

FAST CONVOLUTION-BASED METHODS FOR COMPUTING THE SIGNED DISTANCE FUNCTION AND ITS DERIVATIVES

KARTHIK S. GURUMOORTHY, ANAND RANGARAJAN, AND MANU SETHI

ABSTRACT. We present a fast convolution-based technique for computing an approximate, signed Euclidean distance function at a set of $2D$ and $3D$ grid locations. Instead of solving the non-linear static Hamilton-Jacobi equation ($\|\nabla S\| = 1$), our solution stems from solving for a scalar field ϕ in a *linear* differential equation and then deriving the solution for S from its exponent. In other words, when S and ϕ are related by $\phi = \exp\left(-\frac{S}{\tau}\right)$ and ϕ satisfies a specific linear differential equation corresponding to the extremum of a variational problem, we obtain the Euclidean distance function $S = -\tau \log(\phi)$ in the limit as $\tau \rightarrow 0$. This is in sharp contrast to solvers such as the fast marching and fast sweeping methods which directly solve the Hamilton-Jacobi equation by the Godunov upwind discretization scheme. Our linear formulation provides us with a closed-form solution to the approximate Euclidean distance function expressed as a discrete convolution, and hence efficiently computed by the Fast Fourier Transform (FFT). Moreover, the solution circumvents the need for spatial discretization of the derivative operator thereby providing highly accurate results. As $\tau \rightarrow 0$, we show the convergence of our results to the true solution and also bound the error for a given value of τ . The differentiability of our solution allows us to compute—using a set of convolutions—the first and second derivatives of the approximate distance function. In order to determine the sign of the distance function (defined to be positive inside a closed region and negative outside), we compute the winding number in $2D$ and the topological degree in $3D$ and again explicitly show that these computations can be performed via fast convolutions.

1. INTRODUCTION

Euclidean distance functions (more popularly referred to as distance transforms) are widely used in image analysis and synthesis [12]. The task here is to assign at each grid point a value corresponding to the Euclidean distance to its nearest neighbor from a given point-set. Formally stated: given a point-set $Y = \{Y_k \in \mathbb{R}^D, k \in \{1, \dots, K\}\}$ where D is the dimensionality of the point-set and a set of equally spaced Cartesian grid points X , the Euclidean distance function problem requires us to assign

$$(1.1) \quad S(X) = \min_k \|X - Y_k\|$$

where $\|\cdot\|$ represents its Euclidean norm. In computation geometry, this is the Voronoi problem [6] and the solution $S(X)$ can be visualized as a set of cones with the centers being the point-set locations $\{Y_k\}$. The Euclidean distance function problem is a special case of the eikonal equation where the forcing function is

2010 *Mathematics Subject Classification.* Primary 65Z05, 35J05; Secondary 35J08.

identically equal to 1 everywhere and hence the distance function S satisfies the differential equation

$$(1.2) \quad \|\nabla S\| = 1,$$

barring the point-set locations and the Voronoi boundaries where it is not differentiable. Here $\nabla S = (S_x, S_y)$ denotes the gradients of S . This is a nonlinear differential equation and an example of a static Hamilton-Jacobi equation.

Since the advent of the fast marching method [13, 12], the literature is replete with pioneering works which have successfully tackled this problem. Fast marching is an elegant technique which solves for S in $O(N \log N)$ time at the given N grid locations using the Godunov upwind discretization scheme. Faster methods like the fast sweeping method [20] employs Gauss-Seidel iterations and solves for S in $O(N)$. The ingenious work in [19] gives an $O(N)$ implementation of the fast marching method with a cleverly chosen *untidy priority queue* data structure. Fast sweeping methods have also been extended to the more general static Hamilton-Jacobi equation [11] and also for the eikonal equation on non-regular grids [9, 10]. A Hamiltonian approach to solve the eikonal equation can be found in [16].

The intriguing aspect of our method is that a nonlinear Hamilton-Jacobi equation is obtained in the limit as $\tau \rightarrow 0$ of a linear equation. When the distance function S is expressed as the exponent of a scalar field ϕ , specifically $\phi(X) = \exp(\frac{-S(X)}{\tau})$, and if $\phi(X)$ is the solution to a variational problem satisfying the linear Euler-Lagrange equation, we show that as $\tau \rightarrow 0$, S satisfies Equation 1.2. Consequently, instead of solving the non-linear Hamilton-Jacobi equation, one can solve for the function ϕ (taking advantage of its linearity), and then compute an approximate S for small values of τ . This computational procedure would be approximately equivalent to solving the original Hamilton-Jacobi equation. Our linear approach to the Euclidean distance function problem results in a closed-form solution which can be expressed as a discrete convolution and computed in $O(N \log N)$ time using a Fast Fourier Transform (FFT) [3] where N is the number of grid points. This is major advantage of our method as the closed-form solution circumvents the need for spatial discretization of the derivative operator in equation (1.2), a problem that permeates the Hamilton-Jacobi solvers [13, 12, 20]. This accounts for improved accuracy of our technique as we see in the experimental section. A minor caveat of our method is that the resultant Euclidean distance function is an approximation since it is obtained for a small but non-zero value of τ , but nevertheless converges to the true solution as $\tau \rightarrow 0$.

The linear approach gives us an unsigned distance function. We complement this by independently finding the sign of the distance function in $O(N \log N)$ time on a regular grid in $2D$ and $3D$. We achieve this by efficiently computing the *winding number* for each location in the $2D$ grid and its equivalent concept, the *topological degree* in $3D$. We show that just as in the case of the unsigned Euclidean distance function, the winding number and the topological degree computations can also be written in closed-form, expressed as discrete convolutions and efficiently computed using FFTs. We are not cognizant of any previous work that uses the winding number and topological degree approaches to compute signed distance functions.

Very often, we also seek the gradient, divergence, curvature and medial axes of the signed distance function which are not easy to obtain by Hamilton-Jacobi approaches due to the lack of differentiability of the signed distance function. But we can leverage the closed-form solution obtained from our method to compute these

quantities. Since our approximate distance function is differentiable everywhere, we can once again write down closed-form expressions for the gradients and curvature, express them as discrete convolutions and efficiently compute these quantities in $O(N \log N)$ using FFTs. We visualize the gradients and the maximum curvature using 2D shape silhouettes as the source. To our knowledge, fast computation of the derivatives of the distance function on a regular grid using discrete convolutions is new.

The paper is organized as follows. In Section 2 we derive the linear equation formalism for the Euclidean distance function problem and show convergence of our solution to the true solution as $\tau \rightarrow 0$. We provide an approximate closed-form solution to compute the distance function and give an error bound between the computed and true distance functions for a given value of τ . Section 3 demonstrates how the closed-form solution can be represented as a convolution and thus efficiently computed using fast Fourier transforms. In Sections 4 and 5, we compute the winding number (in $2D$), the topological degree (in $3D$) and the derivatives of the distance function, by again expressing these quantities using discrete convolutions. In Section 6, we show anecdotal evidences for the usefulness of our method by furnishing both experimental results and comparisons to standard techniques. We finally conclude in Section 7.

2. LINEAR EQUATION APPROACH FOR EUCLIDEAN DISTANCE FUNCTIONS

Consider a variational problem

$$(2.1) \quad I(\phi) = \tau^2 \int \|\nabla \phi\|^2 dX + \int |\phi - \phi_0^\tau|^2 dX$$

where ϕ_0^τ —a function of τ —represents the initial wave front concentrated around the source locations $\{Y_k\}_{k=1}^K$ in the as $\tau \rightarrow 0$. We define $\phi_0^\tau(X)$ as

$$(2.2) \quad \phi_0^\tau(X) = \sum_{k=1}^K \phi_k^\tau(X)$$

where $\phi_k^\tau(X)$ is chosen such that it is *square integrable* to 1 with its support sequentially converging towards the point source Y_k as τ approaches zero and asymptotically behaves like the square-root of a δ function centered around Y_k . The square integrability to 1 constraint changes its functional form in accordance with the spatial dimension, as explicated in the subsequent sections.

The Euler-Lagrange equation corresponding to the extremum of $I(\phi)$ computed over the scalar field ϕ is given by the *linear* equation

$$(2.3) \quad -\tau^2 \nabla^2 \phi + \phi = \phi_0^\tau,$$

where ∇ stands for the Laplacian operator. We may be tempted to replace ϕ_0^τ in Equation 2.3 with a combination of delta functions each centered around Y_k and obtain an *inhomogeneous screened Poisson* partial differential equation. But since the delta functions are not square-integrable, they cannot be incorporated into the variational framework given in Equation 2.1. Defining ϕ_0^τ as in Equation 2.2 forces it to behave like the square-root of a δ function as $\tau \rightarrow 0$ and hence is square-integrable for all values of τ .

To our amazement, we realized that when we relate $\phi(X) = \exp(\frac{-S(X)}{\tau})$ and ϕ satisfies Equation 2.3, S asymptotically satisfies the Hamilton-Jacobi equation 1.2 in the limit $\tau \rightarrow 0$, as described under Section 2.2. This relationship motivates us

to solve the linear equation 2.3 instead of the non-linear eikonal equation and then compute the distance function via

$$(2.4) \quad S(X) = -\tau \log \phi(X).$$

2.1. Solution for the Euclidean distance function. We now derive the solution for $\phi(X)$ (in $1D$, $2D$ and $3D$) satisfying Equation 2.3 and then for $S(X)$ from the relation 2.4.

Since it is meaningful to assume that $S(X)$ approaches infinity for points at infinity, we can use Dirichlet boundary conditions $\phi(X) = 0$ at the boundary of an unbounded domain. Using a Green's function approach [2], we can write expressions for the solution ϕ . The Green's function G satisfies the relation

$$(2.5) \quad (-\tau^2 \nabla^2 + 1)G(X) = -\delta(X).$$

The form of the solution for G [2] in $1D$, $2D$ and $3D$ over an unbounded domain with vanishing boundary conditions at ∞ is given by,

1D:

$$(2.6) \quad G(X) = \frac{1}{2\tau} \exp\left(\frac{-|X|}{\tau}\right).$$

2D:

$$(2.7) \quad \begin{aligned} G(X) &= \frac{1}{2\pi\tau^2} K_0\left(\frac{\|X\|}{\tau}\right) \\ &\approx \frac{\exp\left(\frac{-\|X\|}{\tau}\right)}{2\tau\sqrt{2\pi\tau\|X\|}}, \frac{\|X\|}{\tau} \gg 0.25 \end{aligned}$$

where K_0 is the modified Bessel function of the second kind.

3D:

$$(2.8) \quad G(X) = \frac{1}{4\pi\tau^2} \frac{\exp\left(\frac{-\|X\|}{\tau}\right)}{\|X\|}.$$

The solution for ϕ can then be obtained via convolution as

$$(2.9) \quad \phi(X) = G(X) * \phi_0^\tau(X) = \sum_{k=1}^K G(X) * \phi_k^\tau(X)$$

from which S can be recovered using the mathematical relation 2.4.

2.2. Proofs of convergence. We now prove that as $\tau \rightarrow 0$, $S(X)$ —obtained from the exponent of ϕ which satisfies Euler-Lagrange equation $\text{refeq:phiEquationwithtau}$ —converges to the true solution $r(X) = \min_k \|X - Y_k\| = \|X - Y_{k_0}\|$. We show this explicitly for each spatial dimension.

1D: Since we require $\phi_k^\tau(X)$ to behave like a square-root of the δ function centered at Y_k as $\tau \rightarrow 0$, we define it as

$$\phi_k^\tau(X) = \begin{cases} \frac{1}{\sqrt{\tau}} & \text{for } Y_k - \frac{\tau}{2} \leq X \leq Y_k + \frac{\tau}{2}; \\ 0 & \text{otherwise} \end{cases}$$

Plugging it in Equation 2.9 and using the expression for the 1D Green's function, we solve for ϕ as

$$(2.10) \quad \phi(X) = \frac{1}{2\tau^{\frac{3}{2}}} \sum_{k=1}^K \int_{X-Y_k-\frac{\tau}{2}}^{X-Y_k+\frac{\tau}{2}} \exp\left(\frac{-|Z|}{\tau}\right) dZ.$$

Using the relation (2.4), the Euclidean distance function is given by

$$(2.11) \quad S(X) = C_\tau - \tau \log \left(\sum_{k=1}^K \int_{\mathcal{B}_k^\tau(X)} \exp\left(\frac{-|Z|}{\tau}\right) dZ \right),$$

where $C_\tau = \tau \log(2) + \frac{3}{2}\tau \log(\tau)$ and the integration region $\mathcal{B}_k^\tau(X)$ equals

$$(2.12) \quad \mathcal{B}_k^\tau(X) = \left[X - Y_k - \frac{\tau}{2}, X - Y_k + \frac{\tau}{2} \right].$$

In order to show convergence to $r(X)$ as τ approaches zero, we define

$$\alpha_k \equiv \begin{cases} 1 & \text{if } X > Y_k; \\ -1 & \text{if } X < Y_k \end{cases}$$

for each k . Then for sufficiently small τ we get

$$(2.13) \quad \begin{aligned} S(X) &\leq C_\tau - \tau \log \left\{ \tau \exp\left(\frac{-|X - Y_{k_0} + \alpha_{k_0} \frac{\tau}{2}|}{\tau}\right) \right\} \\ &= C_\tau - \tau \log(\tau) + \left| X - Y_{k_0} + \alpha_{k_0} \frac{\tau}{2} \right|. \end{aligned}$$

as $|X - Y_{k_0} + \alpha_{k_0} \frac{\tau}{2}| \geq |Z|$ for $Z \in \mathcal{B}_{k_0}^\tau$.

On the other hand, since $|X - Y_k - \alpha_k \frac{\tau}{2}| \leq |Z|, \forall Z \in \mathcal{B}_k^\tau$ at small values of τ , we also get

$$(2.14) \quad \begin{aligned} S(X) &\geq C_\tau - \tau \log \left\{ \tau \sum_{k=1}^K \exp\left(\frac{-|X - Y_k - \alpha_k \frac{\tau}{2}|}{\tau}\right) \right\} \\ &\geq C_\tau - \tau \log(\tau) - \tau \log \left\{ K \exp\left(\frac{-|X - Y_{k_0} - \alpha_{k_0} \frac{\tau}{2}|}{\tau}\right) \right\} \\ &= C_\tau - \tau \log(\tau) - \tau \log(K) + \left| X - Y_{k_0} - \alpha_{k_0} \frac{\tau}{2} \right|. \end{aligned}$$

In order to see why the second step in the above relation holds, consider the two scenarios (i) X lies on the Voronoi boundary and (ii) X is not a point on the Voronoi boundary. If X doesn't lie on the Voronoi boundary, then exist a neighborhood $N_p(X)$ around X such that $\forall Y \in N_p(X), |Y - Y_{k_0}| < |Y - Y_k|, \forall k$. Since $|X - \alpha_{k_0} \frac{\tau}{2}| \in N_p(X)$ for sufficiently small values of τ , the aforementioned relation is true. On the flip side if X is a point on the Voronoi boundary, the closest source point Y_k is not uniquely defined. However we can unambiguously choose a source point Y_{k_0} and a τ_0 such that for $\tau \in (0, \tau_0], |X - \alpha_{k_0} \tau - Y_{k_0}| < |X - \alpha_k \tau - Y_k|, \forall k$ and $|X - Y_{k_0}| \leq |X - Y_k|, \forall k$. This observation ascertains the above inequality.

Since $C_\tau, \tau \log \tau$ and $\tau \log K$ approach zero as $\tau \rightarrow 0$, we have $\lim_{\tau \rightarrow 0} S(X) = |X - Y_{k_0}| = r(X)$.

2D: Let the grid location X and the point source Y_k be represented by (x, y)

and (x_k, y_k) respectively. We define $\phi_k^\tau(X)$ as

$$\phi_k^\tau(X) = \begin{cases} \frac{1}{\tau}; & x_k - \frac{\tau}{2} \leq x \leq x_k + \frac{\tau}{2}, \\ & y_k - \frac{\tau}{2} \leq y \leq y_k + \frac{\tau}{2}; \\ 0 & \text{otherwise} \end{cases}$$

such that it is square integrable to 1 and behaves like the square-root of the δ function centered at Y_k as τ tends to zero. Using the expression for the 2D Green's function and the relation 2.4, the Euclidean distance function is given by

$$(2.15) \quad S(X) = C_\tau - \tau \log \left(\sum_{k=1}^K \int_{\mathcal{B}_k^\tau(X)} K_0 \left(\frac{\|Z\|}{\tau} \right) dZ \right),$$

where $C_\tau = \tau \log(2\pi) + 3\tau \log \tau$ and the integral region $\mathcal{B}_k^\tau(X)$ equals

$$(2.16) \quad \mathcal{B}_k^\tau(X) = \left[x - x_k - \frac{\tau}{2}, x - x_k + \frac{\tau}{2} \right] \times \left[y - y_k - \frac{\tau}{2}, y - y_k + \frac{\tau}{2} \right].$$

Defining

$$\alpha_k \equiv \begin{cases} 1 & \text{if } x > x_k; \\ -1 & \text{if } x < x_k \end{cases}$$

and

$$\beta_k \equiv \begin{cases} 1 & \text{if } y > y_k; \\ -1 & \text{if } y < y_k \end{cases}$$

for each k and closely following the arguments illustrated for the 1D case, we get

$$(2.17) \quad \begin{aligned} S(X) &\leq C_\tau - \tau \log \left\{ \tau K_0 \left(\frac{\|X_{1\tau} - Y_{k_0}\|}{\tau} \right) \right\} \\ &= C_\tau - \tau \log(\tau) - \tau \log \left\{ K_0 \left(\frac{\|X_{1\tau} - Y_{k_0}\|}{\tau} \right) \right\} \end{aligned}$$

for small values of τ , where $X_{1\tau} = (x + \alpha_k \frac{\tau}{2}, y + \beta_k \frac{\tau}{2})$. As in the 1D case, note that $\|X_{1\tau} - Y_{k_0}\| \geq \|Z\|$ for $Z \in \mathcal{B}_{k_0}$.

Using the relation $K_0(z) \geq \frac{\exp(-z)}{\sqrt{z}}$ when $z \geq 0.5$, we observe that

$$(2.18) \quad S(X) \leq C_\tau - \tau \log(\tau) + \tau \log \left\{ \sqrt{\frac{\|X_{1\tau} - Y_{k_0}\|}{\tau}} \right\} + \|X_{1\tau} - Y_{k_0}\|$$

as τ approaches zero.

If we let $X_{2\tau} = (x - \alpha_k \frac{\tau}{2}, y - \beta_k \frac{\tau}{2})$, then similarly to the 1D case we arrive at the inequality

$$(2.19) \quad \begin{aligned} S(X) &\geq C_\tau - \tau \log \left\{ \tau \sum_{k=1}^K K_0 \left(\frac{\|X_{2\tau} - Y_k\|}{\tau} \right) \right\} \\ &\geq C_\tau - \tau \log(\tau) - \tau \log \left\{ K K_0 \left(\frac{\|X_{2\tau} - Y_{k_0}\|}{\tau} \right) \right\} \\ &= C_\tau - \tau \log(\tau) - \tau \log(K) - \tau \log \left\{ K_0 \left(\frac{\|X_{2\tau} - Y_{k_0}\|}{\tau} \right) \right\} \end{aligned}$$

As $K_0(z) \leq \exp(-z)$ when $z \geq 1.5$, we get

$$(2.20) \quad S(X) \geq C_\tau - \tau \log(\tau) - \tau \log(K) + \|X_{2\tau} - Y_{k_0}\|$$

at small values of τ . Since $X_{1\tau}, X_{2\tau}$ approach X as $\tau \rightarrow 0$ and the rest of the terms tend to zero in the limit, we arrive at our desired result, namely $\lim_{\tau \rightarrow 0} S(X) =$

$$\|X - Y_{k_0}\| = r(X).$$

3D: Let the grid location X and the source point Y_k be denoted by $X = (x, y, z)$ and $Y_k = (x_k, y_k, z_k)$ respectively. To ensure square-integrability to 1 we define ϕ_k^τ as

$$\phi_k^\tau(X) = \begin{cases} \frac{1}{\tau^{\frac{3}{2}}}; & x_k - \frac{\tau}{2} \leq x \leq x_k + \frac{\tau}{2}, \\ & y_k - \frac{\tau}{2} \leq y \leq y_k + \frac{\tau}{2}, \\ & z_k - \frac{\tau}{2} \leq z \leq z_k + \frac{\tau}{2}, \\ 0 & \text{otherwise} \end{cases}$$

By exactly following the line of argument described for the 1D and the 2D case where we bound $S(X)$ above and below by functions which converge to the true Euclidean distance function $r(X)$ as τ tends to zero, we can establish the result $\lim_{\tau \rightarrow 0} S(X) = \|X - Y_{k_0}\| = r(X)$.

2.3. Closed-form solution and the error bound between the obtained and true Euclidean distance function. Based on the nature of the expression for the Green's function, it is worth accentuating the following *very* important point. Observe from above that the expression for Green's function G in either 1D, 2D or 3D takes the form

$$(2.21) \quad \lim_{\tau \rightarrow 0} \frac{\exp\left\{-\frac{\|X\|}{\tau}\right\}}{c\tau^d \|X\|^p} = 0, \text{ for } \|X\| \neq 0$$

for c, d and p being constants greater than zero. In the limiting case of $\tau \rightarrow 0$, we see that if we define

$$(2.22) \quad \tilde{G}(X) \equiv C \exp\left(\frac{-\|X\|}{\tau}\right),$$

for some constant C , then

$$(2.23) \quad \lim_{\tau \rightarrow 0} |G(X) - \tilde{G}(X)| = 0, \text{ for } \|X\| \neq 0$$

and furthermore, the convergence is *uniform* for $\|X\|$ away from zero. Therefore, $\tilde{G}(X)$ provides a very good approximation for the actual unbounded domain Green's function as $\tau \rightarrow 0$. For a fixed value of τ and X , the difference between the Green's functions is $O\left(\frac{\exp\left(\frac{-\|X\|}{\tau}\right)}{\tau^2}\right)$ which is relatively insignificant for small values of τ and for all $X \neq 0$. Moreover, using \tilde{G} also avoids the singularity at the origin in the 2D and 3D case. The above observation motivates us to compute the solutions for ϕ by convolving with \tilde{G} instead of the actual Green's function G .

Furthermore, our proof technique used to manifest convergence of $S(X)$ to the true solution $r(X)$ —described in the preceding section—also encourages us to supplant the integral $\int_{\mathcal{B}_k^\tau(X)} G(Z) dZ$ —obtained as a result of convolving the Green's function G with ϕ_k^τ —with $\tau^D G(X - Y_k)$ at small values of τ . Here D corresponds to the spatial dimension. This is a cogent approximation for the following reasons. Firstly, the integral region $\mathcal{B}_k^\tau(X)$ is considered around $X - Y_k$ with its area/volume dwindling to zero as τ approaches zero (refer Equations 2.12 and 2.16). Hence we can replace the integral $\int_{\mathcal{B}_k^\tau(X)} G(Z) dZ$ with its Riemann summation by assuming that $G(Z)$ is *constant* over $\mathcal{B}_k^\tau(X)$ and equals the mid-point value $G(X - Y_k)$. Secondly, our proof explicitly corroborates that substituting the integral by either

$\tau^D \sup_{Z \in \mathcal{B}_k^\tau(X)} G(Z)$ or $\tau^D \inf_{Z \in \mathcal{B}_k^\tau(X)} G(Z)$, we can still establish convergence to the true solution. Since $\inf_{Z \in \mathcal{B}_k^\tau(X)} G(Z) \leq G(X - Y_k) \leq \sup_{Z \in \mathcal{B}_k^\tau(X)} G(Z)$, our approximation is sound at small values of τ . Thirdly, it bestows upon us with a *closed-form* solution for ϕ as seen below.

These insights inspires us to solve for the scalar field ϕ as

$$(2.24) \quad \phi(X) \approx \left\{ \tau^D \tau^{-\gamma} C \sum_{k=1}^K \exp\left(\frac{-\|X - Y_k\|}{\tau}\right) \right\}$$

rather than via Equation 2.9. Here $\tau^{-\gamma}$ corresponds to the value of $\phi_k^\tau(Y_k)$, with γ determined by the spatial dimension. The approximate Euclidean distance function computed based on the relation 2.4 is given by

$$(2.25) \quad S(X) \approx (\gamma - D)\tau \log \tau - \tau \log C - \tau \log \left\{ \sum_{k=1}^K \exp\left(\frac{-\|X - Y_k\|}{\tau}\right) \right\}.$$

Since $(\gamma - D)\tau \log \tau$ and $\tau \log C$ are constants independent of X and converges to zero as $\tau \rightarrow 0$, they can be ignored while solving for S at small values of τ . Hence the scalar field $\phi(X)$ corresponding to the Euclidean distance function can be approximated by

$$(2.26) \quad \phi(X) \approx \sum_{k=1}^K \exp\left(\frac{-\|X - Y_k\|}{\tau}\right).$$

and the approximate Euclidean distance function equals

$$(2.27) \quad S(X) = -\tau \log \left\{ \sum_{k=1}^K \exp\left(\frac{-\|X - Y_k\|}{\tau}\right) \right\}.$$

We would like to underscore that the approximate function defined in Equation 2.26 possess all the desirable properties that we need. Firstly, we notice that as $\tau \rightarrow 0$, $\phi(Y_k) \rightarrow 1$ at the given point-set locations Y_k . Using the relation (2.4) we get $S(Y_k) \rightarrow 0$ as $\tau \rightarrow 0$ satisfying the initial conditions. Secondly for small values of τ , $\sum_{k=1}^K \exp\left(\frac{-\|X - Y_k\|}{\tau}\right)$ can be subrogated by $\exp\left(\frac{-r(X)}{\tau}\right)$ where $r(X)$ is the true Euclidean distance given by $r(X) = \min_k \|X - Y_k\|$. Hence

$$(2.28) \quad S(X) \approx -\tau \log \exp\left(\frac{-r(X)}{\tau}\right) = r(X).$$

Thirdly, $\phi(X)$ can be easily computed using the fast Fourier transform as discussed under the subsequent section. Hence for all practical, computational purposes we consider the function shown in Equation 2.26. The bound derived below between the approximate $S(X)$ and the true solution $r(X)$ also unveils the proximity between the computed and the actual Euclidean distance function.

Note from Equation 2.27 that

$$(2.29) \quad S(X) \leq -\tau \log \exp\left(\frac{-r(X)}{\tau}\right) = r(X).$$

Also we get

$$(2.30) \quad \begin{aligned} S(X) &\geq -\tau \log \left\{ K \exp\left(\frac{-r(X)}{\tau}\right) \right\} \\ &= -\tau \log K + r(X) \end{aligned}$$

TABLE 1. Approximate Euclidean distance function algorithm

1.	Compute the function $f(X) = \exp\left(\frac{-\ X\ }{\tau}\right)$ at the grid locations.
2.	Define the function $\delta_{kron}(X)$ which takes the value 1 at the point-set locations and 0 at other grid locations.
3.	Compute the <i>FFT</i> of f and g , namely $F_{FFT}(U)$ and $G_{FFT}(U)$ respectively.
4.	Compute the function $H(U) = F_{FFT}(U)G_{FFT}(U)$.
5.	Compute the inverse <i>FFT</i> of $H(U)$ to obtain $\phi(X)$.
6.	Take the logarithm of $\phi(X)$ and multiply it by $(-\tau)$ to recover the approximate Euclidean distance function.

and hence,

$$(2.31) \quad r(X) - S(X) \leq \tau \log K.$$

From Equations 2.29 and 2.31, we have

$$(2.32) \quad |r(X) - S(X)| \leq \tau \log K.$$

It is worth commenting that the bound $\tau \log K$ is actually very tight as (i) it scales only as the logarithm of the cardinality of the point-set (K) and (ii) it can be made arbitrarily small by choosing a small but non-zero value of τ .

3. EFFICIENT COMPUTATION OF THE APPROXIMATE UNSIGNED EUCLIDEAN DISTANCE FUNCTION

We now provide a fast $O(N \log N)$ convolution-based method to compute the distance transform on a set of N grid locations $\{X_i, i = \{1, \dots, N\}\}$. The solution for $\phi(X)$ in Equation 2.26 at the grid locations can be represented as the *discrete convolution* between the functions

$$(3.1) \quad f(X) \equiv \exp\left(\frac{-\|X\|}{\tau}\right)$$

computed at the grid locations, with the function $g(X)$ which takes the value 1 at the point-set locations and 0 at other grid locations, i.e,

$$(3.2) \quad g(X) \equiv \sum_{k=1}^K \delta_{kron}(X - Y_k)$$

where,

$$(3.3) \quad \delta_{kron}(X - Y_k) \equiv \begin{cases} 1 & \text{if } X = Y_k; \\ 0 & \text{otherwise} \end{cases}$$

By the convolution theorem [3], a discrete convolution can be obtained as the inverse Fourier transform of the product of two individual transforms, which for two $O(N)$ sequences can be computed in $O(N \log N)$ time [5]. One just needs to compute the discrete Fourier transform (DFT) of the sampled version of the functions $f(X)$ and $g(X)$, compute their point-wise product and then compute the inverse discrete Fourier transform. Taking the logarithm of the inverse discrete Fourier transform and multiplying it by $(-\tau)$, gives the approximate Euclidean distance function. The algorithm is adumbrated in Table 1.

3.1. Computation of the approximate Euclidean distance function in higher dimensions. Our technique has a straightforward generalization to higher dimensions. Regardless of the spatial dimension, the approximate Euclidean distance function, S can be computed by exactly following the steps delineated in Table 1. It is worthwhile mentioning that computing the discrete Fourier transform using the *FFT* is always $O(N \log N)$ *irrespective* of the spatial dimension. Hence, for all dimensions, S can be computed at the given N grid points in $O(N \log N)$. This speaks for the scalability of our technique, which is generally not the case with other methods, for example KD-Trees [6]¹.

3.2. Numerical issues. In principle, we should be able to apply our technique at very small values of τ and obtain highly accurate results. But we noticed that a naïve double precision-based implementation tends to deteriorate for τ values very close to zero. This is due to the fact that at small values of τ , $f(X)$ drops off very quickly and hence for grid locations which are far away from the point-set, the convolution done using FFT may not be accurate. To this end, we turned to the GNU MPFR multiple-precision arithmetic library which provides arbitrary precision arithmetic with correct rounding [7]. MPFR is based on the GNU multiple-precision library (GMP) [18]. It enabled us to run our technique at very small values of τ giving highly accurate results. We corroborate our claim and demonstrate the usefulness of our method with the set of experiments described under Section 6.

3.3. Exact computational complexity. More the number of precision bits p used in the GNU MPFR library, better is the accuracy of our technique, as the error incurred in the floating point operations can be bounded by $O(2^{-p})$. But using more bits has an adverse effect of slowing down the running time. The $O(N \log N)$ time complexity of the FFT algorithm [5] for an $O(N)$ length sequence represents only the number of floating-point operations involved, barring any numerical accuracy. The accuracy of the FFT algorithm and our technique entirely depends on the number of precision bits used for computing elementary functions like exp, log, sin and cos and hence should be taken into account while calculating the exact time complexity. If p precision bits are used, these elementary functions can be computed in $O(M(p) \log p)$ [4, 14, 17], where $M(p)$ is the computational complexity for multiplying two p -digit numbers. The Schönhage-Strassen algorithm [15] gives an asymptotic upper bound for $M(p)$ with a run-time bit complexity of $M(p) = O(p \log p \log \log p)$. The actual running time of our algorithm—while taking these p precision bits into account—for computing S at the given N grid locations is then $O(N \log(N)p(\log p)^2 \log(\log p))$ bit-wise operations.

4. FAST COMPUTATION OF SIGNED DISTANCE FUNCTIONS

The solution for the approximate Euclidean distance function in Equation 2.27 is lacking in one respect: there is no information on the sign of the distance. This is to be expected since the distance function was obtained only from a set of *points* Y and not a curve or surface. We now describe a new method for computing the signed distance function in 2D using winding numbers and in 3D using the topological degree.

¹Though the actual number of grid points(N) increases with dimension, the solution is always $O(N \log N)$ in the number of grid points.

4.1. Computing winding numbers. Assume that we have a closed, parametric curve $\{x^{(1)}(t), x^{(2)}(t)\}$, $t \in [0, 1]$. We seek to determine if a grid location in the set $\{X_i \in \mathbb{R}^2, i \in \{1, \dots, N\}\}$ is inside the closed curve. The winding number is the number of times the curve winds around the point X_i (if at all) with counter-clockwise turns counted as positive and clockwise turns as negative. If a point is inside the curve, the winding number is a non-zero integer. If the point is outside the curve, the winding number is zero. If we can efficiently compute the winding number for all points on a grid w.r.t. to a curve, then we would have the sign information (inside/outside) for all the points. We now describe a fast algorithm to achieve this goal.

If the curve is C^1 , then the angle $\theta(t)$ of the curve is continuous and differentiable and $d\theta(t) = \left(\frac{x^{(1)}\dot{x}^{(2)} - x^{(2)}\dot{x}^{(1)}}{\|x\|^2} \right) dt$. Since we need to determine whether the curve winds around each of the points $X_i, i \in \{1, \dots, N\}$, define $(\hat{x}_i^{(1)}, \hat{x}_i^{(2)}) \equiv (x^{(1)} - X_i^{(1)}, x^{(2)} - X_i^{(2)})$, $\forall i$. Then the winding numbers for the grid point X_i is

$$(4.1) \quad \mu_i = \frac{1}{2\pi} \oint_C \left(\frac{\hat{x}_i^{(1)}\dot{\hat{x}}_i^{(2)} - \hat{x}_i^{(2)}\dot{\hat{x}}_i^{(1)}}{\|\hat{x}_i\|^2} \right) dt, \forall i \in \{1, \dots, N\}.$$

As it stands, we cannot actually compute the winding numbers without performing the integral in Equation 4.1. To this end, we discretize the curve and produce a sequence of points $\{Y_k \in \mathbb{R}^2, k \in \{1, \dots, K\}\}$ with the understanding that the curve is closed and therefore the “next” point after Y_K is Y_1 . (The winding number property holds for piecewise continuous curves as well.) The integral in Equation 4.1 becomes a discrete summation and we get

$$(4.2) \quad \mu_i = \frac{1}{2\pi} \sum_{k=1}^K \frac{(Y_k^{(1)} - X_i^{(1)})(Y_{k\oplus 1}^{(2)} - Y_k^{(2)}) - (Y_k^{(2)} - X_i^{(2)})(Y_{k\oplus 1}^{(1)} - Y_k^{(1)})}{\|Y_k - X_i\|^2}$$

$\forall i \in \{1, \dots, N\}$, where the notation $Y_{k\oplus 1}^{(\cdot)}$ denotes that $Y_{k\oplus 1}^{(\cdot)} = Y_{k+1}^{(\cdot)}$ for $k \in \{1, \dots, K-1\}$ and $Y_{K\oplus 1}^{(\cdot)} = Y_1^{(\cdot)}$. We can simplify the notation in Equation 4.2 (and obtain a measure of conceptual clarity as well) by defining the “tangent” vector $\{Z_k, k = \{1, \dots, K\}\}$ as

$$(4.3) \quad Z_k^{(\cdot)} = Y_{k\oplus 1}^{(\cdot)} - Y_k^{(\cdot)}, k \in \{1, \dots, K\}$$

with the (\cdot) symbol indicating either coordinate. Using the tangent vector Z_k , we rewrite Equation 4.2 as

$$(4.4) \quad \mu_i = \frac{1}{2\pi} \sum_{k=1}^K \frac{(Y_k^{(1)} - X_i^{(1)})Z_k^{(2)} - (Y_k^{(2)} - X_i^{(2)})Z_k^{(1)}}{\|Y_k - X_i\|^2}, \forall i \in \{1, \dots, N\}$$

We now make the somewhat surprising observation that μ_i in Equation 4.4 is a sum of two discrete convolutions. The first convolution is between two functions $f_{cr}(X) \equiv f_c(X)f_r(X)$ and $g_2(X) = \sum_{k=1}^K Z_k^{(2)}\delta_{\text{kron}}(X - Y_k)$. The second convolution is between two functions $f_{sr}(X) \equiv f_s(X)f_r(X)$ and $g_1(X) \equiv \sum_{k=1}^K Z_k^{(1)}\delta_{\text{kron}}(X - Y_k)$. The Kronecker delta function $\delta_{\text{kron}}(X - Y_k)$ is defined

Equation 3.3. The functions $f_c(X)$, $f_s(X)$ and $f_r(X)$ are defined as

$$(4.5) \quad f_c(X) \equiv \frac{X^{(1)}}{\|X\|}, f_s(X) \equiv \frac{X^{(2)}}{\|X\|}, \text{ and}$$

$$(4.6) \quad f_r(X) \equiv \frac{1}{\|X\|}$$

with the understanding that $f_c(0) = f_s(0) = f_r(0) = 0$. Here we have abused notation somewhat and let $X^{(1)}$ ($X^{(2)}$) denote the x (y)-coordinate of the grid point X . Armed with these relationships, we rewrite (4.4) to get

$$(4.7) \quad \mu(X) = \frac{1}{2\pi} [-f_{cr}(X) * g_2(X) + f_{sr}(X) * g_1(X)]$$

which can be computed in $O(N \log N)$ time using FFT-based convolution *simultaneously* for all the N grid points $\{X_i, i = \{1, \dots, N\}\}$.

4.2. Computing topological degree. The winding number concept for 2D admits a straight forward generalization to 3D and higher dimensions. The equivalent concept is the topological degree which is based on normalized flux computations. Assume that we have an oriented surface in 3D [8] which is represented as a set of K triangles. The k^{th} triangle has an outward pointing normal P_k and this can easily be obtained once the surface is oriented. (We vectorize the edge of each triangle. Since triangles share edges, if the surface can be oriented, then there's a consistent way of lending direction to each triangle edge. The triangle normal is merely the cross-product of the triangle vector edges.) We pick a convenient triangle center (the triangle incenter for instance) for each triangle and call it Y_k . The normalized flux (which is very closely related to the topological degree) [1] determines the ratio of the outward flux from a point X_i treated as the origin. If X_i is outside the enclosed surface, then the total outward flux is zero. If the point is inside, the outward normalized flux will be non-zero and positive.

The normalized flux for a point X_i is

$$(4.8) \quad \mu_i = \frac{1}{4\pi} \sum_{k=1}^K \frac{\langle (Y_k - X_i), P_k \rangle}{\|Y_k - X_i\|^3}.$$

This can be written in the form of convolutions. To see this, we write Equation 4.8 in component form:

$$(4.9) \quad \mu_i = \frac{1}{4\pi} \sum_{k=1}^K \frac{(Y_k^{(1)} - X_i^{(1)})P_k^{(1)} + (Y_k^{(2)} - X_i^{(2)})P_k^{(2)} + (Y_k^{(3)} - X_i^{(3)})P_k^{(3)}}{\|Y_k - X_i\|^3}$$

which can be simplified as

$$(4.10) \quad \mu(X) = -\frac{1}{4\pi} (f_1(X) * g_1(X) + f_2(X) * g_2(X) + f_3(X) * g_3(X))$$

where $f_{(\cdot)}(X) \equiv \frac{X^{(\cdot)}}{\|X\|^3}$ and $g_{(\cdot)}(X) \equiv \sum_{k=1}^K P_k^{(\cdot)} \delta_{\text{kron}}(X - Y_k)$ where the Kronecker delta function $\delta_{\text{kron}}(X - Y_k)$ is defined Equation 3.3. This can be computed in $O(N \log N)$ time using FFT-based convolution for all the N grid points X_i .

For the sake of clarity we explicitly show the generalization of the winding number to the topological degree by rewriting some of the calculations involved in computing the winding number. Recall that for every point Y_k on the discretized curve, we defined its tangent vector Z_k as in equation (4.3). The *outward pointing normal*

$P_k = (P_k^{(1)}, P_k^{(2)})$, at the point Y_k (P_k will point outwards provided Y_1, Y_2, \dots, Y_k are taken in the anti-clockwise order), is given by $P_k^{(1)} = Z_k^{(2)}, P_k^{(2)} = -Z_k^{(1)}$. Using the normal vector P_k , equation (4.4) can be rewritten as

$$(4.11) \quad \mu_i = \frac{1}{2\pi} \sum_{k=1}^K \frac{\langle (Y_k - X_i), P_k \rangle}{\|Y_k - X_i\|^2}.$$

Notice the similarity between equations (4.11) and (4.8). This manifests that the topological degree is just a generalization of the winding number concept.

We have thus demonstrated that the sign component of the Euclidean distance function can be separately computed (without knowledge of the distance) in parallel in $O(N \log N)$ on a regular 2D and 3D grid.

5. FAST COMPUTATION OF THE DERIVATIVES OF THE DISTANCE FUNCTION

Just as the approximate Euclidean distance function $S(X)$ can be efficiently computed in $O(N \log N)$, so can the derivatives. This is important because fast computation of the derivatives of $S(X)$ on a regular grid can be very useful in medial axes and curvature computations. Below, we detail how this can be achieved. We begin with the gradients and for illustration purposes, the derivations are performed in 2D:

$$(5.1) \quad S_x(X) = \frac{\sum_{k=1}^K \frac{(X^{(1)} - Y_k^{(1)})}{\|X - Y_k\|} \exp\left\{-\frac{\|X - Y_k\|}{\tau}\right\}}{\sum_{k=1}^K \exp\left\{-\frac{\|X - Y_k\|}{\tau}\right\}}.$$

A similar expression can be obtained for $S_y(X)$. These first derivatives can be rewritten as discrete convolutions:

$$(5.2) \quad S_x(X) = \frac{f_c(X)f(X) * g(X)}{f(X) * g(X)}, \quad S_y(X) = \frac{f_s(X)f(X) * g(X)}{f(X) * g(X)},$$

where $f_c(X)$ and $f_s(X)$ are as defined in Equation 4.5 and $f(X)$ and $g(X)$ are given in Equations 3.1 and 3.2 respectively.

The second derivative formulae are somewhat involved. Rather than hammer out the algebra in a turgid manner, we merely present the final expressions—all discrete convolutions—for the three second derivatives in 2D:

$$(5.3) \quad S_{xx}(X) = \frac{\left[-\frac{1}{\tau}f_c^2(X) + f_s^2(X)f_r(X)\right] f(X) * g(X)}{f(X) * g(X)} + \frac{1}{\tau}(S_x)^2(X),$$

$$(5.4) \quad S_{yy}(X) = \frac{\left[-\frac{1}{\tau}f_s^2(X) + f_c^2(X)f_r(X)\right] f(X) * g(X)}{f(X) * g(X)} + \frac{1}{\tau}(S_y)^2(X), \text{ and}$$

$$(5.5) \quad S_{xy}(X) = \frac{-\left[\frac{1}{\tau} + f_r(X)\right] f_c(X)f_s(X)f(X) * g(X)}{f(X) * g(X)} + \frac{1}{\tau}S_x(X)S_y(X)$$

where $f_r(X)$ is as defined in Equation 4.6.

Since we can efficiently compute the first and second derivatives of the approximate Euclidean distance function everywhere on a regular grid, we can also compute derived quantities such as curvature (Gaussian, mean and principal curvatures) for the two-dimensional surface $S(X)$ computed at the grid locations X . In the next section, we visualize the derivatives and maximum curvature for shape silhouettes.

6. EXPERIMENTS

In this section we show the usefulness of our linear approach to computing Euclidean distance functions by running it on a bounded $2D$ and $3D$ grid. As we discussed before, in order to improve the computational accuracy of our technique we are forced to go beyond the precision supported by the double floating-point numbers (64 bits) and make use of arbitrary precision packages like GNU multiple-precision library (GMP) [18] and MPFR [7]. For the following experiments we used $p = 512$ precision bits.

6.1. 2D Experiments. Example 1: We begin by demonstrating the effect of τ on our method and evince that as $\tau \rightarrow 0$, the accuracy of our method does improve significantly. To this end, we considered a $2D$ grid consisting of points between $(-0.121, -0.121)$ and $(0.121, 0.121)$ with a grid width of $\frac{1}{2^9}$. The total number of grid points is then $N = 125 \times 125 = 15,625$. We ran 1000 experiments each time randomly choosing 5000 grid locations as data points (point-set), for 9 different values of τ ranging from 5×10^{-5} to 4.5×10^{-4} in steps of 5×10^{-5} . For each run and each value of τ , we calculated the percentage error as

$$(6.1) \quad error = \frac{100}{N} \sum_{i=1}^N \frac{\Delta_i}{D_i},$$

where D_i and Δ_i are respectively the actual distance and the absolute difference of the computed distance to the actual distance at the i^{th} grid point. Figure 1 shows the *mean* percentage error at each value of τ . The *maximum* value of the error at each value of τ is summarized in table 2. The error is less than 0.6% at $\tau = 0.00005$ demonstrating the algorithm's ability to compute accurate Euclidean distance functions.

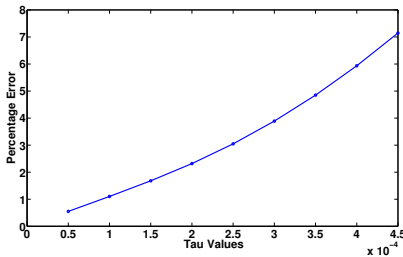


FIGURE 1. Percentage error versus τ in 1000 $2D$ experiments.

Example 2: We then executed our algorithm on a set of 2D shape silhouettes [16]². The grid size is $-0.125 \leq x \leq 0.125$ and $-0.125 \leq y \leq 0.125$ with a grid width of $\frac{1}{2^{10}}$. The number of grid locations equals $N = 257 \times 257 = 66,049$. We set τ for our method at 0.0003. For the sake of comparison, we ran the fast sweeping method for 10 iterations sufficient enough for it to converge. The percentage error calculated according to Equation 6.1 for both our method and fast sweeping in comparison

²We thank Kaleem Siddiqi for providing us the set of 2D shape silhouettes used in this paper.

τ	Maximum error
0.00005	0.5728%
0.0001	1.1482%
0.00015	1.7461%
0.0002	2.4046%
0.00025	3.1550%
0.0003	4.0146%
0.00035	4.9959%
0.0004	6.1033%
0.00045	7.3380%

TABLE 2. Maximum percentage error for different values of τ in 1000 2D experiments.

to the true Euclidean distance function for each of these shapes is adumbrated in Table 3.

Shape	Our linear method	Fast sweeping
Hand	2.182%	2.572%
Horse	2.597%	2.549%
Bird	2.116%	2.347%

TABLE 3. Percentage error for the Euclidean distance function computed using the grid points of these silhouettes as data points

The true Euclidean distance function contour plot and those obtained from our method and fast sweeping is delineated in Figure 2.

In order to differentiate between the grid locations that are either inside or outside each shape, we computed the winding number for all the grid points *simultaneously* in $O(N \log N)$ using our convolution-based winding number method. Grid points with a winding number value greater than zero after rounding were then marked as interior points. In the the left part of Figure 3 we visualize the vector fields (S_x, S_y) for all the interior points (marked in blue). We see that our convolution-based technique for computing the winding number separates the interior grid points from the exterior with almost zero error. In the right part of Figure 3 we plot the histogram of the winding number values computed over all the interior and the exterior locations. Observe that for almost all the grid points, the winding number values are close to *binary*, i.e either 0 or 1, providing anecdotal evidence for the efficacy of our algorithm.

We chose the maximum curvature (defined as $H + \sqrt{H^2 - K}$ where H and K are the mean and Gaussian curvatures respectively of the Monge patch given by $\{x, y, S(x, y)\}$) as the vehicle to visualize the medial axes of each shape. The mean and Gaussian curvatures can be expressed in terms of the coefficients of the first and second fundamental forms (E, F, G and e, f, g [8]) respectively which are in

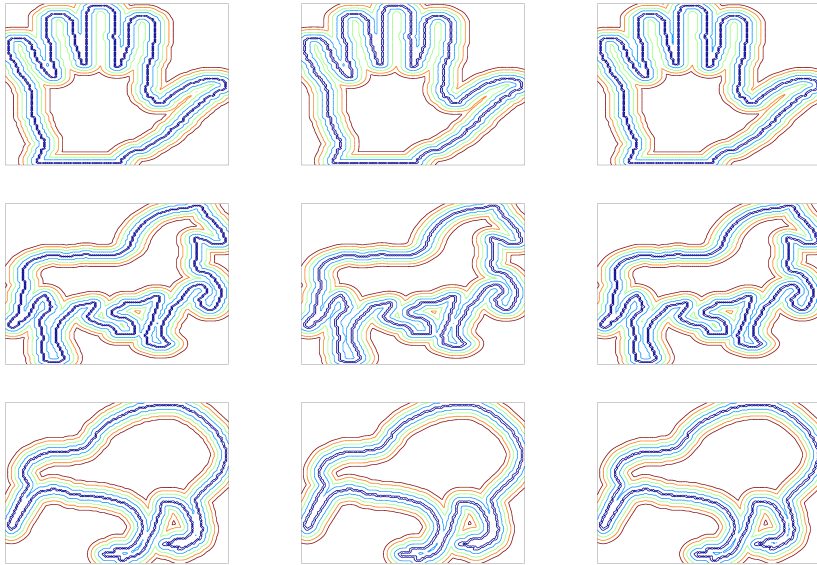


FIGURE 2. Contour plots: (i) Left: True Euclidean distance function, (ii) Center: Our method, (iii) Right: Fast sweeping

turn expressible in closed-form using the first and second derivatives of S . As these derivatives can be written as discrete convolutions (elucidated in Section 5), the max-curvature for the Monge patch can be computed in $O(N \log N)$ using FFT. From the max-curvature we can easily retrieve the medial axes as explained below.

Observe from the quiver plots in Figure 3, that the gradient directions are preserved until they meet with the gradients emanating from other curve locations. A zoomed version of the quiver plot is shown in Figure 4. In places where the gradients meet, their directions change significantly and hence the surface $S(x, y)$ exhibits high max-curvature values at those locations. But these locations exactly correspond to the grid points having more than one closest point on the shape's boundary—also known as the Voronoi boundary points or the medial axes points. Hence a simple thresholding of the max-curvature gives the medial axes, as determined by the points where the max-curvature is greater than (say) τ_1 . The medial axes plots for these shapes are shown in Figure 5 and can also be easily traced from the quiver plots (Figure 3) when viewed in color.

We would like to mention that computing the medial axes using the max-curvature is handicapped by a minor drawback. Using the FFT to compute the distance transform and its derivatives forces the data to sit on a regular grid. Notice from the medial axes plots in Figure 5 that the boundary of these shapes are not smooth but ragged. This results in high max-curvature values at various spurious locations, especially at the grid locations which are very close to the boundary points Y_k and hence will also be labeled as points on the medial axes. To circumvent this, we incorporated a second level of thresholding whereby we consider only those grid locations X where the distance transform $S(X)$ is greater than (say) τ_2 .

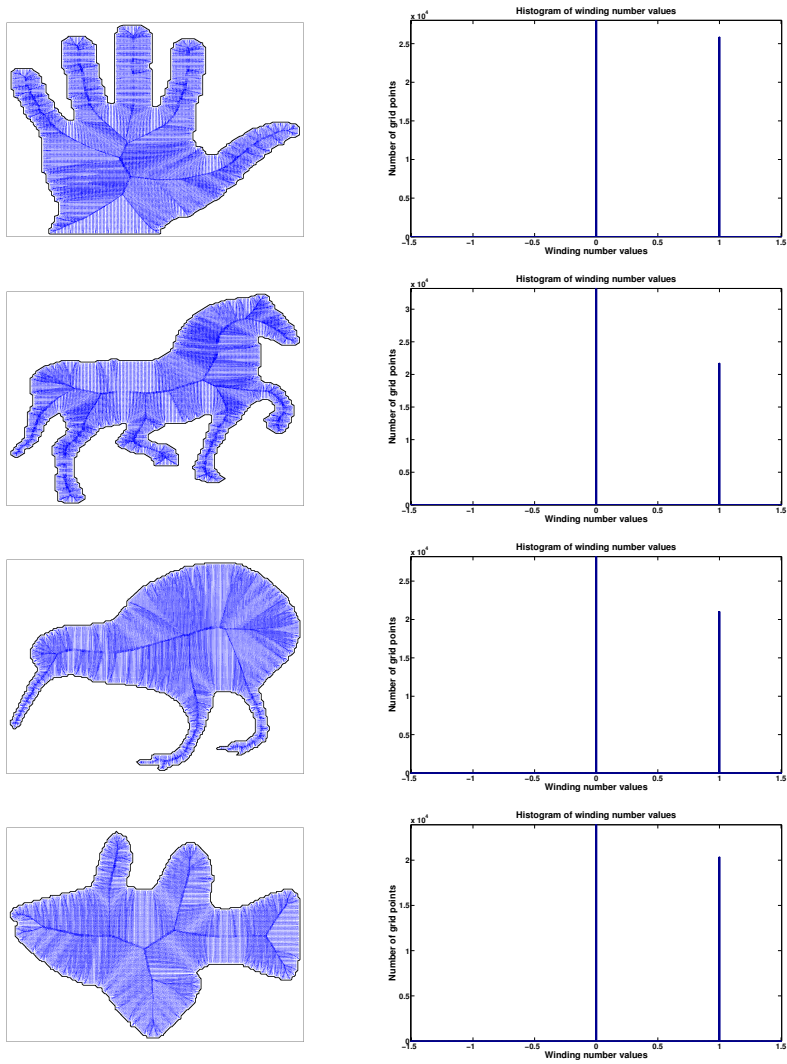


FIGURE 3. (i) Left: Quiver plot of $\nabla S = (S_x, S_y)$ for a set of silhouette shapes (best viewed in color), (ii) Right: Histogram of winding numbers

Depending on the shape, τ_1 was set between 0.09 and 0.12 and τ_2 between 3δ and 6δ where $\delta = 1/2^{10}$ is the grid width.

An easier fix to the aforementioned problem is to run our method on a much finer grid, increasing the number of grid locations thereby smoothing the boundary. But this has an adverse effect of slowing down the running time. A better solution would be to adapt the grid depending upon the data with varying grid width for different locations. Extending our technique to irregular grids is beyond the scope of our current work.

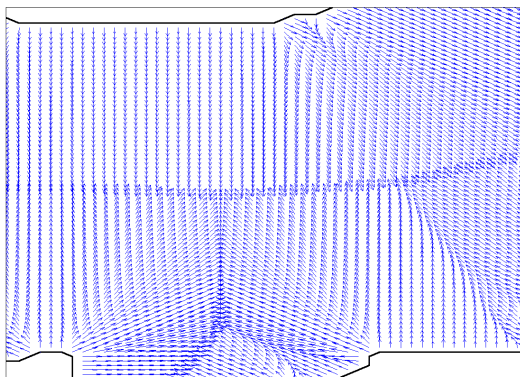


FIGURE 4. Zoomed quiver plot

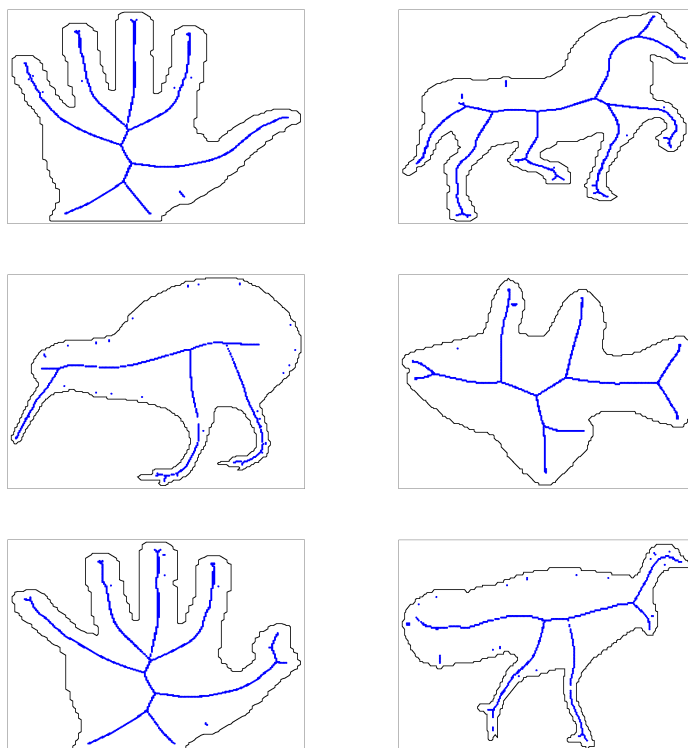


FIGURE 5. Medial axes plots

6.2. **3D Experiments. Example 3:** We also compared our Euclidean distance function algorithm with the fast sweeping method [20] and the exact Euclidean

distance on the “*Dragon*” point-set obtained from the Stanford 3D Scanning Repository³. The common grid was $-0.117 \leq x \leq 0.117$, $-0.086 \leq y \leq 0.086$ and $-0.047 \leq z \leq 0.047$ with a grid width of $\frac{1}{28}$. We ran our approach at $\tau = 0.0004$ and ran the fast sweeping method for 15 iterations which is sufficient for the Gauss-Seidel iterations to converge. We then calculated the percentage error as per Equation 6.1. While the average percentage error in our approach when compared to the true distance function was just **1.306%**, the average percentage error in the fast sweeping method was 6.84%. Our FFT-based approach does not begin by discretizing the spatial differential operator as is the case with the fast marching and fast sweeping methods and this could help account for the increased accuracy.

The isosurface obtained by connecting the grid points at a distance of 0.005 from the point set, determined by the true Euclidean distance function, our algorithm and fast sweeping are shown in Figure 6. The similarity between the plots provides anecdotal visual evidence for the usefulness of our approach.



FIGURE 6. Isosurfaces: (i) Left: Actual Euclidean distance function, (ii) Center: Our algorithm and (iii) Right: Fast sweeping

Example 4:Next, to demonstrate the efficacy of our convolution-based technique for computing the topological degree, we ran the following experiments in $3D$. The grid was confined to the region $-0.125 \leq x \leq 0.125$, $-0.125 \leq y \leq 0.125$ and $-0.125 \leq z \leq 0.125$ with a grid width of $\frac{1}{28}$. The number of grid points was $N = 274,625$. Given a set of points sampled from the surface of a $3D$ object, we triangulated the surface using some of the built-in MATLAB[®] routines. We consider the incenter of each triangle to represent the data points $\{Y_k\}_{k=1}^K$. The normal P_k for each triangle can be computed from the cross-product of the triangle vector edges. The direction of the normal vector was determined by taking the dot product between the position vector \vec{Y}_k and the normal vector \vec{P}_k . For negative dot products, \vec{P}_k was negated to obtain an outward pointing normal vector. We then computed the topological degree for all the N grid locations *simultaneously* in $O(N \log N)$ by running our convolution-based algorithm. Grid locations where the topological degree value exceeded 0.7 were marked as points lying inside the given $3D$ object. Figure 7 shows the interior points for the three $3D$ objects—cylinder, cube and sphere (left to right).

³This dataset is available at <http://graphics.stanford.edu/data/3Dscanrep/>.

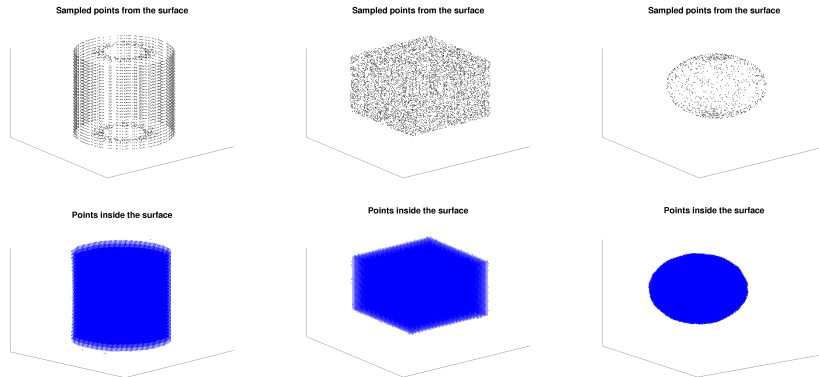


FIGURE 7. Topological Degree: (i) Top: Sampled points from the surface, (ii) Bottom: Grid points lying inside the surface (marked as blue)

7. CONCLUSION

In this work, we furnished a linear, variational formalism for the Euclidean distance function problem where we computed solutions on a bounded 2D and 3D grid. We posed a specific variational problem and evinced that the solution to its linear Euler-Lagrange equation at small values of τ can be used to obtain the Euclidean distance function. The intriguing aspect of our approach is that the non-linear Hamilton-Jacobi equation is embedded inside a *linear* equation and the solution is derived in the limiting case of $\tau \rightarrow 0$. We initially derived the solution for ϕ satisfying the Euler-Lagrange equation via the Green's function approach and later approximated it with a closed-form solution, the major advantage being that the closed-form solution is representable as a discrete convolution which can be efficiently computed in $O(N \log N)$ using FFT. The Euclidean distance is finally recovered from its exponent. Since the scalar field ϕ is computed for a small but non-zero τ , the obtained Euclidean distance function is an approximation. We derived analytic bounds for the error of the approximation for a given value of τ and provided proofs of convergence to the true distance function as $\tau \rightarrow 0$. The differentiability of our solution endows us to compute the gradients and curvature quantities of the distance function S in closed-form, also written as convolutions. We also provided a discrete convolution-based techniques for computing the winding number in 2D and the topological degree in 3D, which are useful in determining the sign of the distance function. Finally, we demonstrated how the gradient and curvature information can aid in the medical axes computation of 2D shape silhouettes.

While Hamilton-Jacobi solvers have gone beyond the eikonal equation and regular grids—by providing efficient solutions even for the more general static Hamilton-Jacobi equation on irregular grids [11, 9, 10]—in the current work we restrict ourselves only to computing the Euclidean distance function on regular grids. In the future, we would like to follow the pioneering works of the fast marching [13] and fast sweeping [20] methods and try to extend our linear formalism to irregular grids.

REFERENCES

1. O. Aberth, *Precise numerical methods using C++*, Academic Press, 1998.
2. M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Government Printing Office, USA, 1964.
3. R.N. Bracewell, *The Fourier Transform and its Applications*, 3rd ed., McGraw-Hill Science and Engineering, 1999.
4. R.P. Brent, *Fast multiple-precision evaluation of elementary functions*, Journal of the ACM **23** (1976), 242–251.
5. J.W. Cooley and J.W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation **19** (1965), no. 90, 297–301.
6. M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, Springer, 2008.
7. L. Fousse, G. Hanrot, V. Lefèvre, P. Pélicissier, and P. Zimmermann, *MPFR: A multiple-precision binary floating-point library with correct rounding*, ACM Trans. Math. Software **33** (2007), 1–15.
8. A. Gray, *Modern differential geometry of curves and surfaces with mathematica*, 2nd ed., CRC Press, 1997.
9. Y.-T. Zhang, J. Qian and H.K. Zhao, *Fast sweeping methods for eikonal equations on triangular meshes*, SIAM Journal on Numerical Analysis **45** (2007), no. 1, 83–107.
10. C.-Y. Kao, S.J. Osher, and J. Qian, *Legendre-transform-based fast sweeping methods for static Hamilton-Jacobi equations on triangulated meshes*, Journal of Computational Physics **227** (2008), 10209–10225.
11. C.-Y. Kao, S.J. Osher, and Y.-H. Tsai, *Fast sweeping methods for static Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis **42** (2004), no. 6, 2612–2632.
12. S.J. Osher and R.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, October 2002.
13. S.J. Osher and J.A. Sethian, *Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations*, Journal of Computational Physics **79** (1988), no. 1, 12–49.
14. T. Sasaki and Y. Kanada, *Practically fast multiple-precision evaluation of $\log(x)$* , Journal of Information Processing **5** (1982), 247–250.
15. A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing **7** (1971), 281–292.
16. K. Siddiqi, A. Tannenbaum, and S.W. Zucker, *A Hamiltonian approach to the eikonal equation*, Energy Minimization Methods in Computer Vision and Pattern Recognition (EMM-CVPR), vol. LNCS 1654, Springer-Verlag, 1999, pp. 1–13.
17. D.M. Smith, *Efficient multiple-precision evaluation of elementary functions*, Mathematics of Computation **52** (1989), no. 185, 131–134.
18. G. Torbjörn and et al., *GNU multiple precision arithmetic library 5.0.1*, June 2010.
19. L. Yatziv, A. Bartesaghi, and G. Sapiro, *$O(N)$ implementation of the fast marching algorithm*, J. Comput. Phys. **212** (2006), no. 2, 393–399.
20. H.K. Zhao, *A fast sweeping method for eikonal equations*, Mathematics of Computation (2005), 603–627.

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE AND ENGINEERING, GAINESVILLE, FL, USA

E-mail address: `sgk@ufl.edu`

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE AND ENGINEERING, GAINESVILLE, FL, USA

E-mail address: `anand@cise.ufl.edu`

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE AND ENGINEERING, GAINESVILLE, FL, USA

E-mail address: `msethi@cise.ufl.edu`