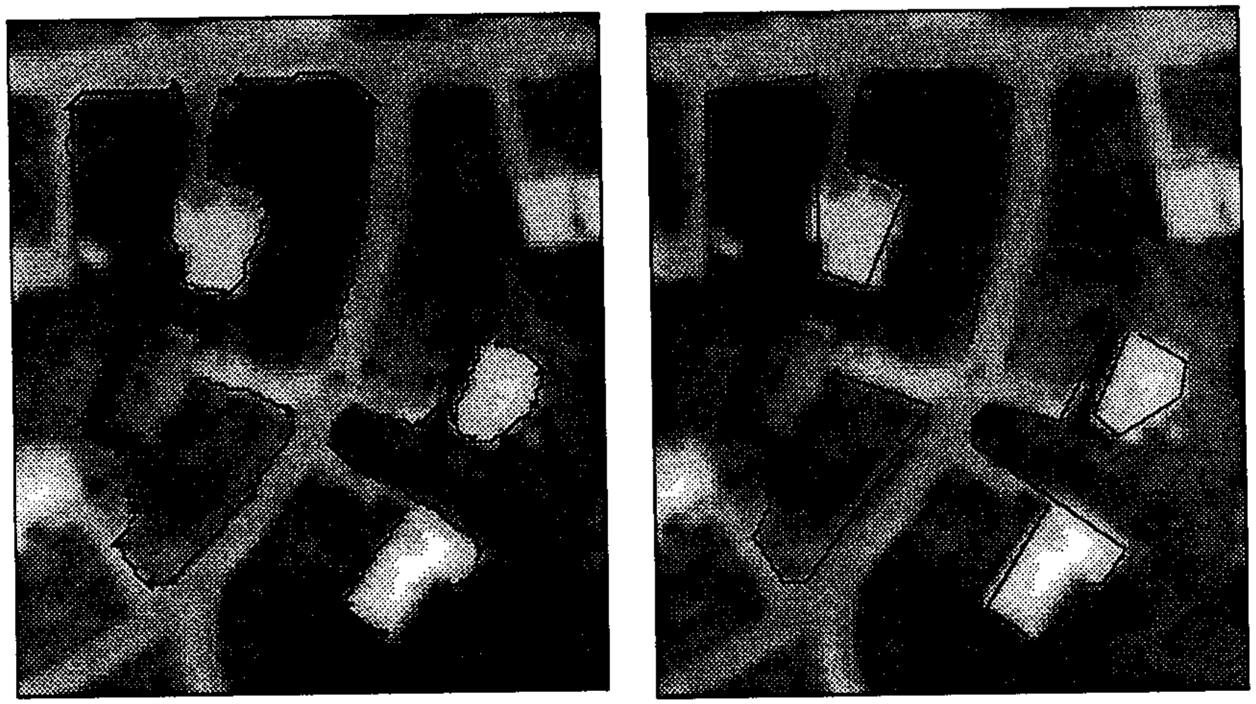


Figure 4-4: Beautification of an edge finder output

In Figure 4-5 three of the regions in 4-3 are shown in detail, and the simplification resulting from our algorithm is compared to Fourier approximations. On the left is the corner detector output, in the middle a Fourier approximation with a comparable number of points and on the right a Fourier approximation with many more points. The gray line is the original curve, the black line is the approximation. All the Fourier approximations used nine terms. As can clearly be seen, the corner detector outperforms the Fourier in its data compression, in its fidelity to the original curve and especially in its corner detection and localization. Furthermore, the corner detector does not have to know the number of corners in advance.

5. Conclusions

We have presented an algorithm that can span the gap between area-based analysis and edge-based analysis of images -- in the case where right angle corners are of interest. Instead of being based on a general theory of line shape analysis, this algorithm is derived from a practical paradigm that has proved itself in a variety of applications. The paradigm calls for designing tests that would solve the problem for noise-free cases and then using a detector of imperfect sequences to relax the stringency of the consecutivity requirement. The resulting algorithm is fast, conceptually simple, and easy to implement, yet performs very well in detecting and localizing significant corners, as is seen from the comparison to

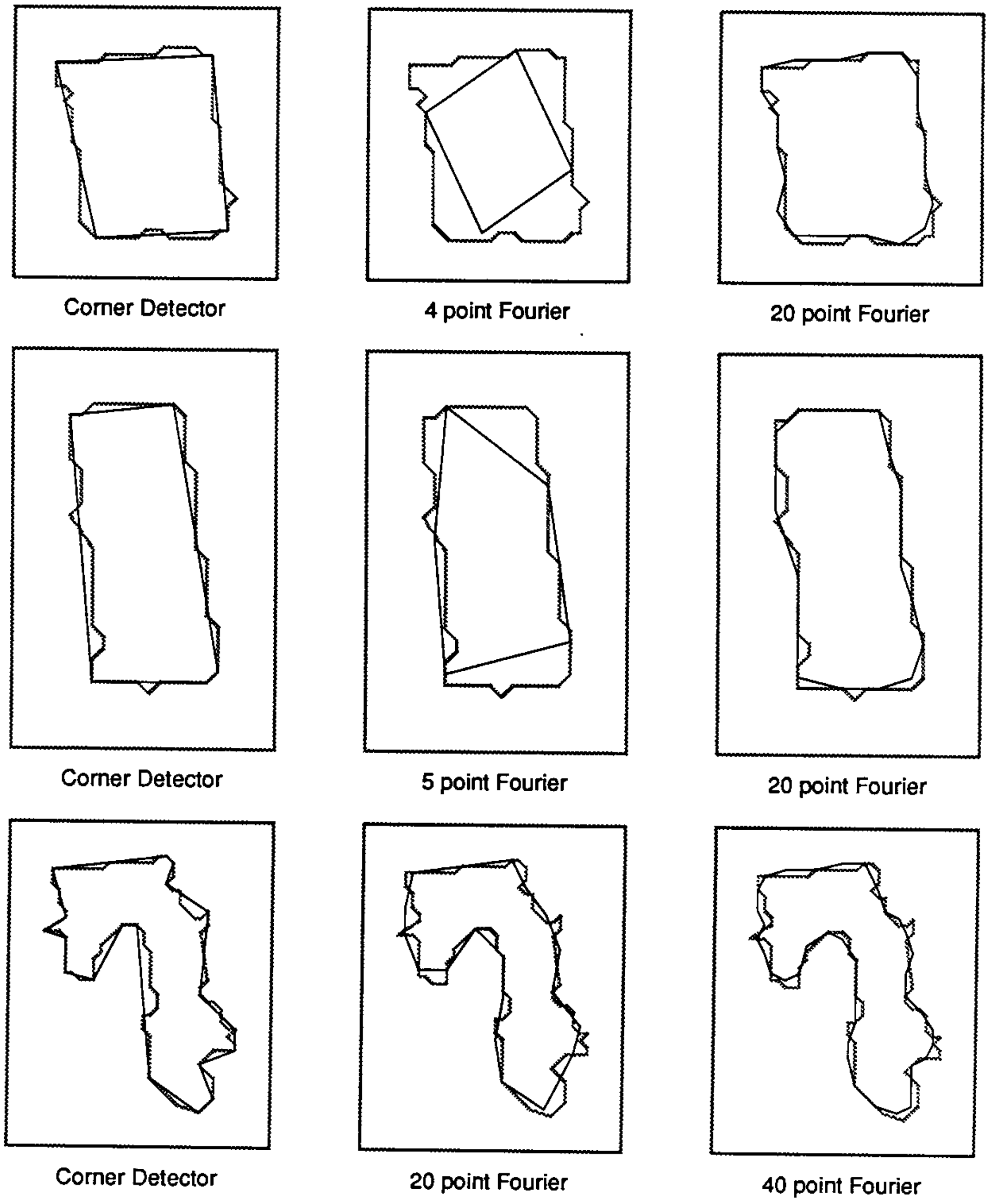


Figure 4-5: Corner Finder compared to Fourier Approximation

Fourier approximations. The simplicity of our method transforms the simplification of machine segmented region boundaries from a project on its own right to a modest step in a larger scope system that is capable of deeper analysis.

Acknowledgement

I would like to thank my colleagues in the Digital Mapping Laboratory for an exceptionally friendly and productive working environment. In particular, I am indebted to Dave McKeown for setting the goals for this research, and to Bruce Irvine who generated many of the test cases.

References

1. Asada, H. and M. Brady. "The curvature primal sketch". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*, 1 (January 1986), 2-14.

2. Aviad, Z. A Discrete Scale-Space Representation. Proc. 1st International Conference on Computer Vision, London, England, June, 1987. Also available as Tech. Report CMU-CS-88-156.

3. Aviad, Z. and S. Ben-Sasson. A Practical Algorithm for Finding RNA Homologies. Unpublished.

4. Aviad, Z. and P. D. Carnine. Road Finding for Road Network Extraction. Proceedings: Computer Vision and Pattern Recognition, Ann Arbor, Michigan, June, 1988, pp. 814-819. Extended version available as Tech. Report CMU-CS-88-157.

5. Aviad, Z. Object Detection by the Analysis of Imperfect Sequences. In preparation.

6. Irvin, R. B., McKeown, D.M. Methods for Exploiting the Relationship between Buildings and their Shadow in Aerial Imagery. SPIE Proceedings Image Understanding and the Man-Machine Interface II, January, 1989. Also available as CMU Computer Science Technical Report CMU-CS-88-200.

7. Leclerc, Y. and W. Zucker. The local structure of image discontinuities in one dimension. Seventh International Conference on Pattern Recognition, Proceedings, Montreal, 1984, pp. 46-48.

8. Lopresti, D. P. "P-NAC: A Systolic Array for Comparing Nucleic Acid Sequences". *IEEE Computer 20*, 7 (July 1987), 98-99.

9. McKeown, D.M., Denlinger, J.L. Map-Guided Feature Extraction from Aerial Imagery. Proceedings of Second IEEE Computer Society Workshop on Computer Vision: Representation and Control, May, 1984. Also available as Technical Report CMU-CS-84-117.

10. Nevatia, R. and K. Ramesh Babu. "Linear Feature Extraction and Description". *Computer Graphics and Image Processing 13* (1980), 257-269.

11. Witkin, A. P. Scale-space filtering. Proc. 7th International Joint Conference on Artificial Intelligence, 1983, pp. 1019-1022.