Euclidean Distance Approximations from Replacement Product Graphs

T. Arthur Terlep, *Member, IEEE,* Mark R. Bell, *Fellow, IEEE,*
Thomas M. Talavage, *Senior Member, IEEE,* and Douglas L. Smith

***Abstract*—We introduce a new chamfering paradigm, locally connecting pixels to produce path distances that approximate Euclidean space by building a small network (a replacement product) inside each pixel. These "RE-grid graphs" maintain near-Euclidean polygonal distance contours even in noisy data sets, making them useful tools for approximation when exact numerical solutions are unobtainable or impractical. The RE-grid graph creates a modular global architecture with lower pixel-to-pixel valency and simplified topology at the cost of increased computational complexity due to its internal structure. We present an introduction to chamfering replacement products with a number of case study examples to demonstrate the potential of these graphs for path-finding in high frequency and low resolution image spaces which motivate further study. Possible future applications include morphology, watershed segmentation, halftoning, neural network design, anisotropic image processing, image skeletonization, dendritic shaping, and cellular automata.**

***Index Terms*—Chamfer, euclidean distance, distance approximation, replacement product, graph theory, path-finding, eikonal equation, Dijkstra's algorithm**

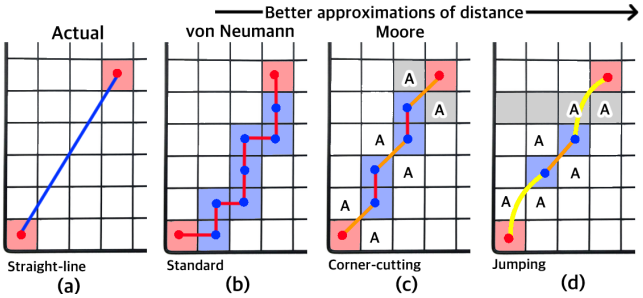


Fig. 1. Measuring the distance traveled between two points (red boxes) in a square grid requires us to construct a path of pixels that links the two points. (a) The actual Euclidean straight-line distance is $\sqrt{5^2+3^2} = \sqrt{34} \approx 5.831$. (b) Choosing a path (blue pixels) that share common edges gives the standard (von Neumann) distance of 8. (c) Allowing diagonal moves between pixels that share corners (Moore) gives a better approximation of $3\sqrt{2}+2 \approx 6.243$. (d) An even closer approximation of $2\sqrt{5}+2 \approx 5.886$ can be achieved if we allow a path that "jumps" pixels in the manner that a knight "jumps" squares on a chessboard. However, the executability of diagonal moves or jumps may depend on the nature of the pixels marked A, whose contents are not queried by these paths.

I. Introduction

NEIGHBORHOOD systems for locally connecting pixels together are limited in their ability to approximate Euclidean distances (Fig. 1 (a)), because of the path constraints imposed by the grid of pixels itself. The two usual ways to connect a pixel to its immediate neighbors are the 4-neighborhood von Neumann system (Fig. 1 (b)) and 8-neighborhood Moore system (Fig. 1 (c)) [1], [2]. However, neither neighborhood system is able to refine the Euclidean distance estimates necessary in techniques like morphology, watershed segmentation, eikonal halftoning, or path-finding, as shown by Maragos in [2]. Such techniques require accommodation via larger neighborhoods of pixels or solutions to eikonal differential equations [3]–[5].

Euclidean distances can be better approximated in the grid with a process called *chamfering*. The term "chamfer" comes from woodworking, where a right-angled corner is repeatedly planed off to approximate a rounded edge. In the earliest chamfers, Rosenfeld and Pfaltz demonstrated that by using compositions of cardinal and diagonal distance functions, more deliberate octagonal distance propagation patterns can be achieved. They applied this to detecting clusters, elongation, regularity, and path-finding [6]. Others have refined this by applying an algorithmic approach to making circles from basic polygons [7], [8]. Borgefors surveyed a number of *ad hoc* chamfering approximations by assigning various weights and scales to masks of neighboring connections to points in $\mathbb{Z}^2$ [9], [10], which has become a common technique for path-finding in images. These weights were optimized in the geometric analyses conducted by Butt and Maragos [4], [5] and similar analyses and applications were conducted in [11]–[14]. Higher dimensional considerations have been made in works like [15].

Chamfers on a grid are representable as graphs [16], in which weighted edges are added symmetrically to the basic grid graph, forming shortcuts to local neighborhoods. These shortcuts help shape the region of points that one can reach within a given radius. Each additional shortcut creates more faces in a polygonal approximation.

While chamfering graphs do provide a useful tool for modeling idealized distances, they have some drawbacks. Consider the Moore neighborhoods in Fig. 1 (c), where diagonal moves are permitted. This can be problematic in some applications such as video games. If the gray pixels in (c) represent walls, a diagonal move between them would be tantamount to a 2-D avatar turning sideways and slipping through the crack between the bricks. Thus, "diagonal moves are only possible if the related cardinal moves are both possible" [16]. Similarly in Fig. 1 (d), executing the knight's move to realize a better

T. A. Terlep and M. R. Bell are with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
e-mail: taterlep@gmail.com.
T. M. Talavage is with the Department of Biomedical Engineering, University of Cincinnati, Cincinnati, OH, USA.
D. L. Smith was with the Division of Engineering and Applied Science, Caltech, Pasadena, CA, USA.

distance estimate would require an act of teleportation through the gray wall, violating the continuity of the grid.

Chamfers also pose topological concerns. In exchange for increased accuracy in approximating Euclidean space, traditional chamfers require connections to distant neighborhoods from each vertex, creating a messy web of crisscrossing edges which rapidly increase the degree (or valency) of vertices. This translates to a large number of edge crossovers. Because each pixel is assumed to have no internal structure, such graphs fail to exploit the tiled modularity of the grid to reduce the global pixel-to-pixel connections required by chamfering. As these edges make ever-larger leaps to incrementally distant neighborhoods, there is a conflicting modeling concern: a path in space must pass through regions that lie between two vertices. This begs the question: Are there better ways to build chamfering graphs which avoid these topological problems of traditional chamfers?

A recent development in graph theory, called the *replacement product*, provides a solution [17]. It converts every vertex of a graph into its own internal network of vertices and edges. The result allows for regions of the graph to behave like "little processors," as suggested by the processor networks of Case [18]. Replacement products, while well understood for generating expander graphs, have not been used in chamfering applications [19]–[21].

This paper presents novel chamfering replacement products and cuts back chamfer reach and edge crossover while reducing individual vertex degree and pixel valency. This product operates on the existing family of chamfers which we call *E-grid graphs*. The result is a new family of chamfering graphs we call *RE-grid graphs*. Although adding additional vertices inside each pixel seems counter-intuitive because it increases the computational complexity in many circumstances, these graphs impose a simpler topology and act as a new geometry, providing superior approximations in some difficult sample spaces as we explore in Section VII and VIII. We conclude with an example which uses the network structure to induce differential behaviors similar to those found in eikonal equations.

II. Background: Traditional Chamfers

The following thought experiment illustrates the basic motivations for chamfering and demonstrates the utility of giving a pixel internal structure by replacing each vertex with a small graph.

A. Simple Example

Colonel Curry, Professor Garnet, and friends sit down to play the Hasbro game *Clue*® (or *Cluedo*®) a boardgame played on a square grid. Players roll a six-sided die to move about a board in which they examine clues. They may only move between squares sharing common edges. Diagonal moves are forbidden.

An argument erupts mid-game over the position shown in Fig. 2 (a). Col. Curry (yellow), who does not care for the notion of metric spaces, points out that although he and Prof. Garnet (red) are the same number of squares from the knife

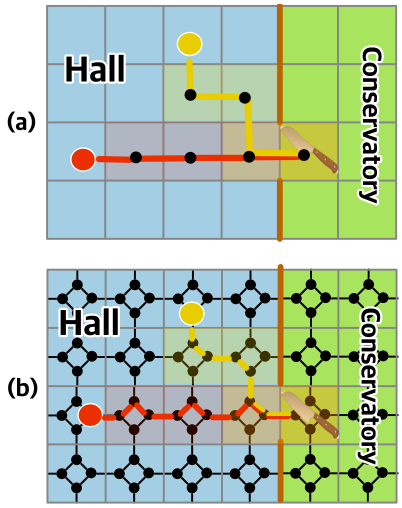


Fig. 2. Col. Curry (yellow) and Prof. Garnet (red) are both vying for the knife just inside the conservatory. (a) The original game board and the standard paths for both players. (b) The board as modified to appease Curry's complaint of an "unfair" distance metric. Note that the new network confers all of the benefits of distance approximation from a Moore neighborhood without cutting corners in the game.

in the conservatory, Prof. Garnet is actually farther from it by the Euclidean "straight-line" distance. In an attempt to placate Curry after a heated and frustrating discussion on the virtues of the city-block metric space, Garnet desperately scribbles in marker the pattern shown in Fig. 2 (b) and announces a new rule called "Col. Curry's chamfer." Players now must move along the black lines and stop on the black dots, treating every line segment as one unit of distance. Although the overall game speed is somewhat slowed, the new rule appeases the colonel's taste for Euclidean geometry well enough to proceed. At this point we will leave the players to finish their game, and will explore the workings of Col. Curry's chamfer in greater detail in Section III.

B. Chamfer Basics

This subsection will review the basics of chamfering from [10], [4], [5]. Effectively, the network or graph used for moving pieces in Clue can be interpreted as the time it takes to travel between two arbitrary pixels (or squares) while traveling at a constant velocity. Chamfering modifies the overlaying network of legal movement so that the collection of points we can reach within a certain time better approximates a circle. That is, the level contours of the minimal time of arrival (ToA) described by traveling optimally through the network are designed by chamfering to closely approximate the true Euclidean distances.

Chamfering improves the accuracy of distance calculations between squares (pixels) when not every direct-line connections can be made. A *neighborhood* is a collection of squares that are connected to a central pixel and share the same distance to it. A *chamfer* connects neighborhoods to that pixel, and defines a distance to each neighborhood. Fig. 3 (a) shows the default neighborhood (red) of the gray pixel. The pixels in this neighborhood are distance $a = 1$ from the

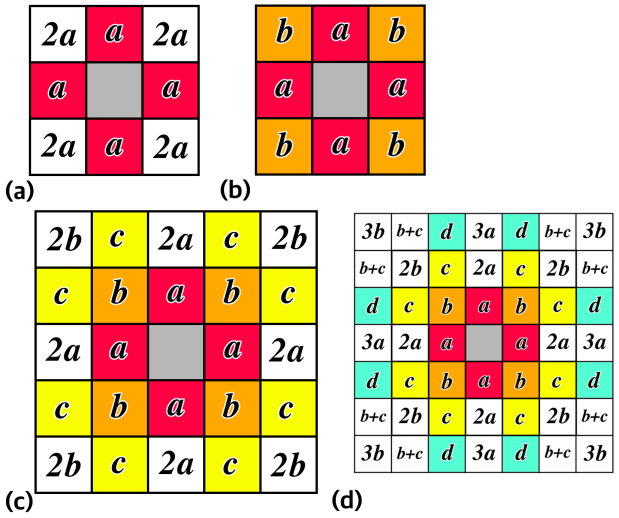


Fig. 3. A common representation of chamfering (or distance masking) using successively farther neighborhoods. Colored squares in the mask have a defined distance and the white squares are the de facto minimum distances. (a) A scaling or non-chamfer on the red neighborhood, where typically the distance $a = 1$. (b) The 3×3 mask defines distances a and b to the red and orange neighborhoods respectively. (c) A 5×5 mask that defines distances a, b, and c to red, orange, and yellow neighborhoods respectively. (d) A 7×7 mask that defines distances to four neighborhoods. Only the eight $b + c$ de facto distances are in need of further chamfer in this mask.

center (note that this is the same as the grid we used in the game board). Fig. 3 (b), (c), and (d) represent successively better chamfering that connect the central pixel to the orange, yellow, and cyan neighborhoods respectively while assigning direct-line distances b, c, and d. Without chamfering, the de facto distances b, c, d in the original grid graph would be the integers 2, 3, and 4 respectively.

A chamfer in a 3×3 mask is fully described by the values of a and b in the 2-tuple (a, b). Some possible values of (a, b) where $a = 1$ include the chessboard, $(1, 1)$; city-block or non-chamfer, $(1, 2)$; Euclidean local distance, $(1, \sqrt{2})$; Borgefors optimal, $(1, 1.3507)$ [9], [10]; Butt and Maragos optimal, $(1, 1.342)$ [5], and scaled integer $(3, 4)/3$.

A chamfer in a 5×5 mask is fully described by the values of a, b, and c in the 3-tuple (a, b, c). Some possible values include Euclidean local distance, $(1, \sqrt{2}, \sqrt{5})$; Borgefors optimal, $(1, \sqrt{2}, 2.19691)$ [9], [10]; Butt and Maragos optimal, $(0.9866, \sqrt{2}, 2.2062)$ [5]; and scaled integer, $(5, 7, 11)/5$.

Similarly, the 7×7 mask can be fully chamfered by a 5-tuple. We will only define and examine a 4-tuple (a, b, c, d), excluding the (2,3) chamfer for later work [4], [10].

Notice that the distances to neighboring squares are not necessarily integers. If this is a problem for a particular application (such as the game of Clue with integer dice rolls), it can be overcome by rounding and scaling up the chamfer to enforce integer values. For example, we could scale the rounded $(3, 4)/3$ to $(a, b) = (3, 4)$.

C. Structure from E-grid Graphs

The square grid and its chamfers are representable as graphs in which square pixels become vertices and connect to neighboring vertices via distance-weighted edges as seen in Fig. 4. For convenience, we call the family of chamfering graphs derived from direct-line connections *Euclidean-grid graphs* or *E-grid graphs*. These graphs have vertices at the exact center of each square pixel. (For a full treatment of graph-theoretic notation and the pixel-to-vertex conversion process, see [22] and [16] respectively).

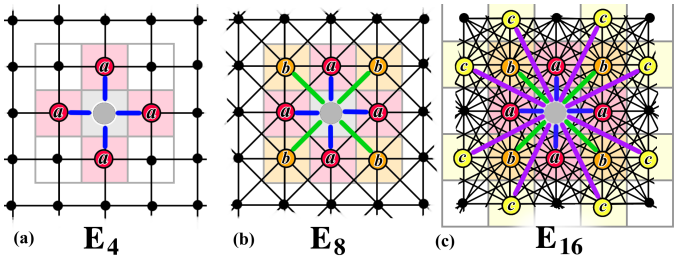


Fig. 4. The graph equivalent of Fig. 3. (a) The E_4-grid graph equivalent of Fig. 3 (a).The central gray vertex is connected via blue edges of length a to the red neighborhood of vertices. The de facto distances of $2a$ inside the mask require chamfering. (b) The E_8-grid graph equivalent to Fig. 3 (b). The central vertex is connected via blue edges of length a to the red neighborhood of vertices. It is also connected via green edges of length b to the orange neighborhood of vertices, chamfering the original grid. (c) The E_{16}-grid graph equivalent of Fig. 3 (c). The central vertex is connected via blue, green, and purple edges of length a, b, and c to the red, orange, and yellow neighborhoods of vertices respectively. Note how each set of additional neighborhoods increases the number of overlapping edges; in fact, E_{24} is not shown because the overlap density would render it unintelligible.

1) Graph definitions: We build our grid graphs as follows. Let $G = \{V_G, \mathcal{E}_G, W_G\}$ be a finite, undirected graph with vertex set V_G of size $m \times n$, edge set $\mathcal{E}_G$, and edge weights W_G.

V_G is a set of *embedded grid vertices* where the vertex set is more formally defined as

$$V_G = \{g\} = \{(x, y) |\ x \in \{1, 2, \ldots, m\}, y \in \{1, 2, \ldots, n\}\}, \tag{1}$$

such that each vertex g_i is embedded at integer (x_i, y_i) coordinates in the real Euclidean plane, $\mathbb{R}^2$.

The edges of an E-grid graph are defined by a subset of distinct coordinate pairs of V_G:

$$\mathcal{E}_G = \{e_{i,j}\} = \{e(g_i, g_j)\} = \{((x_i, y_i), (x_j, y_j))\},\ i \neq j. \tag{2}$$

Note that we do not permit any self-loops in the graph by this definition and that $e_{i,j}$ acts as a shorthand notation for an edge of a known graph G.

A weighted edge is one which assigns a real value cost associated with traversing it (ie. adding it to a path). Thus we define the edge weights of graph G as

$$W_G : \mathcal{E}_G \to \mathbb{R}, \tag{3}$$

$$W_G = \{w_{i,j}\} = \{w(e(g_i, g_j))\}. \tag{4}$$

For convenience and clarity, we will pictorially assign all edge weights with letters $a, b, c, \ldots$ and assign their values in listed equations such as (7). Fig. 4 shows the weight assignments.

A vertex g_i is a *neighbor* of g_j if they share an edge $e_{i,j} \in \mathcal{E}_G$. The set of neighbors with the same weight is collectively called a *neighborhood* of g_i as show by the colored squares and their associated vertices in Fig. 3 and 4.

When we think of time of arrival (ToA) to a pixel, what we really mean is the minimum path distance through the network. Any further weighting of edges changes the local velocity as we go through a pixel. The minimum path distance or simply *graph distance*, $d(g_i, g_k)$, over the edges of the graph is the length of a shortest (minimal) path between g_i and g_k as measured by the sum of the weighted edges, $w_{i,j_1} + w_{j_1,j_2} + \cdots + w_{j_p,k}$, used in that path through the intervening vertices $g_{j_1}, \ldots, g_{j_p}$. By contrast, the *Euclidean distance* (or straight-line distance) between two embedded grid vertices g_i and g_k is defined as

$$u(g_i, g_k) = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}. \tag{5}$$

This definition of Euclidean distance does not require an edge $e_{i,k}$ between g_i and g_k a priori. Note that we will use the generic term "distance" to mean graph distance and specify "Euclidean distance" when we mean the distance in $\mathbb{R}^2$ without necessarily using the paths in the graph.

The *boundary* between two vertex subsets $V_{G,S}, V_{G,T} \subset V_G$, written $\mathcal{E}(V_{G,S}, V_{G,T})$, represents the set of all edges $e_{s,t} = \{((x_s, y_s), (x_t, y_t))\}$ s.t. $e_{s,t} \in \mathcal{E}_{\mathcal{G}}$ and $(x_s, y_s) \in V_{G,S}$ and $(x_t, y_t) \in V_{G,T}$.

The *edge boundary* of a vertex subset $V_{G,S} \subset V$ is the boundary between $V_{G,S}$ and its complement set, $\overline{V_{G,S}}$.

The *Edge Boundary Degree (EBD)* of a subset of vertices $V_{G,S} \subset V_G$ is defined as $\mathrm{EBD}(V_{G,S}) = |\mathcal{E}(V_{G,S}, \overline{V_{G,S}})|$. The EBD is useful in naming the members of our family of E-grid graphs.

A vertex $g_i \in V_G$ is a *full vertex* if $\mathrm{EBD}(g_i) \geq \mathrm{EBD}(g)$ $\forall g \in V_G$. The *interior* of the graph is the set of all full vertices in V_G. A vertex is an *open vertex* if it is not a full vertex. The *exterior* of the graph is the set of all open vertices in V_G.

2) The E_4-grid Graph: Fig. 4 (a) shows the first *E-grid graph* which we call E_4. The subscript implies that each vertex in the graph has at most four natural neighboring connections (*i.e.* is degree 4). The weight of every edge is $a = 1$.

3) Successive E-grid Graphs using MESS: The *MESS* of a given graph grabs the next closest set of neighbors at any vertex whose graph distance is further than their Euclidean distance and adds those edges as a single neighborhood to all vertices, giving them a common Euclidean edge weight. The formal definition follows. Let $G = (V_G, \mathcal{E}_G, W_G)$ be a graph with vertices embedded in $\mathbb{R}^2$ and Euclidean edge weights. If $\exists \omega \in \mathbb{R}$ where ω is the smallest value for which $\exists g_i, g_k \in V_G$ s.t. $d(g_i, g_k) > \omega = u(g_i, g_k)$, then G' is the *Minimal Euclidean Spanning Supergraph* (MESS) of G, denoted $G \triangleleft G'$, with the same vertex set, $V_G = V_{G'}$. The edge set of G' is defined as

$$\mathcal{E}_{G'} = \mathcal{E}_G \cup \{e_{i,j} \mid \omega = u(g_i, g_j) < d(g_i, g_j)\} \tag{6}$$

with associated Euclidean edge weights.

In this context, the term *spanning* simply means that the vertex set $V_{G'}$ is unchanged between G and G'. The additional edges are minimal in that ω is only as large as necessary to include a single new neighborhood of weight ω around all vertices. In the case of the 2D grid, the sequence of ω for successive E-grid graphs is $1, \sqrt{2}, \sqrt{5}, \sqrt{10}, \sqrt{13}, \ldots$ which is sequence `A008784` in the Online Encyclopedia of Integer Sequences (OEIS) and can also be found in the works of Conway and Sloane in [23], [24].

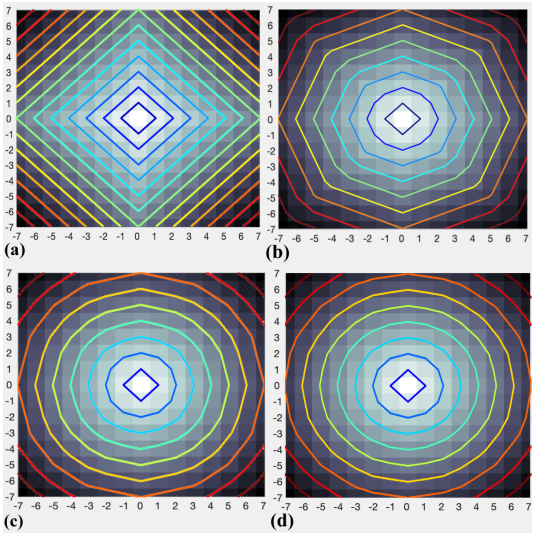


Fig. 5. Distance contour lines for Euclidean chamfers. (a) E_4 where $a = 1$. (b) E_8 where $(a, b) = (1, \sqrt{2})$. (c) E_{16} where $(a, b, c) = (1, \sqrt{2}, \sqrt{5})$. (d) E_{24} where $(a, b, c, d) = (1, \sqrt{2}, \sqrt{5}, \sqrt{10})$. A grid overlays the squares shaded to correspond to graph distances from the central vertex. The multicolored rings denote distance contour lines of unit distance from the center. Note that with each successive chamfer, the polygon gains more edges and becomes more circular, thus more closely approximating the Euclidean metric.

Finally we define E_k as an *E-grid graph* if it is E_4 or the MESS of another E-grid graph, where k is the EBD of any vertex in the infinite grid, that is, k is the number of neighbors we expect to have for an interior vertex of a sufficiently large grid.

The second member of the E-grid family is E_8, shown in Fig. 4 (b). The edge set is extended to include the closest diagonal and antidiagonal connections. This graph is commonly called the *king graph*, for the movement of the king in chess [25]. The weights W_{E_8} are defined by (7):

$$W_{E_8} : \Big\{ a = 1, b = \sqrt{2}. \tag{7}$$

The next member of the E-grid family is E_{16}, shown in Fig. 4 (c). This graph includes the edges from the king graph and adds the *knight graph*, so called for the movement of the knight in chess [26]. With E_{16} we skip over those connections that are redundant and smaller than $\sqrt{5}$, namely those that have a Euclidean distance of 2 in the horizontal or vertical directions, because their graph distances are already minimized. This keeps our vertex valency as minimal as possible. The edge weights $W_{E_{16}}$ are defined by (8):

$$W_{E_{16}} : \Big\{ a = 1, b = \sqrt{2}, c = \sqrt{5}. \tag{8}$$

The fourth extension of the E-grid family is E_{24} which connects via an elongated knight's move. The edge weights $W_{E_{24}}$ are defined by (9):

$$W_{E_{24}} : \Big\{ a = 1, b = \sqrt{2}, c = \sqrt{5}, d = \sqrt{10}. \tag{9}$$

D. Distance Contour Maps

Fig. 5 shows the distance contour maps for the first four E-grid graphs. The shades of gray reflect the graph distances from the central square. The chamfers in (b), (c), (d) generate irregular octagon, 16-gon, and 24-gon propagation patterns respectively, each of which provides a successively better distance approximation in $\mathbb{R}^2$. The shape of the contour map polygon can be mildly adjusted by changing any edge weights. This will push or pull the corners of the polygon along that weight's associated axes of symmetry allowing for optimization as in [5] or more deliberate anisotropic deformation.

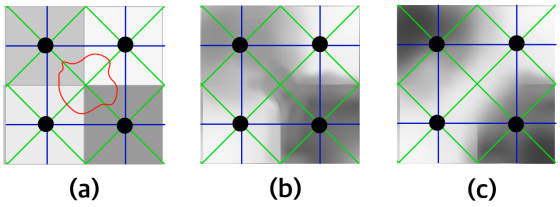


Fig. 6. An ambiguous corner-to-corner connection. (a) E_8 over an example of a sampled ground truth. (b) and (c) represent two reasonable possibilities for the ground truth source image. If we assume only that only the light regions are connected, then a true antidiagonal path through the red bounded region in (a) becomes dubious.

E. Chamfer Connectivity Concerns

All the chamfering MESSes of E_4 share a common feature: a diagonal connections between pixels. Are two pixels adjacent to each other if they only share a single point in common? In many algorithms this detail may be trivial, but for algorithms involving some kind of local connectivity or path-finding, we may not be able to reasonably assume that every diagonal connection is viable. In these cases, we must defer to E_4, a poor facsimile when direct Euclidean distances are required, or manipulate higher order E-grid graphs. Fig. 6 shows how ambiguity may arise between antidiagonal regions in a sample of a high-frequency image space. In Section VII we will show how this ambiguity has a significant impact on distance contour maps.

As seen in Fig. 4, the finer the desired approximation of Euclidean space, the farther we must reach for neighbors. This creates a denser networks of messy crisscrossing edges as we continue to improve our chamfers with direct-line connections. In networks, a high degree at any node can arbitrarily increase its perceived importance in network analysis. Replacement product graphs may help solve this issue by turning a single-vertex pixel into its own network of lower valency vertices. Not only does this reduce the individual vertex valency, but it also decreases the number of pixel-to-pixel connections, as we will see in Section IV.

Creating neighborhoods of vertices separated by large jumps may also not make modeling sense, bypassing the intuition that any connection between two non-adjacent squares must somehow travel through other intervening squares. While this is not an issue in alternative transit models like [27] and [28], if we model a basic path from California to New York, we would expect it to pass through other states along the way.

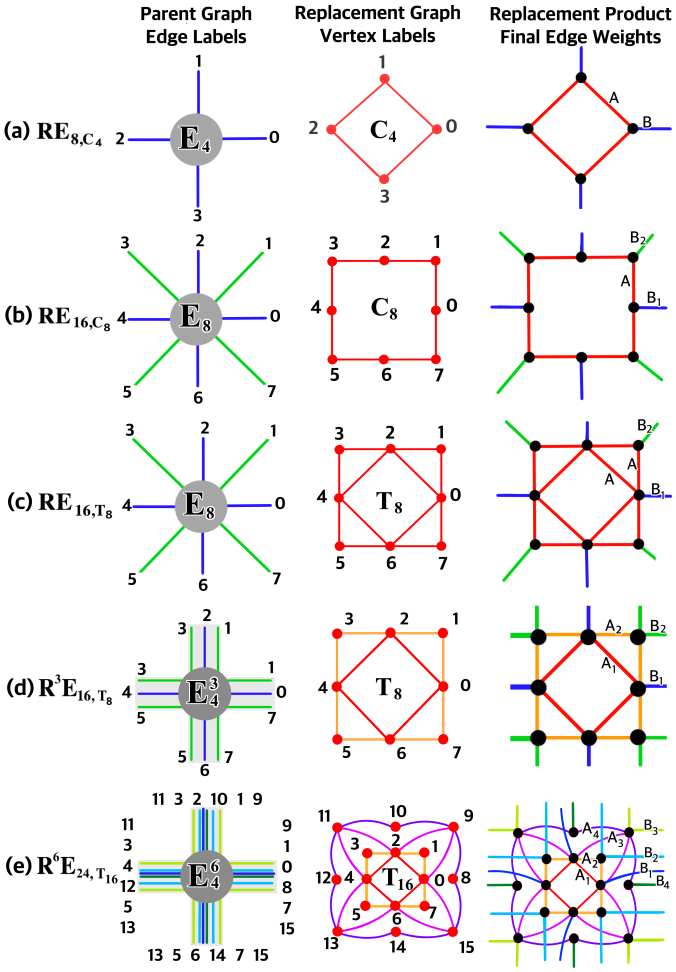


Fig. 7. (a)-(e) show our five graphs which chamfer like like traditional E-grid graphs. Left column: parent G, center column: replacement H, right column: resultant replacement product R. (b) and (c) are alternate 16-gon chamfers with only eight neighbors. The first, second and third Curry chamfers, shown in rows (a), (d), and (e) respectively, are three replacement graph chamfers which connect to only the four adjacent pixels. The elegant lotus-patterned 24-gon chamfer in row (e) shows that high-order chamfering can be accomplished locally and without excessive crossovers.

In the next section, we will explore how our replacement product chamfers mitigate or overcome these concerns in exchange for increased computational complexity.

III. A New Chamfer: RE-grid Graphs

Replacement products take a *parent* graph G and replace each vertex with another graph H. The graph H is called the *replacement* graph. In all constructions presented here, G is a kind of E-grid graph but need not be the case for chamfering replacements. H is typically a much smaller graph than G. How the vertices of H inherit the edges around each vertex in G can vary depending on the application for which it is intended. In the first few cases, we will match the edges and vertices one-to-one, and we will do this surgery on every vertex in the E-grid. Later, we will slice the parent edges lengthwise into channels and match multiple edges per vertex in the replacement.

Fig. 7 provides a practical guide for all of our constructions throughout the section. We begin by looking at each vertex in the original E-grid graph (left column) and labeling all of its edges so that every vertex has a name for each edge around it. Note that each edge will have two names depending on which vertex is queried. Next we decide on a graph (middle column) which we will plug into each vertex of the original parent graph. We also label each vertex of this replacement graph. The resultant product (right column) replaces each vertex of the parent graph with the replacement graph, matching the labeled edges to their same-name vertices. The rigid orientation of E-grid graphs ensures that this replacement can be done unambiguously throughout the plane.

The replacement product is useful for constructing expander graphs and relevant information about their construction and other applications can be found in [17], [19], [20]. The zig-zag product is a kind of replacement product, that when it operates on $\mathbb{Z}^2$ with the 4-cycle replacement graph, superficially resembles our construction of Col. Curry's chamfer [21]. By applying a novel weighting scheme, this zig-zag product graph takes on a new role as a chamfer. Our subsequent replacement product constructions do not coincide with any other well known expander graphs nor are there any other known chamfering approaches which deliberately increase the complexity of the network structure inside a pixel in order to address the concerns listed in the previous section. Because we are effectively reshaping the shape of space inside each pixel with the replacement product, we will need to assign our own new weights to the resultant replacement product graphs to make them meet our chamfering requirements.

A. Replacement Products on E-grid Graphs

Replacement products are typically defined over degree-regular graphs, where each vertex has the same number of neighbors. For our engineering applications in a finite $m \times n$ grid, this is not the case. When assigning labels in the boundary of a finite grid, we can exploit the fact that all vertices are basically translated copies of one another and simply omit the missing connections from the replacement.

1) Parent Graph Edge Label: The left column in Fig. 7 shows the edge labels for each case. Let $G = \{V_G, \mathcal{E}_G\}$ be an unweighted graph. Then a *generalized edge labeling* of the edges $e_{i,j}$ associated with the vertex g_i is defined as

$$\mathcal{L}_{\mathcal{E}_G} : \{(g_i, e_{i,j})\} \to \mathbb{Z}. \tag{10}$$

2) Replacement Graph Vertex Labeling: The graph H that replaces the vertices of the parent E-grid graph must have its vertices labeled to tie them to the appropriate edges in the parent. The center column of Fig. 7 shows the labeling scheme for each replacement graph.

A *vertex labeling* on the vertices $h_i \in V_H$ of a graph H is defined as the injective (one-to-one) map:

$$\mathcal{L}_{V_H} : V_H \to \mathbb{Z}. \tag{11}$$

3) Matching Label Replacement Product: The replacement product may be conveniently defined by inserting the replacement graph at every vertex of the parent graph, matching the labels assigned to the edges of the parent graph to the vertices of the replacement.

More formally, let $G = \{V_G, \mathcal{E}_G\}$ be the parent graph with vertices $g \in V_G$ and edge labeling $\mathcal{L}_{\mathcal{E}_G}$. Let $H = \{V_H, \mathcal{E}_H\}$ be the replacement graph with vertices $h \in V_H$ and vertex labeling $\mathcal{L}_{V_H}$. Then the *generalized replacement product graph* of G with H is the non-commutative operation

$$R = G \textcircled{\mathcal{R}} H, \tag{12}$$

where R is the graph with vertices and edges $(V_R, \mathcal{E}_R)$ s.t.

Vertices:

$$V_R = V_G \times V_H = \{(g, h)\}\ g \in V_G,\ h \in V_H. \tag{13}$$

Outside cloud edges, $\mathcal{E}_{R1}$:

$$\begin{gathered}\{(g, h), (g', h')\} \in \mathcal{E}_{R1} \text{ if } e(g, g') \in \mathcal{E}_G, \\ \text{and } \mathcal{L}_{\mathcal{E}_G}(g, e(g, g')) = \mathcal{L}_{V_H}(h), \\ \text{and } \mathcal{L}_{\mathcal{E}_G}(g', e(g', g)) = \mathcal{L}_{V_G}(h').\end{gathered} \tag{14}$$

Inside cloud edges, $\mathcal{E}_{R2}$:

$$\{(g, h), (g', h')\} \in \mathcal{E}_{R2} \text{ if } e(h, h') \in \mathcal{E}_H \text{ and } g' = g. \tag{15}$$

All edges, $\mathcal{E}_R$:

$$\mathcal{E}_R = \mathcal{E}_{R1} \cup \mathcal{E}_{R2}. \tag{16}$$

A *cloud* is the region inside a pixel wherein a vertex of the parent graph G has been replaced by H. In other words, each cloud is a copy of H and is externally connected to other clouds via the set of edges inherited from the parent G.

We call a replacement product graph from an E-grid parent graph a *Replacement Euclidean-grid graph*, or *RE-grid graph*, denoted $RE_{\gamma,H}$. Rather than use the subscript k of the parent graph E_k, the first subscript, γ, indicates the E-grid graph that the replacement product graph most emulates as a chamfer. The second index indicates the replacement graph, H, that was used to construct $RE_{\gamma,H}$.

The vertex set of the RE-grid graph is a 3-tuple, (x, y, z), where x and y are the coordinates of the parent vertex and z is the label of the replacement graph vertex. We refer to the label z as a *state* within a cloud.

B. Identity and the Replacement Distance

By convention, we restrict ourselves to starting and stopping in the same state when measuring distances between pixels. Since the cloud has no central vertex, we call a selected start-and-stop state an *identity* of the pixel. For RE_{8,C_4}, any vertex may be the identity. For all other RE-grid graphs shown in center column of Fig. 7, we restrict our identities to the vertices labeled 0, 2, 4, and 6.

The distance between pixels in our replacement graphs is called the *replacement distance* and we define it as the graph distance over the replacement product graph when we start and stop on the same identity. For all of our RE-grid graphs, all four identities will provide the same value of replacement distance by symmetry. The selection of identity can become non-trivial in situations where the edges are further weighted as a function of pixel values. We will explore one such function in Section VI. It is important to note that our weights internal to the pixel do not conform to flat surfaces per se, thus physical

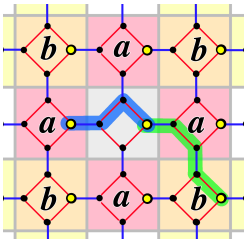


Fig. 8. Examples of minimal distance paths in Col. Curry's chamfer. Path lengths a (blue) and b (green) correspond to the same color scheme in Fig. 4 (b). All other minimal paths may be found by symmetry and combinations of basic pathways. Starting and stopping at the identity (yellow vertex) maintains metric consistency between clouds.

manifestations would rely on curved edges or be built on 2D Riemannian manifolds.

We formally define the replacement distance (pixel-to-pixel) as $\rho(g_i, g_k, z) = d(r_i, r_k)$ where $g_i = (x_i, y_i)$, $g_k = (x_k, y_k)$, for $g_i, g_k \in V_{E_k}$, and $r_i = (x_i, y_i, z)$, $r_k = (x_k, y_k, z)$, for $r_i, r_k \in V_{RE_{\gamma,H}}$, and z is a fixed identity for R. It is worth noting that the replacement distance is designed as a new distance function for the *original* parent vertex set and a fixed z.

C. Col. Curry's Chamfer: RE_{8,C_4}

We construct the RE-grid graph associated with Col. Curry's chamfer as shown in Fig. 7 (a) by first taking the parent graph E_4 (left) and replacing each vertex with the replacement graph C_4 (middle). The right column shows the resultant chamfering replacement product graph, RE_{8,C_4}.

Formally, we call this RE_{8,C_4} because it emulates the chamfers of E_8 under a replacement product with C_4. The weights of this graph are defined as follows:

$$W_{RE_{8,C_4}} : \begin{cases} A = \frac{2-\sqrt{2}}{2} \\ B = \sqrt{2} - 1. \end{cases} \tag{17}$$

The final product in the right of Fig. 7 (a) identifies the locations of some of the edge weights, labeled A and B, which are applied symmetrically throughout the replacement product graph. Note that in this and all subsequent replacement product graphs, A will denote edges internal to a cloud (pixel) and B will denote edges between clouds.

Fig. 8 shows examples of minimal identity-to-identity paths between adjacent and diagonal clouds. To create the equivalence to the Euclidean E_8 chamfer, the blue paths composed of $\{A, A, B\} = 2A + B$ must be of length $a = 1$ and the green paths of $\{B, A, B, A\} = 2A+2B$ must be of length $b = \sqrt{2}$. The weights (17) are derived in (18) below. Selecting any feasible (a, b) will transform a desired chamfer from E_8 into RE_{8,C_4}:

$$\begin{cases} 2A + B = a = 1 \\ 2A + 2B = b = \sqrt{2}. \end{cases} \tag{18}$$

The distance contour map for the Euclidean chamfer in RE_{8,C_4} is identical, by design, to E_8 shown in Fig. 5 (b).

We may refer to any version of RE_{8,C_4} as an argument on its 2-tuple of weights (A, B). Col. Curry's chamfer in the boardgame example used the chamfer $RE_{8,C_4}(1, 1)$. Using the transform in (18) we discover that this is equivalent to the $(a, b) = (3, 4)$ chamfer on E_8! For both graphs, adjusting their weights will push or pull the corners of the polygonal propagation pattern. A sufficiently small b with respect to a will produce a more square pattern until $b < a$ as seen in [5].

D. King and Castle Chamfers

The king's chamfer, Fig. 7 (b), and the castle chamfer, Fig. 7 (c), are two 16-gon replacement product chamfers using E_8 (the king's graph) as the parent graph. They demonstrate two evolutionary tracks in our replacement product chamfers, however both still require a diagonal connection, so we will only briefly mention them in passing here. The king's chamfer demonstrates how we might develop a family of RE-grid graphs from ever larger cycle graphs and specially designed set of parent graphs related to E-grids (which is outside the scope of this paper). The castle chamfer introduces the T_8 replacement graph which is used in the construction the remaining two chamfering graphs. The respective Euclidean edge weights for king and castle chamfers are listed below in (19) and (20):

$$W_{RE_{16,C_8}} : \begin{cases} A = \frac{2-\sqrt{2}}{4} \\ B_1 = \sqrt{2} - 1, B_2 = \frac{2\sqrt{5}+\sqrt{2}-4}{2}, \end{cases} \tag{19}$$

$$W_{RE_{16,T_8}} : \begin{cases} A = \frac{2-\sqrt{2}}{4} \\ B_1 = \sqrt{2} - 1, B_2 = \sqrt{5} + \sqrt{2} - 3. \end{cases} \tag{20}$$

E. Channeling

It is possible to create more refined chamfers while only connecting clouds to their four immediate neighbors. We define a *channeled replacement product* as a replacement product wherein we lengthwise split, or *channel*, the edges of the parent graph into a multigraph. The new edges of this parent graph are called *channels*. We denote the number of channels per edge as a superscript of R, as shown in Fig. 7 (d) and (e). A c-channeled version of E_4, denoted E_4^c, can now be the parent for an arbitrarily large replacement graph as long as the labeled channels of the parent have a corresponding labeled vertex in the in that replacement. We also note that we will allow for multiple channels to connect to the same vertex.

F. Curry's Second Chamfer: R^3E_{16,T_8}

Our construction of Curry's second chamfer uses the replacement graph T_8 and the three-channel E_4^3 graph shown in Fig. 7 (d).

$$W_{R^3E_{16,T_8}} : \begin{cases} A_1 = \frac{2-\sqrt{2}}{2}, A_2 = \frac{3-\sqrt{5}}{2} \\ B_1 = \sqrt{2} - 1, B_2 = \sqrt{5} - 2. \end{cases} \tag{21}$$

G. Curry's Third Chamfer: $R^6E_{24,T_{16}}$ (RE_{24})

Building from the T_8 graph, we introduce T_{16} in the center of Fig. 7 (e). To accommodate the large number of vertices, we use the six-channeled E_4^6. For shorthand, we will refer to

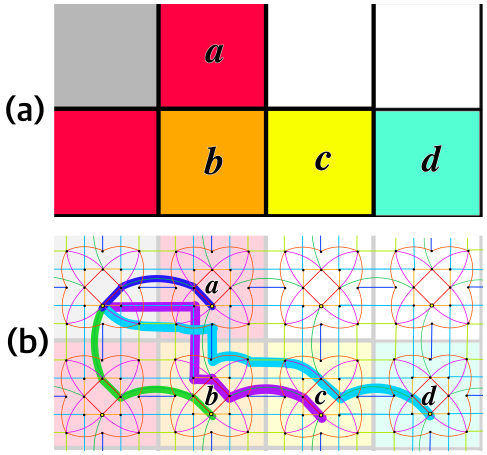


Fig. 9. A canonical set of paths for a single identity in RE_{24}. (a) A portion of Fig. 3 (d) showing the four types of neighborhoods chamfered from the gray pixel. (b) The four highlighted paths connect one identity of the RE-grid graph to the same identity in each of those four neighborhoods via a minimal-distance path. Other minimal path in RE24 can be derived by symmetries and alternatives of these basic pathways.

this simply as RE_{24}. Fig. 9 (a) and (b) show minimal paths used by RE_{24} to the four neighborhoods of Fig. 3 (d).

$$W_{R^6E_{24,T_{16}}}:$$
$$\begin{cases} A_1 = \frac{2-\sqrt{2}}{2},\ A_2 = \frac{3-\sqrt{5}}{2},\ A_3 = \frac{4-\sqrt{10}}{2},\ A_4 = \frac{4-\sqrt{10}}{2} - \frac{\sqrt{10}-3}{4}, \\ B_1 = \sqrt{2}-1,\ B_2 = \sqrt{5}-2,\ B_3 = 3\frac{\sqrt{10}-3}{2},\ B_4 = \frac{\sqrt{10}-3}{2}. \end{cases} \tag{22}$$

IV. Topological Analysis

This section will demonstrate the structural benefits of RE-grid graphs by examining how the modular pixel networks connect to one another tiled over the grid. We will also consider the planarity of the graphs we have examined so far. We define *planarity* as the minimum number of sheets, or planes, required to draw the graph without any overlapping edges where the resultant graph is the union of those planar graphs. One can imagine pinning these 2D sheets together at each vertex.

TABLE I
Topological Properties

Graph	Vertex Degrees	Pixel EBD	n-gon	Planarity
E_4	4	4, 4	4	1
E_8	8	8	2	8, 8
E_{16}	16	16, 32	16	6
E_{24}	24	24, 56	24	10
RE_{8,C_4}	3	4	8	1
RE_{16,C_8}	3	8	16	1
RE_{16,T_8}	3,5	8	16	1
R^3E_{16,T_8}	4,5	12	16	1
$R^6E_{24,T_{16}}$	3,4,6,7	24	24	2

A. Pixel-to-Pixel Valency: Pixel EBD

Table I lists the properties of our E-grid and RE-grid graphs related to topology. When two Pixel EBD values are shown, the first shows the valency where E-grid graph edges are allowed to "jump" over intervening pixels without intersecting them, ignoring sheet topology as shown in Fig. 1. The second Pixel EBD value disallows jumps, counting each intersected entry and exit to a pixel. For RE-grid graphs, these values are one and the same since all the edges of those graphs within a von Neumann or Moore neighborhood.

If we disallow jumping, all RE-grid graphs are better than or equal to their associated E-grid graph counterpart of the same contour n-gon for having a smaller pixel EBD. The fact that our RE-grid graphs require less connections between clouds (pixels) in a square-tiled array may reduce cost for implementation where the cost for traversing or constructing edges *inside* the cloud are trivially small compared to external connections.

B. E-grid planarity problems

Physical implementations of these networks may necessitate the removal of crossovers from E-grid graphs. To accomplish this, we can either use additional sheets to deconflict crossovers, or we can turn the many overlaps into nodes of intersection, far surpassing the vertex counts of RE-grid graphs. The traditional E-grid graph chamfers are simply not designed with topological or physical applications in mind. RE-grid graphs make it possible to consider the practical design of a passive sensor array for near-Euclidean distances. In Sensor Network Localization (SNL), a set of embedded points is used to preserve the distance information in a low-dimensional space. This is useful for source localization, molecular connectivity in three dimensions, graph embedding, and data visualization [29], [30]. Further exploration of the spectral and resonance characteristics of RE-grid graphs is required to discover applications in these areas.

V. Complexity Analysis

This section will analyze the limitations of RE-grid graphs that spring from their increased computational complexity, and demonstrate how to mitigate them by using a selected canonical subset of pathways as shown in Fig. 9. As an example, we examine the challenges of using Dijkstra's algorithm.

A. Dijkstra's Algorithm

The known complexity for Dijkstra's algorithm using a Fibonacci heap is [31]

$$\Theta(|E| + |V| \log |V|) \tag{23}$$

For an $m \times n$ grid, Table II gives the vertex and edge counts for E-grid and RE-grid graphs. It is important to note that any path-finding algorithm on basic E-grid graphs do not act as a function of the intervening pixels for the longer edges, as seen in Fig. 1. RE-grid graphs computationally perform poorly against their E-grid graph counterparts because the $|E|$ and $|V| \log |V|$ terms of (23) are worse in every RE case. We mitigate this in the next subsection.

TABLE II
VERTEX AND EDGE COUNT

Graph	# Vertices	# Edges ([external] + [internal])
E_4	nm	$2mn - m - n$
E_8	nm	$4mn - 3m - 3n + 2$
E_{16}	nm	$8mn - 9m - 9n + 10$
E_{24}	nm	$12mn - 17m - 17n + 22$
RE_{8,C_4}	$4nm$	$[2nm - n - m] + [4mn]$
RE_{16,C_8}	$8nm$	$[4mn - 3m - 3n + 2] + [8mn]$
RE_{16,T_8}	$8nm$	$[4mn - 3m - 3n + 2] + [12mn]$
R^3E_{16,T_8}	$8nm$	$[6nm - 3n - 3m] + [12mn]$
$R^6E_{24,T_{16}}$	$16nm$	$[12nm - 6n - 6m] + [28mn]$

B. Simple RE-grid Graphs

We can mitigate the RE-grid graph's computational complexity by selecting a single minimum path in RE to each neighboring pixel in the associated E-grid graph. This reduces the complexity to that of the E-grid case when the minimal-path edge weights are calculated prior to the execution of Dijkstra's algorithm. We will call this bundle of canonical paths a *simple RE-grid graph*. Fig. 9 shows such a set of minimal pathways for the identity labeled "6" in Fig. 7 (e), center column. Note that some of the simplified paths will reuse the same edges in the overall graph structure. These overlaps provide the key structural novelty which will lead to all of the differences in distance calculation between E_{24} and RE_{24} in Sections VI, VII, and VIII. In addition to overlaps, the choice of identity will result in paths going through alternative sets of pixels. We examine how this occurs more closely in Section VI-B. Simple RE_{24} with pre-calculated path weights renders a computational complexity of

$$\Theta((12mn - 17m - 17n + 22) + mn \log(mn)) \qquad (24)$$

which is equal to that of the E_{24} case.

The complexity of using any RE-grid graphs depends on the structure of an algorithm and this simple RE-grid graph mitigation may not be suitable for all applications. If for example there is an assumed directionality such as we have in the case of the A^* algorithm [16], then there may be pruning or rank ordering of the edge search preferences on RE_{24} which we can apply to the network using the Euclidean direction to target since the simple RE-grid graphs essentially make the assumption that we are maintaining a given heading as we transit through a pixel. In that case, this simplification may not be reasonable.

VI. WEIGHTED IMAGE SPACES

As stated in Section II-B, the distance we travel in the graph (network) can be thought of as the minimum travel time between two pixels. But what if it is difficult to travel through certain regions of the space? Returning to Clue, suppose that we want to double the travel time through Conservatory tiles; perhaps the Conservatory is cluttered with plants and mud which make it less traversable. Then we do so by multiplying all edges inside there by two. To generalize this, we define a *weighted image space* where the pixel value will apply a weight function to the edges of that pixel. We define the weighted image space as a zero-to-one-valued grayscale image where the value of a pixel in the weighted image space is $x \in [0, 1]$. Each pixel's edges are weighted by a factor of $1/x$, representing velocity just as we have in the eikonal equations [3], [4]. Thus the pixels of the Clue weighted image space are 0.5 inside the conservatory and the edge weights therein are multiplied by $\frac{1}{0.5}$. We define *traversibility* through a pixel as the value of the weighted image space. If the value of the pixel is 0, then the region is not traversable and all relevant edge weights are multiplied by infinity. *Thus the graph distances in a weighted image space represent an approximate travel time through a roughly traversable image space*.

A. Segmented E-grid graphs

To apply the weighted image space to graphs like E_{24}, we need to segment the edges of the graph when they cross over a pixel boundary and multiply the weight of each segment by the $1/x$ value of its associated pixel. We call this a *segmented* E-grid graph. We sum over these segments to produce the final distance. While we could segment the edges of RE_{24} which equally stride between two pixels, we can weight them using the harmonic mean with the same mathematical outcome.

B. RE Identities in Weighted Images

The choice of identity determines the pixels used in minimum pathways for RE-grid graphs. This is demonstrated by tracing a minimal diagonal path with the *southern* identity in RE_{8,C_4} in Fig. 8. Instead of going right-then-down, as we have with the green path from the *eastern* identity, the southern identity path in green would go down-then-right. If the application uses long distances ($\gg 10$ pixels) over a smooth weighted image space, the differences between identities, and between E-grid and RE-grid graphs, are all trivially small. In cases with large fluctuations, ie. high-frequency behavior or very coarse resolution of a truly smooth surface, RE-grid and E-grid graphs can provide very different distance estimates from each other which we will exploit in subsequent sections. Furthermore, our identities may also produce path and distance variations from each other. Thus all four identities of the RE-grid graph should be considered, as well will demonstrate in Section VIII.

VII. A SQUARE PEG IN A ROUND HOLE

Suppose we cut up the Clue game board from Fig. 2 into squares and rearrange the Conservatory and Hall tiles into a checkerboard pattern of 1.0 (empty) and 0.5 (cluttered) pixels. Intuitively, since both smooth and random arrangements would produce near-circular shapes for level contours of travel time, we may expect, and indeed desire, near-circular shapes over the checkerboard as well, but sadly this not the case when we calculate using any traditional E-grid graphs. The arrangement of tiles has a significant impact on spacial traversability for higher-order E-grid graphs.

Fig. 10 (a) shows a 50×50 pixel checkerboard region with alternating values of 1.0 (smooth) and 0.5 (difficult). Patterns like this can easily occur as a common aliasing effect of poorly

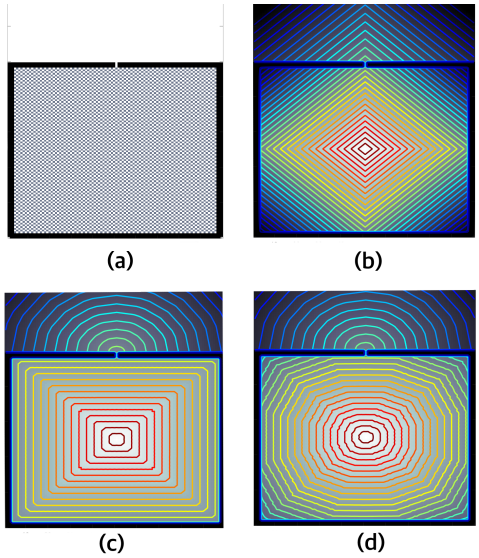


Fig. 10. (a) A 50×50 checkerboard region alternating between 1.0 and 0.5. The top region is an open weighted region of 1. The interface between the two has a hole which is a single pixel wide. All distances are measured from the center of the checkered region. (b) E_4 distance map (grayscale) with contour lines (jet). Although E_4 uniformly scales back its rate of propagation in the checkered space, it retains an undesirable and unrealistic diamond shape in all situations. (c) Segmented E_{24} distance map and contour lines shows how higher order E-grid graphs exploit the use of diagonal paths, collapsing distances into a kind of square metric space. (d) Simple RE_{24} produces a more reasonable compromise between E_4 and E_{24}, boasting a roughly circular polygonal shape which is resistant to the challenging high frequency checkerboard pattern. The normalized distance from the central white pixel to a white pixel (white-to-white) or gray pixel (white-to-gray) in the due north direction is shown in Fig. 11 for variable shades for gray pixels ranging from near-black (0) to white (1).

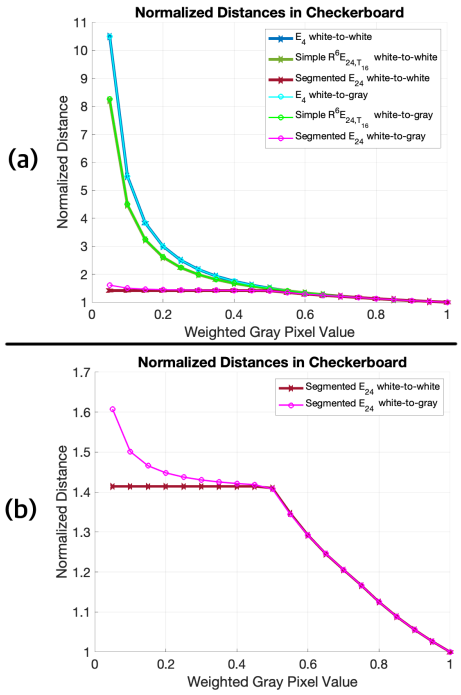


Fig. 11. Normalized distance to a point directly north from the center of the 50×50 checkerboard region in Fig. 10 as the gray value weight is changed. Distances are segregated by those paths which connect from the central white pixel to another white pixel (white-to-white) and those which connect the central white pixel to a gray pixel (white-to-gray). (a) Smooth transition for RE_{24} (green) and E_4 (blue) as the the gray-value in the checkerboard goes from a near black (x-axis left) to white (x-axis right). (b) Zoom-in of the drastic white-to-white bias and discontinuity using E_{24} (magenta).

sampled data or as a synthetic effect. The top wall has a 1-pixel-wide slit above which is empty white space. The open region to the north through the 1 pixel slit serves to show the level-time contour shape over a smooth space. (b)-(d) show level curves of time from a single pixel in the center of that checkerboard space using E_4, segmented E_{24}, and simple RE_{24} respectively. We see free-space contour patterns return after passing through the northern slit. Astonishingly, the simple RE_{24} maintains a reasonably circular shape for timing despite the difficulties encountered by the stubbornly diamond and square shapes derived from E-grid graphs. This suggests that distances measured with RE-grid graphs may provide more practical estimates in some structured image environments since they carefully blend the assumptions of cardinal (E_4) and corner-to-corner (E_8 through E_{24}) connectivity. Briefly, we note that E-grid graphs are not "wrong" per se as they are valid metrics in their own right, but we must ask: Are they a good model of reality in cases when spatial resolution limits continuity?

To press this point further let us compare the behavior of these three graphs as we change the value of the gray pixels in the checkerboard from near-black to white as shown in Fig. 11 (a). We normalize pixel-to-pixel distance by dividing the graph distance in each case by the Euclidean distance. As the gray pixel value goes from black to white we find that there is an incongruity between paths which connect from the central white pixel to another white pixel (white-to-white) and those which connect the central white pixel to a gray pixel (white-to-gray). In Fig. 11 (b), E_{24} plateaus in white-to-white normalized distances as it exploits the diagonal connections in the difficult image space without any consideration of bordering pixels! We might expect that the difficulty of distance increases with the dark-level of the gray pixels, but this is only true over a certain region of grey-levels and does not vary continuously in white-to-gray paths. Thus the once nearly circular E_{24} has completely collapsed into a kind of square metric after a measly gray-weight of roughly 0.5. Furthermore, the inflection point and plateau in (b) show how arbitrary brightness levels in the weighted image space could induce severe discontinuities in distance calculations when using any E-grid chamfers with 8 or more direct-line neighbors. This kind of behavior shows that assuming corner-to-corner connections is a dangerous modeling decision. By comparison, RE_{24} retains a higher order polygonal propagation pattern through the space and acts as a compromise between the E_4 distance estimate and the successive E-grid graphs, as seen by the green lines in Fig. 11 (a). This occurs in the image space without any ad hoc interpolating or averaging methods, as it is an inherent feature of the graph structure! Thus RE-grid graphs maintain the intuitive sense of realism.

All E-grid graphs propagate like squares (or diamonds, ie. rotated squares) in the checkerboard space. Because tiled squares only offer the von Neumann and Moore neighborhood

systems, many medical imaging techniques apply pixel-by-pixel functions without concern for these kinds of errant behaviors that can occur in the metric space, shoving a square peg (contours of E-grid graphs) into a round hole (circular propagation). RE-grid graphs are an alternative way to connect the image space and largely avoid this problem. More advanced techniques, like [32], use higher order neighborhoods such as the E_{16}-grid graph on which they apply a windowed function [33], [34]. If this neighborhood system encounters a regularly ordered, high-frequency image space, like the weighted checkerboard presented here, the techniques and algorithms which use them could break down by measuring along the aliased short cuts, as shown in our simple example.

VIII. Case Study: Rover Traversal

The checkerboard case study suggests that RE_{24} may provide better corner-to-corner estimations when a sparsely sampled weighted image space has large discontinuities. As a thought experiment, we'll examine a very sparse sampling averaged on four smooth sheets: two hyperbolic planes and two flat planes.

Suppose we are pre-planning a rover mission to a distant world, for which only low-resolution images are available, and we wish to estimate how long it will take the rover to traverse the various landforms we can make out.

We have specifically chosen manifolds which highlight the successes and failures of RE-grid graphs.

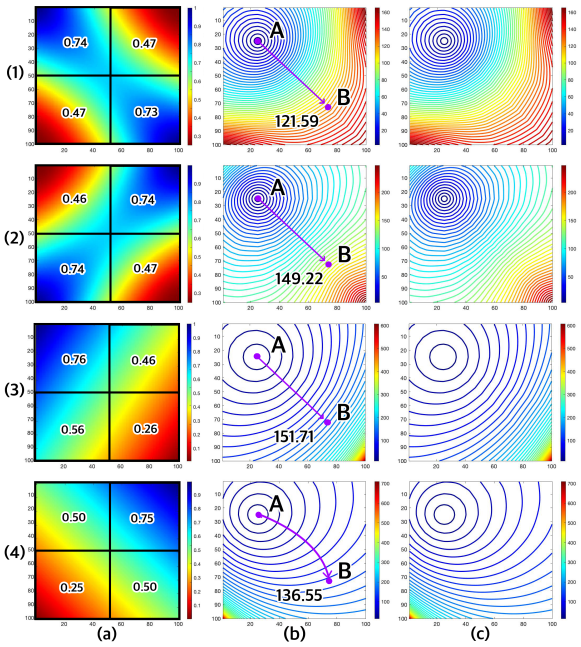


Fig. 12. The rover traversal test. (a) The 100×100 ground-truth matrices, color-coded from easy driving (blue) to hard (red), overlain by the corresponding 2 x 2 test spaces synthesized by averaging each quadrant of the ground-truth. (b) ToA distance contours of E_{24} with the calculated geodesic time from Point A to Point B. (c) ToA produced by RE_{24}. The two sets of contours (b) and (c) are indistinguishable, demonstrating that the two metrics are comparable over nearly-continuous spaces.

A. Methods

Four 100×100 ground-truth matrices were designed in which each cell was assigned a weight, x, ranging from 1.0 (smooth) to 0.0 (impassable) effectively representing the maximum relative velocity of the rover through that cell. Each matrix was then "down-rezzed" to a 2×2 test image matrix in which each element was assigned the arithmetic average of the corresponding ground-truth quadrant. Fig. 12 (a) shows the four 2×2 test matrices written overlay of their respective ground-truth sources. The underlay truth weighted image space is color-coded to denote the traversal difficulty at each pixel, with smooth as blue and impassable as red.

Segmented E_{24} and simple RE_{24} chamfers were used to generate the true ToA contour maps in Fig. 12 (b) and (c) respectively for a hypothetical rover driven across each terrain from Point A to Point B [elements (25, 25) and (75, 75), respectively, in the ground-truth matrices] along a geodesic path. There was no discernible difference between E_{24} and RE_{24} over the ground-truth manifold, as shown in the figure between (b) and (c), because the ground-truth is a smooth and nearly-continuous function, allowing both chamfers to behave nicely. Thus the common ground-truth ToA was used. The ToA estimated from the discontinuous 2×2 matrices were compared to the ground-truth ToA between A and B. We calculate the error in the ToA estimates by subtracting the ground-truth ToA from the estimate ToA. The results are shown in Table III. Because RE_{24} has four identities, we will will consider the mean in the cases where the identities provide differing estimates. Note that positive values indicate a conservative overestimate of the time needed to traverse the area, while negative values reflect underestimated times. Just as in the checkerboard example, this thought experiment is designed to use the choppiness in the weighted image space to reveal the underlying differences between the estimations produced by the two chamfering techniques.

B. Results

TABLE III
Pre-mission Errors
from 1/50th scaled, averaged 2×2 weighted image

	E_{24}	$RE_{24}, 1$	$RE_{24}, 2$	$RE_{24}, 3$	$RE_{24}, 4$
Case 1	-25.69	2.51	2.51	2.17	2.17
Case 2	2.79	-25.45	-25.45	-25.09	-25.09
Case 3	33.77	37.75	23.75	-14.28	0.28
Case 4	4.87	-18.70	39.89	39.89	-18.70

C. Discussion

RE_{24} and E_{24} generate indistinguishable ToA contours over large, smooth manifolds but the differences between these two methods became clear in the small windows of choppy data. Although RE_{24} does not unilaterally solve the corner-cutting problem from E_{24}, it does produce better estimates under some circumstances. This might be adapted by combining the raw estimates from the two techniques.

Cases 1 and 2 illustrate the corner-cutting problem, which might represent moving along a ridge line and crossing a saddle, respectively. The best course of action in Case 1 is for the rover to drive a straight diagonal along the easy path, avoiding the rougher terrain to either side. However, the true

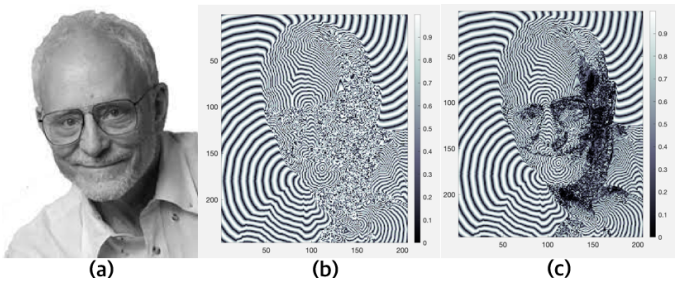


Fig. 13. (a) Original image of Elwyn Berlekamp. (b) Using E_{24}, the absolute value of a modular distance transform (sinusoid of frequency $(2-\sqrt{2})/8$) over the weighted image space from an interior point. (c) Using RE_{24}, the same technique as (b) is applied but we also average the distances provided by each of the four identities, all originating from the same point. The averaging of RE_{24} produces a kind of low-pass filter, creating image coherence in the high frequency (low traversibility) regions of the image. Notably this is accomplished without the use of differential equations or Fourier transforms.

traversibility in the middle of the path is less than 0.74. Thus E_{24} underestimated the timing due to averaging with a corner-cutting assumption. In Case 2, the rover must cross a band of relatively easy terrain midway to its destination. Here, all identities of RE_{24} underestimated the timing by a wide margin, because it estimates the optimal path as having a larger-than-actual component of the 0.74 region.

In Case 3, one of RE_{24}'s four identities provided the most accurate prediction; however, the choice of starting node was also extremely important as identity 1 produced the worst estimate. However, the mean of all four identities was roughly three times more accurate than E_{24}.

In Case 4, E_{24} performed best with a straight-line estimate despite the curved geodesic shortest path in the truth source, as seen by the contours in Fig. 12 (4)-(b). Again, RE_{24} seems to be over-weighting the portion of path contributed by the 0.75 region.

IX. Conclusion

RE-grid graphs should continue to be developed for their wide array of potential applications. The curious case of the weighted checkerboard shows that these little processors have a lot to offer image processing when it comes to describing and shaping space with their unique flavor of discrete geometry. While a more robust characterization of performance over weighted image spaces is necessary, the data here shows that RE-grid graphs provide sufficiently better estimates in some cases over E-grid graphs. These might be refined by composing the estimates of the four identities. RE-grid graphs are not a perfect predictor in low resolution space and therefore should not unilaterally replace E-grid graphs, but they may provide better estimates when used with them.

In closing, Fig. 13 demonstrates how the estimates from the four identities in RE_{24} can be averaged to create image coherence by effectively low-pass filtering the high-frequency areas of a modular distance transform. This echos the insights from Cases 3 and 4 of the rover traversal, where we see the identities behave like a 4-directional perturbation of center-to-center distances. Thus, these identities of RE-grid graphs can be used to imply differential forms as an inherent property of the RE-grid graph. Future work will exploit this chamfering network transform in the areas of morphology, watershed segmentation, wavelet transforms, and halftoning as these graphs seem to be providing a differential element necessary to approximating eikonal equations [2]–[4], [35]–[38]. By extension, RE-grid graphs may also improve neural network architecture with the intrinsic differential behaviors that ultimately inform many image processing techniques [39]–[41]. Additional imaging applications arise from scalar manipulations of the canonical sets of edges *internal* to each pixel. These induce anisotropic image deformations fields which can be used in image skeletonization, dedritic shaping, space-filling curves, and a new class of cellular automata [14], [42]–[44].

Acknowledgments

The authors would like to thank Joseph Rispoli, Colin Wright, Amit Patel, Jason Williford, Eric Moorhouse, Brad Fitzgerald, and Elwyn Berlekamp for helpful discussions. Funding was made possible by the National Science Foundation through the Graduate Research Fellowship Program.

References

[1] A. Rosenfeld and A. C. Kak, *Digital Picture Processing: Volume 1 and 2*, 2nd ed., ser. Computer Science and Applied Mathematics. Orlando, FL: Academic Press, 1982.

[2] A. C. Bovik, *Handbook of Image and Video Processing (Communications, Networking and Multimedia)*. USA: Academic Press, Inc., 2005.

[3] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996. [Online]. Available: https://www.pnas.org/content/93/4/1591

[4] P. Maragos, "Differential morphology and image processing," *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 922–937, 1996.

[5] M. Akmal Butt and P. Maragos, "Optimum design of chamfer distance transforms," *IEEE Transactions on Image Processing*, vol. 7, no. 10, pp. 1477–1484, 1998.

[6] A. Rosenfeld and J. Pfaltz, "Distance functions on digital pictures," *Pattern Recognition*, vol. 1, no. 1, pp. 33 – 61, 1968.

[7] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227 – 248, 1980.

[8] D. G. Bailey, "An efficient euclidean distance transform," in *Combinatorial Image Analysis*, R. Klette and J. Žunić, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 394–408.

[9] G. Borgefors, "Distance transformations in arbitrary dimensions," *Computer Vision, Graphics, and Image Processing*, vol. 27, no. 3, pp. 321 – 345, 1984.

[10] ——, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344 – 371, 1986.

[11] F. Leymarie and M. Levine, "Fast raster scan distance propagation on the discrete rectangular lattice," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 84 – 94, 1992.

[12] S. Marchand-Maillet and Y. M. Sharaiha, "Euclidean ordering via chamfer distance calculations," *Computer Vision and Image Understanding*, vol. 73, no. 3, pp. 404 – 413, 1999.

[13] A. Hajdu, H. Lajos, and R. Tijdeman, "Approximation of the euclidean distance by chamfer distances," *Acta Cybernetica*, vol. 20, 01 2012.

[14] C.-S. Chiang, "The euclidean distance transform," Ph.D. dissertation, Purdue University, USA, 1992.

[15] C. Fouard and G. Malandain, "3-d chamfer distances and norms in anisotropic grids," *Image and Vision Computing*, vol. 23, no. 2, pp. 143 – 158, 2005, discrete Geometry for Computer Imagery.

[16] A. Patel. (2013) Grids and Graphs. [Online]. Available: https://www.redblobgames.com/pathfinding/grids/graphs.html

[17] R. H. Hammack, W. Imrich, and S. Klavžar, *Handbook of Product Graphs*. CRC Press, 2016.

[18] J. Case, D. S. Rajan, and A. M. Shende, "Lattice computers for approximating euclidean space," *Association for Computing Machinery*, vol. 48, no. 1, p. 110–144, Jan. 2001.

[19] O. Goldreich, *Basic Facts about Expander Graphs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 451–464.

[20] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 4, pp. 439–561, 2006.

[21] A. Donno, "Replacement and zig-zag products, cayley graphs and lamplighter random walk," *International Journal of Group Theory*, vol. 2, pp. 11–35, 2013.

[22] O. Lézoray and L. Grady, *Image Processing and Analysis with Graphs: Theory and Practice*. CRC Press, 2012.

[23] J. H. Conway and N. J. A. Sloane, *Sphere packings, Lattices, and Groups*. Springer, 1999.

[24] J. H. Conway, *Book of Numbers*. Springer, 2012.

[25] E. W. Weisstein. (2021) "King Graph." from mathworld–a wolfram web resource. [Online]. Available: https://mathworld.wolfram.com/KingGraph.html

[26] ——. (2021) "Knight Graph." from mathworld–a wolfram web resource. [Online]. Available: https://mathworld.wolfram.com/KnightGraph.html

[27] J. Arz, D. Luxen, and P. Sanders, "Transit node routing reconsidered," in *Experimental Algorithms*, V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 55–66.

[28] H. Bast, S. Funke, D. Matijević, C. Demetrescu, A. Goldberg, and D. Johnson, "Transit: Ultrafast shortest-path queries with linear-time preprocessing," *9th DIMACS Implementation Challenge — Shortest Path, DIMACS (2006)*, 01 2006.

[29] J. Fliege, H.-D. Qi, and N. Xiu, "Euclidean distance matrix optimization for sensor network localization," in *Cooperative Localization and Navigation: Theory, Research and Practice*. CRC, 2019. [Online]. Available: https://eprints.soton.ac.uk/426671/

[30] H.-D. Qi and X. Yuan, "Computing the nearest euclidean distance matrix with low embedding dimensions," *Mathematical Programming*, vol. 147, pp. 351–389, November 2014. [Online]. Available: https://eprints.soton.ac.uk/361847/

[31] M. Fredman and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," in *25th Annual Symposium onFoundations of Computer Science, 1984.*, 1984, pp. 338–346.

[32] A. Salazar, D. Kaba, Y. Li, and X. Liu, "Segmentation of blood vessels and optic disc in retinal images." *IEEE journal of biomedical and health informatics*, vol. 18, 01 2014.

[33] J.-S. Kim and K.-S. Hong, "A new graph cut-based multiple active contour algorithm without initial contours and seed points," *Machine Vision Applications*, vol. 19, pp. 181–193, 05 2008.

[34] V. Kolmogorov and Y. Boykov, "What metrics can be approximated by geo-cuts, or global optimization of length/area and flux," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, 2005, pp. 564–571 Vol. 1.

[35] J. S. Walker, *A primer on wavelets and their scientific applications*. Chapman & Hall/CRC, 2008.

[36] P. S. Addison, *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering medicine and finance*. CRC Press, 2020.

[37] B. Cancela and A. Alonso-Betanzos, "Wavefront marching methods: a unified algorithm to solve eikonal and static hamilton-jacobi equations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[38] M. Algarni and G. Sundaramoorthi, "Surfcut: Surfaces of minimal paths from topological structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 3, pp. 726–739, 2019.

[39] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, "Eikonet: Solving the eikonal equation with deep neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–12, 2020.

[40] M. Lichtenstein, G. Pai, and R. Kimmel, "Deep eikonal solvers," *Scale Space and Variational Methods in Computer Vision*, pp. 38–50, 06 2019.

[41] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[42] S. A. Mihelic, W. A. Sikora, A. M. Hassan, M. R. Williamson, T. A. Jones, and A. K. Dunn, "Segmentation-less, automated vascular vectorization robustly extracts neurovascular network statistics from in vivo two-photon images," *bioRxiv*, 2020.

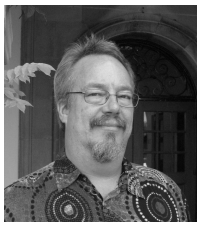
[43] L. Xu and D. Mould, "Modeling dendritic shapes - using path planning," in *GRAPP*, 2007.

[44] S. Wolfram, *A New Kind of Science*. Wolfram Media, 2002. [Online]. Available: https://www.wolframscience.com

T. Arthur Terlep T. Arthur Terlep (M'21) graduated Phi Beta Kappa, Magna Cum Laude, and A&S Top 20 Graduate from the University of Wyoming with a B.S. in Mathematics and Honors in 2011 supported by the McNair scholars program. He served in the United States Air Force as a science officer, operations research analyst for the MALD-J and QF-16 systems at AFOTEC DET 2, Eglin AFB, FL, USA from 2012 to 2014 and as a flight test analyst for the ALCM system at the 49th TES, ACC, Barksdale AFB, LA, USA from 2014 to 2017. He is currently a Graduate Research Fellow under the National Science Foundation pursuing a PhD in Electrical and Computer Engineering at Purdue University, West Lafayette, IN, USA with research interests in digital signals processing, graph theory, and biomedical imaging.

Mark R. Bell Mark R. Bell (F'11) received the B.S. degree in electrical engineering from California State University, Long Beach, CA, USA, in 1981 and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 1982 and 1988, respectively. From 1979 to 1989, he was employed by Hughes Aircraft Company, Fullerton, CA, USA. From 1981 to 1989, he was affiliated with the Radar Systems Laboratory at Hughes, where he held the positions of Member of the Technical Staff and Staff Engineer and worked in the areas of radar signal processing, electromagnetic scattering, radar target identification, and radar systems analysis. While at Caltech, he was a Howard Hughes Doctoral Fellow. Since 1989, he has been on the Faculty of Purdue University, West Lafayette, IN, USA, where he is a Professor in the School of Electrical and Computer Engineering. His research interests include the areas of radar and sonar, information theory, detection and estimation theory, and communications.

Thomas M. Talavage Thomas M. Talavage (M'89–SM'11) received the B.S. and M.S. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1992 and 1993, respectively, and the Ph.D. degree in speech and hearing sciences from the Harvard–MIT Division of Health Sciences and Technology, Cambridge, MA, USA, in 1998. He joined the faculty of Purdue University, West Lafayette, IN, USA, in 1998, and was a Professor with the School of Electrical and Computer Engineering and the Weldon School of Biomedical Engineering, and from 2007-2020 served as Founding Co-Director of the Purdue MRI Facility, Weldon School of Biomedical Engineering, West Lafayette, IN, USA. He has been recognized nationally for his work since 2009 as one of the core PIs of the Purdue Neurotrauma Group. In 2020 he joined the Department of Biomedical Engineering at the University of Cincinnati and is currently serving as the department head.



Douglas L. Smith Douglas L. Smith received the B.S. degree in chemistry from Waynesburg University, Waynesburg, PA, USA, in 1980, and the graduate certificate in science communication from the University of California, Santa Cruz, Santa Cruz, CA, USA in 1982. He is retired from the California Institute of Technology (Caltech), Pasadena, CA, USA, where he has variously been a Lecturer in Engineering (2013–17, 2019) and writer, managing editor, and editor of the Caltech research magazine, *Engineering & Science* (1987–2012) for which he won six international Circle of Excellence Silver Medals from the Council for the Advancement and Support of Education (CASE). He is a member of the National Association of Science Writers and is currently researching a book on Caltech history.