Extending the Power of Snap Rounding Variants

Eli Packer*

Abstract

Snap Rounding (SR for short) is a method for converting arbitrary-precision arrangements of line segments into a fixed-precision representation. In the previous years two variants of SR were presented: Iterated Snap Rounding (ISR) and Iterated Snap Rounding with Bounded Drift (ISRBD). Their goal was to eliminate an undesirable property that SR possesses. Prior to their appearances, the capabilities of SR were extended in two publications. One showed how to support dynamic rounding (insertion and deletion of line segments). The other extended SR to $\mathbb{R}^3$ (limited to arrangement of line segments). In this work we show how to extend ISR and ISRBD to support these capabilities.

1 Introduction

Snap Rounding (SR for short) is a method for converting geometric input into finite precision representation in order to achieve more robustness in the computations that follow. Most of the concentration of SR has been devoted to arrangements of line segments in $\mathbb{R}^2$. In the previous two decades, several publications have dealt with SR [6, 7, 8, 9, 11, 12].

SR proceeds as follows. The plane is tiled with a grid of unit squares, *pixels*, each centered at a point with integer coordinates. A pixel is *hot* if it contains vertices of the arrangement. Then the output of each line segment is the polygonal chain through the centers of the hot pixels met by it, in the order along the line segment. See Figure 1(a-b) for an illustration.

Halperin and Packer [10] observed that a vertex of the output may be extremely close to a non-incident edge[1], inducing possible robustness problems when handling corresponding queries. They further proposed an augmented procedure, *Iterated Snap Rounding* (ISR, for short), that eliminates this undesirable property. The main idea is that whenever a *link* (a segment in a polygonal chain) passes through a hot pixel h, but not ends at its center, it is further broken and rounded to the center of h. As a result, all links that penetrate a hot pixel, necessarily end at its center. Thus, the distance between any vertex (center of hot pixel) and a non-incident link is at least half a unit.

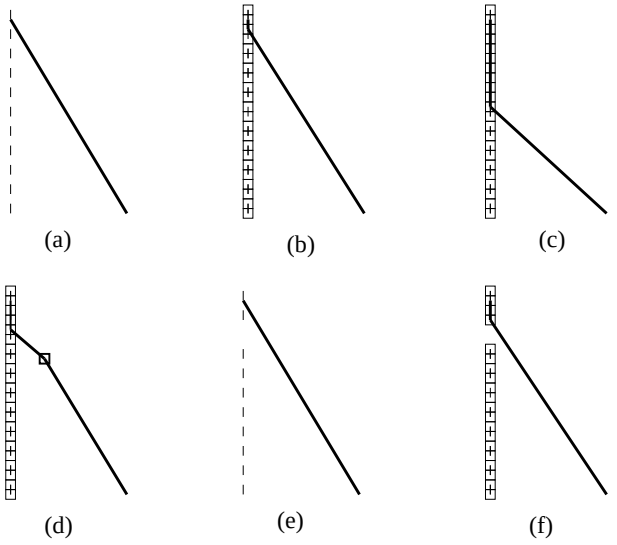

Figure 1: Results of SR, ISR and ISRBD. All the squares in sub figures (b),(c), (d) and (f) are hot pixels. The line segment which demonstrates the difference between the methods is marked in bold. (a) Input (b) SR output (c) ISR output (d) ISRBD output. The effect of deleting/inserting a line segment is illustrated in sub figures (e) and (f). In (e), an input line segment is deleted. As a result, the output of the bold line segment changes, both in ISR and ISRBD. The result (same in ISR and ISRBD) is illustrated in (f). Alternatively, one can view this process as adding a new line segment if the original input is as in (e).

ISR, however, may round line segments very far from their origin[2] [14]. This property motivated another variant, *Iterated Snap Rounding with Bounded Drift* (ISRBD, for short) [15]. Its main idea is to control the deviation of the input by introducing new hot pixels beyond the ones originated by SR. For each input line segment s, ISRBD finds a list of hot pixels Γ that must be the ones that cause s to drift beyond a distance threshold, if at some point it does. It was shown that the centers of these pixels must be located within two thin

*Department of Computer Science, Stony Brook University, epacker@cs.sunysb.edu. Work on this paper has been partially supported by the National Science Foundation (CCF-0431030).

[1] The distance between a vertex and a non-incident edge can be made as small as $1/\sqrt{(2^b-1)^2+1} \approx 2^{-b}$, where b is the number of bits in the representation.

[2] ISR may round the output as far as $\theta(n^2)$ units, where n is the input size. Figure 1(c) illustrates such a large deviation.

stripes, called the *forbidden loci*. Then, by exploiting the weakly-monotone property that the output polygonal chains possess, ISRBD heats pixels in such a way that it is impossible for the hot pixels of Γ to round the output of s. Figure 1 illustrates the differences between the three variants.

De Berg et al. [3] showed that degree-2 vertices that do not correspond to endpoints can be safely removed from the output while maintaining the nice properties of SR. Packer [15] adapted a similar routine in ISRBD. In this routine, vertices that correspond to hot pixels that actually constrain the deviation of the line segments beyond some threshold are not removed. He also showed that this removal may even be crucial to asymptotically decrease the complexity of the output.

Some of the papers that were devoted to SR offered extensions. Guibas and Marimont [9] showed how to support dynamic rounding (the ability to insert and delete line segments after the Snap Rounding representation has been computed). Goodrich et al. [6] showed how to deal with arrangements of line segments in $\mathbb{R}^3$ and Fortune [5] generalized the input to polyhedral subdivisions.

In this work, we propose extensions for both ISR and ISRBD. First, we propose dynamic rounding (Section 2). Then, we show how to support line segments in $\mathbb{R}^3$ (Section 3). We conclude in Section 4.

2 Dynamic Rounding

Guibas and Marimont [9] showed how to maintain hierarchical vertical cell decomposition of line segments in $\mathbb{R}^2$ (VCD) [13] with Snap-Rounded representation. They further showed how to support insertions and deletions of line segments. We use the VCD analogously, but will not attempt to describe how we use it in detail. This information is mainly technical and will be described in a detailed report.

Let S be the set of input line segments and let $\mathcal{A}(S)$ be the output arrangement of S (in either variant). We next show how ISR and ISRBD proceed.

2.1 Dynamic Iterated Snap Rounding

Insertion. Let s be the line segment we insert. We find all the pixels that need to be heated as a result of inserting s: one or two for its endpoints and others that correspond to intersections with the **input** line segments (these are found with the VCD). We collect only the above pixels that are not hot yet. Let H denote this set. Except of rounding s, we also round any link of $\mathcal{A}(S)$ that intersects the pixels in H; we find these links (using the VCD) and use the rounding routine of ISR to round links [10].

Deletion. We delete a line segment s from $\mathcal{A}(S)$. By doing so, it is possible that hot pixels *cool down* (cease to be hot). These are the hot pixels that would not have been heated if s had not been inserted to $\mathcal{A}(S)$. This information can be easily saved under the hot pixel data structure. Then we process each cooling of a hot pixel by modifying the output of some of the input line segments as follows. Let s' be any line segment. We construct a hierarchical tree of links for s', denoted by $T(s')$. In the first level, $T(s')$ contains s'. In the second, $T(s')$ contains the links through the center of hot pixels that s' intersects in the order of intersections and so on. Note that the output of s' is the links at the bottom of $T(s')$, ordered by the DFS. Let h be a hot pixel that is cooled down. Let l be the top link in $T(s')$ that intersects with h but does not end there (assume that such a link exists). Consider the two links l_1 and l_2, one level below l, that are adjacent to h. We first delete both subtrees of l_1 and l_2 from $T(s')$. Let h_1 and h_2 be the hot pixels that contain the other endpoints of l_1 and l_2, respectively. We insert the link that connects the centers of h_1 and h_2 to $T(s')$ by calling the rounding routine of ISR. The new representation of s' will correspond to the modified $T(s')$.

Let A be complexity of $\mathcal{A}(S)$. Let $\mathcal{H}$ denote the set of hot pixels and let $\mathcal{W}$ denote the set of pixels that either contain an endpoint of a vertical attachment in the VCD or is a neighbor of such a pixel. For any pixel p, its degree will be denoted by $|p|$. Note that this quantity may be larger in ISR than in SR, as links are broken to create multiple ones. The following is a theorem that bounds the total insertion time of n line segments. It follows from the analysis in [9].

Theorem 1 *Performing n insertions using Dynamic Iterated Snap Rounding takes $O(n \log n + A + \Sigma_{h\in\mathcal{H}}|h| \log |h| + \Sigma_{w\in\mathcal{W}}|w|)$ time.*

2.2 Dynamic Iterated Snap Rounding with Bounded Drift

Insertion. The insertion of line segments is similar to the insertion in the previous subsection, with the modifications that ISRBD requires (see Section 1). When inserting a line segment s, we proceed as follows. We find the hot pixels that are centered within the forbidden loci of s. For each, we heat a corresponding pixel as explained in [15]. This process may cascade as the heated pixel may be centered within a forbidden loci of another line segment. Then we round s and the links that intersect with the hot pixels in the way ISR proceeds (see the previous section). Finally, we delete some of the degree-2 hot pixels similarly to ISRBD (see section 1 and [15] for details).

Deletion. When deleting a line segment s, we need to do both the work of the deletion that is presented in Section 2.1 and the work required by ISRBD which we describe next. For each line segment, we collect the new pixels that were heated before to constrain its deviation (denote this set by Ξ). The pixels of Ξ have to be cooled down when deleting s, unless they were used to constrain the deviation of other line segments as well. As we mentioned above, the process of heating pixels may cascade. Thus, we also need to delete all the pixels that were heated as a result of cascading from the pixels of Ξ (we save the corresponding data). After all of the hot pixels are cooled down, we proceed in the same way as the removal of ISR, namely reconstructing the hierarchy of the output of the line segments that are effected by this process.

Figure 1 illustrates the results of inserting and deleting a line segment.

The following theorem bounds the total insertion time of n line segments. It is mostly dominated by the routines of ISRBD. In this case $\mathcal{H}$ denotes the set of all hot pixels, including the ones ISRBD heats to constrain the deviation of the output.

Theorem 2 *Performing n insertions using Dynamic Iterated Snap Rounding with Bounded Drift takes $O(n^3(\log n + |\mathcal{H}|^{\varepsilon}) + \kappa(\kappa \log n + |\mathcal{H}|^{\varepsilon}) + \Sigma_{w \in \mathcal{W}}|w|)$ time for any $\varepsilon > 0$ where κ is the number of degree two vertices that do not correspond to endpoints.*

3 Line Segments in $\mathbb{R}^3$

Goodrich et al.[6] extended SR to line segments in $\mathbb{R}^3$. In $\mathbb{R}^3$, the space is partitioned into a set of voxels (unit-cubes) centered at points with integer coordinates. Similarly to $\mathbb{R}^2$, voxels which contain line segment endpoints are hot. The rest of the hot voxels are determined as follows. For each pair of line segments s_1 and s_2, find $l = (p_1, p_2)$, the shortest link that connects them with length d in the L_∞ metric. If $d \leq 1$, we heat the voxels that contain p_1 and p_2 (possibly the same voxel contains both). To test whether two line segments are close, they define a *tube* $\tau(s)$ to be the Minkowski sum of s and an axis-oriented unit cube centered at the origin. It is easy to verify that two line segments are within an L_∞ distance of 1 if and only if their tubes intersect. To report the tubes that intersect, they use a range searching technique for semi-algebraic varieties [1] (denoted by ϕ). The main idea is to define a search structure of the faces of the tubes that finds all the segments of other tubes that penetrate the faces. We use ϕ for the same purpose. Then the output of any line segment is the polygonal chain through the centers of the hot voxels it meets, in the order of appearance. We next present how ISR and ISRBD proceed.

3.1 Iterated Snap Rounding

We first find the hot voxels using ϕ. Then we round links to hot voxels in the way ISR proceeds [10], with the analogous changes. To find the hot voxels that intersect a given line segment, we use the multi level partition tree data structure [2] on the hot voxels. We project the hot voxels onto the three planes $x = 0$, $y = 0$ and $z = 0$, and build the multi level partition tree for each. Then an $\mathbb{R}^3$ query is carried out by intersecting the query results of the three $\mathbb{R}^2$ trees; each queries the projection of the line segment.

The next theorem summarizes ISR in $\mathbb{R}^3$. It follows from analyzing the algorithm of ISR with the necessary modifications we described above.

Theorem 3 *Iterated Snap Rounding of line segments in $\mathbb{R}^3$ can be computed in $O(n^{\frac{8}{5}+\varepsilon} + K + L^{\frac{2}{3}}N^{\frac{2}{3}+\varepsilon} + L)$ time and $O(n^{\frac{8}{5}} + K + L^{\frac{2}{3}}N^{\frac{2}{3}+\varepsilon} + N)$ space for any $\varepsilon > 0$ where K denotes the number of intersecting tubes, L is the overall number of links in the chains produced by the algorithm and N is the number of hot voxels.*

3.2 Iterated Snap Rounding with Bounded Drift

Let δ be the deviation bound. We first decide which voxels to heat in order to restrict the deviation of any line segment. We then cool down degree-two hot voxels which are both not associated with endpoints and will not cause line segments to drift too much if deleted (analogous to the work in [15]).

Note that in $\mathbb{R}^2$ we define forbidden loci to be the two stripes to the left and right of a line segment s, that necessarily contain the center of any hot pixel that may be the first to round s too far. In $\mathbb{R}^3$, the forbidden loci are constructed analogously as follows. For any line segment s, let v_1 and v_2 be the voxels that contain its endpoints. Let $B(s)$ be the bounding box of the centers of v_1 and v_2. Due to the weakly-monotone property, the output of s will be located within $B(s)$. Let $C(s)$ be the Minkowski sum of s with a ball of radius δ centered at the origin. It follows that the output of s must be located within $D(s) = B(s) \cap C(s)$. Note that the maximum deviation any link may undergo during one rounding step is $\frac{\sqrt{3}}{2}$ units. It follows that if any hot voxel rounds the output of s beyond $D(s)$, it must be centered at a point that is within distance $0 < d \leq \frac{\sqrt{3}}{2}$ from $D(s)$, and inside $B(s)$. We define $C'(s)$ to be the Minkowski sum of s with a ball of radius $\delta + \frac{\sqrt{3}}{2}$ centered at the origin. It follows that the forbidden loci of s is $F(s) = (C'(s) \setminus C(s)) \cap B(s)$.

The routine for heating voxels that restrict the deviation proceeds as follows. For each hot voxel v, we find the set of line segments whose forbidden loci contain its center. Let s be a line segment in this set. We heat a voxel which both intersects s and will guarantee that s

will not be rounded to v. There are three questions to address at this point.

First, which voxel do we heat on s? For each voxel v, we partition $\mathbb{R}^3$ into 8 octants using the three orthogonal walls through the center of v. Ignoring degenerate cases (which are not difficult to handle), s will pass through three or four octants. Let $\Psi(s, v)$ be an octant that does not contain any of the endpoints of s and let $s' = s \cap \Psi(s, v)$. We heat the voxel which contains the middle of s'. By heating it we are guaranteed that s will never be rounded to v; otherwise it would contradict the weakly-monotone property[3]. Note however, that if there is already a hot voxel that intersects s', such that its 3 coordinates differ from the corresponding coordinates of v, we need not heat any voxel.

The second question is how to find the line segments whose forbidden loci contain the center of v. It would be desirable to use a range search data structure of $\{F(s)|s \in S\}$. However, to the best of our knowledge such a data structure is yet not available in the literature. So instead we will test each pair of line segment-hot voxel.

Next we ask what would be the complexity of the output. In [15], the selection of pixels to heat for restricting the deviation was such that the output complexity of ISRBD is not increased[4]. It is not clear how to do the same in $\mathbb{R}^3$ and whether it is possible. We leave this task for a future work. We believe that the output complexity does not increase unless the input is pathologic, if at all. We note that the number of new voxels is finite as we deal with finite input of line segments.

The next theorem summarizes ISRBD in $\mathbb{R}^3$. It follows from analyzing the algorithm of ISRBD in $\mathbb{R}^2$ with the necessary modifications we described above.

Theorem 4 *Iterated Snap Rounding with Bounded Drift of line segments in* $\mathbb{R}^3$ *can be computed in* $O(n^3 + K + L^{\frac{2}{3}}N^{\frac{2}{3}+\varepsilon} + L + \kappa(\kappa \log n + N^{\varepsilon}))$ *time and* $O(n^{\frac{8}{5}} + L^{\frac{2}{3}}N^{\frac{2}{3}+\varepsilon} + N + K^{1+\varepsilon})$ *space for any* $\varepsilon > 0$. *The parameters were defined in Theorem 3.*

4 Conclusions

ISR and ISRBD are variants of SR. Both may be more beneficial to use than SR in different situations. In this work we expanded the capabilities of both ISR and ISRBD to support both dynamic rounding and line segments in $\mathbb{R}^3$.

Snap Rounding has drawn attention in the last two decades. Most recently, Snap Rounding was extended to support Bézier curves [4]. We believe that providing the options of ISR and ISRBD variants to other versions of SR may be useful for users when choosing the rounding model.

[3] Analogously to [15], we require $\delta > \frac{3\sqrt{3}}{2}$ to make this idea works. We postpone this discussion to the detailed report.

[4] It is still open if this is the case when links that are shared by multiple polygonal chains are counted only once.

References

[1] P. K. Agarwal and J. Matousek. On range searching with semialgebraic sets. In *Mathematical Foundations of Computer Science*, pages 1–13, 1992.

[2] P. K. Agarwal and M. Sharir. Applications of a new space-partitioning technique. *Discrete Comput. Geom.*, 9:11–38, 1993.

[3] M. de Berg, D. Halperin, and M. Overmars. An intersection-sensitive algorithm for snap rounding. *Comput. Geom. Theory Appl.*, 36(3):159–165, 2007.

[4] A. Eigenwillig, L. Kettner, and N. Wolpert. Snap rounding of bzier curves. In *Proc. 23th ACM Symposium on Computational Geometry, SoCG*, 2007.

[5] S. Fortune. Vertex-rounding a three-dimensional polyhedral subdivision. *Discrete Comput. Geom.*, 22(4):593–618, 1999.

[6] M. Goodrich, L. J. Guibas, J. Hershberger, and P. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 284–293, 1997.

[7] D. H. Greene. Integer line segment intersection. Unpublished Manuscript.

[8] D. H. Greene and F. F. Yao. Finite-resolution computational geometry. In *Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 143–152, 1986.

[9] L. Guibas and D. Marimont. Rounding arrangements dynamically. *Internat. J. Comput. Geom. Appl.*, 8:157–176, 1998.

[10] D. Halperin and E. Packer. Iterated snap rounding. *Computational Geometry: Theory and Applications*, 23(2):209–225, 2002.

[11] J. Hershberger. Improved output-sensitive snap rounding. In *Proc. 22th ACM Symposium on Computational Geometry, SoCG*, pages 357–366, 2006.

[12] J. Hobby. Practical segment intersection with finite precision output. *Comput. Geom. Theory Appl.*, 13:199–214, 1999.

[13] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[14] E. Packer. Finite-precision approximation techniques for planar arrangements of line segments. In *Master's thesis, Dept. Comput. Sci., Tel-Aviv Univ.*, 2002.

[15] E. Packer. Iterated snap rounding with bounded drift. In *Proc. 22th ACM Symposium on Computational Geometry, SoCG*, pages 367–376, 2006.