THEORY OF COMPUTING, Volume 8 (2012), pp. 415–428
www.theoryofcomputing.org

Distance Transforms of Sampled Functions

Pedro F. Felzenszwalb   Daniel P. Huttenlocher

Received: December 18, 2009; revised: May 30, 2012; published: September 2, 2012.

Abstract: We describe linear-time algorithms for solving a class of problems that involve transforming a cost function on a grid using spatial information. These problems can be viewed as a generalization of classical distance transforms of binary images, where the binary image is replaced by an arbitrary function on a grid. Alternatively they can be viewed in terms of the minimum convolution of two functions, which is an important operation in grayscale morphology. A consequence of our techniques is a simple and fast method for computing the Euclidean distance transform of a binary image. Our algorithms are also applicable to Viterbi decoding, belief propagation, and optimal control.

ACM Classification: F.2.1, I.4

AMS Classification: 68T45, 68W40

Key words and phrases: distance transform, minimum convolution, image processing

1 Introduction

Distance transforms are an important tool in computer vision, image processing and pattern recognition. A distance transform of a binary image specifies the distance from each pixel to the nearest "on" pixel. Distance transforms play a central role in the comparison of binary images, particularly for images resulting from local feature detection techniques such as edge or corner detection. For example, both the Chamfer [6] and Hausdorff [16] matching approaches make use of distance transforms in comparing binary images. Distance transforms are also used to compute the medial axis (boundaries of Voronoi cells) of digital shapes [4].

In this paper we consider a generalization of distance transforms to arbitrary functions on a grid rather than binary-valued ones (i. e., real-valued images rather than binary images). There is a simple intuition underlying this generalization. Binary images are often used to specify the locations of certain features in a picture. Rather than using a binary array to specify the presence or absence of a feature at each pixel, it

DOI: 10.4086/toc.2012.v008a019

can be useful to have a real-valued array specifying a cost (or strength) for a feature at each pixel. For such more general situation it is again useful to compute a type of distance transform, which in this case should reflect a combination of distances and feature costs.

Let $\mathcal{G}$ be a regular grid and $f : \mathcal{G} \to \mathbb{R}$ a function on the grid. We refer to a function on a grid as a *sampled function*. Sampled functions are usually defined by arrays. We define the distance transform of f to be a another function $\mathcal{D}_f : \mathcal{G} \to \mathbb{R}$ where

$$\mathcal{D}_f(p) = \min_{q \in \mathcal{G}} \big(d(p,q) + f(q) \big) . \tag{1.1}$$

Here $d(p,q)$ is some measure of the distance (not necessarily a metric) between p and q. Intuitively, for each point p we find a point q that is close to p, and for which $f(q)$ is small. Note that if f has a small value at some location, $\mathcal{D}_f$ will have small value at that location and any nearby point, where nearness is measured by the distance $d(p,q)$. When $d(p,q)$ is the Euclidean distance, we refer to $\mathcal{D}_f$ as the *Euclidean distance transform* (EDT) of f.

The definition in equation (1.1) is closely related to the traditional distance transform of a set of points $P \subseteq \mathcal{G}$, which associates to each grid location the distance to the nearest point in P,

$$\mathcal{D}_P(p) = \min_{q \in P} d(p,q) .$$

The grid $\mathcal{G}$ and the set of points P are often defined by a binary image, in which case the distance transform $\mathcal{D}_P$ is a real-valued image of the same size. Many algorithms for computing the distance transform of point sets use the equivalent definition

$$\mathcal{D}_P(p) = \min_{q \in \mathcal{G}} \big(d(p,q) + 1(q) \big) ,$$

where $1(q)$ is an indicator function for membership in P:

$$1(q) = \begin{cases} 0 & \text{if } q \in P , \\ \infty & \text{otherwise.} \end{cases}$$

This definition is almost the same as the definition for the distance transform of a sampled function in equation (1.1), except that it uses the indicator function $1(q)$ rather than an arbitrary function $f(q)$.

Efficient algorithms for computing the distance transform of a binary image using the L_1 and L_∞ distances were developed by Rosenfeld and Pfaltz [27]. Similar methods described in [5] have been widely used to compute approximations to the EDT of a binary image. These algorithms can be easily adapted to compute the distance transform of a sampled function, as we show in Section 3 for the case of the L_1 distance.

Our main result is a new linear-time algorithm for computing the distance transform of a sampled function when distance is measured by the squared Euclidean distance. This in turn provides a new technique for computing the exact EDT of a binary image, by computing the transform of the corresponding indicator function and taking square roots of the result. There are a number of other algorithms for computing the EDT of a binary image in linear-time (e. g., [18, 7, 17]); however these methods are quite involved and are not widely used in practice. In contrast, our algorithm is relatively simple, easy to implement, and very fast in practice. In the Conclusions (Section 5) we indicate the range of areas of science and technology to which our algorithm has already been applied.

A sampled function is equivalent to a real-valued image. We use the terminology "distance transform of a sampled function" for two reasons. First, we want to stress that the input is generally some kind of cost function that is being transformed so as to incorporate spatial (or distance) information. Second, there are methods for computing transforms of real-valued images based on minimum distances along paths, where the cost of a path is the sum of values along the path [28]. We want to avoid confusion with these methods which compute something quite different from what we consider here.

1.1 Minimum convolution

The distance transform of a function is closely related to the *min-convolution* operation. This operation and its continuous counterpart play an important role in grayscale morphology [21]. The min-convolution of two functions f and g is defined as,

$$(f \otimes g)(p) = \min_q \big(f(q) + g(p-q)\big)\,.$$

Just like standard convolution, this operation is commutative and associative,

$$\begin{aligned} f \otimes g &= g \otimes f, \\ (f \otimes h) \otimes g &= f \otimes (h \otimes g). \end{aligned}$$

When $d(p,q) = g(p-q)$, the distance transform of f under the distance d is the min-convolution of f and g. The algorithms we describe in this paper can be seen as a algorithms for computing min-convolutions with restricted functions g. Our basic algorithms solve one-dimensional problems. The multi-dimensional case is handled by decomposing it into a sequence of one-dimensional problems.

In the case of the squared Euclidean distance we are computing the min-convolution of a one-dimensional function f and a parabola. We solve the problem by noting a relationship to lower envelopes of parabolas. Lower envelopes of n parabolas can be computed in $O(n \log n)$ time using the method in [8]. That method operates by sorting the parabolas in an appropriate order to be inserted in the lower envelope in amortized constant time. In our case we get an $O(n)$ algorithm because the parabolas have the same shape and are already sorted in an appropriate order. Here n is the size of the domain of f. In the case of the L_1 distance we compute the min-convolution of a one-dimensional function f and a cone using the classical approach of Rosenfeld and Pfaltz [27].

When d is the squared Euclidean and L_1 distance, the corresponding g is convex. As described by Eppstein [9] the problem of computing one-dimensional min-convolutions between arbitrary functions f and convex functions g can be solved in $O(n)$ time using the totally monotone matrix search algorithm of [1]. When g is concave the problem can be solved in $O(n\alpha(n))$ time using the matrix search algorithm of [20], where α is the extremely slowly growing inverse Ackermann function.

Our algorithm for one-dimensional min-convolution with a parabola generalizes to computing the min-convolution of an arbitrary one-dimensional function f and a convex or concave function g. If an intersection point between shifted copies of g can be computed in constant time we get an $O(n)$ method for computing the min-convolution (the resulting algorithm is simpler than the linear-time method based on matrix search, and also applies to the case when g is concave). Otherwise binary search can be used to find an intersection in $O(\log n)$ time, yielding an $O(n \log n)$ min-convolution algorithm.

1.2 Optimization problems

Distance transforms of sampled functions arise in the solution of a number of optimization problems. For instance in the widely used Viterbi algorithm for hidden Markov models [24], in max-product belief propagation [23], in optimal control methods [3] and in resource allocation problems [2].

In these problems we typically have a discrete state space S, a cost $b(p)$ for each state $p \in S$, a transition cost $a(p,q)$ for changing from state p to state q, and a dynamic programming equation

$$\delta'(q) = b(q) + \min_{p \in S}(\delta(p) + a(p,q)) . \tag{1.2}$$

For our purposes a detailed understanding of this equation is not as important as observing its form. The minimization in the second term of the right hand side is closely related to the distance transform of a function. If S is a grid, then equation (1.2) can be rewritten in terms of a distance transform

$$\delta'(q) = b(q) + \mathcal{D}_\delta(q) .$$

Thus algorithms for computing distance transforms of functions apply to certain problems of the form in (1.2). We have used these methods to develop improved algorithms for recognition of articulated (non-rigid) objects [11], for inference using large state-space hidden Markov models [13], and for the solution of low-level vision problems such as stereo, image restoration and optical flow using loopy belief propagation [12]. For instance, in the case of a hidden Markov model with n states the standard computation of the Viterbi recurrence takes $O(n^2)$ time which is not practical for large values of n, while the computation using our distance transform techniques takes $O(n)$ time.

1.3 Classical EDT

An important consequence of our results is a new linear-time algorithm for computing the classical Euclidean distance transform of a binary image. The method works by decomposing the problem into a sequence of one-dimensional problems. This decomposition relies on the solution of the more general problem, involving transforms of sampled functions. Thus, by solving a more general problem for the one-dimensional case we obtain a simple algorithm for the multi-dimensional EDT.

While other methods for computing the EDT of binary functions existed before, these methods have not been used in practice due to their practical complexity. In contrast, our algorithm has been used in a wide variety of applications. For example, it was applied to study urban change in [15], motion planning in [25], object recognition in [29] and image analysis PDEs in [22]. This illustrates the significance of developing algorithms that are not only asymptotically optimal, but are also easy to implement and have low complexity in practice.

2 Squared Euclidean distance

2.1 One dimension

Let $\mathcal{G} = \{0,\dots,n-1\}$ be a one-dimensional grid, and $f : \mathcal{G} \to \mathbb{R}$ be a function on the grid. The squared Euclidean (or quadratic) distance transform of f is given by

$$\mathcal{D}_f(p) = \min_{q \in \mathcal{G}}\big((p-q)^2 + f(q)\big) . \tag{2.1}$$

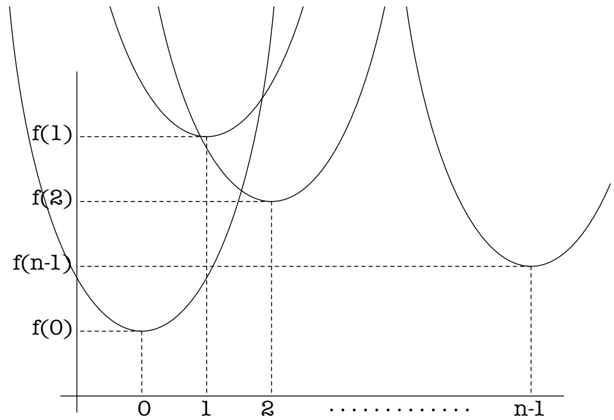


Figure 1: The distance transform as the lower envelope of n parabolas.

Note that for each point $q \in \mathcal{G}$ there is a constraint that the distance transform of f be bounded from above by a parabola rooted at $(q, f(q))$. In fact the distance transform is defined by the lower envelope of these parabolas, as shown in Figure 1. The value of the distance transform at p is simply the height of the lower envelope at that point.

Our algorithm for computing this distance transform has two distinct steps. First we compute the lower envelope of the n parabolas just mentioned. We then fill in the values of $\mathcal{D}_f(p)$ by checking the height of the lower envelope at each grid location p. This is a unique approach because we start with something defined on a grid (the values of f), move to a combinatorial structure defined over the whole domain (the lower envelope of the parabolas) and move back to values on the grid by sampling the lower envelope. Pseudocode for the procedure is shown in Algorithm 1.

The main part of the algorithm is the lower envelope computation. Note that any two parabolas defining the distance transform intersect at exactly one point. Simple algebra yields the horizontal position of the intersection between the parabolas coming from grid positions q and r as,

$$s = \frac{(f(r) + r^2) - (f(q) + q^2)}{2r - 2q} .$$

If $q < r$ then the parabola coming from q is below the one coming from r to the left of the intersection point s, and above it to the right of s.

We compute the lower envelope by sequentially computing the lower envelope of the first q parabolas, where the parabolas are ordered according to the horizontal locations of their vertices. The algorithm works by computing the combinatorial structure of this lower envelope. We keep track of the structure by using two arrays. The horizontal grid location of the i-th parabola in the lower envelope is stored in $v[i]$. The range in which the i-th parabola of the lower envelope is below the others is given by $z[i]$ and $z[i+1]$. The variable k keeps track of the number of parabolas in the lower envelope.

When considering the parabola from q, we find its intersection with the parabola from $v[k]$ (the rightmost parabola in the lower envelope computed so far). There are two possible cases, as illustrated in Figure 2. If the intersection is after $z[k]$, then the lower envelope is modified to indicate that the parabola

Algorithm $DT(f)$

1. $k \leftarrow 0$ (∗ Index of rightmost parabola in lower envelope ∗)
2. $v[0] \leftarrow 0$ (∗ Locations of parabolas in lower envelope ∗)
3. $z[0] \leftarrow -\infty$ (∗ Locations of boundaries between parabolas ∗)
4. $z[1] \leftarrow +\infty$
5. **for** $q = 1$ **to** $n-1$ (∗ Compute lower envelope ∗)
6. $\quad s \leftarrow ((f(q)+q^2)-(f(v[k])+v[k]^2))/(2q-2v[k])$
7. $\quad$ **if** $s \leq z[k]$
8. $\quad\quad$ **then** $k \leftarrow k-1$
9. $\quad\quad\quad$ **goto** 6
10. $\quad\quad$ **else** $k \leftarrow k+1$
11. $\quad\quad\quad v[k] \leftarrow q$
12. $\quad\quad\quad z[k] \leftarrow s$
13. $\quad\quad\quad z[k+1] \leftarrow +\infty$
14. $k \leftarrow 0$
15. **for** $q = 0$ **to** $n-1$ (∗ Fill in values of distance transform ∗)
16. $\quad$ **while** $z[k+1] < q$
17. $\quad\quad k \leftarrow k+1$
18. $\quad \mathcal{D}_f(q) \leftarrow (q-v[k])^2 + f(v[k])$

Algorithm 1: One-dimensional distance transform under the squared Euclidean distance.

from q is below all others starting at the intersection point. If the intersection is before $z[k]$ then the parabola from $v[k]$ should not be part of the new lower envelope, so we decrease k to delete that parabola and repeat the procedure.

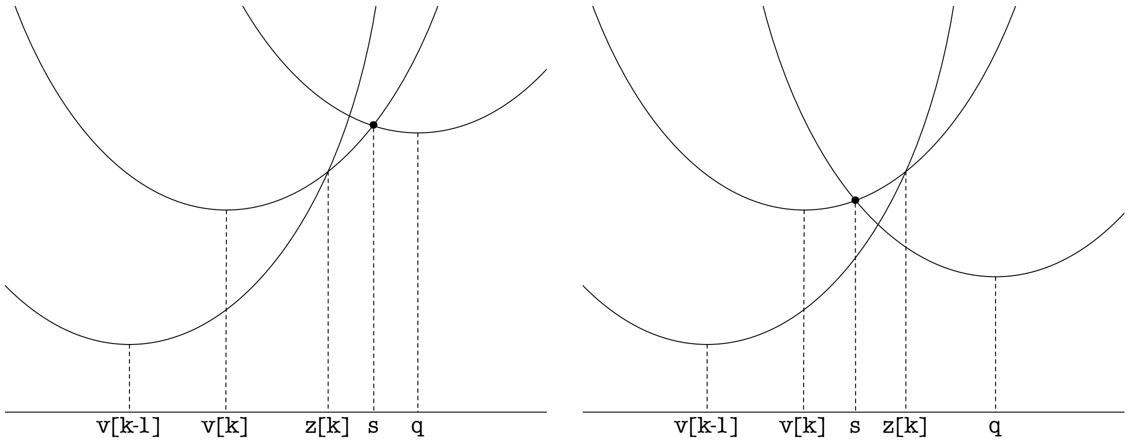
Theorem 2.1. *Algorithm 1 correctly computes the distance transform of a one-dimensional sampled function under the squared Euclidean distance in $O(n)$ time.*

Proof. We start by showing that the algorithm correctly computes the lower envelope of the first q parabolas by induction. The base case is trivial. The lower envelope of the first parabola is represented by a single interval from $-\infty$ to $+\infty$ dominated by the parabola from grid position 0.

Let s be the horizontal position of the intersection between the q-th parabola and the one from $v[k]$ as computed in line 6. The parabola from q is above the one from $v[k]$ to the left of s, and below it to the right of s. By the induction hypothesis the parabola from $v[k]$ is above at least one other parabola in the lower envelope to the left of $z[k]$ and below all of them to the right of $z[k]$.

Suppose $s > z[k]$ (as in Figure 2a). The parabola from q is below the one from $v[k]$ to the right of s, which in turn is below all others everywhere after $z[k]$. We see that the parabola from q dominates the lower envelope after s. The parabola from $v[k]$ continues to be the lowest between $z[k]$ and s. The algorithm updates the lower envelope to reflect these changes by creating a new interval from s to ∞ dominated by the q-th parabola.

Suppose $s \leq z[k]$ (as in Figure 2b). Now the parabola from q is below the parabola from $v[k]$ in the whole interval previously dominated by the parabola from $v[k]$. This means that the parabola from $v[k]$



(a) The case when $s > z[k]$.

(b) The case when $s \leq z[k]$.

Figure 2: The two possible cases considered by the algorithm when adding the parabola from q to the lower envelope constructed so far.

is not part of the lower envelope of the first q parabolas. The algorithm modifies the lower envelope to remove the parabola from $v[k]$ and proceeds to add the parabola from q to the remaining structure. This process eventually terminates since $z[0] = -\infty$.

Once the lower envelope is computed it remains to fill in the distance transform values by sampling the height of the lower envelope at each grid location. This is done from left to right on lines 14 through 18. To understand the running time of the algorithm note that we consider adding each parabola to the lower envelope exactly once. The addition of one parabola may involve the deletion of many others, but each parabola is deleted at most once. So the overall computation of the lower envelope in lines 1 through 13 takes $O(n)$ time. The computation of the distance transform values from the lower envelope in lines 14 through 18 considers each grid position and each parabola in the lower envelope at most once, so the second part of the algorithm also takes $O(n)$ time. $\square$

2.2 Arbitrary dimensions

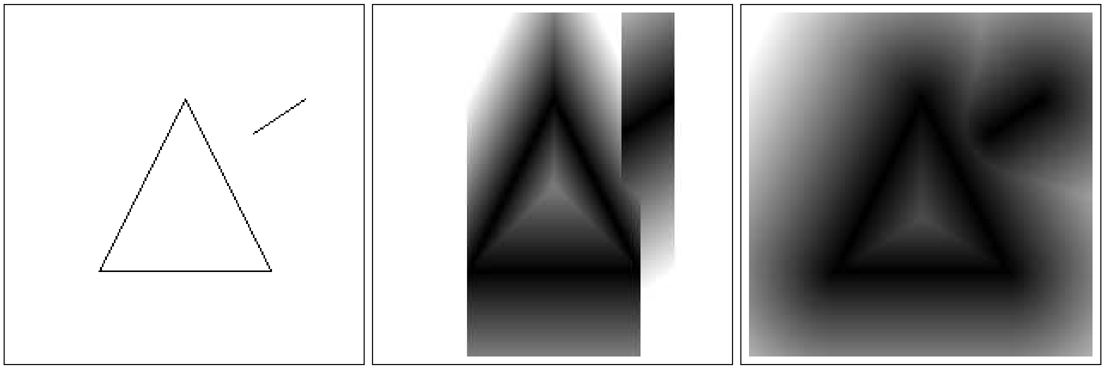
Let $\mathcal{G} = \{0, \dots, n-1\} \times \{0, \dots, m-1\}$ be a two-dimensional grid, and $f : \mathcal{G} \to \mathbb{R}$ a function on the grid. The two-dimensional distance transform of f under the squared Euclidean distance is given by,

$$\mathcal{D}_f(x,y) = \min_{x',y'} \left((x-x')^2 + (y-y')^2 + f(x',y') \right).$$

The first term does not depend on y' so we can rewrite this equation as,

$$\begin{aligned} \mathcal{D}_f(x,y) &= \min_{x'} \left((x-x')^2 + \min_{y'} \left((y-y')^2 + f(x',y') \right) \right), && (2.2) \\ &= \min_{x'} \left((x-x')^2 + \mathcal{D}_{f|_{x'}}(y) \right), && (2.3) \end{aligned}$$

where $\mathcal{D}_{f|_{x'}}(y)$ is a one-dimensional distance transform of f restricted to the column indexed by x'.



(a) (b) (c)

Figure 3: Illustration of the distance transform of a binary image using the squared Euclidean distance. The input image is shown in (a). The transform of the input along each column is shown in (b). The final distance transform, obtained by transforming the intermediate result along each row, is shown in (c). Dark pixels correspond to low values while bright pixels correspond to high values.

Thus a two-dimensional distance transform can be computed by first computing one-dimensional distance transforms along each column of the grid, and then computing one-dimensional distance transforms along each row of the result.

This argument extends to arbitrary dimensions, resulting in the composition of transforms along each dimension of the underlying grid. Note that changing the order of these transforms yields the same result, as can be seen readily for the two-dimensional case above. The multi-dimensional algorithm runs in $O(dN)$ time, where d is the dimension of the grid and N is the overall number of grid locations ($d = 2$ and $N = nm$ for the grid defined above). Therefore we obtain the following result.

Theorem 2.2. *The squared Euclidean distance transform of a sampled function defined on a d dimensional grid with N sample points can be computed in $O(dN)$ time.*

Figure 3 illustrates the computation of the classical EDT of a binary image using our method, where we start with the indicator function for the points on the grid. Note that in equation (2.3) the function that must be transformed along the second dimension is no longer an indicator function. Thus the notion of a distance transform for arbitrary functions is important here. The separation of the multi-dimensional transform into multiple one-dimensional ones makes our method substantially simpler than previous techniques for computing the classical EDT.

3 L_1 distance

For a set of points on a one-dimensional grid, the distance transform under the L_1 distance can be computed using a simple two-pass algorithm (e. g., [27]). Essentially the same algorithm can be used to compute the distance transform of a one-dimensional sampled function under the L_1 distance. Pseudocode for the method is shown in Algorithm 2.

Algorithm $DT(f)$

$\mathcal{D}_f \leftarrow f$ (∗ Initialize $\mathcal{D}_f$ with f ∗)

for $q = 1$ **to** $n-1$ (∗ Forward pass ∗)

  $\mathcal{D}_f(q) \leftarrow \min(\mathcal{D}_f(q), \mathcal{D}_f(q-1)+1)$

for $q = n-2$ **to** 0 (∗ Backward pass ∗)

  $\mathcal{D}_f(q) \leftarrow \min(\mathcal{D}_f(q), \mathcal{D}_f(q+1)+1)$

Algorithm 2: One-dimensional distance transform under the L_1 distance.

The values of the distance transform are initialized to the values of f itself. In the forward pass, each successive element of $\mathcal{D}_f(q)$ is set to the minimum of its own value and one plus the value of the previous element. This is done "in place" so that updates affect one another. In the backward pass each item is analogously set to the minimum of its own value and one plus the value of the next element. For example given the input $(4,2,8,6,1)$, after the first pass $\mathcal{D}_f$ will be $(4,2,3,4,1)$, and after the second pass it will be $(3,2,3,2,1)$.

Theorem 3.1. *Algorithm 2 correctly computes the distance transform of a one-dimensional sampled function under the L_1 distance in $O(n)$ time.*

Proof. Let

$$a(p) = \begin{cases} |p| & \text{if } p \geq 0\,, \\ \infty & \text{otherwise,} \end{cases} \qquad \text{and} \qquad b(p) = \begin{cases} |p| & \text{if } p \leq 0\,, \\ \infty & \text{otherwise.} \end{cases}$$

It's not hard to check that $(a \otimes b)(p) = |p|$. We claim that the forward pass of the algorithm computes the minimum convolution of f with a while the backward pass computes the minimum convolution of the result with b. Since minimum convolution is an associative operation the algorithm computes $f \otimes (a \otimes b)$, which as discussed in Section 1.1 is the distance transform of f under the L_1 distance.

Now we show that the forward pass of the algorithm does indeed compute the minimum convolution of f and a.

$$\begin{aligned}
(f \otimes a)(p) &= \min_q \big(f(q) + a(p-q)\big) && [\text{Definition of } \otimes] \\
&= \min_{q \leq p} \big(f(q) + a(p-q)\big) && [a(x) = \infty \text{ when } x < 0] \\
&= \min_{q \leq p} \big(f(q) + p - q\big) && [\text{definition of } a] \\
&= \min\big(\min_{q \leq p-1} \big(f(q) + p - q\big), f(p)\big) && [q \leq p-1 \text{ or } q = p] \\
&= \min\big(\min_{q \leq p-1} \big(f(q) + (p-1) - q\big) + 1, f(p)\big) && [\text{simple algebra}] \\
&= \min\big((f \otimes a)(p-1) + 1, f(p)\big)\,. && [\text{recursive substitution}]
\end{aligned}$$

The last equation is exactly the computation performed by the algorithm. The convolution of the resulting function with b in the backward pass is analogous. Both the forward and backward passes take a constant number of operations per grid position, yielding a $O(n)$ algorithm overall. □

As with the squared Euclidean distance case considered in the previous section, a multi-dimensional L_1 distance transform can be computed by successive transforms along each dimension of the grid.

Theorem 3.2. *The L_1 distance transform of a sampled function defined on a d dimensional grid with N sample points can be computed in $O(dN)$ time.*

4 Other distances

We can compute distance transforms of functions under other distances not discussed so far. An important case is the distance transform under the box distance defined by $d(p,q) = 0$ when $|p-q| < T$ and ∞ otherwise (here T is a threshold). This transform can be computed using a linear time algorithm for the min-filter described in [14] but our approach is simpler and more general.

Algorithm 1 can be easily modified to use any distance of the form $d(p,q) = g(p-q)$ as long as g is convex. We only need to be able to compute intersection points between shifted copies of g defined by $g_q(p) = f(q) + g(p-q)$ for $q \in \{0,\ldots,n-1\}$. This can often be done in constant time if g is given in analytic form. In the worst case an intersection point can be found in $O(\log n)$ time using binary search. The key is to note that $\mathcal{D}_f(p) = \min_q g_q(p)$ and for any convex function g, if $q < r$ we have that $g_q(p)$ is below $g_r(p)$ before their intersection point, and above afterwards.

Algorithm 1 can also be modified to handle the case when g is concave. The only difference is that we have to consider the shifted copies of g in reverse order. This is because when g is concave and $q < r$ we have that $g_q(p)$ is above $g_r(p)$ up to their intersection point and below afterwards.

These observations lead to the following result in terms of min-convolutions.

Theorem 4.1. *Let f be an arbitrary one-dimensional function on a grid of size n, and g be a concave or convex function. The min-convolution $f \otimes g$ can be computed in $O(nt(n))$ time where $t(n)$ is the time required to computed an intersection between shifted copies of g.*

Another way to obtain fast distance transform (and min-convolution) algorithms is to use the relationships described below. For example, the distance $d(p,q) = \min\big(c(p-q)^2, a|p-q|+b\big)$ is commonly used in robust estimation and is very important in practice. Intuitively this distance is robust because it grows slowly after a certain point. We can compute the distance transform of a function under the robust distance by computing both a linear and a quadratic distance transform and combining the results.

Observation 1. If $d(p,q) = \min\big(d^1(p,q), d^2(p,q)\big)$ then $\mathcal{D}_f(p) = \min\big(\mathcal{D}^1_f(p), \mathcal{D}^2_f(p)\big)$.

Observation 2. If $d^{(a,b)}(p,q) = ad(p,q) + b$ then $\mathcal{D}^{(a,b)}_f(p) = a\mathcal{D}_{f/a}(p) + b$.

Many state transition costs which appear in dynamic programming equations similar to (1.2) can be written as a combination of a small number of linear and quadratic and box distances using these relations. In such cases our algorithms can be used to compute the dynamic programming equation in time linear in the number of possible states.

5 Conclusions

Classical Euclidean distance transforms of binary images have many important applications. Previous exact algorithms have not been used in practice because they were too complex despite having linear-time worst case running time. In contrast we have described simple methods for solving more general problems. These problems are equivalent to min-convolutions with special functions. The results described here first appeared in a technical report [10].

Our approach has lead to a practical algorithm for the classical EDT that has been widely used by other researchers. Our EDT algorithm has been used in robotics for motion planning [25], in cartography for understanding urban development [15], in image processing for image segmentation [22], in medical image analysis for prenatal genetic testing [26], and in information interfaces for sonification of images for the visually impared [30].

The generalization of the distance transform to real-valued functions has proven to have important practical applications as well. It has been used in computer vision for object recognition [11] and depth estimation [12] as well as in machine learning for maximum a posteriori inference [13, 19].

References

[1] ALOK AGGARWAL, MARIA M. KLAWE, SHLOMO MORAN, PETER SHOR, AND ROBERT WILBER: Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987. Preliminary version in SCG'86. [doi:10.1007/BF01840359] 417

[2] RICHARD BELLMAN AND WILLIAM KARUSH: Functional equations in the theory of dynamic programming XII: An application of the maximum transform. *J. Mathematical Analysis and Applications*, 6(1):155–157, 1963. [doi:10.1016/0022-247X(63)90099-4] 418

[3] DIMITRI BERTSEKAS: *Dynamic Programming and Optimal Control*. Athena Scientific, 2001. 418

[4] HARRY BLUM: Biological shape and visual science (part I). *J. Theoretical Biology*, 38(2):205–287, 1973. [doi:10.1016/0022-5193(73)90175-6] 415

[5] GUNILLA BORGEFORS: Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986. [doi:10.1016/S0734-189X(86)80047-0] 416

[6] GUNILLA BORGEFORS: Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(6):849–865, 1988. [doi:10.1109/34.9107] 415

[7] HEINZ BREU, JOSEPH GIL, DAVID KIRKPATRICK, AND MICHAEL WERMAN: Linear time Euclidean distance algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(5):529–533, 1995. [doi:10.1109/34.391389] 416

[8] OLIVIER DEVILLERS AND MORDECAI J. GOLIN: Incremental algorithms for finding the convex hulls of circles and the lower envelopes of parabolas. *Information Processing Letters*, 56(3):157–164, 1995. Preliminary version in CCCG'94. [doi:10.1016/0020-0190(95)00132-V] 417

[9] DAVID A. EPPSTEIN: *Efficient algorithms for sequence analysis with concave and convex gap costs*. PhD thesis, Columbia University, 1989. 417

[10] PEDRO F. FELZENSZWALB AND DANIEL P. HUTTENLOCHER: Distance transforms of sampled functions. Technical Report 2004-1963, Cornell University, 2004. eCommons@Cornell. 425

[11] PEDRO F. FELZENSZWALB AND DANIEL P. HUTTENLOCHER: Pictorial structures for object recognition. *Internat. J. Computer Vision*, 61(1):55–79, 2005. Preliminary version in CVPR'00. [doi:10.1023/B:VISI.0000042934.15159.49] 418, 425

[12] PEDRO F. FELZENSZWALB AND DANIEL P. HUTTENLOCHER: Efficient belief propagation for early vision. *Internat J. Computer Vision*, 70(1):41–54, 2006. Preliminary version in CVPR'04. [doi:10.1007/s11263-006-7899-4] 418, 425

[13] PEDRO F. FELZENSZWALB, DANIEL P. HUTTENLOCHER, AND JON M. KLEINBERG: Fast algorithms for large-state-space HMMs with applications to web usage analysis. In *Advances in Neural Information Processing Systems 16 (NIPS'03)*, 2003. NIPS. 418, 425

[14] JOSEPH GIL AND MICHAEL WERMAN: Computing 2-D min, median, and max filters. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):504–507, 1993. [doi:10.1109/34.211471] 424

[15] ALEX HAGEN-ZANKER AND HARRY TIMMERMANS: A metric of compactness of urban change illustrated to 22 European countries. In *Proc. 11th Conf. Association of Geographic Information Laboratories for Europe (AGILE'08)*, pp. 181–200. Springer, 2008. [doi:10.1007/978-3-540-78946-8_10] 418, 425

[16] DANIEL P. HUTTENLOCHER, GREGORY A. KLANDERMAN, AND WILLIAM RUCKLIDGE: Comparing images using the Hausdorff distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(9):850–863, 1993. [doi:10.1109/34.232073] 415

[17] CALVIN R. MAURER JR., RENSHENG QI, AND VIJAY 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):265–270, 2003. Preliminary version in IPMI'01. [doi:10.1109/TPAMI.2003.1177156] 416

[18] ALEXANDER V. KARZANOV: Quick algorithm for determining the distances from the points of the given subset of an integer lattice to the points of its complement. *Cybernetics and System Analysis*, pp. 177–181, 1992. Translation from the Russian. 416

[19] MIKE KLAAS, DUSTIN LANG, AND NANDO DE FREITAS: Fast maximum *a posteriori* inference in Monte Carlo state spaces. In *Proc. 10th Internat. Workshop on Artificial Intelligence and Statistics*, 2005. (AISTATS'05). 425

[20] MARIA M. KLAWE AND DANIEL J. KLEITMAN: An almost linear time algorithm for generalized matrix searching. *SIAM J. Discrete Math.*, 3(1):81–97, 1990. [doi:10.1137/0403009] 417

[21] PETROS MARAGOS: Differential morphology. In MITRA AND SICURANZA, editors, *Nonlinear Image Processing*, pp. 289–329. Academic Press, 2001. [doi:10.1016/B978-012500451-0/50010-2] 417

[22] GEORGE PAPANDREOU AND PETROS MARAGOS: Multigrid geometric active contour models. *IEEE Trans. Image Processing*, 16(1):229–240, 2007. [doi:10.1109/TIP.2006.884952] 418, 425

[23] JUDEA PEARL: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. 418

[24] LAWRENCE R. RABINER: A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–289, 1989. [doi:10.1109/5.18626] 418

[25] NATHAN RATLIFF, MATT ZUCKER, J. ANDREW BAGNELL, AND SIDDHARTHA SRINIVASA: CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE Internat. Conf. on Robotics and Automation (ICRA'09)*, pp. 489–494. IEEE Comp. Soc. Press, 2009. [doi:10.1109/ROBOT.2009.5152817] 418, 425

[26] CHRISTOPHE RESTIF: Towards safer, faster prenatal genetic tests: Novel unsupervised, automatic and robust methods of segmentation of nuclei and probes. In *European Conf. on Computer Vision*, number 3954 in Lecture Notes in Computer Science, pp. 437–450. Springer, 2006. [doi:10.1007/11744085_34] 425

[27] AZRIEL ROSENFELD AND JOHN L. PFALTZ: Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, 1966. [doi:10.1145/321356.321357] 416, 417, 422

[28] DENIS RUTOVITZ: Data structures for operations on digital images. In CHENG ET AL., editor, *Pictorial Pattern Recognition*, pp. 105–133. Thomson Book, WA, 1968. 417

[29] JAMIE SHOTTON, ANDREW BLAKE, AND ROBERTO CIPOLLA: Contour-based learning for object detection. In *10th IEEE Internat. Conf. on Computer Vision (ICCV'05)*, pp. 503–510. IEEE Comp. Soc. Press, 2005. [doi:10.1109/ICCV.2005.63] 418

[30] TSUBASA YOSHIDA, KRIS M. KITANI, HIDEKI KOIKE, SERGE BELONGIE, AND KEVIN SCHLEI: EdgeSonic: image feature sonification for the visually impaired. In *Proc. 2nd Augmented Human Internat. Conf.*, pp. 11:1–11:4. ACM Press, 2011. [doi:10.1145/1959826.1959837] 425

AUTHORS

Pedro F. Felzenszwalb
Associate Professor
Brown University, Providence, RI
pff@brown.edu
http://www.cs.brown.edu/~pff

Daniel P. Huttenlocher
John P. and Rilla Neafsey Professor of Computing, Information Science and Business
Cornell University, Ithaca, NY
dph@cs.cornell.edu
http://www.cs.cornell.edu/~dph

ABOUT THE AUTHORS

PEDRO F. FELZENSZWALB received his Ph. D. from M.I.T. in 2003; his advisor was W. Eric L. Grimson. He is currently an Associate Professor of Engineering and Computer Science at Brown University. His research interests include computer vision, artificial intelligence, machine learning and discrete algorithms.

DANIEL P. HUTTENLOCHER received his Ph. D. from M.I.T. in 1988; his advisors were W. Eric L. Grimson and Shimon Ullman. He is currently the John P. and Rilla Neafsey Professor of Computing, Information Science and Business at Cornell University. He is also Dean of Computing and Information Science at Cornell. His research interests include computer vision, geometric algorithms and large-scale networks.