Available online at www.sciencedirect.com
ScienceDirect
Image and Vision Computing 27 (2009) 37–44
www.elsevier.com/locate/imavis

New sequential exact Euclidean distance transform algorithms based on convex analysis

Yves Lucet *

Computer Science, I. K. Barber School of Arts and Sciences, University of British Columbia Okanagan, 3333 University Way, Kelowna, BC, Canada V1V 1V7

Received 10 November 2005; received in revised form 24 August 2006; accepted 20 October 2006

Abstract

We present several sequential exact Euclidean distance transform algorithms. The algorithms are based on fundamental transforms of convex analysis: The Legendre Conjugate or Legendre–Fenchel transform, and the Moreau envelope or Moreau-Yosida approximate. They combine the separability of the Euclidean distance with convex properties to achieve an optimal linear-time complexity.

We compare them with a Parabolic Envelope distance transform, and provide several extensions. All the algorithms presented perform equally well in higher dimensions. They can naturally handle grayscale images, and their principles are generic enough to apply to other transforms.

Keywords: Distance transform; Euclidean distance; Feature transform; Fast Legendre transform; Legendre–Fenchel transform; Fenchel conjugate; Moreau envelope; Moreau–Yosida approximate; Computational convex analysis

1. Introduction

Consider an $n \times m$ binary image IMG stored as a 0–1 matrix. Its squared Euclidean distance transform is an $n \times m$ image where each pixel $p \in \{1, \ldots, n\} \times \{1, \ldots, m\}$ has the value

$$\mathrm{DT}^2(p) = \min_q \{\|p - q\|^2;\ \mathrm{IMG}(q) = 0\}.$$

In other words, the EDT computes a new image in which the value at each pixel is equal to the Euclidean distance from that pixel to the background. To avoid unnecessary floating point operations, the square EDT, which we denote DT^2, is usually computed. For example, the DT^2 of the binary image in Fig. 1 is displayed in Fig. 2. If you consider the upper right pixel (5,1), its value in the transformed image becomes its square distance to the background pixel (3,3): $(5-3)^2 + (1-3)^2 = 8$. The value of each pixel is computed similarly.

A distance transform algorithm takes advantage of the fact that the values for all the pixels have to be computed (and there are relations between adjacent pixels) to speed up the computation Fig. 3a shows a binary image and Fig. 3b shows its distance image.

Distance transforms have long been recognised for their importance [1–3], and their applications [4–7]. Starting from non-Euclidean metrics like the city-block distance, and Chamfer distances, several algorithms have been proposed. Computational algorithms for the exact Euclidean distance transform (EDT) appeared later [4,8–14], and several linear-time algorithms are now known. Current research focuses on providing simpler algorithms, now that we have a better understanding of the properties of the EDT, and on extending the distance transform to a more general setting [6,10,13–20].

The three algorithms presented are optimal. They do not rely on complex data structure like polygonal chain [14]

* Tel.: +1 250 807 9505; fax: +1 250 470 6001.
E-mail address: yves.lucet@ubc.ca

0262-8856/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2006.10.011

$$\mathrm{IMG} = \begin{array}{|c|c|c|c|c|}\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline 1&1&0&1&1\\\hline 1&1&1&1&1\\\hline 1&1&1&1&1\\\hline\end{array}$$

Fig. 1. Example of a binary image IMG. The value 0 is associated with the background of the image.

$$\mathrm{DT}^2 = \begin{array}{|c|c|c|c|c|}\hline 8&5&4&5&8\\\hline 5&2&1&2&5\\\hline 4&1&0&1&4\\\hline 5&2&1&2&5\\\hline 8&5&4&5&8\\\hline\end{array}$$

Fig. 2. The Square Euclidean Distance transform of IMG. The value of each pixel is now its square Euclidean distance to the closest background pixel with value 0.

or on other concept like Voronoï diagram [8,10]. They manipulate one-dimensional arrays and are easily parallelizable since the computation on one row only depends on that row and not on values in adjacent rows contrary to for example [11].

Additionally, we consider an extension to the Euclidean distance transform to handle grayscale images instead of binary images. Our framework draws from convex analysis as developed in [21–23]. Following standard convex analysis formulations, we rewrite the constrained optimization problem defining DT^2 with an infinite penalization

$$\mathrm{DT}^2(p) = \min_q\{\|p-q\|^2 + I(q)\},$$

where $I(q) = 0$ if q is a background pixel ($\mathrm{IMG}(q) = 0$) and $+\infty$ otherwise (I is called the indicator function of the set $\{q; \mathrm{IMG}(q) = 0\}$). In that formulation, a natural extension is to consider a general function f instead of the indicator function I. The problem then becomes

$$\mathrm{DT}^2(p) = \min_q\{\|p-q\|^2 + f(q)\}. \tag{1}$$

The function f could be naturally chosen as the graylevel of the image for example. The resulting extended function Eq. (1), which we still denote DT^2, is well known in convex analysis as the Moreau envelope (also named Moreau–Yosida approximate, or Moreau–Yosida regularisation). The difference between our first formulation of DT^2 and the Moreau envelope Eq. (1) is that the minimum is taken over all the $\mathbb{R}^2$ plane in the later while it is taken only over the pixel defining the image in the former.

Going back to the work of Yosida [24], the Moreau envelope regularisation properties have been studied extensively in convex and variational analysis [25–27,22]. Computing the Moreau envelope is equivalent to computing another fundamental transform in convex duality: The Legendre conjugate (also called Legendre–Fenchel conjugate, or Legendre–Fenchel transform)

$$f^*(s) = \sup_{x\in\mathbb{R}^2}[\langle s,x\rangle - f(x)], \tag{2}$$

where $\langle\cdot,\cdot\rangle$ denotes the standard dot product in $\mathbb{R}^2$. Historically, its computation was first motivated by the study of Hamilton–Jacobi equations, and gave birth to several numerical algorithms [28–32], and later to a linear-time algorithm [33], with applications in numerous fields: numerical simulation of Burger's equation (see for example [34–37]), robotics [38], network communication [39], pattern recognition [40], numerical simulation of multiphasic flows [41], and analysis of the distribution of chemical compounds in the atmosphere [42].

The algorithms presented here rely on convex analysis properties to achieve a worst-case linear computation time. They either compute the Fenchel conjugate Eq. (2) or (equivalently) the Moreau envelope Eq. (1). They include the Linear-time Legendre Transform (LLT) [33], the parabolic envelope (PE) [12,13,20], and the non-expansive proximal mapping (NEP) [20]. The algorithms were imple-

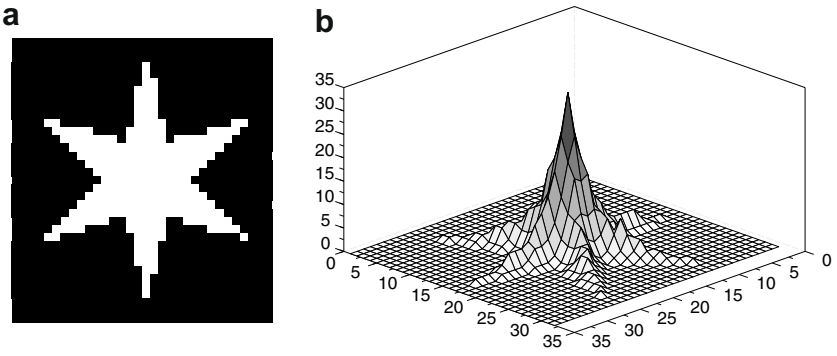

Fig. 3. 35×32 binary image of a star and its distance transform. (a) Binary image of a star. Background pixels are in black. (b) Distance transform of the image in (a).

mented in Scilab to allow for future comparison with known distance transform algorithms such as those available in the Scilab Image Processing toolbox (SIP) [43].

2. The LLT algorithm

We summarise the algorithm detailed in [40] to compute DT^2 the square EDT using the LLT algorithm. We write

$$DT^2(p) = \|p\|^2 - 2g^*(p) \tag{3}$$

where $g^*(p) = \max_q[\langle p,q\rangle - g(q)]$ is the Legendre conjugate of the function $g(q) = \|q\|^2/2 + I(q)$, and $\langle .,. \rangle$ is the standard scalar product. So the algorithm amounts to computing g for all pixels q, then applying the LLT algorithm to obtain g^* for all pixels p, and finally deducing DT^2 at all pixels p by Eq. (3).

The LLT algorithm uses convexity to obtain a linear running time. First, it factors the two-dimensional conjugate as several one-dimensional conjugates, which amounts to processing rows in a first pass, and columns in a second pass. Next, the computation of each one-dimensional transform involves computing the lower convex envelope of the function in linear time using the Beneath-Beyond algorithm, and then merging two increasing sequences. More precisely, to compute

$$g^*(p_1,p_2) = \max_{q_1}[p_1q_1 + \max_{q_2}[p_2q_2 - g(q_1,q_2)]]$$

in linear time, we only need to be able to compute u^* the conjugate of a convex real-valued function u in linear time, which can be achieved using the following property [33, Lemma 3]: Assume u is a convex univariate function, and define the finite difference slopes $c_i = (u(x_{i+1}) - (u(x_i))/(x_{i+1} - x_i)$. Then if $c_{i-1} < p_1 < c_i$, the maximum is attained at x_i, and if $c_i = p_1$, the maximum is attained at both x_i and x_{i+1}. Since u is convex the sequence c_i is non-decreasing, so computing u^* at all the x_i, amounts to merging the sequences c_i and x_i.

Figs. 4–9 illustrate the LLT algorithm and its partial computations. The binary image is transformed first into another (binary) image with pixel values in $\{0, +\infty\}$, then each background pixel (i,j) (pixels with value *0*) is set to the value $(i^2 + j^2)/2$. The Fenchel conjugate is then applied to the resulting image first row by row, then column by column. Finally Formula Eq. (3) is used to deduced DT^2.

As noted in [40] there is no need to compute the lower convex envelope of the function g due to the particular

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

Fig. 4. The binary image IMG (the background pixels have value 0).

∞	∞	∞	∞	∞
∞	0	∞	0	∞
∞	∞	∞	∞	∞
∞	∞	0	∞	∞
∞	∞	∞	∞	∞

Fig. 5. The function f generated from IMG by replacing any non-background pixel with $+\infty$.

∞	∞	∞	∞	∞
∞	4	∞	10	∞
∞	∞	∞	∞	∞
∞	∞	12.5	∞	∞
∞	∞	∞	∞	∞

Fig. 6. The function g generated by replacing the value of any background pixel (i,j) with the value $(i^2 + j^2)/2$.

∞	∞	∞	∞	∞
2	0	-2	-6	-10
∞	∞	∞	∞	∞
9.5	6.5	3.5	0.5	-2.5
∞	∞	∞	∞	∞

Fig. 7. The partial conjugate of g computed by applying the one-dimensional conjugate to the image row by row.

0	2	4	8	12
2	4	6	10	14
4	6	8.5	12	16
6.5	9.5	12.5	15.5	18.5
10.5	13.5	16.5	19.5	22.5

Fig. 8. The full conjugate of g obtained by applying the conjugacy operation to the partial conjugate column by column.

structure of g (it is the sum of a quadratic and an indicator function, so the vertices of its convex envelope are the points where g is finite). However, the convex envelope of the partial conjugate has to be computed along each columns, which can be done in linear time since the points $(i,g(i,j))_i$ are naturally sorted along the first coordinate.

The complexity of computing DT^2 using the LLT algorithm is linear. The LLT algorithm requires a two-pass scan of the image in addition to computing the function g and applying Eq. (3).

2	1	2	1	2
1	0	1	0	1
2	1	1	1	2
4	1	0	1	4
5	2	1	2	5

Fig. 9. The resulting square Euclidean distance transform DT^2 obtained using Eq. (3): multiply the image in Fig. 8 by -2, then at each pixel (i,j), add (i^2+j^2).

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

Fig. 10. Binary image IMG.

The LLT algorithm can be easily extended to non-indicator functions i.e. to compute the Moreau envelope Eq. (1) for any real-valued function f. For example, the function f can be defined as the gray level in a grayscale image. It also naturally extends to higher dimensions: When the data is d-dimensional, the LLT complexity is $O(d\,N)$, where N is the number of points in the grid ($N=n^2$ for an $n\times n$ image).

3. The NEP algorithm

The Non Expansive Proximal mapping (NEP) algorithm relies on the non-expansiveness[1] of the proximal mapping P defined by

$$P(p)=\mathrm{Argmin}_q\{\|p-q\|^2+f(q)\}. \tag{4}$$

The mapping P associates the closest background pixel to each pixel of the image (computing the distance between the two pixels gives the distance transform).

The non-expansiveness property holds when the function f is finite at some point, lower semi-continuous, and convex. When f is an indicator function of a binary image, it amounts to the object in the image (pixels with zero value) being convex. In that case, the NEP algorithm is extremely simple: Perform two scans first on the rows then on the columns, so we only compute P on $\{1,\ldots,n\}$ and not on the two-dimensional grid. On each scan, we perform a linear search to compute $P(1)$. Then a single loop returns all the values of P since under the assumptions above

$$0\leqslant P(i+1)-P(i)\leqslant i+1-i=1$$

[1] A function P is non-expansive if for any x,y, $\|P(x)-P(y)\|\leqslant\|x-y\|$.

∞	∞	∞	∞	∞
∞	0	∞	∞	∞
∞	∞	0	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞

Fig. 11. The function f obtained by substituting the value $+\infty$ at each pixel with value 1.

5	6	6	6	6
2	2	2	2	2
3	3	3	3	3
5	6	6	6	6
5	6	6	6	6

Fig. 12. The partial transform Pp obtained by applying the operator P to the image in Fig. 11 row by row (only row 2 and 3 are relevant).

∞	∞	∞	∞	∞
1	0	1	4	9
4	1	0	1	4
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞

Fig. 13. The partial distance transform associated with the partial transform in Fig. 12.

2	2	2	3	3
2	2	3	3	3
2	3	3	3	3
3	3	3	3	3
3	3	3	3	3

Fig. 14. The image resulting from applying the operator P to Fig. 13 column by column.

i.e. the value of P at the next pixel $i+1$ is either $P(i)$ or $P(i)+1$. The convexity assumption reduces the global search to a very narrow local search.

We illustrate the algorithm on Figs. 10–15. First Fig. 10 is converted to Fig. 11 by replacing the value of non-background pixels (pixels with value 1) with $+\infty$. Next each row of Fig. 11 is considered independently. The value of each pixel becomes the index in the row to the closest background pixel. For example, for the second row there is only one background pixel at column 2, so all the pixels in that

2	1	2	5	8
1	0	1	2	5
2	1	0	1	4
5	2	1	2	5
8	5	4	5	8

Fig. 15. The resulting DT^2 computed from Figs. 14 and 13 column by column.

row take the value 2. Note that rows 1, 4, and 5 do not have any background pixel. In that case, the algorithm assigns arbitrary values (when the image is considered column by column, there will always be a pixel with a lower value on that column and the arbitrary values will be superceded by a relevant value).

Finally, starting from the partial distance transform in Fig. 13, the algorithm is applied column by column. Again, the value of each pixel becomes the index of the closest background pixel in that column. Consider the second column. Pixel 2 is closest to pixel 1, so the value of pixel 1 becomes 2. Now the value of pixel 2 is either 2 or 3, and a simple comparison gives it the value 2. Similarly, the value of pixel 3 is either 2 or 3, and after comparison it is assigned value 3. The full DT^2 in Fig. 15 is obtained by reading Figs. 14 and 13 column by column and assigning to each pixel (i,j) with value $P_{i,j}$ in Fig. 14 and value $D_{i,j}$ in Fig. 13 the value $(P_{i,j}-i)^2+D_{i,j}$.

When the data is not convex like the example in Fig. 4, the algorithm fails as Fig. 16 shows: Line 2 of the partial feature transform Pp cannot return the index *4* since its distance to the previous value is more than *1*. Hence DT^2 returns errors as shown by Fig. 17 (the right answer is shown in Fig. 9).

5	6	6	6	6
2	2	2	2	2
5	6	6	6	6
3	3	3	3	3
5	6	6	6	6

Fig. 16. Partial feature transform *Pp* of the image in Fig. 4.

2	1	2	5	10
1	0	1	4	9
2	1	2	5	10
5	4	5	8	13
10	9	10	13	18

Fig. 17. Erroneous DT^2 computed from Fig. 16. Fig. 9 shows the right answer.

The NEP principle extends to any transform having a non-expansive argmin or argmax. In effect, the NEP principle reduces a global search to a local search. As soon as the distance between the pixels realising the minimum is bounded by a constant, a linear-time algorithm exists.

4. The PE algorithm

To initiate comparison with other distance transform algorithms, we recall the parabolic envelope (PE) algorithm [12,13,20]. It reduces computation to one dimension and uses properties of parabolas to obtain a linear-time algorithm. Namely, given two parabolas $p_1=(.-i)^2+j$ and $p_2=(.-i')^2+j'$, we can compute their intersection in constant time $O(1)$. So computing the lower envelope of the family of parabolas $(.-i)^2+j$ takes linear time. The algorithm is a two-pass scan. The first pass considers each row and computes the parabolic envelope, then evaluates the distance transform on that parabolic envelope. The second pass performs the same operations on the columns resulting from the first pass.

For the binary image shown in Fig. 4, Fig. 18 shows the results of the algorithm after the first pass: The parabolic envelope computed for each row is evaluated on the grid. Fig. 19 shows the resulting exact square EDT.

The PE algorithm runs in linear time. When the data are d dimensional, its complexity becomes $O(d\,N)$, for a grid containing N points (for an $n\times m$ binary picture $N=nm$). Its underlying principle is applicable to the computation of any family of functions provided the intersection of two such functions can be computed in constant time. Felzenszwalb [13] gave several examples of such

∞	∞	∞	∞	∞
1	0	1	0	1
∞	∞	∞	∞	∞
4	1	0	1	4
∞	∞	∞	∞	∞

Fig. 18. Partial DT^2 resulting from applying the PE algorithm to the rows of Fig. 4.

2	1	2	1	2
1	0	1	0	1
2	1	1	1	2
4	1	0	1	4
5	2	1	2	5

Fig. 19. DT^2 computed by applying the PE algorithm column by column to Fig. 18.

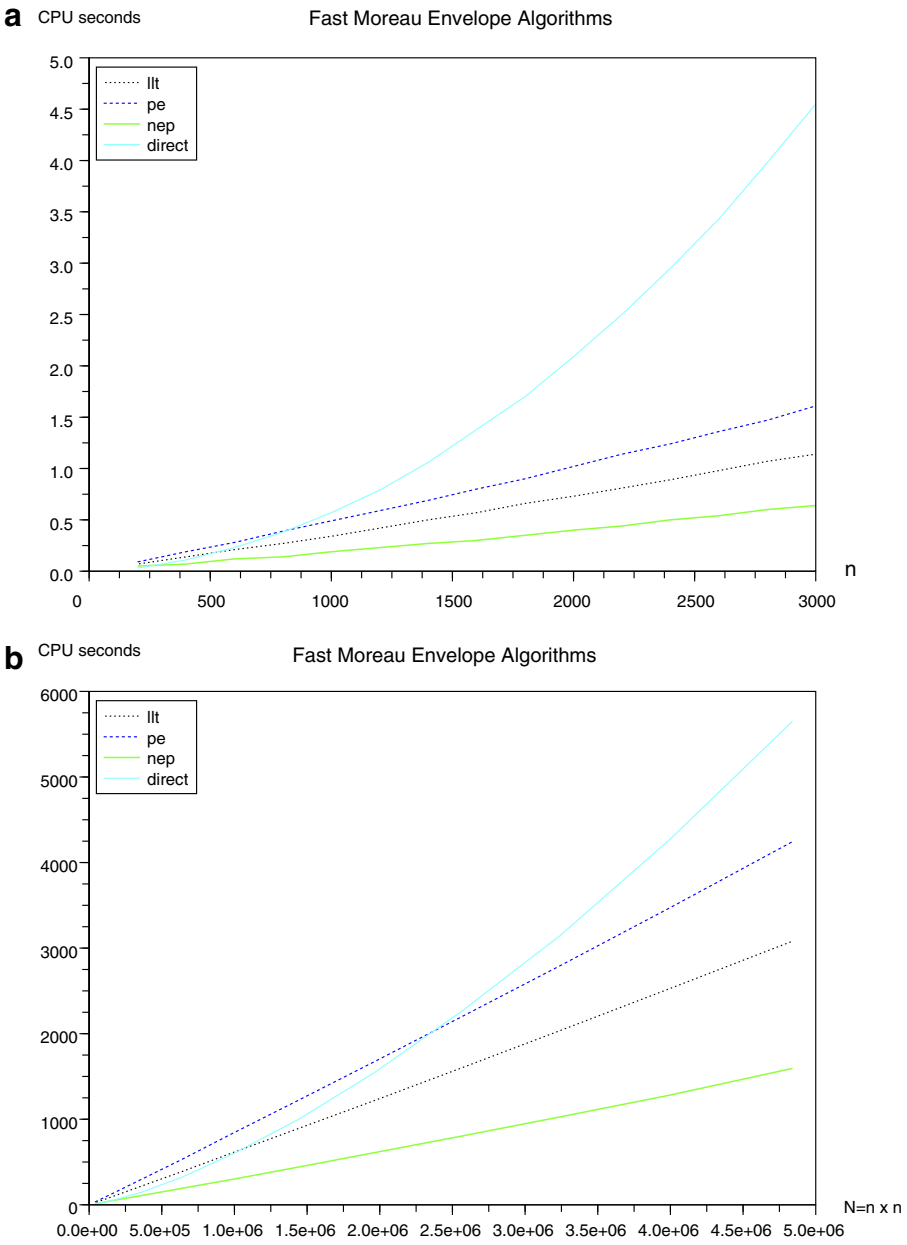

Fig. 20. Numerical validation of the linear-time complexity of the LLT, NEP, and PE algorithms. (a) Complexity for one-dimensional data. (b) Complexity for two-dimensional data such as distance transforms.

cases, including replacing the Euclidean distance with the l_1 distance.

Note that no assumption is made on the function f, all assumptions are on the distance function.

While the PE algorithm considers a family of parabolic functions, one can consider a family of affine functions and apply the same principle as the PE algorithm to compute the lower envelope. The resulting envelope is the lower convex envelope of the points. Hence, the LLT algorithm can be seen as following the same principle for the family of affine functions going through two consecutive pixels of the image. The only difference lies in how much computation is performed during each pass.

5. Numerical comparisons and complexity

Computing the EDT of an $n \times n$ binary image having $N = n^2$ pixels can be achieved by brute force at an $O(N^2) = O(n^4)$ cost as the following function shows.

```
0    function M = brute2d (Xr, Xc, f, Sr, Sc)
1      [nl,n2]=size(f);
2      ml=length(Sr);m2=length(Sc);
3      M=zeros(ml,m2);
4      for pl=l:ml
5        for p2=l:m2
6          tl = (Xr-Sr(pl)).^2 * ones(l,n2);
7      t2 = ones(nl,l) * ((Xc-Sc(p2)).^2)';
8          t = tl + t2 + f;
9        M(pl,p2) = min(t);//O(n^2) cost
10       end;
11     end;
12   endfunction
```

Using the fact the Euclidean distance is a separable function (it can be written as the sum of two one-dimensional functions) one can build a direct computation algorithm with complexity $O(N^{3/2}) = O(n^3)$. Compare the following with the `brute2d` function above (the function `me_direct(X,f,S)` is the one-dimensional brute force computation and runs in $O(n^2)$ when X and S have size n).

```
0    function M = direct2d (Xr, Xc, f, Sr, Sc)
1      for i=l:length(Xr)
2        F = me_direct(Xc, f(i,:)', Sc);
3          rl(i,:) = F';
4      end
5      M = ones(size(Sr, l), size(Sc, l));
6      for i=l:size(Sc, l)
7      M(:,i) = me_direct(Xr, rl(:,i), Sr);
8      end
9    endfunction
```

Fig. 20(a) numerically validates the linear-time complexity of the algorithms for the univariate quadratic function $f(x) = x^2$ while Fig. 20(b) illustrates the same algorithms for two-dimensional data such as computing the EDT. In the later figure, the brute force algorithm is not represented since its cost is too prohibitive.

6. Conclusion

We have presented new exact Euclidean Distance Transform (EDT) algorithms for binary images which all share an optimal linear-time complexity of $\Theta(N) = \Theta(n^2)$ for an $n \times n$ image. By taking advantage of the separability of the Euclidean distance, and using convex properties, the algorithms reduce to simple calculations on a line. In addition to scaling to higher dimension and being trivially parallelizable, all the algorithms compute the more general Moreau envelope, of which the distance transform is a special case.

The principles used in each algorithm can be applied to different transforms. The LLT algorithms uses convex properties, the NEP algorithm uses non-expansiveness, and the PE algorithm uses a $O(1)$ intersection cost. The last two properties are easy to identify and are sufficient to guarantee a linear-time algorithm.

Our implementation of the algorithms show that the NEP algorithm is generally faster than the LLT algorithm, which is faster than the PE algorithm. So if one has an image with a convex object in the background, the NEP algorithm is indicated. In the absence of convexity, the LLT algorithm performs quite well even though it introduces intermediate steps. However, the intermediate steps are very quick to perform in a matrix-optimized language such as Scilab. Another implementation in a non matrix-optimized language will probably find the PE algorithm very competitive.

Acknowledgements

The present research was partly supported by a Natural Sciences and Engineering Research Council of Canada Discovery Grant (NSERC), and by the Okanagan University College Grant-in-Aid program. All computations were done on the mathematical software Scilab using the SIP (Scilab Image Processing) package on a Pentium IV 1.8 GHz computer. Our Scilab implementation of the LLT algorithm presented in the paper is available through the GPL license.

References

[1] A. Rosenfeld, J.L. Pfaltz, Sequential operations in digital picture processing, J. ACM 13 (4) (1966) 471–494.
[2] P.-E. Danielsson, Euclidean distance mapping, Comput. Graph. Image Process. 14 (1980) 227–248.
[3] G. Borgefors, Distance transformations in digital images, Comput. Vis. Graph. Image Process. 34 (3) (1986) 344–371.
[4] G. Borgefors, Distance transformations on hexagonal grids, PRL 9 (1989) 97–105.
[5] O. Cuisenaire, Distance transformations: fast algorithms and applications to medical image processing, Ph.D. thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium (Oct. 1999).
[6] J. Toriwaki, K. Mori, Distance transformation and skeletonization of 3d pictures and their applications to medical images, in: R.K.E.G. Bertrand, A. Imiya (Eds.), Digital and Image Geometry: Advanced Lectures, Lecture Notes in Computer Science, vol. 2243/2001, Springer-Verlag Heidelberg, 2001, pp. 412–429.
[7] S. Prohaska, H.C. Hege, Fast visualization of plane-like structures in voxel data, in: VIS '02: Proceedings of the Conference on Visualization '02, IEEE Computer Society, Boston, MA, 2002, pp. 29–36.
[8] H. Breu, J. Gil, D. Kirkpatrick, M. Werman, Linear time Euclidean distance transform algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 17 (5) (1995) 529–533.
[9] L. Chen, H.Y.H. Chuang, A fast algorithm for Euclidean distance maps of a 2-d binary image, Inform. Process. Lett. 51 (1) (1994) 25–29.
[10] C.R. Maurer Jr., R. Qi, V.V. Raghavan, A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions, IEEE Trans. Pattern Anal. Mach. Intell. 25 (2) (2003) 265–270.
[11] F.Y. Shih, Y.-T. Wu, Fast Euclidean distance transformation in two scans using a 3×3 neighborhood, Comput. Vis. Image Underst. 93 (2) (2004) 195–205.
[12] L. Deniau, J. Blanc-Talon, Fractal analysis with Hausdorff distance under affine transformations, Tech. rep., ETCA-CREA-SP (1995).

[13] P.F. Felzenszwalb, D.P. Huttenlocher, Distance transforms of sampled functions, Tech. Rep. TR2004-1963, Cornell Computing and Information Science (Sep. 2004).
[14] M. Gavrilova, M.H. Alsuwaiyel, Two algorithms for computing the Euclidean distance transform, Tech. Rep. 2000-661-13, Computer Science Technical Reports, University of Calgary (2000).
[15] O. Cuisenaire, B. Macq, Fast and exact signed Euclidean distance transformation with linear complexity, in: Proc. of ICASSP'99, vol. 6, IEEE, Phoenix, AZ, USA, 1999, pp. 3293–3296, Int. Conference on Acoustics, Speech and Signal Processing.
[16] D.G. Bailey, An efficient Euclidean distance transform, in: R. Klette, J. Žunić (Eds.), Combinatorial Image Analysis: 10th International Workshop, IWCIA 2004, Auckland, New Zealand, December 1–3, 2004. Proceedings, Lecture Notes in Computer Science, vol. 3322, Springer-Verlag GmbH, 2004, pp. 394–408.
[17] W. Hesselink, A linear-time algorithm for euclidean feature transform sets, http://www.cs.rug.nl/~wim/pub/mans.html/ (May 2005).
[18] S. Mauch, A fast algorithm for computing the closest point and distance transform, http://www.acm.caltech.edu/seanm/projects/cpt/cpt.pdf/ (Dec. 2000).
[19] C. Sigg, R. Peikert, M. Gross, Signed distance transform using graphics hardware, in: Proceedings of IEEE Visualization '03, ETH Zürich, 2003.
[20] Y. Lucet, Fast Moreau envelope computation I: Numerical algorithms, Tech. rep., University of British Columbia Okanagan (2005).
[21] R.T. Rockafellar, R.J.-B. Wets, Variational Analysis, Springer-Verlag, Berlin, 1998.
[22] R.T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, New york, 1970.
[23] J.-B. Hiriart-Urruty, C. Lemaréchal, Convex Analysis and Minimization Algorithms, vol. 305–306 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1993, vol I: Fundamentals, vol II: Advanced theory and bundle methods.
[24] K. Yosida, Functional analysis, Classics in Mathematics, Springer-Verlag, Berlin, 1995, reprint of the sixth (1980) edition.
[25] J.-J. Moreau, Propriétés des applications "prox", C. R. Acad. Sci. Paris 256 (1963) 1069–1071.
[26] J.-J. Moreau, Proximité et dualité dans un espace Hilbertien, Bull. Soc. Math. France 93 (1965) 273–299.
[27] J.-J. Moreau, Convexity and duality, in: Functional Analysis and Optimization, Academic Press, New York, 1966, pp. 145–169.
[28] Y. Brenier, Un algorithme rapide pour le calcul de transformées de Legendre–Fenchel discrètes, C. R. Acad. Sci. Paris Sér. I Math. 308 (1989) 587–589.
[29] L. Corrias, Fast Legendre–Fenchel transform and applications to Hamilton–Jacobi equations and conservation laws, SIAM J. Numer. Anal. 33 (4) (1996) 1534–1558.
[30] Y. Lucet, A fast computational algorithm for the Legendre–Fenchel transform, Comput. Optimizat. Appl. 6 (1) (1996) 27–57.
[31] A. Noullez, M. Vergassola, A fast Legendre transform algorithm and applications to the adhesion model, J. Scient. Comput. 9 (3) (1994) 259–281.
[32] Z.-S. She, E. Aurell, U. Frisch, The inviscid Burgers equation with initial data of brownian type, Comm. Math. Phys. 148 (3) (1992) 623–641.
[33] Y. Lucet, Faster than the fast legendre transform, the linear-time legendre transform, Numer. Algorithms 16 (2) (1997) 171–185.
[34] J. Bec, U. Frisch, K. Khanin, Kicked Burger turbulence, J. Fluid Mech. 416 (2000) 239–267.
[35] U. Frisch, J. Bec, Burgulence, in: A.M. Lesieur, E.F. David (Eds.), Les Houches 2000: New Trends in Turbulence, Springer EDP-Sciences, 2001, pp. 341–383.
[36] U. Frisch, J. Bec, B. Villone, Singularities and the distribution of density in the Burgers/adhesion model, Phys. D 152-153 (2001) 620–635.
[37] A. Noullez, S.N. Gurbatov, E. Aurell, S.I. Simdyankin, The global picture of self-similar and not self-similar decay in Burgers turbulence, Tech. Rep. nlin.CD/0409022, arXiv.org eprint archive (Sep. 2004).
[38] B. Koopen, Contact of bodies in 2D-space: Implementing the Discrete Legendre Transform, AI Master's thesis, Intelligent Autonomous Systems Group, University of Amsterdam (Feb. 2002).
[39] T. Hisakado, K. Okumura, V. Vukadinovic, L. Trajkovic, Characterization of a simple communication network using Legendre transform, in: Proc. IEEE Int. Symp. Circuits and Systems, vol. 3, 2003, pp. 738–741.
[40] Y. Lucet, A linear Euclidean distance transform algorithm based on the Linear-time Legendre Transform, in: Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV 2005), IEEE Computer Society Press, Victoria BC, 2005.
[41] P. Helluy, Simulation numérique des écoulements multiphasiques : De la théorie aux applications, Ph.D. thesis, Institut des Sciences de l'Ingenieur de Toulon et du Var, Laboratoire Modélisation Numérique et Couplages, BP 56, 83162 La Valette CEDEX, France, habilitation à Diriger des Recherches (Jan. 2005).
[42] B. Legras, I. Pisso, G. Berthet, F. Lefvre, Variability of the Lagrangian turbulent diffusion in the lower stratosphere, Atmospheric Chem. Phys. 5 (2005) 1605–1622.
[43] R. Fabbri, Scilab image processing toolbox, 2005. http://siptoolbox.sourceforge.net/.