SECOND-ORDER MODELS FOR COMPUTING DISTANCE TRANSFORMS

Siddharth Manay *Anthony Yezzi*

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
email: smanay.ece98@gtalumni.org, anthony.yezzi@ece.gatech.edu

ABSTRACT

In this paper we present two different second-order accurate numerical algorithms for computing distance functions (solutions to the Eikonal Equation) around a curve. Both methods rely upon a local approximation of the level curves of the distance function as concentric circles (rather than straight lines implicitly assumed by most fast marching implementations, for example) and use the resulting geometry to yield a more accurate estimate of distance values at nearby gridpoints. The first scheme presented tries to estimate this geometry from an initial first order accurate estimate (yielded by previous fast marching schemes, for example) in order to find an error correction term to improve the initial estimate. The second scheme, on the other hand, directly estimates this second order model from the data in order to extend currently solved distance values to nearby unsolved grid points. Both schemes strictly adhere to an upwind criterion, allowing these schemes to handle shocks in the distance function in the same way they are handled by previous schemes. We show the improved accuracy of the resulting distance functions compared to more standard fast marching methods using ground truth examples.

1. INTRODUCTION

A distance function is a mapping that describes the distance from a point to a given set, usually a surface embedded in R^n. In many applications, the surface is known and does not change, so the distance function can be computed *a priori*. Applications that rely such mappings include registration, as in [1].

There are many algorithms to compute discrete approximations of a distance function, ranging from chamfer algorithms to PDE-based frameworks (as in [6]) and fast marching algorithms. These range in accuracy and differentiability. Even for accurate algorithms, such as the fast marching algorithm (see [5] and [3]), the accuracy of the function is limited by the accuracy of the differencing scheme. Methods that rely on one-sided discretized derivatives will result in distances that are locally accurate only to first order.

We present a local model of distance functions derived by approximating the level sets of the function as concentric circles. We are motivated to use circles as the basis because these level sets are more circular as distance increases.

In the first algorithm presented in this paper, we estimate this osculating circle model from the numerically calculated curvature of an initial first order estimate of the distance function. From this, an estimate of the error in the initial first-order approximation is calculated and then used to correct that approximation.

In the second algorithm, we directly estimate the osculating circle model from the data and use this model to extend already calculated distances to nearby grid points where distances have not yet been calculated. In addition to the increased accuracy of the distance calculation, this second model-based approach gives the advantage of allowing us to obtain the first and second derivatives of our computed distance function directly from the model, rather than through finite differencing methods. This yields improved accuracy and more stable behavior in the derivatives of the distance function, which is advantageous in applications such a registration where quasi-Newton Rhapson schemes require calculation of both the gradient and Jacobian of numerically computed distance functions. Finite differencing schemes, particularly for second derivatives in the vicinity of shocks, tend to yield much noisier results.

2. ERROR ANALYSIS FOR STANDARD FIRST-ORDER FAST MARCHING SCHEMES

Most current PDE-based schemes to numerically calculate distance functions are based upon a first order-accurate discretization of the Eikonal Equation

$$\|\nabla\Phi\| = 1 \tag{1}$$

where Φ is the distance function. Since information propagates away from the points of zero-distance, upwind differencing schemes must be utilized in approximating the spatial derivatives of Φ. Efficient fast-marching schemes are

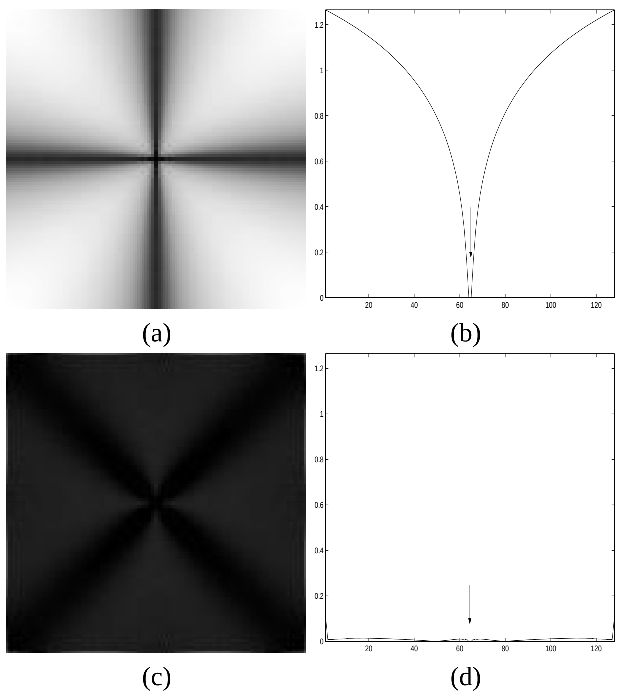

Fig. 1. (a) Magnitude of error of standard fast marching model. White pixels have maximal error (1.2 units) (b) Error on a cross-section of a 2D distance function. The location of the zero-level set is marked with an arrow. A preview of the curvature-corrected distance computation: (c) Magnitude of error of curvature-corrected model. (d) Error on a cross-section of the distance function.

also based upon this causality property by using heap structures to visit pixels in order of increasing distance, allowing us to solve for values of pixels with smaller distances early on and then use these solved values to solve for neighboring pixels with larger distances. We will follow this same procedure in the second-order accurate algorithms presented in this paper.

For an unsolved pixel p with solved neighbors p_x and p_y that meet the upwind criteria, the normal first-order discretized version of Equation 1 is

$$(\Phi[p]-\Phi[p_x])^2+(\Phi[p]-\Phi[p_y])^2=1.$$

(assuming the grid spacing $\Delta x=\Delta y=1$). This yields a quadratic equation for the distance value $\Phi[p]$, whose solution after simplification is given by

$$\Phi[p]=\frac{(\Phi[p_x]+\Phi[p_y])\pm\sqrt{2-(\Phi[p_x]-\Phi[p_y])^2}}{2}. \quad (2)$$

This equation is the basis of the fast marching algorithm. Note that the first-order approximation of the derivatives implies a local planar approximation of the true distance

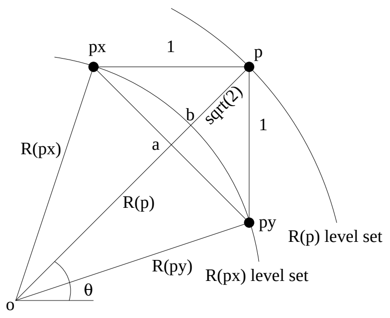

Fig. 2. Geometry of error analysis.

function with straight line level sets[1]. In regions where the true distance function has nonzero second derivatives (regions where the level sets are curved and more circular, as is typical in distance functions to most objects other than straight lines) the standard first-order accurate solution of the Eikonal equation typically used in fast marching techniques has a bias proportional to the curvature of the level sets. Figure 1 shows the deviation of the first-order value from the true value in two dimensions and on a cross-section (taken on a diagonal line through the center of the function) of a two-dimensional distance function around a point.

To bound the error from this first order prediction we exploit the fact that, in the specific case above (which we'll soon relax), a level set through a point is a circle with radius equal to the distance at that point. Consider a two dimensional distance function around a point o, and a point p such that the vector $\overline{p}$ makes an angle of $\pi/4$ with the x-axis. The adjacent downwind gridpoints are p_x and p_y in the x and y directions, respectively. A diagram of this example is shown in Figure 2.

Because $\Phi[p_x]=\Phi[p_y]$, substituting these values into Equation 2 gives the standard fast marching approximation $\Phi'[p]=\Phi[p_x]+\sqrt{2}/2$. We can interpret both terms of the approximation geometrically as the lengths of the segments $\overline{ob}$ and $\overline{ap}$ in Figure 2. However, the actual distance is the sum of the lengths $\overline{ob}$ and $\overline{bp}$, so the error in the approximation is the length $\overline{ab}$. Calculating this length yields the error

$$\begin{aligned} E=\Phi'[p]-\Phi[p]=|\overline{ab}| &= |\overline{ob}|-|\overline{oa}| \\ &= \Phi[p_x]-\sqrt{\Phi[p_x]^2-\frac{1}{2}}. \end{aligned}$$

Note that the error is a function of the radius of curvature of the level set through the neighboring pixel and therefore also the radius of curvature of the level set through the pixel in question (since the radii can be expressed as functions

[1]Of course, one could use higher-order difference approximations of the derivatives, but this would require looking two or more pixels away from the point to be solved. In our second order osculating circle based models, we will constrain ourselves to looking still only one pixel away, but will include the diagonal neighbor as well as the adjacent x and y neighbors

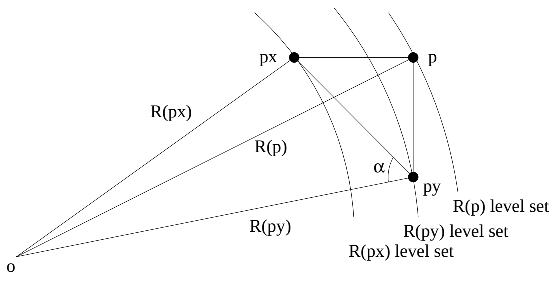

Fig. 3. Geometry of circular approximation for an exterior pixel and diverging characteristics.

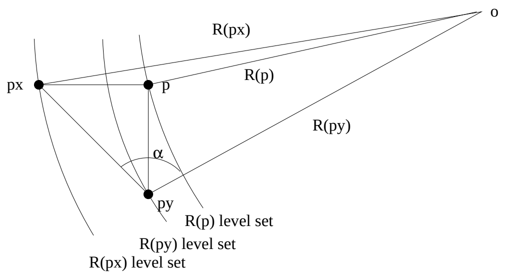

Fig. 4. Geometry of circular approximation for an exterior pixel and converging characteristics.

of one another). We may now relax the assumption of a distance function to a single point by estimating the radii of the osculating circles to the level curves from a calculation of curvature rather than the actual distance value itself. The error in the standard first order fast marching approximation varies continuously with θ from $E = 0$ at $\theta = 0$ (where the upwinding scheme requires the use of a degenerate case of Equation 2, $\Phi[p] = \Phi[p_x] + \Delta x$) to its maximum at $\theta = \pi/4$ and then back to $E = 0$ at $\theta = \pi/2$.

3. AN ERROR CORRECTION SCHEME FOR THE STANDARD FAST MARCHING METHOD

In this section, we augment the standard first-order fast marching technique by adjusting its normally yielded solution by a term to correct its overshoot/undershoot estimated according to the analysis in the previous section by a computation of curvature using adjacent grid points.

We recall the example in Section 2 of a distance function to a single point. In that example, all the level sets of the function were concentric circles, resulting in a simple geometry where the distance value at a point was related to curvature via radius. We use a similar model as an approximation to distance functions in general; that is, we assume that locally the level sets of a distance function may be approximated as concentric circles with radius equal to the reciprocal of their curvature (osculating circles)[2]. In this case the distance value is still related to curvature via radius, even though radii and distance are not equivalent. For this reason we are first concerned with the computation of the radius of the level set at p based on the data at the neighboring gridpoints p_x and p_y.

The geometry of the model is shown in Figure 3. Here we assume that the desired point is outside the level set (and therefore positive) and the that characteristics of the distance function are diverging. The behavior of the characteristics, whether they are converging or diverging, is determined by the sign of the curvature; positive curvature indicates diverging characteristics in the outward-marching case and converging characteristics in the inward-marching case. The known quantities are the radii of the level sets through p_x and p_y ($R[p_x]$ and $R[p_x]$ respectively, calculated from the curvature of the level sets) and the dimensions of the line segments connecting the three given points (again, we assume that the distance between adjacent gridpoints is 1). We note that the point o is unknown, and in general does not correspond to any significant point on the underlying distance function.

From the figure, we derive the unknown radius $R[p]$ using the Law of Cosines.

$$\alpha = \cos^{-1}\left(-\frac{R^2[p_x] - R^2[p_y] - 2}{2R[p_y]\sqrt{2}}\right)$$

$$R[p] = \sqrt{R^2[p_y] + 1 - 2R[p_y]\cos(\alpha + \frac{\pi}{4})}. \qquad (3)$$

We note that this computation is valid only if $|R[p_x] - R[p_y]| < \sqrt{2}$.

The configuration in Figure 4 reflects the case that the calculation is propagating outwards but the characteristics are converging. In this situation, the computation of α is the same, but the computation of $R[p]$ is

$$R[p] \quad = \quad \sqrt{R^2[p_y] + 1 - 2R[p_y]\cos(\alpha - \frac{\pi}{4})}. \qquad (4)$$

Two additional cases occur when the computation is propagating inwards. While the distance is increasing, the signed value of the distance is decreasing. The geometry in the cases with converging and diverging characteristics correspond to the outward-propagating cases with diverging and converging characteristics, respectively. That is, if the characteristics are converging, Equation 3 applies, and if the characteristics are diverging, we use Equation 4.

Lastly, we relate the radius of the level set to the distance. As in the case of the distance function around a circle, when the curvature of the level sets is positive, $\Phi[p] = \mathrm{sgn}(\kappa)(R - R_l)$, where κ is the curvature. In two dimensions, to compute the curvature of the level sets embedded

[2]Note that the osculating circle approximation of the level sets of the distance function contains as a special case the straight line level curves implicit in the standard first-order fast marching technique. In this case, the osculating circles have infinite radii.

in a function we use the expression

$$\kappa = \frac{\Phi_x^2 \Phi_{yy} - 2\Phi_x \Phi_y \Phi_{xy} + \Phi_y^2 \Phi_{xx}}{(\Phi_x^2 + \Phi_y^2)^{\frac{3}{2}}},$$

where Φ_x and Φ_{xx} are first and second derivatives in the x direction, etc. However, due to the marching nature of the algorithm, only points downwind of the current point will be available. This means that the first and cross derivatives must be approximated by one-sided differences, where the appropriate side is chosen based on which neighbors have been computed and whether they are downwind. We avoid the necessity of using two downwind points to calculate the second derivatives and instead compute these quantities analytically from the Eikonal equation. Taking ∇ of both sides of this expression yields $\nabla^2\Phi\nabla\Phi = \bar{0}$, where $\nabla^2\Phi$ denotes the Hessian of Φ. Re-arranging gives expressions for the second derivatives

$$\Phi_{xx} = -\frac{\Phi_{xy}\Phi_y}{\Phi_x},\ \Phi_{yy} = -\frac{\Phi_{xy}\Phi_x}{\Phi_y}. \qquad (5)$$

Radius is calculated by $R = |\frac{1}{\kappa}|$. Given the distance and curvature values at the p's neighbors p_x and p_y, we can compute

$$R_l = \frac{R[p_x] - \mathrm{sgn}(\kappa[p_x])\Phi[p_x] + R[p_y] - \mathrm{sgn}(\kappa[p_y])\Phi[p_y]}{2}.$$

These are all the quantities needed for the curvature-corrected distance calculation.

The curvature-corrected marching algorithm is very similar in structure to the standard fast marching algorithm. In summary, the algorithm for the computation of the values outside the curve proceeds as follows:

1. Initialize the distance values around the curve and place these solved pixels in a minimum heap.
2. Pop the pixel with the smallest value off the heap. Mark this pixel, called pixel q, as “done”.
3. For each of q's neighbors, called pixel p:
   (a) Determine whether p has two downwind neighbors with distance values, p_x and p_y; these must have nonzero, finite curvature and satisfy the condition $|R[p_x] - R[p_y]| < \sqrt{2}$.
   (b) If the neighbors meet these conditions, compute κ and R for p's neighbors, and use these values to compute R_l, $R[p]$, and $\Phi[p]$ as discussed above and in Section 3.
   (c) If not, compute $\Phi[p]$ via the standard fast marching model.
   (d) Place p in the minimum heap.

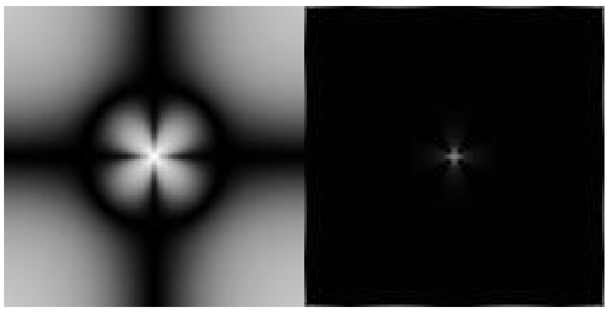
Fig. 5. Error due to standard fast marching [LEFT] and curvature-corrected [RIGHT] marching algorithms.

4. Repeat steps 2 and 3 until the heap is empty.

The algorithm for the interior is similar, but because the distance values are negative the marching order now begins with pixels with the largest distance (smallest magnitude) and proceeds in order of decreasing distance. This requires the use of a maximum heap, rather than a minimum heap.

The curvature-corrected algorithm requires good estimates of curvature. However, the distance (and therefore curvature) values at the points adjacent to the zero level set are very coarsely estimated. More accurate values may be found by first using the standard fast marching algorithm to solve for a narrow band around the zero level set, and then correcting these values with a first order partial differential equation proposed in [6],

$$\frac{\partial\Phi}{\partial t} = \mathrm{sgn}(\Phi)(1 - \|\nabla\Phi\|). \qquad (6)$$

This first-order estimate of the distance function around the zero level set can be used to initialize the curvature-corrected algorithm.

4. RESULTS FOR THE CURVATURE-CORRECTION METHOD

In this section we present some preliminary results demonstrating the benefits of using the curvature-corrected marching technique.

In the first example we wish to compute the distance function around a circle. (We choose this simple function because we can easily derive an analytic solution, giving us a ground truth for comparison). For the reasons discussed above, for these preliminary tests we initialize the pixels adjacent to the curve with the analytical values. We then run the curvature-corrected marching algorithm. For comparison, we also generate a distance function using the standard fast marching algorithm *initialized the same way*.

In Figure 5 we show a comparison of the absolute error of the standard fast marching and curvature-corrected

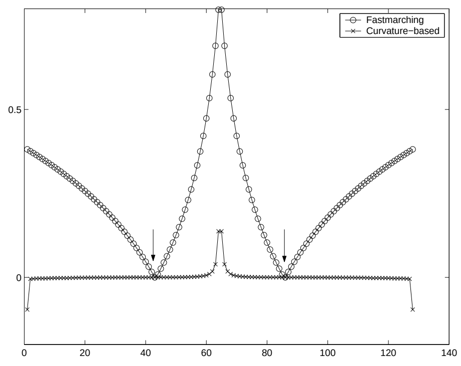

Fig. 6. Error on a cross-section of the distance function.

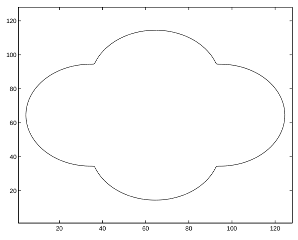

Fig. 7. Initial contour for distance function.

models; in Figure 6 we include a one-dimensional cross-section (taken on a diagonal line through the center of the function) of the error for both methods. The cascading error is clearly visible in the standard fast marching results, increasing further away from the zero level set (where the values were initialized correctly, marked by arrows on the figure). The curvature-corrected algorithm does not display this behavior. The overall squared error for the standard fast marching algorithm is 488, while the squared error for the curvature-corrected algorithm is 2.16, a *two order of magnitude decrease*. The curvature-corrected algorithm is a more expensive algorithm, though, requiring 0.0648 seconds for the above example on a Pentium IV 900 Mhz, compared to 0.0408 seconds for the standard fast marching algorithm.

In Figures 8 and 9 we test the algorithms on a distance function generated around the curve in Figure 7. Again we note that the squared error of the curvature-corrected method is 1.24×10^5, compared to 3.41×10^5 for the standard fast marching method.

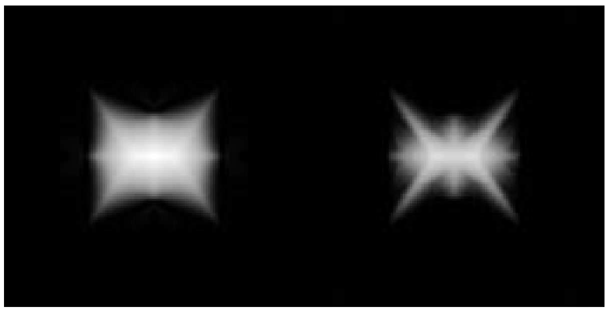

Fig. 8. Error due to standard fast marching [LEFT] and curvature-corrected [RIGHT] algorithms.

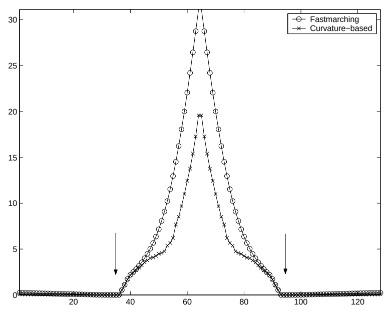

Fig. 9. Error on a cross-section of the distance function.

In Section 3 we noted that the curvature-corrected distance computation was only valid when $|R[p_x] - R[p_y]| < \sqrt{2}$. This condition is violated when the curvature to the level sets of the distance function is ill-defined, i.e. near a shock in the distance function. The other conditions under which the curvature-corrected computation is ill-posed are zero or infinite curvature (although infinite curvature means straight line level sets and therefore implies that the unadjusted first-order accurate estimate needs no correction). The authors are exploring several options based on the osculating circle approximation so that this curvature-corrected method will handle these cases robustly.

5. A DIRECT SECOND-ORDER MODEL

We now put forth another more direct approach utilizing the same osculating circle approximation of the underlying level curves to the distance function. In this case, however, we avoid a numerical curvature computation and directly fit the values of three neighboring grid points (one diagonal, one horizontal, and one vertical) to our osculating circle model. Then, using this model, we can directly calculate the values for neighboring unsolved pixels. Furthermore, one may analytically compute the gradient and Hessian of the distance function constructed in this way using the model parameters, thereby avoiding completely any discrete derivative approximations. This property is particularly attractive for variational applications which utilize derivatives of distance functions.

Given a set of adjacent points on a grid, $p_0, p_1, p_2 \subset \Re^2$ and already computed distance values $\Phi(p_0) = \Phi_0, \Phi(p_2) = \Phi_2, \Phi(p_2) = \Phi_2$, at these points, we wish to construct a local model of the distance function Φ. The model is based on the approximation of Φ as a distance function around a circle with radius R centered at $c \subset \Re^2$ (which is the distance function consistent with the local approximation of the level curves by concentric osculating circles). This model will then be used to calculate the unknown distance at another nearby gridpoint point p.

5.1. The Case of Diverging Characteristics

Given the radius and center of a circle in the plane, the signed distance from the circle to any point p_i is

$$\|c - p_i\| - R = \Phi(p_i) = \Phi_i. \qquad (7)$$

based on the convention that the points inside the circle have negative distance. This model has diverging characteristics; a model for regions of Φ with converging characteristics will be discussed later.

To solve for R and c, we construct the system of three equations based on the known gridpoints and their values.

$$\begin{aligned}\|c - p_0\| - R = \Phi(p_0) = \Phi_0\\ \|c - p_1\| - R = \Phi(p_1) = \Phi_1\\ \|c - p_2\| - R = \Phi(p_2) = \Phi_2\end{aligned}$$

To solve this system, first square both sides,

$$c^Tc - 2c^Tp_0 + p_0^Tp_0 = R^2 + 2R\Phi_0 + \Phi_0^2 \qquad (8)$$
$$c^Tc - 2c^Tp_1 + p_1^Tp_1 = R^2 + 2R\Phi_1 + \Phi_1^2 \qquad (9)$$
$$c^Tc - 2c^Tp_2 + p_2^Tp_2 = R^2 + 2R\Phi_2 + \Phi_2^2 \qquad (10)$$

and then subtract Equation 9 from Equation 8 and Equation 10 from Equation 9 to form the linear set of two equations

$$\begin{aligned}-2c^T(p_0 - p_1) + p_0^Tp_0 - p_1^Tp_1 = 2R(\Phi_0 - \Phi_1) + \Phi_0^2 - \Phi_1^2\\ -2c^T(p_2 - p_1) + p_1^Tp_1 - p_2^Tp_2 = 2R(\Phi_1 - \Phi_2) + \Phi_1^2 - \Phi_2^2\end{aligned}$$

Rewriting,

$$Ac = Rb_0 + \frac{1}{2}b_1 \qquad (11)$$

where

$$\begin{aligned}A &= \begin{bmatrix}(p_0 - p_1)^T\\(p_1 - p_2)^T\end{bmatrix} \text{ a 2x2 array}\\ b_0 &= \begin{bmatrix}\Phi_1 - \Phi_0\\ \Phi_2 - \Phi_1\end{bmatrix} \text{ a vector, and}\\ b_1 &= \frac{1}{2}\begin{bmatrix}p_0^Tp_0 - p_1^Tp_1 - \Phi_0^2 + \Phi_1^2\\ p_1^Tp_1 - p_2^Tp_2 - \Phi_1^2 + \Phi_2^2\end{bmatrix} \text{ a vector.}\end{aligned}$$

Note that the columns of A are the vectors from p_0 to p_1 and from p_1 to p_2. If these points are colinear, A is singular. However, if these vectors are orthogonal and the grid spacing is $\Delta x = \Delta y = 1$, A is orthogonal (so that $A^{-1} = A^T$). The rest of this derivation relies on this assumption, but can be generalized.

By solving Equation 11, c can be written in terms of R

$$c = RA^Tb_0 + \frac{1}{2}A^Tb_1 \qquad (12)$$

This expression can be used to simplify the quadratic expression in Equation 9. Rewriting the some of the terms that appear in that equation

$$\begin{aligned}c^Tc &= R^2b_0^Tb_0 + Rb_1^Tb_0 + \frac{1}{4}b_1^Tb_1\\ c^Tp_1 &= Rb_0^TAp_1 + \frac{1}{2}b_1^TAp_1\end{aligned}$$

After some simplification, the quadratic can now be written as

$$a_2R^2 + a_1R + a_0 = 0 \qquad (13)$$

where

$$\begin{aligned}a_2 &= b_0^Tb_0 - 1\\ a_1 &= b_1^Tb_0 - 2b_0^TAp_2 - 2\Phi_1\\ a_0 &= \frac{1}{4}b_1^Tb_1 - b_1^TAp_1 + p_2^Tp_1 - \Phi_1^2,\end{aligned}$$

which further simplify to

$$\begin{aligned}a_2 &= (\Phi_1 - \Phi_0)^2(\Phi_1 - \Phi_2)^2 - 1\\ a_1 &= (\Phi_1 - \Phi_0)^2(\Phi_1 + \Phi_0) + (\Phi_1 - \Phi_2)^2(\Phi_1 + \Phi_2)\\ &- (\Phi_0 + \Phi_2);\\ a_0 &= \frac{2(1 - \Phi_0^2 - \Phi_2^2) + (\Phi_1^2 - \Phi_0^2)^2 + (\Phi_1^2 - \Phi_2^2)^2}{4}\end{aligned}$$

The solution to this equation is simply

$$R = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2} \qquad (14)$$

This gives two solutions for R and c (via Equation 12). We call the solutions (R_1, c_1) and (R_2, c_2). However, the solution is meaningful only if $R > 0$. Now, the unknown distance is

$$\Phi(p) = \|c - p\| - R.$$

Further, expressions for the first- and second-order derivatives of the distance function at this point can be derived analytically from the model, as follows.

$$\begin{aligned}\nabla\Phi &= \frac{p - c}{\sqrt{(c - p)^T(c - p)}}\\ \nabla^2\Phi &= -\frac{(p - c)(p - c)^T}{\sqrt{(c - p)^T(c - p)}^3} + I\frac{1}{\sqrt{(c - p)^T(c - p)}},\end{aligned}$$

where I is the identity matrix.

5.2. The Case of Converging Characteristics

Now we slightly alter the previous derivation to model regions of Φ with *converging* characteristics. We reverse the convention of the signed distance function around the circle; that is, points inside the circle have positive distance and points outside have negative distance. For this model, the signed distance to any point p_i is

$$R - \|c - p_i\| = \Phi(p_i) = \Phi_i. \qquad (15)$$

This model is a similar to the diverging model (Equation 7) with the exception of a sign change. Because of this, the derivation of the solution is very similar, resulting in

$$R = \frac{a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2} \qquad (16)$$

$$c = -RA^Tb_0 + \frac{1}{2}A^Tb_1 \qquad (17)$$

Again, the quadratic results in two solutions that we call (R_3, c_3) and (R_4, c_4) (via Equation 17).

However, comparing Equation 16 to Equation 14, we see that $R_4 = -R_1$ and $R_3 = -R_2$. Substituting these relations into Equation 17, $c_4 = c_1$ and $c_3 = c_2$. Since we discard solutions with negative R, the two "viable" solutions are

$$(|R_1|, c_1)$$
$$(|R_2|, c_2) \quad .$$

This results in a simpler calculation; instead of calculating four solutions from two models, we calculate two solutions using one model and take the absolute value of R. The two solutions for $\Phi(p)$ are then computed

$$\Phi_j(p) = \mathrm{sgn}(R_j)\|c_j - p\| - R_j,$$

where $\Phi_j(p)$ is the j^{th} solution for the unknown value $\Phi(p)$, $j = 1, 2$.

5.3. Choosing from the Two Solutions

What remains is to choose the correct solution from the two available solutions. We choose the solution that satisfies the upwind criteria. If an additional criterion is needed, we choose the solution that best satisfies the Eikonal condition $\|\nabla\Phi\| = 1$.

6. IMPLEMENTATION

In this section we discuss some implementation details before outlining an algorithm that embeds the computation in Section 5 in a heap-based algorithm to compute a distance function.

In the previous section, the method for constructing the distance function model proceeded on the assumption that the given grid points p_0, p_1, p_2 are not colinear. For ease and speed of computation, we desire $p_0 - p_1$ orthogonal to $p_1 - p_2$ so that A will be orthogonal and easy to invert. To satisfy this condition, we assign the x- and y-neighbors of the current pixel p as p_0 and p_2, and the "diagonal" neighbor adjacent to all these points as p_1. This scenario is shown in Figure 10 (a). However, in a heap-based marching algorithm, it is not always possible to access both the x- and y-neighbors of a given gridpoint. When one of these neighbors is not available, we choose the p_i's as shown in Figure 10 (b) and (c).

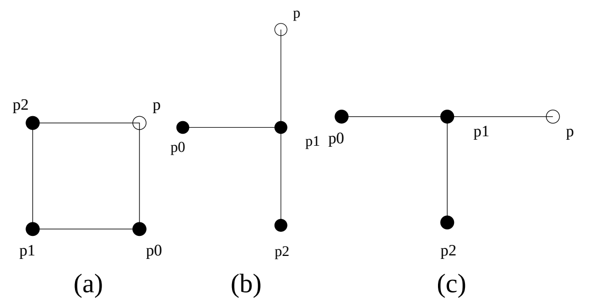

Fig. 10. Geometries of neighboring pixels. (a) x- and y-neighbors are available. (b) Only y-neighbor is available. (c) Only x-neighbor is available.

The calculation of the discriminant criteria in Section 5.3 is also dependent on the availability of neighboring pixels. For the j^{th} solution, we calculate the deviation from the Eikonal condition as

$$e_j = |1 - \sqrt{(\Phi_0 - \Phi_j(p))^2 + (\Phi_2 - \Phi_j(p))^2}|$$

when both neighbors are available, as in Figure 10 (a). For pixels that only have one computed neighbor, we calculate

$$e_j = |1 - \sqrt{(\Phi_1 - \Phi_0)^2 + (\Phi_1 - \Phi_j(p))^2}|$$

for the scenario in Figure 10 (b) and

$$e_j = |1 - \sqrt{(\Phi_1 - \Phi_2)^2 + (\Phi_1 - \Phi_j(p))^2}|$$

for the scenario in Figure 10 (c).

The direct second-order model-based marching algorithm is very similar in structure to the standard fast marching algorithm. In summary, the algorithm for the computation of the values outside the curve proceeds as follows:

1. Initialize the distance values around the curve and place these solved pixels in a minimum heap.
2. Pop the pixel with the smallest value off the heap. Mark this pixel, called pixel q, as "done".
3. For each of q's neighbors, called pixel p:
   (a) Determine whether p has two downwind neighbors with distance values, p_0 and p_2. If so, assign the p_i's as shown in Figure 10 (a); if not, assign the p_i's as shown in Figure 10 (b) or (c).

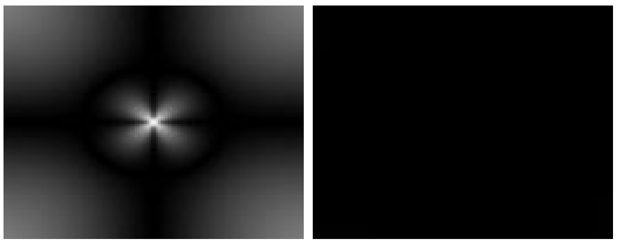

Fig. 11. Error due to two distance transform algorithms. [LEFT] Fastmarching algorithm: total squared error is 488.2. [RIGHT] Second-Order Model-based algorithm: total squared error is 0.098.

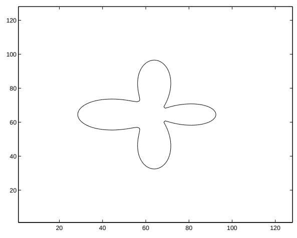

Fig. 12. Initial contour for distance function in.

(b) Compute R, c, and two solutions for $\Phi[p]$ as discussed above and in Section 5. Choose the best solution from the criteria given in that section. Compute $\nabla\Phi$ and $\nabla^2\Phi$ from the same model.

(c) Place p in the minimum heap.

4. Repeat steps 2 and 3 until the heap is empty.

The algorithm for the interior is similar, but because the distance values are negative the marching order now begins with pixels with the largest distance (smallest magnitude) and proceeds in order of decreasing distance. This requires the use of a maximum heap, rather than a minimum heap.

7. RESULTS FOR THE DIRECT SECOND-ORDER MODEL-BASED METHOD

In Figure 11 we demonstrate the distance transform based on the second-order model on a circle. We note that the model does not make use of any *a priori* information about the circularity of the contour. The left image shows the error of the standard fast-marching algorithm compared to the actual ground-truth distance value. For comparison, the right image shows the error for a distance function calculated using our direct second-order model-based marching method. Both methods were initiallized as in Section 4. Again, the higher-order model yields a much better approximation of the distance values.

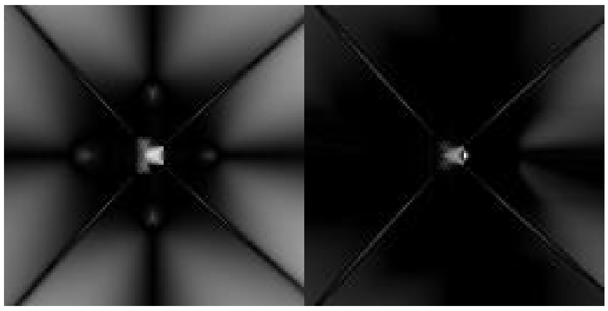

Fig. 13. Error due to two distance transform algorithms. [LEFT] Fastmarching algorithm: total squared error is 2.076×10^3. [RIGHT] Second-Order Model-based algorithm: total squared error is 229.0.

A more complex contour is shown in Figure 12; this contour's parametric representation allows us to compute a ground truth via brute-force for the resulting distance function. We compare the squared error of the fast-marching algorithm and the second-order model-based method in Figure 13. The higher-order method reduces the error by an order of magnitude, at the expense of an increased execution time of 0.27 seconds verses 0.05 seconds for the fast-marching algorithm. We note that the current implementation of this model is not numerically stable everywhere. Currently, we handle these instabilities by using the fast-marching algorithm in these regions; however, a stable implementation of the model-based method is the subject of continuing investigation.

8. CONCLUSION

In this paper we have presented two second-order accurate numerical algorithms for computing distance functions. These methods rely on a local approximation of the level sets of the distance function as concentric circles. The first method estimates this geometry from discrete curvature calculations on an initial distance function to find an error correction term. The second scheme directly estimates this second-order model from the data to extend solved distance values to nearby unsolved grid points. Both schemes are implemented in a heap-based algorithm to demonstrate the improved accuracy of the resulting distance functions.

9. REFERENCES

[1] O. Cuisenaire, “Distance Transformations: Fast Algorithms and Applications to Medical Image Processing”, Ph.D. Dissertation, Laboratoire de Telecommunications et Teledetection, 1999.

[2] C. Y. Kao, S. Osher, and Y. Tsai, “Fast Sweeping Methods for Hamilton-Jacobi Equations”, *UCLA CAM Report*, December 2002.

[3] J. Sethian, “A Fast Marching Level Set Method for Monotonically Advancing Fronts,” *Proc. Natl. Acad. Sci. USA*, Vol. 93, pp. 1591-1595, Feb. 1996.

[4] J. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Cambridge, UK, 1999.

[5] J. Tsitsiklis, “Efficient Algorithm for Globally Optimal Trajectories”, *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528-1538, 1995.

[6] M. Sussman, P. Smereka, and S.J. Osher, “A Level Set Method for Computing Solutions to Incompressible Two-Phase Flow”, it J. Comp. Phys., vol. 114, pp. 146-159, 1994.