1-1-1976

Experiments in Iterative Enhancement of Linear Features

Gordon J. VanderBrug

Reprinted from

Symposium on Machine Processing of Remotely Sensed Data

June 29 - July 1, 1976

The Laboratory for Applications of
Remote Sensing

Purdue University
West Lafayette
Indiana

IEEE Catalog No.
76CH1103-1 MPRSD

EXPERIMENTS IN ITERATIVE ENHANCEMENT OF LINEAR FEATURES

Gordon J. VanderBrug

Computer Science Center
University of Maryland
College Park, Maryland 20742

I. ABSTRACT

Lines and curves in an image are detected locally by a template-matching process which determines the "line-ness" value of the image at each point, in a set of orientations. The output of the detection process is the strongest of these values at each point, and the orientation that gave rise to this value. The results of this approach tend to be noisy, but their noisiness can be reduced by examining, for each point, the values at nearby points, in the direction defined by the preferred orientation, and increasing the point's value if the nearby points have high values and similar orientations. Iteration of this reinforcement process leads to further noise reduction. Several variations on this scheme are presented. The preferred orientations can also be "sharpened" by examining the orientation at nearby points (in the preferred direction) and biasing it toward their average. Experimental results using these methods are obtained for LANDSAT and SKYLAB images containing many linear features.

II. INTRODUCTION

Lines and curves in an image can be detected locally by a template-matching process, in which a set of operators, each having a preferred orientation, is applied to the image at each point. The operators may be linear or nonlinear, but it has been found that a nonlinear technique generally yields better detection results [1]. The output of the detection process at each point is a vector whose magnitude is the strongest of the operators' outputs at that point, and whose orientation is the one for which this strongest output was obtained. This method of line detection has been tested on LANDSAT and SKYLAB imagery containing linear features such as roads, rivers, and lineaments [2].

The results of any local line detection operation will be somewhat noisy, since image points not lying on lines or curves may appear locally line-like, and vice versa. This report describes some methods of reducing the noisiness by examining, for each point, the output values at nearby points, in the direction defined by the preferred orientation, and increasing the point's value if the nearby points have high values and similar orientations. Further noise reduction is obtained by iterating this process. Several variations on this approach are described, and results are presented for LANDSAT and SKYLAB images containing many linear features. A method of "sharpening" the preferred orientation at each point, by examining the orientations at nearby points (in the preferred direction) and biasing it toward their average, is also tested.

A more elaborate method of improving line detection results by iterative local processing is presented in [3]. Here a probability vector is initially associated with each point; its components are the probabilities that a line passes through the point in each of a set of orientations, or that no line is present. At subsequent steps, the probability vectors in the neighborhood of the point are examined, and the line and no-line probabilities at the point are adjusted accordingly: high line probabilities in a given direction and orientation reinforce the line probability for that orientation, while high no-line probabilities reinforce the no-line probability. This method has led to successful results in a variety of tests, as described in [3]. The methods described in the present report, which were developed concurrently, can be regarded as simplifications of the approach used in [3]; the present methods reinforce line intensity, and sharpen line direction, independently of one another.

Section III describes several variations on the line intensity enhancement process, and the results obtained using them. Section IV discusses line orientation sharpening. Section V recapitulates the results and recommends directions for future studies.

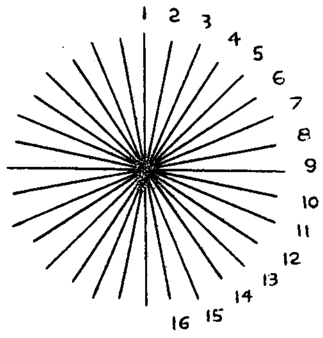
III. LINE INTENSITY REINFORCEMENT

The reinforcement algorithm is designed to iteratively enhance the output of a line detector by rewarding (increasing the intensity of) those points on a line, and punishing (decreasing the intensity of) those points which are not on a line.

The algorithm assumes the existence of a line detector that produces line orientation values as well as line intensity values. At each point in the array of line intensity values, it examines neighboring points in the directions indicated by the orientation value. It increases the intensity value according to the number and intensity of the points that it sees which are reasonably well aligned with that orientation. Similarly, it decreases the point's intensity value according to the number and intensity of the points that it sees which are not well aligned with that orientation. Thus, the amount of reward or punishment depends on:

1) the number of points seen
2) the intensities of these points
3) the alignment of these points

The reinforcement algorithm was written to accept the output of a nonlinear line detection operation [2]. This nonlinear detector generates output in 16 orientations, as follows:



These orientations are used to define masks, which are regions (or neighborhoods) of a picture point within which the point looks for information to use as a basis for rewarding or punishing its line intensity value. The masks used for Version I of the reinforcement program are shown in Figure 1. (Versions II, III, and IV will be described later.) Only half of the masks for orientations 1 through 5 are shown. The other half of each mask is constructed by reflecting across the line orthogonal to the orientation. Masks 6 through 16 are constructed by rotating and reflecting the masks shown in Figure 1.

For each picture point the reinforcement algorithm examines all of the points within the region defined by its orientation (i.e., within the mask). A vector $(w_0,\ldots,w_8)$ is used to compute the measure of reward/punishment defined by the points within the mask region. Let $p_1,\ldots,p_n$ and $q_1,\ldots,q_n$ be the line value intensities and orientations of the points within the mask region defined by a point whose intensity is p and orientation is q. The quantity

$$R = \sum_{i=1}^{n} w_{||q_i-q||} \cdot p_i ,$$

where $0 \le ||q_i-q|| \le 8$ represents the absolute difference in orientation between the points p_i and p, is a sum of the intensity of the points "seen by p", weighted according the differences in orientation. The weights contribute to the extent to which p is reinforced as follows:

$w_j > 0$ - reward

$w_j = 0$ - neither reward nor punish

$w_j < 0$ - punish

A possible (symmetric) set of weights might be

w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
2	2	2	1	0	-1	-2	-2	-2

Thus, if R is a large positive number, the point p should be rewarded substantially; if it is a large negative number, it should be punished substantially; and similarly for intermediate values.

The actual change in p is computed from R as follows. Let

$$d_{pos} = \frac{1}{n_{pos}} \sum_{w_i>0} w_i$$

$$d_{neg} = \frac{1}{n_{neg}} \sum_{w_i<0} |w_i|$$

where n_{pos} and n_{neg} are the numbers of positive and negative weights, respectively. If $R \geq 0$ then

$$p \leftarrow p + \frac{R}{d_{pos} \cdot PFAC} \cdot (63-p)$$

If $R < 0$ then

$$p \leftarrow p + \frac{R}{d_{neg} \cdot NFAC} \cdot p$$

where PFAC and NFAC are parameters which are used to control the magnitude of the reward and punishment. If $\frac{R}{d_{pos} \cdot PFAC} \geq 1$ then p is set to 63; similarly, if $\frac{R}{d_{neg} \cdot NFAC} < -1$ then p is set to 0. (Here 0 and 63 are the limits of the grayscale used to represent line intensity values.)

The program was run on a 127x127 point portion of a LANDSAT image of Kentucky (Figure 2) using the following choices of weights and parameters:

1) Symmetric weights:

   2 2 2 1 0 -1 -2 -2 -2

   PFAC = NFAC = 630

2) Same, except with NFAC = 320

3) Negatively biased weights:

   3 2 1 0 -1 -2 -3 -3 -3

   PFAC = NFAC = 630

The value 630 chosen for PFAC and NFAC (except in case (2)) was arrived at as follows: For a strong line in the preferred orientation through the point, about 15 of the 24 mask points should have this orientation (i.e., zero orientation difference); if these points have intensity 42, we have an R contribution of about 15x42.

Results of experiments using these three sets of weights and parameter values are shown in Figures 3-5. The results are all quite similar, though there is a slightly greater elimination of noise when the negatively biased weights are used (compare Figure 5d with Figure 3d).

Three other versions of the reinforcement scheme were also tested:

Version II uses the same masks as Version I (as shown in Figure 1), but it only rewards, never punishes -- i.e., it uses non-negative weights of the form
2 2 2 1 0 0 0 0 0.

Version III uses the shorter, wider masks shown in Figure 6, and only rewards, never punishes.

Version IV uses masks that do not involve close neighbors of the given point (see Figure 7); otherwise, it is identical to Version I.

Results of experiments using these versions are shown in Figures 8-10. As would be expected, Version II does not suppress noise (Figure 8a-b), but the curves that it enhances can be extracted by thresholding (Figure 8c-d). Similar remarks apply to Version III (Figure 9), which seems to do somewhat better than Version II (compare Figures 8c-d and 9c-d). Version IV does not appear to do as well as Version I (compare Figures 3a-d and 10a-d). But all the versions perform quite comparably, which is an indication of the robustness of the iterative approach.

Figure 11a shows a Skylab image (from the S190B camera) of suburban Washington, D. C. The output of the line detector and four iterations of Version II, which are shown in Figure 11b-f, demonstrate that the major roads and segments of the suburban streets are enhanced. A version with no punishment was chosen here because the detector output appears to have very little noise.

The progress of the reinforcement scheme can be monitored by examining the histogram of intensity values at each iteration. Such histograms for the case in Figure 3 are shown in Figure 12; note that they become more bimodal with each iteration.

IV. LINE ORIENTATION ADJUSTMENT

The orientational adjustment algorithm presented here is a parallel, iterative algorithm which uses information about the line orientations at a point's neighbors to adjust the line orientation at the point so as to improve the alignment. The neighbors that are used to obtain this information are defined by masks such as those shown in Figures 1, 6, and 7.

The computation at a point p proceeds as follows. The orientations of the lines at all the points of p's mask are determined, and those orientations which do not differ from p's orientation by too much are averaged. Points whose orientations differ from that at p by large amounts are excluded from this computation because their influence on p would lead to undesired results. However, if p's mask contains too many such points, one must question the existence of a line at p itself. Also, p's existence is questionable when its mask contains too few points whose orientations are reasonably well aligned with its own. Therefore, the algorithm deletes a point if the number of nonaligned mask points is too large, or if the number of aligned points is too small. Each point which survives has its orientation adjusted by some fraction of the difference between the original orientation and the computed average orientation over the mask.

The iterative application of the above algorithm allows for increasingly more global knowledge about orientation to be used in the orientational alignment process. Since the magnitude of the adjustment at each stage will in general be a fraction of a discrete orientation (16 orientations were used), the array produced at each iteration is real valued. The progress at each iteration can be monitored by examining 1) the number of points set to 0 because of too few aligned points, 2) the number of points set to 0 because of too many nonaligned points, 3) the number of points set to 0 because of both (1) and (2), and 4) the average adjustment in the orientation of the points which survive.

The input picture points have the 16 orientations shown in Section 2. To average two orientations, one cannot simply add and divide by 2: the average of orientations 2 and 16 is not 9, but 1. Rather, one must average in such a way that the smaller angle between the orientations (which is less than $\pi/2$) is bisected. This is equivalent to adding the orientations modulo 16, so that the sum of orientations 2 and 16 is 2, rather than 18, and their average is $2/2 = 1$. Alternatively, we can average the orientations in the ordinary way, and then represent the average modulo 8, so that the average of 2 and 16 is 9, which is the same as 1 modulo 8.

To compute the average of n orientations in this way, we must maintain a partial average value, so that at each stage the decision as to whether or not the next orientation should be adjusted modulo 16 before averaging can be made. Suppose that a_i is the average of the first i orientations, and that b is the i+1st orientation. The following algorithm computes a_{i+1}:

```
/* Adjust b if necessary */
    If $(a_i-b) > 8$ then $b \leftarrow b + 16$, else if
                        $(a_i-b) < -8$
            then $b \leftarrow b - 16$

/* Average in b */
    $a_{i+1} \leftarrow \frac{i \cdot a_i + b}{i+1}$

/* Adjust computed average so that
            $1 \le a_{i+1} \le 16$ */

    If $a_{i+1} > 16$ then $a_{i+1} \leftarrow a_{i+1} - 16$,

    If $a_{i+1} < 1$ then $a_i+1 \leftarrow a_{i+1} + 16$
```

This algorithm can be viewed as a procedure, Aver, which computes a_{i+1} from a_i and b; that is,

$$a_{i+1} \leftarrow \text{Aver}\ (a_i, b)^*$$

In addition to the computation of the average orientation, the alignment algorithm uses a difference-in-orientation computation which we denote by $||q-m||$. Here $||q-m||$ represents the signed difference modulo 16 between orientations q and m. For example,

$$||16 - 2|| = -2$$

$$||2 - 16|| = 2$$

$$||8 - 4|| = 4$$

This computation is defined as:

```
$||q-m|| \equiv$ if $(q-m) > 8$ then $q-m-16$
        else if $(q-m) < -8$ then $q-m+16$
            else $q-m$
```

<u>Input Variables</u>

d - if the absolute value of the difference in orientations at two points is $\le$ d the points are considered aligned, otherwise they are nonaligned.

t_1 - if the number of aligned points seen is $\le t_1$ then delete the point.

*Note that the average of two perpendicular orientations is ambiguous; e.g., the average of 4 and 12 could be either 8 or 16. The program resolves this ambiguity in an arbitrary manner.

t_2 - if the number of nonaligned points seen is $\geq t_2$ then delete the point.

w - the adjustment fraction.

Intermediate Variables

q - orientation at the point p being adjusted.

m - orientation at a point in the mask of p.

a - average (partial or final) of orientations of the points in the mask of p.

i,j - counters used to determine whether or not a point should be deleted.

For each point p we proceed as follows:

```
/* Compute average of points in p's mask */
   $a \leftarrow 0$; $i \leftarrow 0$; $j \leftarrow 0$
For each point m of the mask
If $\mathrm{abs}(||q-m||) \leq d$
   then $a \leftarrow \mathrm{Aver}(a, i, m)$
         $i \leftarrow i+1$
   else $j \leftarrow j+1$

/* Delete points which do not have enough
   aligned points, or have too many non-
   aligned points, in their masks */
   If $i \leq t_1$ or $j \geq t_2$ then $q \leftarrow 0$
/* Adjust the orientation of the point */
   $q \leftarrow q + w \cdot ||a-q||$
   If $q > 16$ then $q \leftarrow q - 16$
      else if $q < 1$ then $q \leftarrow q + 16$
```

The program also maintains the following statistics for each iteration:

counter 1 - number of points set to 0 because there were not enough aligned points in the mask

counter 2 - number of points set to 0 because there were too many nonaligned points in the mask

counter 3 - number of points set to 0 for both of these reasons

counter 4 - number of points whose orientations were adjusted

aver. diff. - the average orientation difference, $||a-q||$, for the points that were adjusted; w times aver. diff. gives the average adjustment

Several versions of the alignment algorithm have been implemented. The first uses the masks shown in Figure 1, while the second uses the shorter, wider masks shown in Figure 13. A third version, designed but not implemented, weights the difference between the orientations at p and each mask point by the line value of the mask point. Specifically, let q, m, a, i be as above, let A be the partial sum of the line values, and let M be the line value of m; then we define Procedure Aver (a, i, m, M) as follows:

```
If $(a-m) > 8$ then $m \leftarrow m + 16$
   else if $(a-m) < -8$ then $m \leftarrow m - 16$

$a \leftarrow \frac{A \cdot a + M \cdot m}{A+M}$

If $a > 16$ then $a \leftarrow a - 16$
   else if $a < 1$ then $a \leftarrow a + 16$
```

The results of running the first two versions of the algorithm on the window shown in Figure 2 are summarized in Figures 14 and 15. In all of these examples, the DIFF parameter was set at 4, and the weight factor w at 1/2. It is seen that the number of points adjusted (Counter 4), and the average difference, decrease in every case, so that the process appears to be convergent. Pointwise numerical results for a portion of the window are shown in Figure 16; here the orientations 1,...,16 are represented by the characters 1,...,9,A,...,G. Some "sharpening" of the orientations is apparent in these examples.

V. CONCLUSIONS

The experiments reported here confirm the utility of local iterative techniques for enhancing line intensities, and adjusting line orientations, in the output of a line detection algorithm. Variations in the masks and reinforcement schemes used seem to have little effect on the results, which indicates that the approach is robust. It could be implemented very straightforwardly using parallel hardware. Even on a sequential machine, it requires only a single pass through the line value (or orientation) array per iteration, so that its computational cost per iteration is comparable to that of the original line detection process -- namely, order (picture area) operations.

There are a number of possibilities for extending this work. One extension is to combine the two approaches by having a point reinforce its intensity and align its orientation at the same time. The alignment algorithm can easily be made a function of the intensities of the points seen, as well as their orientations. One might consider making the level of reinforcement inversely proportional to the variance of orientations of the points seen.

Another possibility for future work is to give the algorithm some capability for extending lines and filling gaps. Presently no such capability is present because neither algorithm does any computation at points where the line detector output (i.e., the intensity) is zero. Providing for a line extension capability in a brute force manner by examining all orientations at all points with zero intensity would be computationally very expensive. A less costly approach would be based on the idea that a point which lies at the end of a line can recognize itself as an endpoint, and one can then invoke a (local) procedure, at the points in the picture which are natural extensions of the line, in an attempt to extend the line. This approach would be computationally feasible because, (1) it would not be invoked at every zero-intensity picture point, and (2) it has some orientational information at points where it is invoked. There are some potential pitfalls in this approach, especially where sharply curved lines are present. These problems may be manageable by (1) providing wide fields of view at the endpoints, and (2) insisting that lines cannot be extended for more than a certain number of steps except in instances of gap filling.

Another aspect of line enhancement which can be incorporated into these iterative techniques is thinning. In thinning one looks in directions that are orthogonal to the directions used for reinforcement and alignment. The action taken should serve to enhance those points which are local maxima, since they lie at the center of the line. This approach has been used successfully on edge detector output in[4].

Our experiences indicate that iterative enhancement techniques appear promising, and extensions and additional experiments, using a wider variety of input data, are warranted.

REFERENCES

[1] G. J. VanderBrug, Semilinear line detectors, Computer Graphics and Image Processing 4, 1975, 287-293.

[2] G. J. VanderBrug, Line detection in satellite imagery, Proc. Symp. on Machine Processing of Remotely Sensed Data, June 1975; IEEE Trans. on Geoscience Electronics, vol. GE-14, no. 1, January 1976.

[3] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, Applications of relaxation labelling, 1: Line and curve enhancement, TR-419 Computer Science Center, University of Maryland, October 1975.

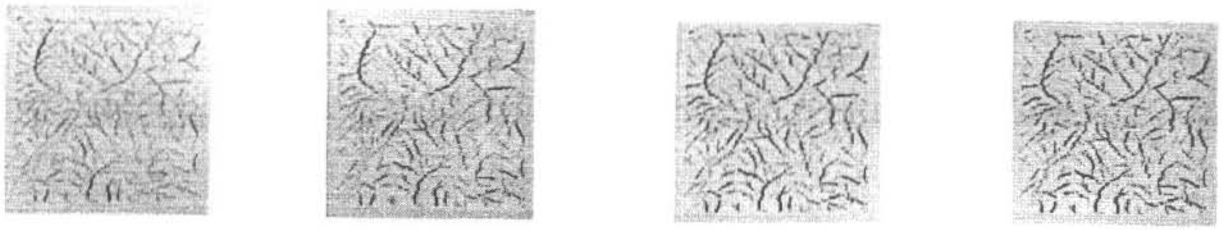
[4] R. B. Eberlein, An iterative gradient edge detection algorithm, TR-382, Computer Science Center, University of Maryland, May 1975.

```
xxx    xx       xx        xx         xx
xxx    xx      xx        xx         xxx
 x     xx      xx        x          xx
 x     x       x        xx         x
 x     x     xx        xx        x
 x    xx     x         x        x
 x    x      x       x         x
 x    x     x        x       x
 p    p     p        p       p
```

Figure 1. Masks used in Version I of the line intensity enhancement algorithm.

(a) (b)

Figure 2. LANDSAT image used in experiments. a) - Original image. b) - Output of nonlinear line detection process applied to (a).

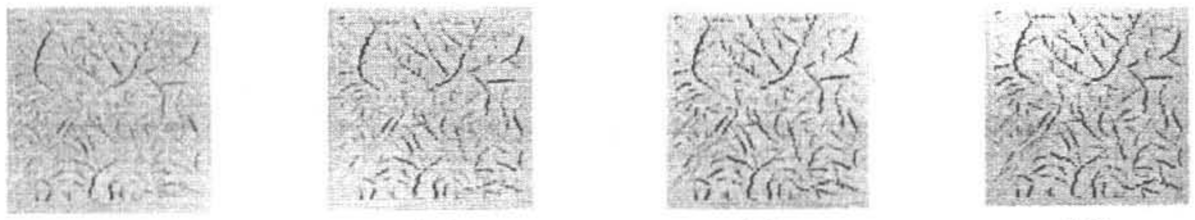


(a) (b) (c) (d)

Figure 3. Four iterations of Version I of the line intensity enhancement algorithm, applied to Figure 2b:
Weights - 2, 2, 2, 1, 0, -1, -2, -2, -2
PFAC = NFAC = 630

(a) (b)

Figure 4. Two iterations of Version I of the line intensity enhancement algorithm, applied to Figure 2b:
Weights same as in Figure 3; PFAC = NFAC = 320.



(a) (b) (c) (d)

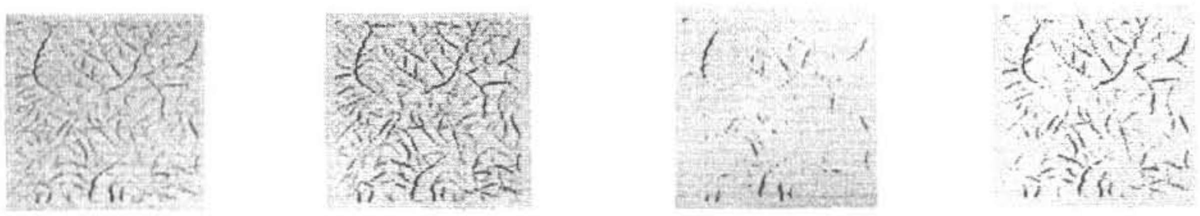
Figure 5. Four iterations of Version I of the line intensity enhancement algorithm, applied to Figure 2b:
Weights - 3, 2, 1, 0, -1, -2, -3, -3, -3
PFAC = NFAC = 630

```
xxx        xxx          xxx           xxx           xx
xxx        xx          xxx           xxx           xxx
 x         xx          xx            xx           xxx
 x         x           xx           xx           xx
 p         p           p            p           p
```

Figure 6. Masks used in Version III of the line intensity enhancement algorithm.

```
xxx           xx               xx                xx                   xx
xxx           xx              xx                xx                   xxx
 x            xx              xx                x                    xx
 x            x               x                xx                   x
 0            0             00                00                  0
 0           00             0                 0                  0
 0           0              0                0                  0
 0           0             0                 0                 0
 p           p             p                p                 p
```

Figure 7. Masks used in Version IV of the line intensity enhancement algorithm. Points indicated by 0's were not used.



(a) (b) (c) (d)

Figure 8. Two iterations of Version II of the line intensity enhancement algorithm, applied to Figure 2b:
Weights - 2, 2, 2, 1, 0, 0, 0, 0, 0.
a) First iteration; b) Second iteration;
c) Result of thresholding (a) at 35;
d) Result of thresholding (b) at 35.

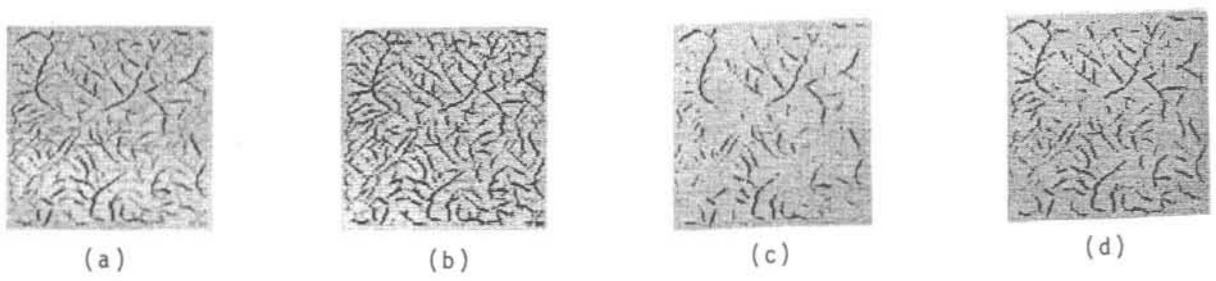


Figure 9. Two iterations of Version III of the line intensity enhancement algorithm, applied to Figure 2b; weights and thresholds same as in Figure 8.

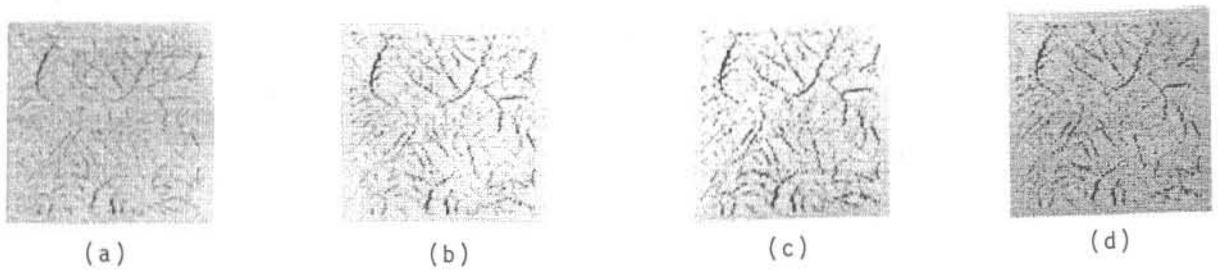


Figure 10. Four iterations of Version IV of the line intensity enhancement algorithm, applied to Figure 2b; weights same as in Figure 3.

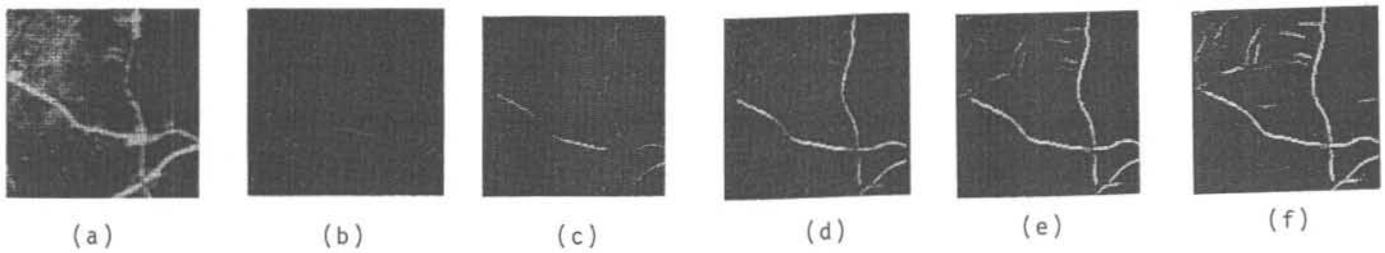


Figure 11. Experiments on a SKYLAB image of suburban Washington DC. a) Original image; b) Output of nonlinear line detector; c) - f) Four iterations of Version II of the line intensity enhancement algorithm.

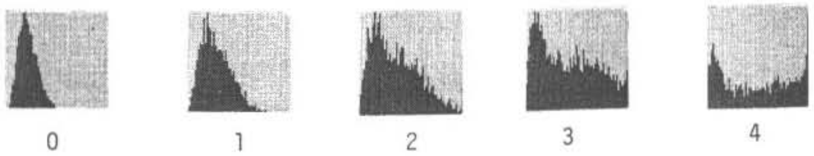


Figure 12. Histograms of the intensities in Figures 2b and 3 (= iterations 0-4).

```
xxx        xxxx        xxxx          xxx          xx
xxx        xxxx        xxx          xxx          xxx
xxx        xxx         xxx          xxx         xxx
xxx        xxx        xxx          xxx         xxx
xxx        xx         xxx         xxx         xxx
xxx        xx         xx          xx         xxx
 p         p          p           px         px
```

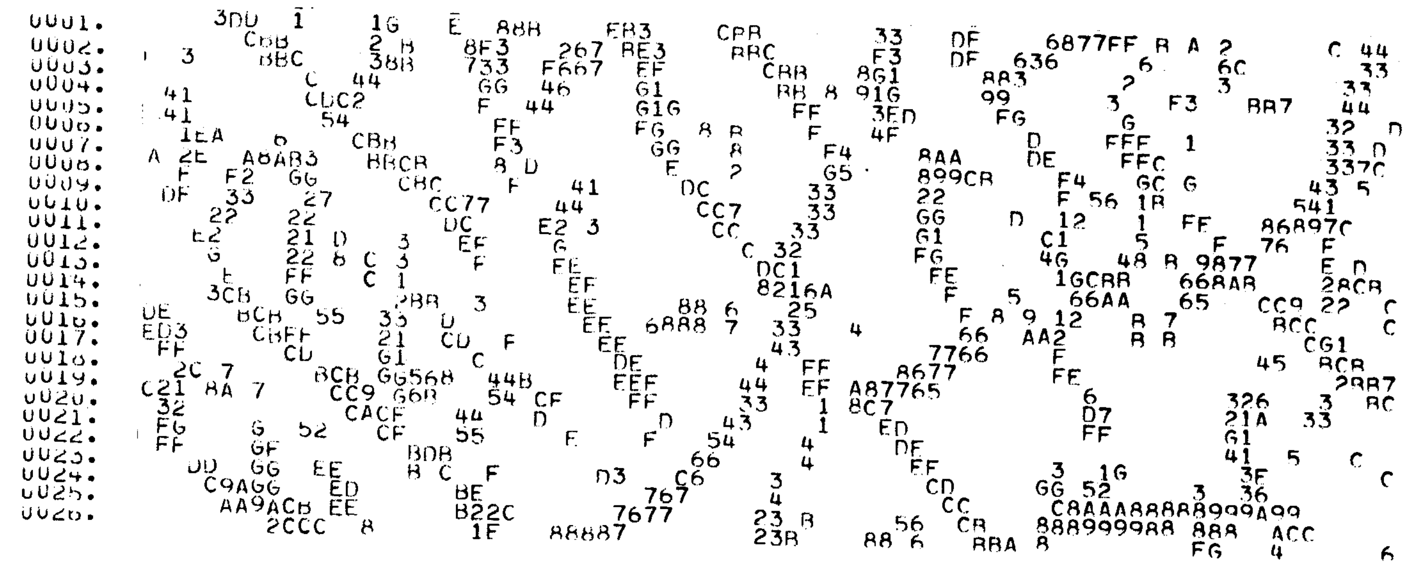
Figure 13. Masks used in Version II of the line orientation alignment algorithm.

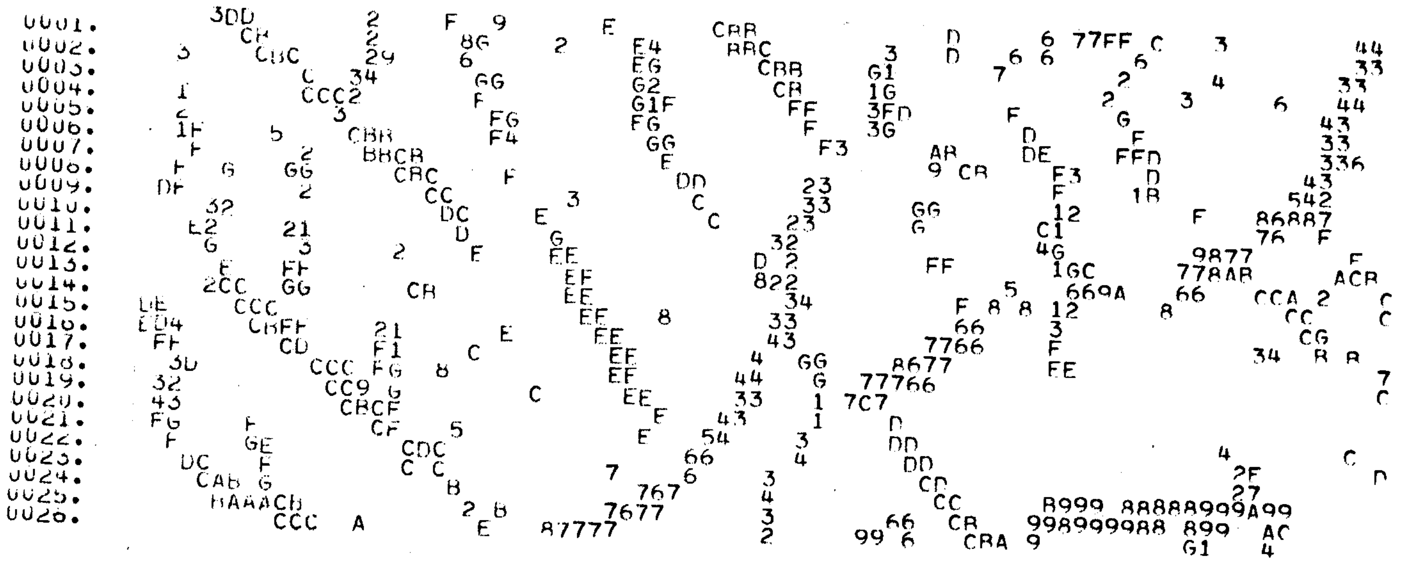
Parameter values		Iteration	Counters				AVERAGE DIFFERENCE
t_1	t_2		1	2	3	4	
7	5	1	1718	155	671	1586	.702
		2	1240	0	7	339	.475
3	5	1	383	545	281	2921	.859
		2	700	47	60	2114	.772
0	24	1	33	0	0	4097	.999
		2	63	0	0	4034	.862

Figure 14. Experiments with the first version of the alignment algorithm.

Parameter values		Iteration	Counters				AVERAGE DIFFERENCE
t_1	t_2		1	2	3	4	
0	36	1	5	0	0	4474	.874
		2	6	0	0	4462	.736
6	12	1	699	10	35	3735	.786
		2	613	1	7	3114	.684
		3	340	1	3	2770	.531
		4	250	0	0	2520	.415
7	7	1	584	503	424	2968	.750
		2	840	12	29	2087	.658

Figure 15. Experiments with the second version of the alignment algorithm.





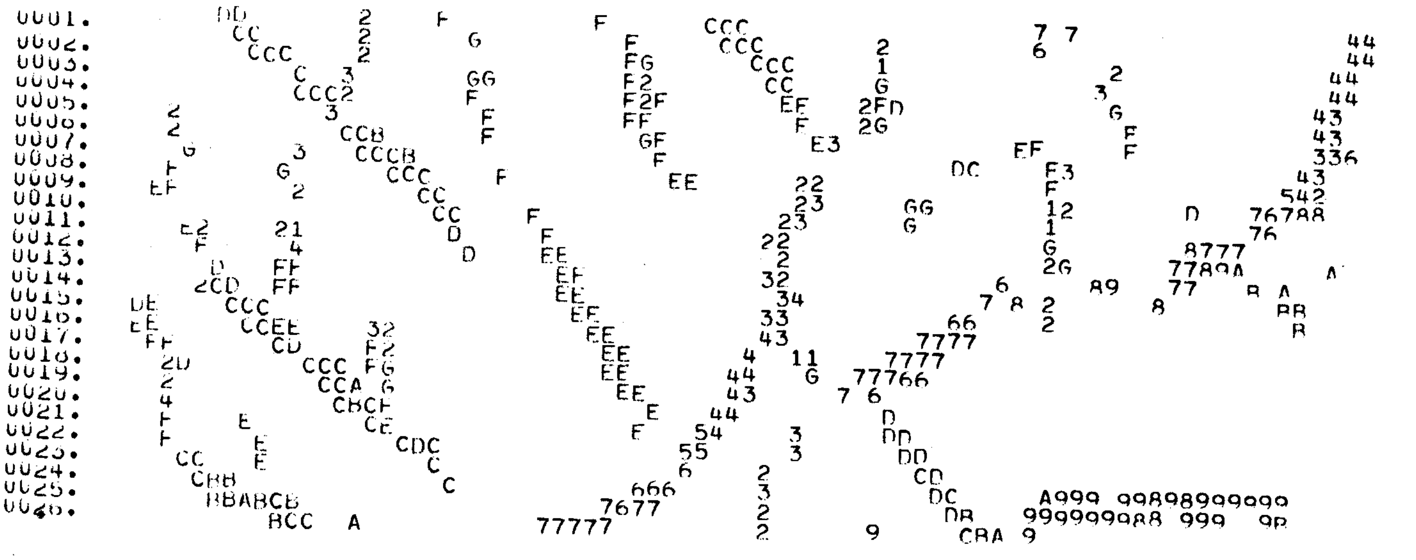


Figure 16. Results (from top to bottom) of two iterations of Version II of the line orientation alignment algorithm. Parameters $t_1 = 7$, $t_2 = 7$.

APPENDIX. COMPUTATION OF 16 ORIENTATIONS

The nonlinear line detectors used in these experiments were based on comparisons between average gray levels measured over 2-by-2 cells. For example, the vertical line detector was computed as follows: Let A,B,...,I be a block of nine 2-by-2 cells as illustrated;

```
A B C
D E F
G H I
```

then we say that a vertical line has been detected if six conditions are satisfied:

$|B|>|A|$, $|B|>|C|$, $|E|>|D|$, $|E|>|F|$, $|H|>|G|$, and $|H|>|I|$, where the bars denote "average gray level of".

The 16 orientations were obtained by using similar configurations of cells aligned in other directions. The following table shows the arrangements of the cells corresponding to each orientation.

```
Orientation           Arrangement(s)

    1                     A B C
                          D E F
                          G H I

    2        A B C                      A B C
             D E F          or         D E F
            G H I                      G H I

    3        A B C                       A B C                     A B C
             D E F          or         D E F           or         D E F
            G H I                      G H I                     G H I

    4        A B C                        A B C
            D E F           or          D E F
          G H I                        G H I

    5         A B C                         A
            D E F           or            D B
          G H I                         G E C
                                        H F
                                        I

                                            A
                                          D
    6             A                     G   B
                D                         E
                  B                     H   C
              G E           or            F
                  C                     I
              H F
              I

                                                                     A
                                                                   D
    7           D A                         A                    G   B
                                                                   E
              G E B         or          G D B          or        H   C
                                                                   F
              H F C                     H E C                    I
              I                         I F

                                                                     A
                                          D A                    G D
    8                                   G                            B
                                          E B          or        H E
                                        H                            C
                                          F C                    I F
                                        I

    9                                   G D A
                                        H E B
                                        I F C

   10         G D                       G
                  A                       D A
              H E           or          H
                  B                       E B
              I F                       I
                  C                       F C
```

```
Orientation        Arrangements(s)

   11      G D                       G                         G
                                                                 D
           H E A         or          H D A          or         H   A
                                                                 E
           I F B                     I E B                     I   B
                                                                 F
               C                       F C                         C

   12      G
             D
           H   A         or          G
             E                         D
           I   B                     H   A
             F                         E
               C                     I   B
                                       F
                                         C

   13      G                         G H I
           H D                         D E F
           I E A         or              A B C
             F B
               C

   14      G H I                     G H I
             D E F       or           D E F
              A B C                     A B C

   15      G H I                     G H I                     G H I
             D E F       or          D E F          or          D E F
             A B C                     A B C                     A B C

   16        G H I                   G H I
              D E F      or          D E F
              A B C                   A B C
```