INPE ePrint: sid.inpe.br/yolanda/2004/05.24.18.48 v1 2004-05-25

Distance transform as a basis for contour line interpolation and intermediate line generation

ARLEY F. DE SOUZA, GERALD J. F. BANON

INPE - Instituto Nacional de Pesquisas Espaciais
Av. Dos Astronautas, 1758, Jd. Da Granja, 12227-010 São José dos Campos, SP, Brasil
{arley, banon}@dpi.inpe.br

Abstract. The objective of this work is to show how the distance transform can be used to perform contour line interpolation, and intermediate line generation. More precisely, it is shown that these problems can be solved by computing two sets of local image distances, in only two steps: one in raster mode and another in antiraster mode. Besides its efficiency, the algorithm leads to contour line interpolation without flat zones.

Keywords. Image processing, mathematical morphology, distance transform, contour line interpolation, intermediate line.

1. Introduction

The *distance transform* (DT) is an operator that converts binary images into grey-level images, where each foreground pixel is assigned to its distance to the background. The DT result is named *image distance* (ID). The DT is a classical operator in image processing, and its use is of fundamental importance in the implementation of some mathematical morphology operators like the binary image dilations and erosions [1], the medial axis extraction [2-6] for skeletonization, and the watershed for segmentation [6]. In the case of binary dilation and erosion, the DT allows their sequential implementation without the overhead of successive iterations in the image, even though parameterized by large structuring elements.

The DT can be implemented exploring different kind of metrics: City-Block, Chessboard, Octagonal [7] (the Octagonal metric is obtained from the combination of the City-Block and Chessboard metrics), Chamfer [8, 9], and Euclidean [10]. There is yet the quasi-Euclidean DT [11] that is much faster than the Euclidean DT, and has the Euclidean metric precision for almost all the mapped pixels.

The paper main objective is to show how the combination of IDs of binary images representing a color image can be used, in order to perform contour line interpolation and intermediate line generation.

An illustrative example in one dimension is shown in Figure 1(a - c). In these figures, the foreground is the segment between p and q, resulting in an image with a background consisting of two *connected components* (CC), one on the left of p (denoted by P) and another on the right of q (denoted by Q).

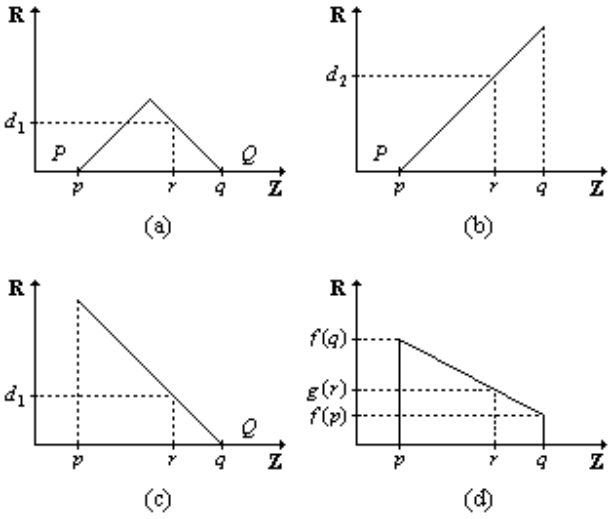

Figure 1 (a - c) Euclidean images distance of $[p, q]$, $[p, +\infty]$ and $[-\infty, q]$, and (d) linear interpolation between p and q.

Figure 1a shows the Euclidean ID of the foreground. Figure 1b shows the Euclidean ID of the image, which background is just the first CC P (ignoring the second CC Q), and Figure 1c shows the opposite. One observes that Figure 1a is precisely the intersection of Figures 1b and 1c. Nevertheless, Figure 1a is useless in order to perform interpolation between the pixels $(p, f(p))$ and $(q, f(q))$, because the ID just gives the distance to $P \cup Q$, and not the distance to P and Q. Actually, what is needed are both IDs of Figures 1b and 1c as it can be seen from the linear interpolation expression $g(r)$ at a point r in the foreground:

Equation 1 $g(r) = f(p) + (f(q) - f(p)) * d(r, P) / (d(r, P) + d(r, Q))$,

where $d(r , X) = \min\{d(r, x): x \in X\}$, and d is a metric (for more details see Section 2).

Figure 1d shows the interpolation result of Equation 1.

In the next section the concept of distance and metric are recalled and a DT implementation is given. The third section introduces the color image decomposition in terms of binary images. The forth presents the interpolation for contour line images, and the fifth the generation of intermediate lines in contour line images.

2. Distance Transform

The DT is based on the concept of distance and metric which are recalled bellow.

The mapping d of $\mathbf{Z}^2$ to $\mathbf{R}$ is a *distance* if for any x and $y \in \mathbf{Z}^2$, the following conditions are satisfied:

(i) $d(x, y) = d(y, x)$;

(ii) $d(x, y) \geq 0$;

(iii) $d(x, y) = 0 \Leftrightarrow x = y$.

A distance d is a *metric* if for any x, y and $z \in \mathbf{Z}^2$, it satisfies the triangle inequality:

(iv) $d(x, y) \leq d(x, z) + d(z, y)$.

Examples of metric are the City-Block, Chessboard and Euclidean metrics, d_4, d_8 and d_E, defined by, for any $x = (x_1, x_2)$ and $y = (y_1, y_2)$ of $\mathbf{Z}^2$,

$d_4(x, y) = |x_1 - y_1| + |x_2 - y_2|$;

$d_8(x, y) = \max\{|x_1 - y_1|, |x_2 - y_2|\}$;

$d_E(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

Based on a metric d one can define the respective distance transform.

Let E be a rectangle of $\mathbf{Z}^2$, and $K = \{0, 1\}$. A binary image f is a mapping from E to K, where for any point $(x, y) \in E$, $f(x, y) = 1$ if (x, y) belongs to the foreground and $f(x, y) = 0$ otherwise.

Let A be a subset of E, and A^c its complement. The ID of A with respect the metric d is the mapping DT_A from E to $\mathbf{R}$ given by, for any $x \in E$,

$$\mathrm{DT}_A(x) = \begin{cases} d(x, A^c) & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

The mapping $A \mapsto \mathrm{DT}_A$ is called the *distance transform*.

In this paper, the DT implementation is based on the Chessboard metric. The corresponding pseudocode is presented in Pseudocode A below.

The distance propagation occurs in two iterations over the image, one in raster mode (from left to right and from top to bottom) and the other in antiraster mode (from right to left and from bottom to top).

Let m be the number of lines of E, let n be the number of columns, and let (0, 0) be the upper left point of E.

Let denote $N_r((x, y), f) = \{f(x - 1, y - 1), f(x - 1, y), f(x - 1, y + 1), f(x, y - 1)\}$ and $N_a((x, y), f) = \{f(x, y + 1), f(x + 1, y - 1), f(x + 1, y), f(x + 1, y + 1)\}$ two subsets of pixel values.

Pseudocode A

```
(1) create a matrix M of the same size as E;
/* fill out the first and last columns */
(2) for x varying from 0 to m – 1 do
      if f(x, 0) is equal to 1 then M(x, 0) = 1;
      else  M(x, 0) = 0;
      if f(x, n – 1) is equal to 1 then M(x, n – 1) = 1;
      else  M(x, n – 1) = 0;
    end do;
/* fill out the first and last rows */
(3) for y varying from 1 to n – 2 do
      if f(0, y) is equal to 1 then M(0, y) = 1;
      else  M(0, y) = 0;
      if f(m – 1, y) is equal to 1 then M (m – 1, y) = 1;
      else  M(m – 1, y) = 0;
    end do;
/* process in raster mode */
(4) for x varying from 1 to m – 2 do
      for y varying from 1 to n – 2 do
        if f(x, y) is equal to 1 then
          M(x, y) = min{Nr((x, y), M)} + 1;
        else  M(x, y) = 0;
      end do;
    end do;
/* process in antiraster mode */
(5) for x varying from m – 2 to 1 do
      for y varying from n – 2 to 1 do
        if M(x, y) is greater than 1 then
          M(x, y) = min{M(x, y), min{Na((x, y), M)} + 1};
      end do;
    end do.
```

Pseudocode A returns the ID of the foreground $f(\{1\})$ of f, that is, $M = \mathrm{DT}_{f(\{1\})}$.

Moreover, it can be easily expanded for color images. In this case, two substitutions are necessary. The first one consists of substituting (2) and (3) by a code that fills out the border of matrix (M) with value 1. The second one consists of changing the *if* and *else* condition in (4) by:

```
if f(x, y) is equal to Nr((x, y), f) then
  M(x, y) = min{Nr((x, y), M)} + 1;
else M(x, y) = 1.
```

There is no change in (5). One should observe that, differently of Pseudocode A, this last pseudocode applied to binary images returns the ID of both the foreground $f(\{1\})$ and background $f(\{0\})$ of f, that is, $M = DT_{f(\{1\})} \vee DT_{f(\{0\})}$.

3. Color image decomposition

A color image is a mapping from E to a set of colors.

By background of a color image, it is meant here, the set of positions of pixels having a predefined color (for example, the set of positions of pixels represented by – is considered as the background of the Figure 2a image). Furthermore, to each other color in the image, one assigns the set of positions of pixels having that color. These sets are called color objects in the color image domain (Figure 2a shows two color objects, one is the set of positions of pixels represented by □, and the other by ■).

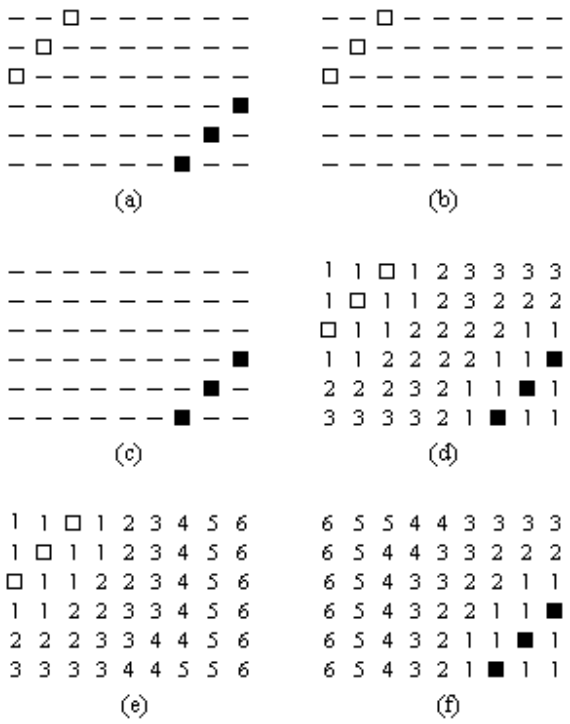

Figure 2 Color image decomposition and Chessboard IDs. (a) Part of a color image, (b) first object binary image, (c) second object binary image, and (d - f) Chessboard IDs of the (a – c) backgrounds.

A color image can be decomposed into a set of binary images whose foregrounds are its color objects.

The decomposition of the Figure 2a color image consists of the two binary images of Figures 2b and 2c. The Figures 2 (d – f) show respectively the Figures 2 (a – c) background Chessboard IDs.

The same decomposition can be applied to images with more than two color objects. In this case, there will be as many binary images as objects.

The pair of IDs of Figures 2e and 2f gives the distance values from any background pixel to the two color objects. This is exactly what will be used to perform the interpolation between contour lines in the next section.

4. Contour line interpolation

It is assumed that contour line data are raster data, that is images, otherwise they must be converted, for example from vector data to raster data.

Let $f : E \rightarrow K = \{0, 1, \ldots, m\}$ be a color image of m contour lines. For the sake of simplicity, it is assumed here that the contour line increment is unitary.

Let denote by $C_i = f(\{i\})$, with $i = 0, \ldots, m$. Following the definitions of the previous section, C_0 is the background of f and the C_i's ($i = 1, \ldots, m$) are its color objects, one for each contour line. Let $B_1, \ldots, B_n$ be the n CCs of the background C_0.

By properties of contour line images, every B_k ($k = 1, \ldots, n$) is adjacent to at most two C_i's.

Let r be a background point, $r \in C_0$, and let k be such that $r \in B_k$, k is unique. If B_k has two adjacent contour lines, and if i is such that C_i and C_{i+1} are these two contour lines, then, following Equation 1, the linear interpolation expression $g(r)$ at point r is given by

$g(r) = i + 1 * d(r, C_i) / (d(r, C_i) + d(r, C_{i+1}))$.

Figure 3a shows five contour lines $C_1, \ldots, C_5$, and six CCs $B_1, \ldots, B_6$. Figure 3b and 3c show $d(r, C_i)$ and $d(r, C_{i+1})$ for any points r in B_{i+1} ($i = 1, \ldots, 4$) along the dashed line of Figure 3a.

The above interpolation includes the following tasks: finding the CCs of the background, finding their adjacent contour lines and computing the two IDs over these CCs.

Actually, in the case of the Chessboard metric, all these tasks can be computed at once using Pseudocode B below.

Pseudocode B is divided into two parts. The first one simulates the distance propagation from the contour lines with the smallest quote values (as in Figure 3b), and the second one with the greatest quote values (as in Figure 3c).

Pseudocode B

Part one:

(1) create a matrix $M1$ of the same size as E, and fill $M1$ border with a large value;

(2) create an image $f1$, and copy f into $f1$;

/* process in raster mode */

```
(3) for x varying from 1 to m – 2 do
      for y varying from 1 to n – 2 do
        if f(x, y) ≠ 0 /* i.e. (x, y) is a contour line point */
          M1(x, y) = 0;
        else
          choose the color with smallest value in N_f((x, y),
          f1), in case of more than one color with the
          smallest value, choose that one with the smallest
          distance value in M1;
          set M1(x, y) to this distance value plus one;
          set f1(x, y) to this color value;
        end else;
      end do;
    end do;
/* process in antiraster mode */
(4) for x varying from m – 2  to 1 do
      for y varying from n – 2 to 1 do
        if f(x, y) = 0 /* i.e. (x, y) is a background point */
          choose the color with smallest value in N_a((x, y),
          f1), in case of more than one color with the
          smallest value, choose that one with the smallest
          distance value in M1;
          if f1(x, y) is greater than this color value then
            set M1(x, y) to this distance value plus one;
            set f1(x, y) to this color value;
          end if;
          else
            if M1(x, y) is greater than this distance value
            plus one and f1(x, y) is equal to this color value
            then
              set M1(x, y) to this distance value plus one;
          end else;
        end if;
      end do;
    end do.
```

Part two:

Because of the duality between the first and second part, the second one is obtained from the first one by replacing the variable names $M1$ and $f1$ by $M2$ and $f2$, and the boldface words by their dual.

By applying Pseudocode B to the Figure 3a example,

- the image $f1$ is filled out with 1 over B_2, 2 over B_3, 3 over B_4, and 4 over B_5;
- the image $f2$ is filled out with 2 over B_2, 3 over B_3, 4 over B_4, and 5 over B_5;
- one gets the images M_1 and M_2 partially depicted in Figures 3b and 3c respectively.

Pseudocode B differs from the color DT implementation of Section 2, for two reasons. The first one is that the DT applies now to the background (and not to each colored CC). The second one is that the background CCs must be identified by propagating the contour line colors.

The linear interpolation g of image f is given by, Pseudocode C below.

Pseudocode C

```
for (x, y) varying in E do
  if f(x, y) = 0 /* i.e. (x, y) is a background point */
    g(x, y) = f1(x, y) + (f2(x, y) – f1(x, y)) * M1(x, y) /
              (M1(x, y) + M2(x, y));
  else  g(x, y) = f(x, y);
end do.
```

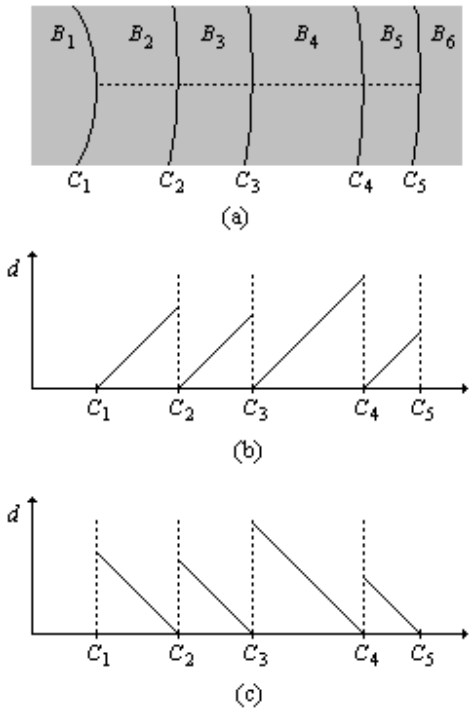

Figure 3 (a) Five contour line image, (b and c) distance values to the contours along the dashed line of Figure a.

Pseudocode B could be reused to implement a more accurate (non linear) interpolation by applying it, besides the original contour lines image, to the even and odd contour line images.

The interpolation could be still improved by using more accurate metrics, like the Octagonal, Chamfer and Euclidean ones.

An interesting feature of the suggested algorithm is its ability to interpolate properly areas like the one marked by × in Figure 4. In this case, where most of the interpolators do produce flat zones, the algorithm doesn't, since it is based on pairs of contour lines.

Figure 4 Potential flat zone.

This is the case of the interpolators based on Delaunay triangulation and its dual, the Voronoi diagram [12], in situations like in the Figure 4 example, these interpolators produce flat zones, since the triangles are constructed from a same contour line. In order to solve this problem some authors [12] introduce intermediate lines and additional points between the contour lines.

Intermediate lines are usually obtained from skeletons [13, 14]. At the contrary of the skeleton, which generates only one intermediate line between two contours, in the next section more than one intermediate line are generated using Pseudocode B.

5. Intermediate line generation in contour lines

The intermediate line generation becomes a simple operation by using two IDs.

This is shown for example, by applying Pseudocode B to the Figure 5a image.

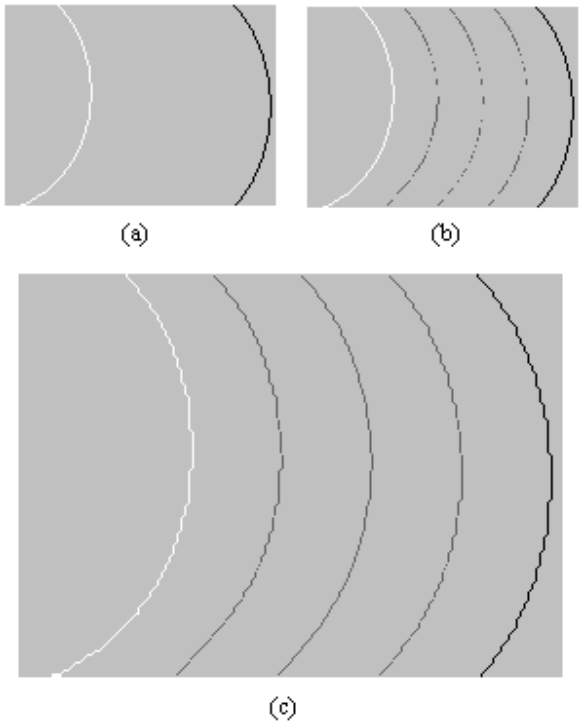

Figure 5 Intermediate lines in contour lines. (a) Part of a contour line image, (b) three intermediate contours, and (c) three intermediate contour lines of the image expanded by dilatation.

The generation of an image *g* containing the contour lines of image *f* plus three (for example) intermediate lines in between, is given in Pseudocode D below.

Pseudocode D

```
for (x, y) varying in E do
  if f(x, y) = 0 /* i.e. (x, y) is a background point */
    if M1(x, y) = M2(x, y) and f1(x, y) != f2(x, y) then
      g(x, y) = (f1(x, y) + f2(x, y))/2;
    else
      if M1(x, y) = M2(x, y)/3 then
        g(x, y) = f1(x, y) + (f2(x, y) – f1(x, y)/3;
      else
        if M1(x, y)/3 = M2(x, y) then
          g(x, y) = f2(x, y) – (f2(x, y) – f1(x, y)/3;
      end else;
    end else;
  end if;
  else  g(x, y) = f (x, y);
end do.
```

Figure 5b shows the Figure 5a image with 3 intermediate lines. Because of the discrete nature of the image domain, the intermediate lines are generally not connected. In order to solve this problem one can apply an expansion by dilation [15] in the image. Figure 3c shows the Figure 5a image expanded by dilatation with 3 intermediate lines. For better contrast between the background and intermediate lines, the three intermediate lines have been plotted using the same color.

6. Conclusions

In this paper, it was shown that the contour line interpolation and intermediate line generation can be solved by applying the DT to parts of the original image.

Furthermore, it was shown that the image segmentation can be done while computing the IDs, resulting in a very simple and fast algorithm.

In the future, applications of contour line interpolation to image compression could be investigated. In this case, the contour lines will be the image cross section borders, and the reconstruction will be obtained by interpolation.

Acknowledgements

The first author is supported by FAPESP, process: 02/02975-4.

References

[1] H. J. A. M. Heijmans, *Morphological image operators*, Boston: Academic Press, 1994.

[2] C. Arcelli, L. P. Cordella and S. Levialdi, "From local maxima to connected skeletons", *IEEE Transactions on Pattern Recognition and Machine Intelligence* PAMI-3:2 (1981), 134--143.

[3] C. Arcelli and G. Sanniti di Baja, "A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11:4 (1989), 411--414.

[4] F. Y. Shih and C. C. Pu, "A skeletonization algorithm by maxima tracking on Euclidean distance transform", *Pattern Recognition* 28:3 (1995), 331--341.

[5] A. F. Souza, "Esqueletos 8-isotrópicos", *Msc. Thesis*, National Institute for Space Research, INPE, Brazil, 2002.

[6] F. Preteux, "Watershed and skeleton by influence zones: a distance-based approach", *Journal of Mathematical Imaging and Vision* 1 (1992), 239--255.

[7] P. P. Das, "Lattice of octagonal distances in digital geometry", *Pattern Recognition Letters* 11:10 (1990), 663--667.

[8] G. Borgefors, "Distance transformations in digital images", *Computer Vision, Graphics, and Image Processing* 34 (1986), 344--371.

[9] M. A. Butt and P. Maragos, "Optimum design of chamfer distance transforms", *IEEE Transaction on Image Processing* 7:10 (1998), 1477--1484.

[10] P. E. Danielsson, "Euclidean distance mapping", *Computer Graphics and Image Processing* 14 (1980), 227--248.

[11] F. Y. Shih and O. R. Mitchell, "A mathematical morphology approach to Euclidean distance transformation", *IEEE Transactions on Image Processing* 1:2 (1992), 197--204.

[12] D. Thibault and C. M. Gold, "Terrain reconstruction from contours by skeleton retraction", *GeoInformatica 4* (2000), 349--373.

[13] C. M. Gold and J. Snoeyink, "A one-step crust and skeleton extraction algorithm", *Algorithmica* 30:2 (2001), 144--163.

[14] N. Amenta, M. Bern and D. Eppstein, "The crust and the beta-skeleton: combinatorial curve reconstruction," *Graphical Models and Image Processing* 60:2 (1998), 125--135.

[15] G. J. F. Banon, "New insight on digital topology", in *Mathematical morphology and its applications to image processing*, L. Vicent and J. Goutsias Eds., Palo Alto, CA, USA, Kluwer Academic, 2000.