Pattern Recognition Letters 168 (2023) 1–7

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

An efficient region expansion algorithm for regular triangulated meshes

Brian Ondov[a,*], Hanan Samet[b]

[a] National Library of Medicine, National Institutes of Health, Bethesda, MD, USA
[b] Department of Computer Science, University of Maryland, College Park, MD, USA

A R T I C L E I N F O

Article history:
Received 14 November 2021
Revised 28 May 2022
Accepted 13 February 2023
Available online 15 February 2023

Edited by Prof. S. Sarkar

MSC:
41A05
41A10
65D05
65D17

Keywords:
Geographic information systems
Quadtrees
Triangulated meshes

A B S T R A C T

Region expansion—the growth of regions to include all points within a certain distance of their perimeters—is a basic, widely applicable operation, but is expensive to perform exactly. It has been shown that, if the solution is approximated by relaxing the distance metric to the L_∞-norm, efficiency can be greatly improved using properties of quadtrees. The method as described, however, requires the quadtrees to be square, both for the metric and the particular details of the algorithm. In some cases, such as spherical surface approximation, it is desirable for the quadtree nodes to be triangular instead. In this work, we thus describe an adaptation of the L_∞-norm metric and the previously described algorithm to allow efficient approximation of region expansion in images represented as regular triangulated meshes. Like the original method for square quadtrees, our algorithm achieves sublinear time with respect to expansion radius.

1. Introduction

Region expansion (also called image dilation) is the process of growing all instances of a specific class of region—e.g. black pixels in a binary image—to include all points within a certain distance metric to the original regions. Applications for this basic operation include the creation of buffers or corridors in Geographic Information Systems (GIS) and pick operations for user interfaces, with further uses in photo editing, computer vision, and robot motion planning. A naïve, though commonly used, method for region expansion in raster images is to apply a square or disc kernel, which effectively grows each pixel independently, using the union as the result. Though speed can be gained by leveraging matrix operations with dedicated graphics hardware, this method still uses significant computing resources and does not scale well due to a quadratic dependence on the expansion radius. Ang et al., and others, have been able to instead achieve constant complexity when the distance metric is the L_∞-norm (also known as the "chessboard" distance) by exploiting the dimension-reducing properties of quadtrees [1–3,10,15,17,19]. Because of the requirement of the L_∞-norm, however, the method as published pertains only to square quadtrees and not those of other shapes. Triangular quadtrees, for example, are commonly used in geodesic approximations [7,8,20], which could benefit from optimizations in region expansion (but see [9] for applications in higher dimensions). We thus report the adaptation of the general principles of the Ang et al. method to triangular quadtrees and show that similar gains in efficiency are possible by using a hexagonal norm as the distance metric.

2. Methods

The components of the algorithm are fairly straightforward adaptations of the original L_∞-norm algorithm as defined by Ang et al., with some exceptions. Throughout, we assume that our image is stored in a regular triangular quadtree, and, without loss of generality, that it is binary, comprising only black and white triangles (more generally, black triangles could represent, for example, a particular color, or all those triangles that meet a certain luminance threshold, or that match a query of associated spatial data).

Note that, unlike square quadtrees, triangular quadtrees contain nodes with two orientations, which we will refer to as *tip-up* and *tip-down*. Since these two orientations are symmetrical, we will depict below only the *tip-up* case; operations for *tip-down* nodes are

* Corresponding author.
E-mail address: brian.ondov@nih.gov (B. Ondov).

https://doi.org/10.1016/j.patrec.2023.02.014
0167-8655/© 2023 Elsevier B.V. All rights reserved.

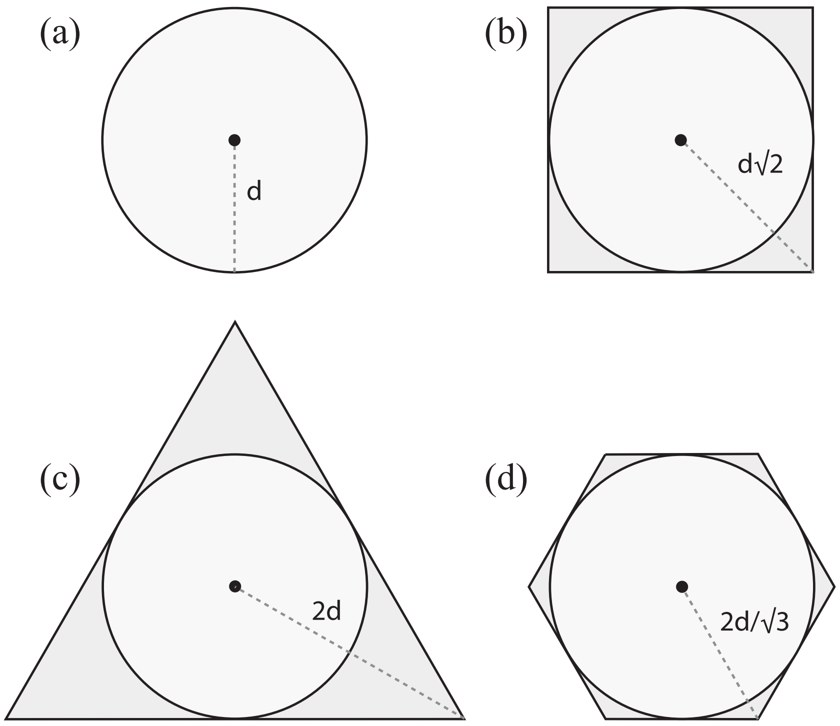

Fig. 1. Loci of points for different shapes. In (a), the locus of points for euclidean (L_2) distance of d is a circle. In (b), the locus of points within the LL_∞-norm, as in Ang et al., is a square. A distance metric that creates a triangle (c), would enable triangular mesh optimization but would be a poor approximation of the euclidean distance. Our proposed metric creates a hexagonal locus of points (d), which is a much better approximation.

simply mirror images, reflected vertically.

$$D_{hex}(\Delta_x, \Delta_y) = \max\left(|\Delta_y|, \frac{|\Delta_y| + \sqrt{3}|\Delta_x|}{2}\right) \tag{1}$$

2.1. Hexagonal distance metric

We start from the notion that the (regular) hexagon is the ideal kernel shape for region expansion in a triangular mesh (Fig. 1). This is because, though we desire to use the shape of the mesh to gain efficiency, the triangle itself is not a good approximation of a circle (Fig. 1c). The hexagon, however, is a much better approximation (Fig. 1d), with a 10% over-estimation of area (compared to 65% for a triangle, or 27% for a square) and a 13% under-estimation of distance at the corners of the kernel (compared to 50% for a triangle, or 29% for a square). Further, since the hexagon decomposes into equilateral triangles, it can easily be mapped onto a regular triangulated mesh without weighted distances such as [11].

A distance metric D_{hex} producing a hexagon as its unit circle can be formally defined by Eq. (1). However, we can also describe this metric in terms of the L_∞-norm by (conceptually) adding a dimension to the expansion space and rotating it by $\pi/2$ radians on two axes [14]. We can then expand by the L_∞-norm in three dimensions, producing a cube for each point. When rotated back to the original orientation, and projected back to two dimensions, the locus of all points within the L_∞-norm of each original point is a hexagon, as illustrated for a single point in Fig. 2. Note that this metaphor is used only for the formal metric definition; in practice, we do not need to perform any rotations or dimensional increases. Instead, we simply count by triangle edges, which correspond to distance maxima (the corners of the cube), or by triangle altitudes (segments perpendicular to one edge and incident to the opposite vertex), which correspond to minima (the midpoints of the cube edges). As with the standard L_∞-norm, the result is an underestimation of the euclidean distance, or L_2-norm, producing an overestimated area of expansion. When all points within this metric are included for a single triangle, the result is not a regular hexagon, but a truncated equilateral triangle, as shown in Fig. 3.

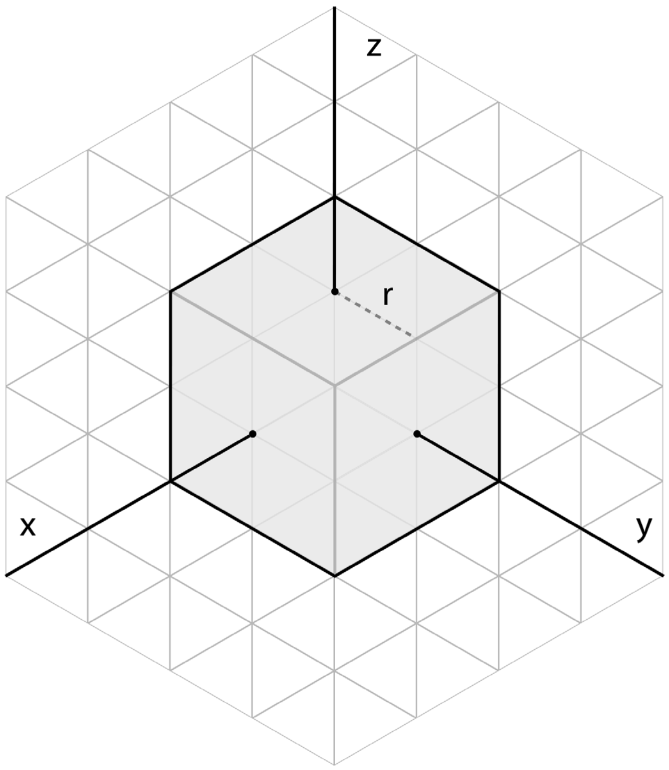

Fig. 2. The locus of points within the three-dimensional L_∞-norm for distance r is a cube of width $2r$. If obliquely projected into two dimensions, the resulting shape is a hexagon whose corners are $2r$ from the original point, and whose edges are least $r\sqrt{3}$ from the point.

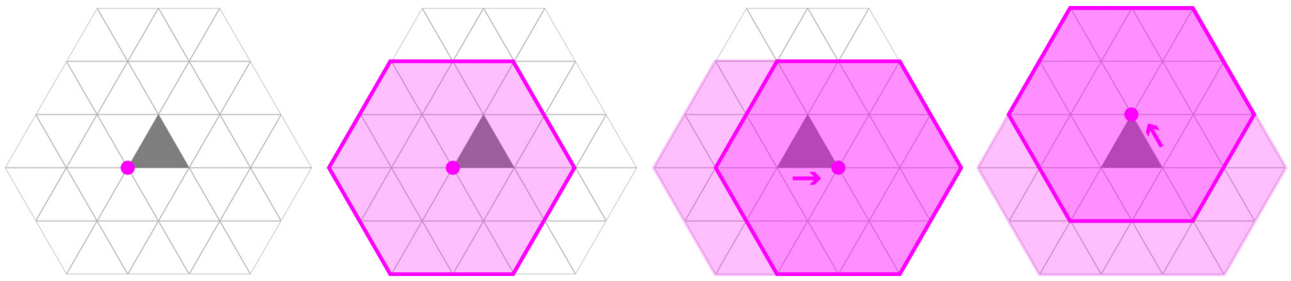
Fig. 3. The perimeter resulting from expanding a single black block of a triangular mesh with a hexagonal kernel becomes a truncated equilateral triangle. Here the resulting shape is illustrated as if generated by moving a hexagonal kernel around the perimeter of the black block.

2.2. Merging clusters

The first key observation of Ang et al. is that a "gray" quadtree node (i.e. one that is split and therefore has both black and white children) whose edge length is less than or equal to $R+1$, where R is the expansion radius, can be directly set to black. This is because all white children of that node must be within the radius of expansion of some black sibling according to the distance metric. Such nodes are called *merging clusters*, and are defined as internal quadtree nodes of edge size

$$W(R) = 2^r, 2^r \leq R+1 \; < \; 2^{r+1}$$

for expanding by radius R. In other words, the merging cluster node size is the largest integral power of 2 less than or equal to $R+1$. This operation is identical for the triangular case as for the square case (Fig. 4).

2.3. Vertex sets and frontiers

With the triangular merging cluster set to black, we next must expand outward from this node based on the positions of its black internal nodes (descendants). The second key observation of Ang et al. is that the expansion from a merging cluster depends only on certain vertices from black child nodes within it, and that these vertices can be further subdivided to correspond to different directions of expansion. Ang et al. define two different cases for expansion directions. The first case is expanding along orthogonal corridors, projecting N, E, S, and W of a square node. The "frontier," or leading edge of growth, in these corridors will simply be a line segment. The position of this frontier is only dependent on the black node vertex closest to the relevant edge of the merging cluster, with ties broken arbitrarily. The second case is expanding in diagonal corridors in between the orthogonal corridors, which project from the corners of a square merging cluster. The frontiers of growth in these regions will be staircase-like, and depend on groups of vertices called *vertex sets* (Fig. 6). Vertex sets are filtered using the *opposite quadrant* operation, which, for a given vertex and expansion direction, returns whether another vertex is in the quadrant opposite the expansion direction. For example, when expanding SW, any vertex in the closed NW quadrant of any other vertex can be excluded from the set, since that other vertex will push the frontier at least as far in the SW direction (See Fig. 2 and Table 1 of Ang et al.). For triangles, the two cases are analogous—that in which expansion will fall on parallel lines, and that in which the expansion must represent the more complex patterns of the internal nodes, taking on staircase-like patterns. However, for triangles, there are six directions for each case, rather than the four, owing to the hexagonal kernel shape, and thus six vertex sets. Vertex sets are used for expanding in the magenta regions. The triangular *opposite quadrant* operation differs from the rectilinear version of Ang et al. in that we must choose two of the three available axes to define the quadrants. These will be the two axes that are not parallel to the axis of expansion in the relevant region. Similarly to the rectilinear case, in the complex frontier regions (shown in magenta in Fig. 5), a projection of the vertex set for the appropriate quadrant will always form a monotonic, staircase-like boundary, allowing a simple loop to fill in the region without redundancy.

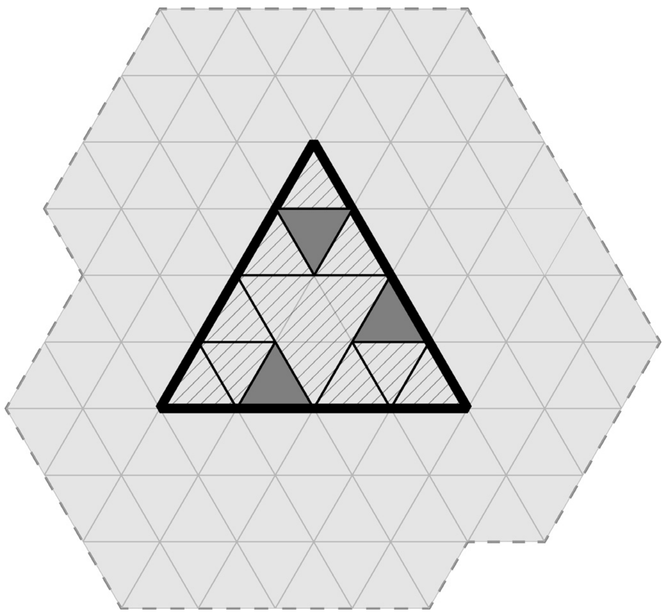
Fig. 4. A merging cluster (shown within heavy outline) of size 4 and its expanded region (dashed line) for radius 3. All the "white" blocks (shown hashed) within the cluster must be within 3 units of one of the "black" blocks (dark gray); thus the entire parent node (heavily outlined triangle) can be set to black, reducing complexity.

2.4. Vertex set bounds

Here we bound the sizes of the vertex sets based on the merging cluster size. Note that one vertex of a black node may appear in multiple vertex sets, in which case we consider it to be multiple distinct items when unioning the sets.

Lemma 1. *The maximum number of vertices in a vertex set is $W(R)/2$, and this bound is attainable.*

Proof. Consider the two axes of an *opposite quadrant* operation (e.g. the E-W and NE-SW axes for the SE expansion direction) to be an affine transformation of a Cartesian grid. Clearly, no two vertices can share the same x or y value, due to the closed nature of the *opposite quadrant* operation. Now consider an axis orthogonal to the expansion direction, which corresponds to altitudes of the

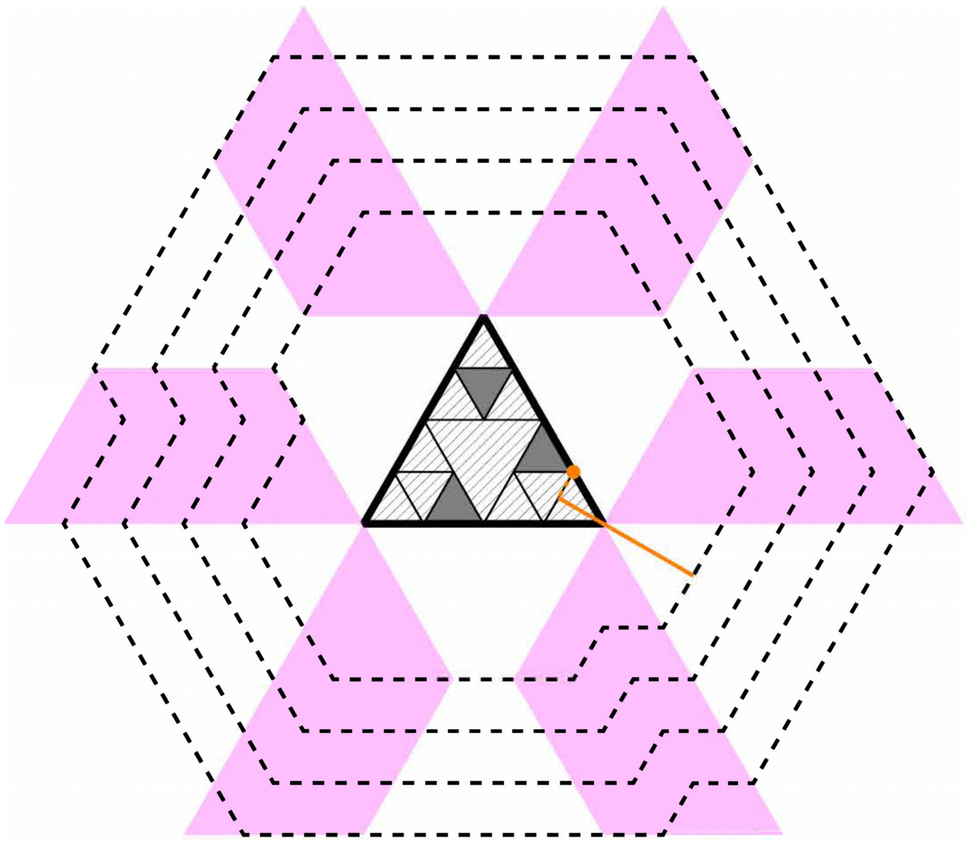

Fig. 5. The black blocks of the merging cluster (center triangle) have been expanded (dotted lines) to all possible distances (3,4,5,6) for a merging cluster of size 4. Only the trapezoidal regions in magenta will have nontrivial frontiers. These frontiers will have staircase patterns and must be monotonic. The triangular and hourglass-shaped regions in between the magenta regions will have trivial frontiers parallel to a gridline. The orange line illustrates offsetting within one of these trivial regions using a vertex set, which in this case is the single orange vertex.

triangular nodes. Because the vertices in the set cannot share x or y value, each one adds at least two altitudes on the orthogonal axis (dashed line in Fig. 7a). By definition of the merging cluster and distance metric D_{hex}, the altitude of the merging cluster is $W(R)$, and thus the maximum number of vertices in a vertex set for a given direction is $W(R)/2$. Figure 7a is an example of attaining this bound. □

Lemma 2. *The size of the union of any two adjacent vertex sets (i.e. in directions that share one opposite quadrant axis) is bounded by* $W(R)/2+1$*, and this bound is attainable.*

Proof. Consider adjacent vertex sets V_{SW} and V_{SE}, for the directions SW and SE, respectively. Let v'_{SW} denote the easternmost vertex in V_{SW}. Now consider a vertex v'_{SE} in V_{SE} that is to the west of v'_{SW}. This vertex must be to the south of v'_{SW}, or it would have been excluded by a vertex of the same black node that donated v'_{SW}. However, v'_{SE} must come from a black node that would have excluded v'_{SW} from V_{SW}. Thus, vertex v'_{SE} cannot exist, and all vertices in V_{SE} are to the east of those in V_{SW}. Lemma 1 combined with the allowance that a single black node can contribute to both of two adjacent vertex sets achieves the bound of $W(R)/2+1$. It is easy to find examples that attain this bound (e.g. Fig. 7a) By symmetry, the same holds for any two adjacent vertex sets. □

Claim The size of the union V_M of all vertex sets for a merging cluster is bounded as a function $V_{\max}(R)$ of expansion radius R by

$$V_{\max}(R) = \frac{3}{2}W(R) + 3. \tag{2}$$

Proof: Any two adjacent vertex sets can achieve a total of $W(R)/2+1$ (Lemma 2). The six vertex sets needed for expanding from a given merging cluster are arranged hexagonally, each with two neighbors. Any two vertex set sizes that add to $W(R)/2+1$ could be tiled around this hexagon in an alternating pattern. Adding any additional vertex to such a tiling would violate Lemma 2. Thus, the bound is $3[W(R)/2+1] = \frac{3}{2}W(R)+3$, leading to Eq. (2).

Attainability: It is as of yet unknown whether the bound in Eq. (2) is attainable. Here we explore configurations of vertex sets that approach the theoretical worst case. We will discuss two strategies for maximizing the size of the union V_M of all vertex sets, in which we maximize either (1) two opposite vertex sets or (2) three regularly spaced vertex sets (Fig. 7).

Strategy (1) only allows two of six sets (e.g. NW and SE in the example in Fig. 7a) to achieve the individual maximum of $W(R)/2$ (Lemma 1), because each set with this maximum needs to have both of its neighbors restricted to one vertex as a consequence of Lemma 2. The bound for this strategy is $W(R)/2*2+1*4 = W(R)+4$

Strategy (2) is equivalent to decomposing the cluster into 4 equal triangles, and using the altitudes of the 3 outer triangles to reach maxima of $W(R)/4$ for 3 directions (e.g. W, SE, and NE in the example in Fig. 7b). The other 3 interstitial directions can only have 2 in this case, creating a bound of $W(R)/4*3+2*3 = \frac{3}{4}W(R)+6$.

The two strategies intersect at $W(R)=8$, with strategy (1) dominant when $W(R)>8$ and strategy (2) dominant when $W(R)<8$.

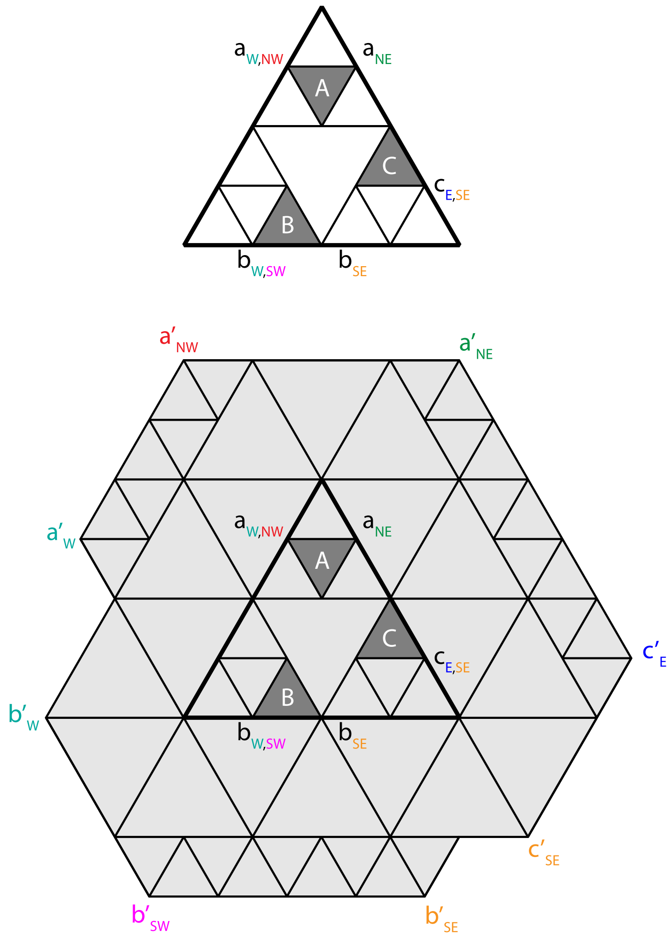

Fig. 6. Top, the vertex sets for the six directions (E,NE,NW,W,SW,SE) of an example *tip-up* node. Bottom, the result of expanding by 3 units, showing the frontier vertices corresponding to the vertex sets.

The currently proven attainable maximum $V'_{\max}(R)$ is thus the piecewise function:

$$V'_{\max}(R) = \begin{cases} W(R) + 4, & W(R) \geq 8 \\ \frac{3}{4}W(R) + 6, & W(R) < 8 \end{cases}.$$

3. Empirical results

We examine the performance of the expansion algorithm empirically with an implementation in c++. For data, we use a bitmap representation of a map overlay depicting the floodplain of the Russian River in northern California, which has been used previously to benchmark quadtree operations [3,16,18]. The bitmap was converted to a binary regular triangulated mesh with the same number of rows (such that the altitude of each triangle corresponded to the height of an original square pixel), using a nearest neighbor approximation. A triangular quadtree was then built using the mesh triangles as leaf nodes and merging hierarchically. The original and expanded meshes with quadtrees overlaid can be seen in Fig. 8.

As a baseline, we also implemented the naïve, kernel-based algorithm described earlier. For each black triangle of the grid, this amounts to setting all surrounding triangles to black within the shape of a truncated triangle (or irregular hexagon) similar to that shown in the rightmost panel of Fig. 3. While it is not highly optimized and does not utilize dedicated hardware, this implementation should faithfully represent complexity with respect to expansion radius and thus provides a good baseline against which to compare the new algorithm. Both implementations were tested by growing the same region with various radii. As expected, the naïve algorithm exhibits quadratic growth in time with increases in radius, while the new algorithm is sublinear (Fig. 9). Code and data for this experiment are available at https://github.com/ondovb/triangle.

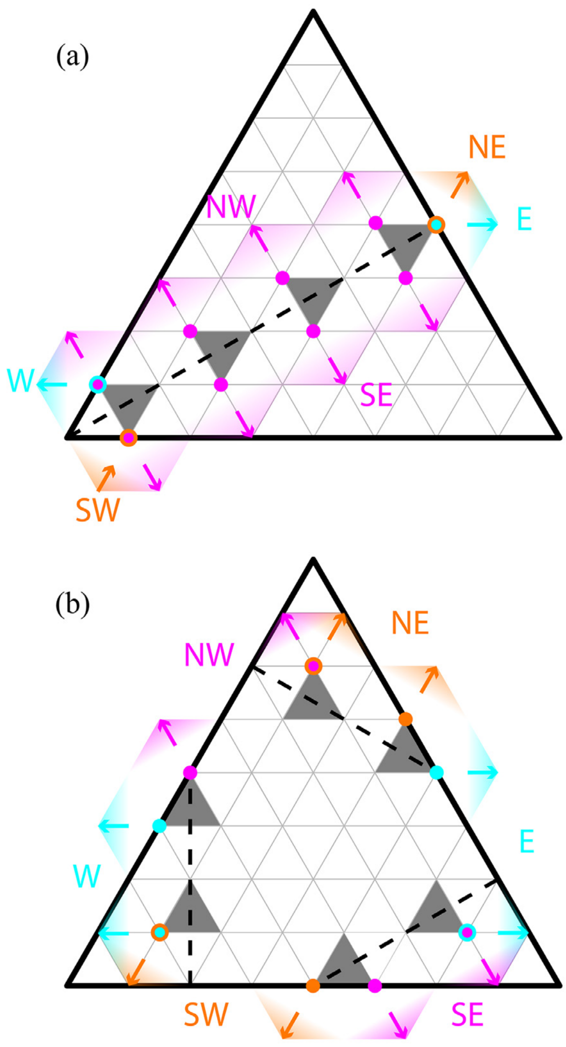

Fig. 7. The two types of worst case arrangements for vertex set complexity. The arrangement in (a) produces the maximum vertex set size for $W(R) \geq 8$, while the arrangement in (b) produces the maximum vertex set size for $W(R) \leq 8$. The two arrangements produce the same size vertex set (12), when $W(R) = 8$, as shown here.

4. Discussion

This work shows that efficient methods for approximating region expansion can be adapted from square quadtrees to triangular quadtrees, by adopting a norm whose locus of points is hexagonal. While the implementation tested already achieved sublinear time, additional gains can likely be made by avoiding expansion into neighboring blocks that are already black, which can be queried quickly using a linear quadtree implementation [4,5,12] or coloring [13]. As the original square quadtree expansion algorithm was able to achieve constant time, we expect this to be the theoretical bound of improving the triangular algorithm as well.

While many GIS applications use square quadtrees, one potential application of this work is in systems for querying astronom-

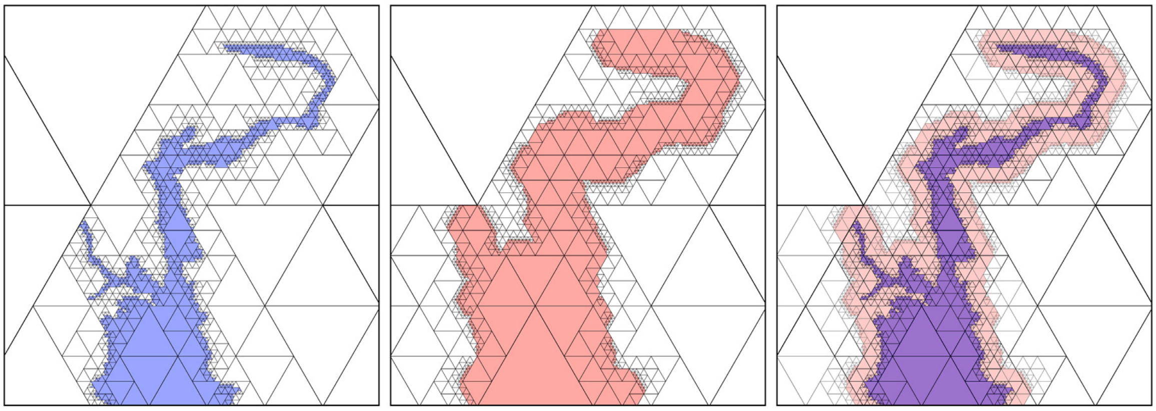

Fig. 8. Left, a binary image stored in a triangular quadtree. Center, the same image expanded by 5 units. Right, a composite of the original and expanded images. Note that all colored regions correspond to "black" nodes in the quadtree; we vary the color here for the sake of compositing.

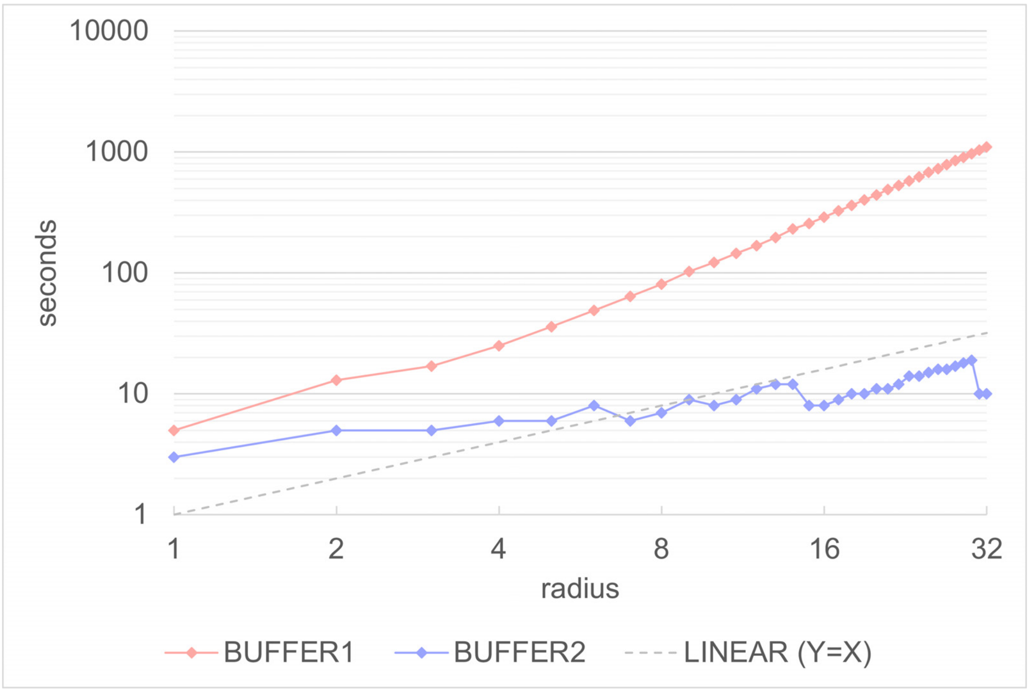

Fig. 9. Runtime comparison of a naïve algorithm (BUFFER1, red) and the adapted quadtree algorithm (BUFFER2, blue). Both were implemented in c++ and used to expand the mesh in Fig. 8, repeating 10,000 times for each radius, running on a 2.3Ghz Intel core i5 processor. BUFFER1 exhibits quadratic complexity with respect to radius, while BUFFER2 is sublinear. The x-axis is log base 2 scale; the y-axis is log base 10.

ical footprint databases [6], which use triangular quadtrees to index regions of the night sky for searches and boolean operations. These systems already provide region expansion functionality but could benefit from optimization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was sponsored in part by the NSF under Grants IIS-18-16889, IIS-20-41415, and IIS-21-14451 and by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health.

References

[1] A. Amir, A. Efrat, P. Indyk, H. Samet, Efficient regular data structures and algorithms for dilation, location, and proximity problems, Algorithmica 30 (2) (2001) 164–187.
[2] C. Ang, Analysis and applications of hierarchical data structures, University of Maryland, College Park, MD, 1989 Ph.D. thesis.
[3] C.-H. Ang, H. Samet, C.A. Shaffer, A new region expansion for quadtrees, IEEE Trans. Pattern Anal. Mach. Intell. 12 (7) (1990) 682–686.
[4] F.W. Burton, V.J. Kollias, J.Y.G. Kollias, Expected and worst-case storage requirements for quadtrees, Pattern Recognit. Lett. 3 (2) (1985) 131–135.
[5] P.-M. Chen, Variant code transformations for linear quadtrees, Pattern Recognit. Lett. 23 (11) (2002) 1253–1262.
[6] L. Dobos, E. Varga-Verebélyi, E. Verdugo, D. Teyssier, K. Exter, I. Valtchanov, T. Budavári, C. Kiss, The footprint database and web services of the Herschel space observatory, Exp. Astron. 42 (2) (2016) 139–164.
[7] G. Dutton, Geodesic modelling of planetary relief, Cartographica 21 (1984) 188–207.
[8] G. Fekete, Rendering and managing spherical data with sphere quadtrees, in: Proceedings of the First IEEE Conference on Visualization: Visualization '90, 1990, pp. 176–186.
[9] R. Fellegara, F. Iuricich, L. De Floriani, U. Fugacci, Efficient homology-preserving

simplification of high-dimensional Simplicial shapes, Comput. Graphics Forum 39 (1) (2020) 244–259.
[10] G.M. Hunter, Efficient computation and data structures for graphics, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, 1978 Ph.D. thesis.
[11] G. Kovács, B. Nagy, B. Vizvári, Weighted distances on the truncated hexagonal grid, Pattern Recognit. Lett. 152 (2021) 26–33.
[12] M. Lee, H. Samet, Navigating through triangle meshes implemented as linear quadtrees, ACM Trans. Graph. 19 (2) (2000) 79–121.
[13] S. Menon, P. Gao, T. Smith, Multi-colored quadtrees for GIS: exploiting bit-parallelism for rapid Boolean overlay, Pattern Recognit. Lett. 8 (3) (1988) 171–179.
[14] B. Nagy, A family of triangular grids in digital geometry, in: Proceedings of the 3rd International Symposium on Image and Signal Processing and Analysis, Vol. 1, IEEE, 2003, pp. 101–106.
[15] H. Samet, A quadtree medial axis transform, Commun. ACM 26 (9) (1983) 680–693.
[16] H. Samet, Data structures for quadtree approximation and compression, Commun. ACM 28 (9) (1985) 973–993.
[17] H. Samet, Reconstruction of quadtrees from quadtree medial axis transforms, Comput. Vis. Graph. Image Process. 29 (3) (1985) 311–328.
[18] H. Samet, A. Rosenfeld, C.A. Shaffer, R.E. Webber, Quadtree region representation in cartography: experimental results, IEEE Trans. Syst. Man Cybern. (6) (1983) 1148–1154.
[19] C.A. Shaffer, H. Samet, Algorithm to expand regions represented by linear quadtrees, Image Vis. Comput. 6 (1988) 162–168.
[20] R. Sivan, H. Samet, Algorithms for constructing quadtree surface maps, in: Proceedings of the 5th International Symposium on Spatial Data Handling, Vol. 1, 1992, pp. 361–370.