



RAYMOND AND BEVERLY SACKLER
FACULTY OF EXACT SCIENCES
THE BLAVATNIK SCHOOL OF COMPUTER SCIENCE

Snap Rounding on the Sphere

Thesis submitted in partial fulfillment of the requirements for the M.Sc.
degree in the School of Computer Science, Tel-Aviv University

by

Boris Kozorovitzky

This work has been carried out at Tel-Aviv University
under the supervision of Prof. Dan Halperin

September 2010

Acknowledgements

I deeply thank my advisor, Prof. Dan Halperin, for his help in guidance, support, and encouragement, and for introducing me to the field of applied computational geometry.

I wish to thank Ophir Setter, Efi Fogel, and Eric Berberich for sharing priceless knowledge and helping me at the expense of their own free time. I would also like to thank all other members of the applied computational geometry lab at the computer science school of Tel-Aviv University who provided support, useful suggestions and interesting conversations.

I wish to thank all the people involved in the development of the great open source and free tools I used throughout the making of this thesis. Tools such as Inkscape, WinShell, MiKTeX, and many more that saved me a lot of time and effort.

Finally, I wish to thank my family and coworkers for allowing me spend so much time working on this thesis and for their help and support.

Memoriam

In memoriam of Eshed Zachevsky, a dear friend whom I miss so much.

Abstract

Snap rounding (SR for short) is a well known method for transforming a planar arrangement of segments given in some arbitrary-precision coordinates into a fixed-precision representation. We extend the method to transforming an arrangement of geodesic arcs on the sphere. We present two approaches for solving the problem. A simple approach of enclosing the sphere in an isocube and projecting the arrangement onto its faces and a more complex approach that makes use of tools from *Discrete Global Grid Systems* (DGGS) to create a better approximation to the sphere. We also generalize the Guibas-Marimont proof of the topological properties preserved by the standard SR for segments in the plane; the generalization is needed for the DGGS approach. We present in detail the implementation of both approaches and give rounding results for both methods, obtained with our CGAL (Computational Geometry Algorithms Library) based implementation.

Contents

1	Introduction	1
1.1	Related Work	4
1.2	Introduction to Spherical Geometry	5
2	The Isocube Approach	9
2.1	Preliminaries and Notation	11
2.2	The Spherical SR Process	12
2.3	The Topological and Geometric Properties of Isocubical SSR	13
3	The DGGS Approach	17
3.1	The Subdivision Process	18
3.2	Grid Types for which SR Preserves Topology	19
3.3	Defining Pixels	21
3.4	Additional Notation	23
3.5	The Spherical SR Process	24
3.6	The Topological and Geometric Properties of DGGS SSR	24
4	SSR With Labeled Pixels	27
4.1	Labeling the Pixels	27
4.2	Labeled Spherical SR Process	28
4.3	Conclusion	29
5	Implementation	31
5.1	CGAL Packages Used by the Implementation	31
5.1.1	2D Arrangement	31
5.1.2	2D Snap Rounding	32
5.1.3	Arrangement of Geodesic Arcs on the Sphere	32
5.2	Implementation Details	32
5.2.1	Input	33
5.2.2	Distribution to Faces	33

5.2.3	Snap Rounding on Faces	34
5.2.4	Adding Connection Arcs	35
5.2.5	Output	35
5.3	Measuring the Directed Hausdorff Distance on the Sphere	36
6	Experimental Results	37
6.1	Robustness	39
6.2	Rounding Distance	41
6.3	Random Input	42
6.4	Real World Input	43
7	Conclusion and Future Work	49
A	Appendix	51
A.1	Convex Sets on a Sphere	51
A.2	Circumcenter Bit Length	51
A.3	Input Files Example	53

List of Figures

1.1	Geometric rounding compared to regular rounding	2
1.2	SR illustration	3
1.3	An illustration of grids on the sphere	7
2.1	A unit sphere enclosed by an isocube	10
2.2	The 2D coordinate system on each face	11
2.3	The forbidden region of a square face	14
2.4	Possible connection arcs between four pixels.	15
2.5	Possible connection arcs between three pixels.	16
3.1	The octahedron	17
3.2	Triangular face subdivision	18
3.3	Shrinking edge during deformation	21
3.4	The 2D coordinate system on each face	22
3.5	Creating parallelogram p-pixels	23
3.6	Forbidden region of a triangular face	25
4.1	Assigning labels to new pixels after a subdivision step	28
4.2	Icosahedral net	28
6.1	Spherical snap rounding of real world inputs	46
6.2	Spherical snap rounding of real world inputs 2	47
6.3	Spherical snap rounding of real world inputs 3	48
A.1	The bit length of circumcenter	52

List of Tables

2.1	Isocube 2D coordinate axes	10
3.1	Octahedron 2D coordinate axes	22
6.1	Examples of spherical grids	38
6.2	Examples of how degeneracies are handled	39
6.3	Abbreviations.	41
6.4	Rounding results of a single arc with respect to its distance from the spherical face boundary on a face of the isocube	42
6.5	Rounding results of a single arc with respect to its distance from the spherical face boundary on a face of the octahedron	42
6.6	200 random arcs results 1	43
6.7	200 random arcs results 2	43
6.8	Increasing number of arcs results	44
6.9	Rounding the border of USA 1	44
6.10	Rounding the border of USA 2	44
6.11	Rounding the map of major roads in North America 1	45
6.12	Rounding the map of major roads in North America 2	45

1

Introduction

In computational geometry, geometric objects and algorithms are often described using infinite precision arithmetic (the so-called real RAM model [PS85]) for exact calculation of predicates and new geometric objects. In general assuming infinite precision and exact arithmetic is possible, yet it is often too slow and space consuming to run on real world inputs. In some cases the input is represented by low precision coordinates but after some manipulations (e.g., building Voronoi Diagrams [OBSC00,AK00]) the coordinates may grow substantially in their bit length making further calculations cumbersome and time consuming. Furthermore, sometimes the calculated data is to be transferred to a different system that does not support exact calculations and may even receive its input in single-precision or in integer arithmetics (graphical frameworks often work this way).

To produce finite-precision approximation of geometric objects several methods of *geometric rounding* can be used. Geometric rounding for planar straight edge graphs can be defined as follows: We have a set of line segments on the plane that may intersect only at their endpoints. The goal is to round the vertices to integer coordinates while maintaining the topology as much as possible. There are several definitions on what it means to preserve the topology. In general, the geometric rounding process creates a polygonal chain, *polysegment* in place of every original input segment and we expect crossings of these polysegments to correspond to vertices in the original input.

We review common methods for geometric rounding of an *arrangement* (see definition in Section 1) of segments in the plane. (i) The Greene and Yao [GY86] method. Consider the grid with pegs at all the integer points and the segments as rubber bands on the plane. Move the vertices of the segments to the closest peg and keep the rubber band tight. The result is a polysegment through integer points. This method clearly maintains topology but also introduces many unnecessary vertices in the output. (ii) The Milenkovic shortest path geometric rounding [Mil00] method. Milenkovic gives his own definition to topological consistency preservation and gives a general rounding algorithm. The algorithm requires the

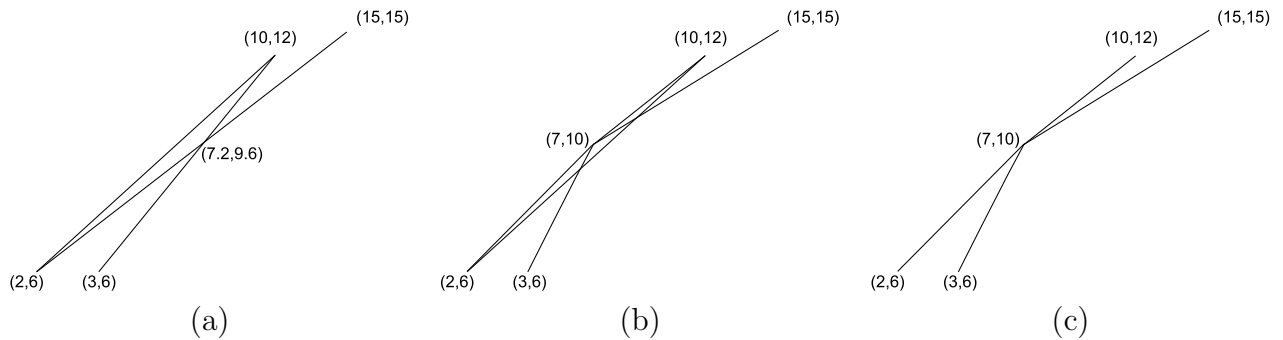


Figure 1.1: An example of an arrangement (a) whose vertices are rounded to integer coordinates with regular rounding (b) and with snap rounding (c). Regular rounding does not preserve topology, as shown in the figure a rounded vertex “jumps” over a segment. Furthermore, two new unrounded intersections are created. These issues are not present in the snap rounded arrangement¹.

definition of a lattice of reference points. Each reference point defines a simply connected cell such that every point in the plane is in one and only one cell. In the algorithm each vertex in the input is rounded to the reference point of the cell that contains this vertex. The algorithm is hard to implement because it requires the vertices to “move” within the cell and “push” the segments in its path until it reaches its final position. Finally, (iii) the Snap Rounding method which is the focus of this thesis and is described in detail below. Figure 1.1 shows why geometric rounding is needed over trivial rounding of vertex coordinates in cases where it is important to preserve the topology of the subdivision induced by the segments.

Snap Rounding

Snap Rounding (SR for short) is a method for finite-precision approximation of arrangements of segments in the plane: It transforms the arrangement whose segment endpoints’ coordinates are given in some arbitrary-precision, into a low precision representation.

Given a finite set of segments \mathcal{S} , the arrangement $\mathcal{A}(\mathcal{S})$ is the subdivision of the plane into vertices, edges, and faces induced by \mathcal{S} . The vertices of $\mathcal{A}(\mathcal{S})$ are either endpoints or intersection points of segments in \mathcal{S} . For a given arrangement whose vertices are specified in arbitrary precision snap rounding is the following process:

1. Tile the plane with a grid of unit squares centered at integer coordinates. We refer to each square as a *pixel*.
2. Define a pixel to be *hot* if it contains a vertex of the given arrangement.
3. Replace each vertex by the center of the hot pixel containing it.
4. Replace each original input segment e by a polygonal chain e' going through the centers of the hot pixels intersected by e in the same order the pixels are met by e .

¹Figure based on [Hob99]

See Figure 1.2 for an illustration.

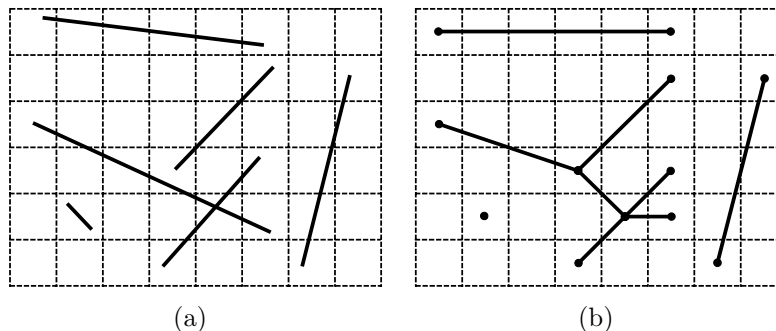


Figure 1.2: An arrangement of segments before (a) and after (b) snap rounding.

We refer to an original (unrounded) segment and to a resulting polygonal chain as *ursegment* and polysegment, respectively [GM98]. Note that in the process, vertices, edges, and faces of the original arrangement may collapse. At the end of the process all the vertices in the snap rounded arrangement are at integer coordinates. The snap rounded arrangement preserves topological and geometric properties with respect to the original arrangement. **Geometric similarity**— the rounded polysegment e' is within the Minkowski sum of the original ursegment e and a unit square centered at the origin. **Topological similarity**— there is a continuous deformation of the segments in \mathcal{S} to their snap-rounded counterparts such that no segment ever crosses over a vertex of the arrangement.

In the thesis we consider a variation of the rounding problem. In our case the input is an arrangement of *geodesics* (arcs of great circles) on a sphere with vertices given in arbitrary precision. Our *spherical snap rounding* (SSR for short) transforms this arrangement into low precision representation, while preserving topological and geometric properties similar to those preserved by the original SR scheme for segments in the plane. The properties of the SSR process and the rounded arrangement are as follows:

1. The rounded arcs drift such that the directed Hausdorff distance on the sphere (see formal definition in Section 1.2) between the rounded polyarc and the original arc is no larger than the diameter of the circumcircle of the largest spherical pixel in the defined grid.
2. The original and the rounded arrangement are topologically equivalent up to the collapsing of features [GM98].
3. The centers of the spherical pixels can be represented using small (with respect to bit length) rational values.
4. We also require that: The spherical pixels should have similar shape and area and should be as regular as possible.
5. The grid should be refinable (to allow for increasing the approximation quality).

1.1 Related Work

SR was independently introduced by Hobby [Hob99] and by Greene (unpublished manuscript) as an alternative to the works of Greene and Yao [GY86] and of Victor Milenkovic [Mil89, Mil90]. Hobby relies on the Bentley-Ottmann sweep [BO79] and his approach is the basis for the SR work that followed. This basic scheme was then generalized by Guibas and Marimont [GM98] to a dynamic SR algorithm where the authors also introduce some elementary proofs regarding the topological and geometric properties of the snap rounded arrangement. Two additional algorithms by Goodrich *et al.* [GGHT97] and by de Berg *et al.* [dBHO07] give two versions of the SR algorithm. The first algorithm takes advantage of the fact that a hot pixel can be discovered by finding only one intersection or endpoint within the pixel. All other intersections and endpoints can be ignored within this pixel. The second algorithm takes special care with multiplicity of rounded arcs (namely, many ursegments collapsing to the same rounded segment) and uses “bundles” and a special sweep technique to handle multiple arcs emanating from the same hot pixel. Hershberger [Her08] presents an improvement to both algorithms by introducing the $is(h)$ factor. $is(h)$ is the number of segments that have an intersection or an endpoint within the hot pixel h and it is clearly less or equal to the number of segments incident to h . Hershberger’s SR algorithm performs well in all the cases. Bhattacharya and Samber [BS07] modify the definition of a hot pixel slightly and produce an algorithm which works on columns instead of single pixels.

Another approach to SR comes in the form of iterated SR presented by Packer and Halperin [HP02]. In their paper the authors notice that after SR a segment can still be very close to a vertex in the rounded arrangement. They show that if the input arrangement is contained in a $2^b \times 2^b$ grid of unit pixels the distance between a rounded segment and a vertex can be almost 2^{-b} . To alleviate this problem they suggest running SR iteratively until no change is made in the rounded arrangement and show an efficient way of doing so using oriented kd-trees. In a later paper [Pac08], Packer deals with the fact that in iterative SR a rounded segment might drift up to $\Theta(n^2)$ units from the original segment. He suggests introducing an additional parameter to the original algorithm which bounds the drift of the rounded segment by artificially heating specific pixels around the unrounded segment.

The subdivision of the sphere that we use is based on known methods in the fields of Geodesy and Cartography. In these fields there is a requirement for a discrete representation of the sphere. Such a representation allows classification of discrete areas (forests, rivers, wheat field, etc.) and the coloring of pixels for display. We concentrate on the Discrete Global Grid Systems (DGGS) surveyed by Sahr *et al.* [SWK03]. They show several ways to build a discretization of the sphere and show the favorable properties of such subdivisions.

Contribution of the Thesis

In this work we describe two approaches of discretizing the sphere and dividing it to spherical pixels. Chapter 2 describes a discretization based on an isocube which creates six identical spherical faces and present a method of snap rounding geodesic arcs on the sphere by subdividing these faces into spherical pixels. We prove the topological and the geometric

properties of this method with respect to the list of properties given above. Chapter 3 presents a more elaborate method based on DGGs that uses the octahedron to divide the sphere into eight identical spherical faces which results in smaller spherical pixels and therefore a better approximation of the sphere. We present insights regarding the grids that can be used for planar SR and still preserve the topological property. We continue and prove the topological and geometric properties for the DGGs-based method. In Chapter 4 we show how the DGGs method can be extended to provide an even better approximation of the sphere but only when the limit on the bit length of the coordinates is dropped. Chapter 5 describes in some detail our implementation of both approaches. We use CGAL [2] to build an arrangement of geodesic arcs and then round it using the underlying construction of each approach. In Chapter 6 we report on experimental results obtained with our implementation on various types of synthetic and real world inputs. Finally, in Chapter 7 we present our conclusions and suggestions for future work.

In recent years more and more services and applications use data represented on the sphere. The best known examples of this trend are utilities such as NASA World Wind [9], ESRI Arc GIS Explorer [3] and Google Earth [6] that represent all the data on the sphere. Traditionally running geometric algorithms on this data requires projecting it onto the plane and running the planar algorithm, a method that introduces deformations and other undesired artifacts to the data. Sometimes, such projections require complex calculation which at times create incorrect and unusable results. For example, convex shapes become concave or new intersections which do not exist in the original input are introduced. We present a spherical variation on a widely used method of planar snap rounding that can be run directly on spherical input and result in rounded spherical output. Much like in the planar version, the spherical algorithm can be run with grids of various pixel sizes that increase the geometric quality of the output but has a negative effect on the bit length of the coordinates in the output.

Our discussion on possible grids for planar SR provides the groundwork needed for implementing variations of the basic algorithm which can include different pixels shapes.

1.2 Introduction to Spherical Geometry

Spherical geometry differs from the Euclidean geometry in several ways, most notably the sphere is a finite closed surface while the plane is an infinite surface. A geodesic is the shortest curve between two points on a given surface. On the plane a geodesic is simply a line segment, on the sphere it is the section between two points supported by a great circle containing the two points. This arc is made unique by selecting the shorter arc, unless, the points are antipodal in which case there are infinitely many such geodesics with equal length. Throughout this thesis we use the unit sphere centered at the origin, we consider the length of a geodesic between the points A and B on the sphere as the angle $\angle AOB$ (in radians) where O is the origin. This is the standard definition for a geodesic arc on a unit sphere.

A spherical triangle is an area enclosed by three pairwise intersecting geodesics. The sum of angles in such a triangle is between π and 3π radians (A nice anecdote is that a right spherical triangle is one with three $\frac{\pi}{2}$ angles). The perimeter of a spherical triangle is the

sum of lengths of the geodesics inducing it and therefore can be easily calculated once the vertices of the triangle are known. The area of a spherical triangle can be derived from its angles and is given by Girard’s “spherical excess” formula [Cox69] as $\Delta = \alpha + \beta + \gamma - \pi$ where α , β , and γ are the spherical angles of the triangle. Throughout this thesis we will use the spherical coordinate system (θ, ϕ) , $0 \leq \theta < 2\pi$, $0 \leq \phi \leq \pi$ to represent coordinates on the unit sphere given by two angles.

A Note on Spherical Grids

Spherical grids were the subject of research along the years (and still are) but perhaps the best known grid is the one induced by the geographic coordinate system (ϕ, λ) , $-\frac{\pi}{2} \leq \phi < \frac{\pi}{2}$, $-\pi \leq \lambda \leq \pi$ and commonly known as the Latitude-Longitude grid [4]. The grid lines are formed by keeping the radius and one of the angles constant while changing the other angle in fixed interval. The main advantage of this grid is that every cell is a square in the parameter space, making it easy to use. But such a grid has several critical disadvantages. The grid cells have inconsistent size and shape on the sphere and the pole cannot be uniquely represented in the coordinate system. A slightly different approach generates a grid using Voronoi partitioning of the sphere. First we select points on the sphere with some predefined distribution and define the grid cells as the Voronoi partitioning of the sphere with the points as sites [JDG02, DJ05]. Recent research created yet another kind of grid with beneficial qualities, the HEALPix grid [GHB⁺05]. The HEALPix grid satisfies three criteria. It is hierarchical (can be easily refined to produce a grid with smaller cells), it has equal area cells and the cell centers are distributed on an isolatitude grid (a grid with equal latitude for each row of pixels).

Sometimes it is important for the cells on the grid to be of equal area. This means that the cells will be very distorted as in the Snyder Equal Area grid [Sny92] or the grid can be constructed using small circle arcs [SKS02].

A completely different way to define a spherical grid is by projecting the facets of a polytope inscribed in the sphere onto the sphere. An example of this method, which is commonly used, is the DGGs [SWK03]. To build a DGGs, five parameters are selected: the base regular polytope, the orientation of the polytope with respect to the sphere, the transformation method of points on the unit sphere to and from the polytope, the subdivision method of the faces of the polytope and the selection of the final grid cells. More information on DGGs is given in Chapter 3.

Figure 1.2 shows four examples of grids on the sphere.²

Hausdorff Distance on the Sphere

We define the distance $d(A, B)$ between the points A and B on the sphere as the length of the geodesic between these points. On the unit sphere $0 \leq d(A, B) \leq \pi$. This allows us to

²This figure and other 3D figures in this thesis were created using an interactive viewer for an extended VRML format called “Player” (see footnote in Section 5.2.5)

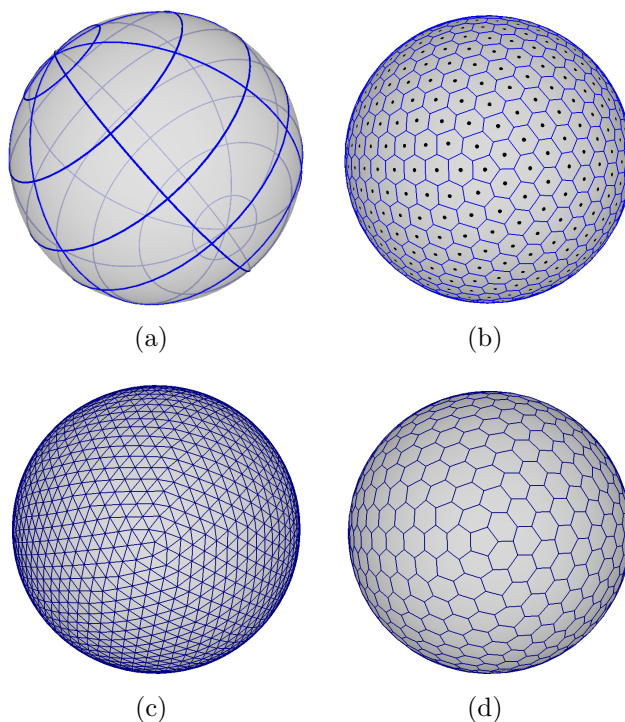


Figure 1.3: An illustration of grids on the sphere: (a) the lat-lon geographic grid (b) Voronoi cells of points (c) DGGs with triangular pixel selection (d) DGGs with hexagonal pixel selection.

use the directed Hausdorff distance on the sphere as $d_H(X, Y) = \max_{x \in X} \left(\min_{y \in Y} (d(x, y)) \right)$.

In the plane, calculating the Hausdorff distance between two segments with Euclidean metrics can be done by considering only the endpoints of the two segments. To calculate the Hausdorff distance for two geodesic arcs χ and ψ on the unit sphere, $d_H(\chi, \psi)$ we must determine which points on the two arcs are candidates to determine this distance. The basic block of the calculation is finding the minimal distance between two points. We know that this distance is the length of the geodesic arc between the points (by definition). The next step is to determine the minimum distance from a point to an arc. For simplicity and w.l.o.g., we assume that ψ is on the equator. We calculate the minimum distance between two points on the unit sphere centered at the origin. A general point (a, b, c) (given in Cartesian coordinates) and a point on the equator, $(x, \pm\sqrt{1-x^2}, 0)$. Consider the first point as the vector \vec{p} originating at the origin and the second point as the vector \vec{q} also originating at the origin. The angle between the vectors can be expressed as $\vec{p} \cdot \vec{q} = \cos(\alpha)$, or $f(x) = ax \pm b\sqrt{1-x^2} = \cos(\alpha)$. To find the minimum we find the derivative $f'(x) = a \pm \frac{b}{2} \frac{-2x}{\sqrt{1-x^2}}$ which equals 0 when $\frac{a}{b} = \pm \frac{x}{\sqrt{1-x^2}}$. This means that the point (a, b, c) is on the same longitude as the point on the equator.

Definition 1.1. Let $\alpha = ((\theta_1, \pi/2), (\theta_2, \pi/2))$ be a geodesic arc on the unit sphere (assume w.l.o.g. that $\theta_1 < \theta_2$) which coincides with the equator. The vertical lune of α , $VL(\alpha)$ is the set of all points (θ, φ) such that $\theta_1 \leq \theta \leq \theta_2$ and $0 \leq \varphi \leq \pi$

Since ψ coincides with the equator then we can divide the arc χ into two parts. The first

part is $\hat{\chi} = \chi \cap VL(\psi)$ where we consider only the point with the greatest absolute latitude. Note that this does not have to be an endpoint of $\hat{\chi}$, if the apogee of the underlying great circle of χ is in $VL(\psi)$ then it has the greatest absolute latitude. The second part is $\chi \setminus \hat{\chi}$, the remainder of χ where we can calculate the required distance by using only the endpoints (the distance is between two endpoints).

2

The Isocube Approach

We wish to adapt the planar snap rounding algorithm for the sphere, but there is no known way of generating recursive tiling of the sphere with convex identical cells [SWK03, WKSS98] (see Appendix A.1 for a definition of spherical convexity) which are required by the rounding properties we suggested above. Instead, we adapt the input by projecting it on several planes and get a planar arrangement on each plane. Then we apply an augmented version of the standard planar SR algorithm. We start by enclosing the unit sphere with an isocube (the facets of which are oriented parallel to the axes). We introduce a label for each face of the isocube: The coordinates of the centroid of that face. The coordinates of the centroids in \mathbb{R}^3 are $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, and $(0, 0, \pm 1)$. To maintain consistency we assign each face its inclusive and exclusive boundary. This is important in order to make the SR process well defined: Each point on the cube (and therefore each point on the sphere) will be thus eventually assigned to a unique pixel. The following list is the assignment of the inclusive boundary for each face:

- The face labeled $(1, 0, 0)$ is associated with the open edges $((1, 1, -1), (1, 1, 1))$, $((1, -1, 1), (1, 1, 1))$ and the vertex $(1, 1, 1)$.
- The face labeled $(0, -1, 0)$ is associated with the open edges $((1, -1, -1), (1, -1, 1))$, $((-1, -1, 1), (1, -1, 1))$ and the vertex $(1, -1, 1)$.
- The face labeled $(-1, 0, 0)$ is associated with the open edges $((-1, -1, -1), (-1, -1, 1))$, $((-1, 1, -1), (-1, -1, 1))$ and the vertex $(-1, -1, -1)$.
- The face labeled $(0, 1, 0)$ is associated with the open edges $((-1, 1, 1), (-1, 1, -1))$, $((1, 1, -1), (-1, 1, -1))$ and the vertex $(-1, 1, -1)$.
- The face labeled $(0, 0, -1)$ is associated with the open edges $((-1, -1, -1), (1, -1, -1))$, $((1, 1, -1), (1, -1, -1))$ and the vertices $(1, -1, -1)$, $(1, 1, -1)$.

Face	X	Y
(1,0,0)	-Y	-Z
(0,1,0)	X	Z
(0,0,1)	-Y	X
(-1,0,0)	Y	Z
(0,-1,0)	-X	-Z
(0,0,-1)	Y	-X

Table 2.1: The definition of 2D coordinate axes on each face of the isocube.

- The face labeled $(0, 0, 1)$ is associated with the open edges $((1, 1, 1), (-1, 1, 1))$, $((-1, -1, 1), (-1, 1, 1))$ and the vertices $(-1, 1, 1), (-1, -1, 1)$.

The selection of the open edges and vertices is arbitrary with respect to the 3D axes. The only guideline it follows is that every face must be associated with two edges whose closure share a vertex, and that vertex. Since there are six faces and eight vertices, two vertices were assigned to faces without specific consideration.

We can increase the precision of the output by subdividing the face of the isocube. Actually, we perform the subdivision on each face of the enclosing isocube by extending perpendicular bisectors from its edges. Each face is subdivided by performing ρ iterations creating 4^ρ square cells on every initial face. We refer to each cell as a planar pixel (p-pixel). Every p-pixel is defined to include only the top and left edges and their incident vertex, this is consistent with the planar snap rounding scheme.

We define a new 2D coordinate system on every face of the isocube relative to the 3D coordinate system of the isocube and the sphere. The 2D coordinate system of each face is presented in Figure 2.2 and is summarized in Table 2.1.

We choose a point in \mathbb{R}^3 and shoot a ray from the origin through the point. The coordinate on the sphere in this representation is the intersection of the ray and the sphere surface. It is clear that in this representation each point on the sphere has infinitely many coordinates but this fact does not pose a problem in our discussion. With this convention we can represent the initial coordinates of the face centers on the sphere with small rational numbers (in terms of bit length). Our initial face centers are represented using the numbers $-1, 0, 1$, which requires only 2 bits per axis and 6 bits per coordinate. In each subdivision the denominator is a power of 2 with bit length ρ and the numerator is in the range of $(-2^\rho, 2^\rho)$; thus having the bit length of $\rho + 1$. This means that the bit length of the centroid coordinates is in the order of $O(\rho)$.

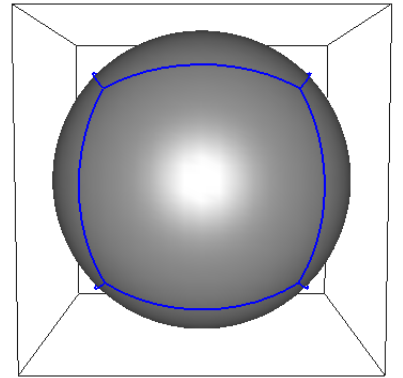


Figure 2.1: A unit sphere enclosed by an isocube whose edges are projected onto the sphere.

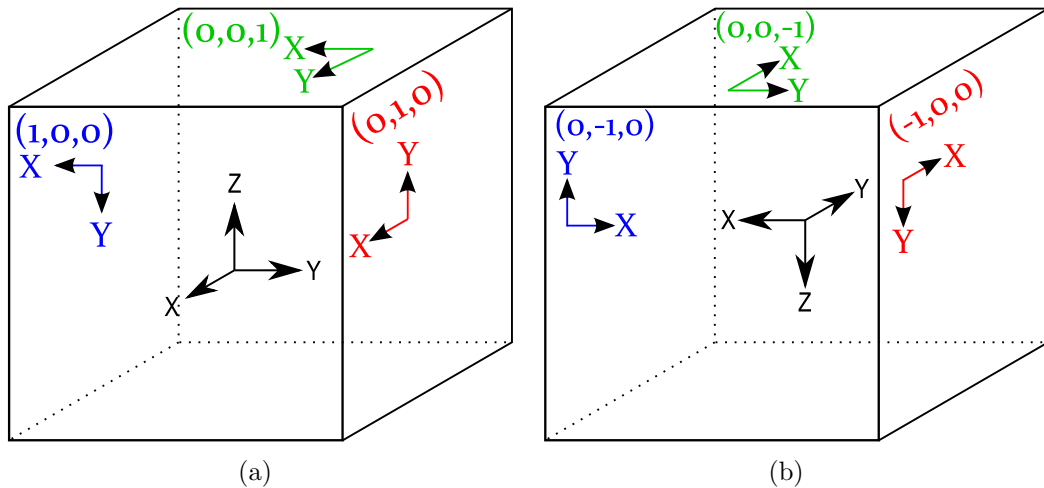


Figure 2.2: An illustration of the 2D coordinate axes on each face relative to the 3D axes of the isocube.

2.1 Preliminaries and Notation

The input to Spherical Snap Rounding (SSR) is a set C of geodesic arcs on a sphere. For convenience we use the term *arc* to refer to a great circle arc; when we use a more general definition of arc, we explicitly say so. The input arc-endpoints are given in some arbitrary precision by specifying their start and end vertices. The coordinates of the vertices are given in our coordinate scheme described above. We define an additional vector to determine the direction of the arc (two non antipodal points on the sphere can be connected with two arcs, while two antipodal points can be connected with an infinite number of arcs). The arrangement of arcs on the sphere is denoted by $\mathcal{A}(C)$ and it comprises the vertices, edges and faces induced by the arcs on the sphere. The goal of our algorithm is to round all the vertices of the arrangement (arc endpoints and arc intersection-points) into some fixed coarser precision: In our case the vertices of the arrangement will be rounded to the centers of the pixels that we define on the faces of the isocube and project onto the sphere. For the rest of the thesis we will refer to these points as the *reference points* of the pixels as in further discussion they are not at the geometric center of the pixels. We use the term *urarc* for an original unrounded input arc, and the term *polyarc* for the rounded version, which is a series of arcs concatenated at the endpoints. The input to the algorithm is a collection of urarcs C and a parameter ρ , and the output is the arrangement, which is the spherical snap rounded version of $\mathcal{A}(C)$.

The following is the notation that we will use hereafter:

- $\mathcal{A}(C)$ - The arrangement of urarcs on the sphere.
- ρ - The number of subdivision steps on each face of the isocube.
- f_i - A face of the isocube.
- G_P - The Gnomonic projection from the unit sphere onto the faces of the isocube. For

some point p on the sphere, the intersection point of the ray \vec{op} from the center o of the sphere with a face of the cube is the Gnomonic projection of p , denoted $G_P(p)$.

- G_P^{-1} - The inverse transformation of G_P : the Gnomonic projection from the faces of the isocube onto the sphere.
- ϕ_i - $G^{-1}(f_i)$, the spherical face corresponding to the face f_i .
- p_i - A planar pixel i (also referred to as a p-pixel) on a face of the isocube.
- ξ_i - $G^{-1}(p_i)$, a spherical pixel (also referred to as an s-pixel). The reference point of the s-pixel is the inverse projection of the reference point of the corresponding p-pixel and it is represented with the exact same coordinate in our coordinates scheme.
- $SR(s)$ - The polysegment induced by the ursegment s after applying the standard version of planar snap-rounding algorithm.
- $SSR(\sigma)$ - The polyarc induced by the urarc σ after applying the isocubical version of spherical snap-rounding.

2.2 The Spherical SR Process

The following steps give an overview of the SSR process:

1. Project the arrangement $\mathcal{A}(\mathcal{C})$ onto the faces of the isocube using G_P .
2. Run planar SR on each face of the isocube with consideration to the boundary conditions of that face.
3. Project the resulting arrangement back onto the sphere using G_P^{-1} .
4. Connect the projected arrangements using *connecting arcs* (see below).

In more detail, given an arrangement $\mathcal{A}(\mathcal{C})$ of geodesic arcs on the unit sphere centered at the origin and the integer ρ , the spherical snap rounding proceeds as follows: We enclose the unit sphere with an isocube aligned to the coordinate axes. On every face of the isocube we perform ρ steps of the previously mentioned subdivision creating a total of $6 \times 4^\rho$ square p-pixels. Now, we project the arrangement $\mathcal{A}(\mathcal{C})$ onto the faces of the isocube using G_P . We define a p-pixel to be *hot* if it contains a vertex in the projected arrangement or it has a segment crossing its boundary into a different face of the isocube. We call a segment on the face of the cube that intersects the face boundary and extends to another face a “broken” segment. In this case we create two vertices infinitesimally close to the boundary edge intersected by the segment, one in each p-pixel, each p-pixel on a different face, and register each of these p-pixels as *boundary-connected* and **hot**. We refer to the intersection point itself as the planar *connection point* of the two p-pixels and its projection onto the unit sphere as the spherical connection point. Note that if a projected segment ends exactly on the boundary we use the inclusion rules previously defined to decide to which pixel(s) it

belongs. Each vertex of the projected arrangement is replaced by the reference point of the hot p-pixel containing it and each projected segment s is replaced by a polygonal chain going through the hot p-pixels in the same order they are met by s on a single face. In other words, we run the planar SR on every face. After the rounding process, all the rounded segments are projected back onto the sphere using G_P^{-1} . For each registered pair of connected p-pixels, their corresponding s-pixels reference points are connected with a small geodesic arc.

2.3 The Topological and Geometric Properties of Isocubical SSR

We show that in SSR each polyarc is close to its inducing urarc and that we satisfy Property 1 presented in the Introduction.

Observation 2.1. *On a single face of the isocube the geometric property of the planar SR holds. Each segment is rounded within the Minkowski sum of the segment and a p-pixel centered at the origin.*

This follows from the properties of the standard planar SR.

Lemma 2.2. *Given a spherical face ϕ_i and an urarc σ that is contained entirely within ϕ_i . Let $\xi_j \subseteq \phi_i$ be the largest s-pixel crossed by σ and let $\hat{\sigma}$ be the spherically snap rounded polyarc induced by σ . The directed Hausdorff distance $d_H(\hat{\sigma}, \sigma)$ is no larger than the diameter of the circumcircle of ξ_j .*

Proof. Let $s = G_P(\sigma)$ be the projection of an urarc to a face of the isocube. From the properties of the projection, s is a line segment on the face f_i . The polysegment $\hat{s} = SR(s)$ is contained in the Minkowski sum $s \oplus p_k$ for any p-pixel on f_i as this is one of the properties of planar SR. It is also known that the Gnomonic projection preserves topology for features on a single hemisphere so $\hat{\sigma} \in G_P^{-1}(s \oplus p_k)$. For any point $p_1 \in \hat{s}$ there exist a point $p_2 \in s$ such that $p_1 \in p_2 \oplus p_k$ so $G_P^{-1}(p_1) \in G_P^{-1}(p_2 \oplus p_k)$ and $G_P^{-1}(p_2 \oplus p_k)$ is contained in the circumcircle of ξ_j centered at $G_P^{-1}(p_1)$ \square

Lemma 2.3. *Let ξ_k and ξ_j be two registered boundary-connected s-pixels that were registered due to the arc σ . Let $\hat{\sigma}_c$ be the small geodesic arc connecting the reference points of ξ_k and ξ_j with respect to the arc σ . Let ψ_k and ψ_j be the circumcircles of the s-pixels ξ_k and ξ_j respectively and let $\psi = \max(\psi_k, \psi_j)$, the circumcircle with the maximal diameter. The directed Hausdorff distance $d_H(\hat{\sigma}_c, \sigma)$ is no larger than the diameter of ψ .*

Proof. By contradiction, assume that there is a point p on $\hat{\sigma}_c$ from which we measure $d_H(\hat{\sigma}_c, \sigma)$ and this distance is larger than the diameter of ψ . Place ψ on the sphere such that its center coincides with the spherical connection point for which we created $\hat{\sigma}_c$. The reference points of ξ_k and ξ_j are contained within the placed ψ and it is convex (see Appendix A.1) therefore ψ contains $\hat{\sigma}_c$. Place ψ such that its center coincides with p , it still contains the spherical connection point. When we calculated the directed Hausdorff distance between p and σ we found the shortest distance between them and it was larger than the

diameter of ψ but there is a point within ψ which is on σ (the spherical connection point) leading to a contradiction. \square

Theorem 2.4. *Let σ be an urarc and $\hat{\sigma} = SSR(\sigma)$ and let ξ_i be the s -pixel with the largest circumcircle crossed by σ . The directed Hausdorff distance $d_H(\hat{\sigma}, \sigma)$ is no larger than the diameter of the circumcircle of ξ_i .*

Proof. In lemma 2.2 we show this for an urarc that is contained entirely inside a face. Lemma 2.3 shows this for a connection arc. A snap rounded arc can be described as alternating concatenation of polyarcs that are contained entirely within a face and connection arcs. The faces are disjoint except at the edges or vertices of the isocube, therefore we get a simple polyarc within the asserted directed Hausdorff distance from the given urarc. \square

It remains to show that this scheme preserves the topological property as it was defined in the Introduction. On each face we apply the planar SR so all that remain to show is that the connection arcs do not intersect the rounded arcs or themselves (except at the endpoints).

Let f_i be a square face of the isocube and g_i be the closed square whose corners are the reference points of the corner pixels (pixels that touch at least two edges) of that face. We define the *forbidden region* of the face as $f_i^{\text{forb}} = f_i \setminus g_i$ (see Figure 2.3, g_i is the white portion of the grid).

Lemma 2.5. *After SR on the face f_i the forbidden region f_i^{forb} does not contain vertices or segments of the rounded arrangement on f_i .*

Proof. By contradiction. It is impossible for f_i^{forb} to contain a segment without at least one of its endpoints due to the way we project the arrangement to f_i . If the forbidden region contains an endpoint, during SR this boundary p-pixel was heated and the endpoint was moved to the reference point of the pixel which is outside the forbidden region. \square

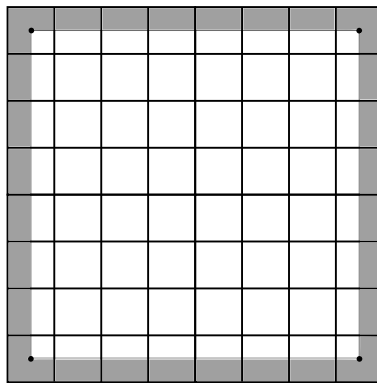


Figure 2.3: The forbidden region of a square face (marked in gray).

Lemma 2.6. *No two connection arcs intersect except at the endpoints.*

Proof. By contradiction, assume that there are two connection arcs that intersect at their interiors. The grid on the sphere within the forbidden region (can be considered as $G_{\mathcal{P}}^{-1}$ of f_i^{forb}) has two types of vertices. Degree 3 vertices which form an intersection point of three s-pixels, each from a different face of the isocube and degree 4 vertices which form an intersection point between two s-pixels on one spherical face and two s-pixels on another spherical face. In the case of a degree 4 vertex v , there are four possible connection arcs that can be created between the s-pixels. Only two of the possible connection arcs intersect in the interior. Those connection arcs are induced by an urarc that intersects v . It is sufficient to show that both intersecting arcs cannot exist simultaneously. We define v in the inclusive boundary of one of the four s-pixels therefore every connection arc that is created due to an urarc passing through v must start at the containing pixel. Therefore it is impossible to have two such connection arcs simultaneously. Figure 2.4 illustrates the planar view of this case. The degree 3 vertex is relevant only when an urarc intersects the vertex v (see Figure 2.5 for an illustration). In this case a connection arc is created between the s-pixel that contains the vertex and the s-pixels that the urarc intersects around the vertex. There are two such connection arcs that intersect only at the endpoint. \square

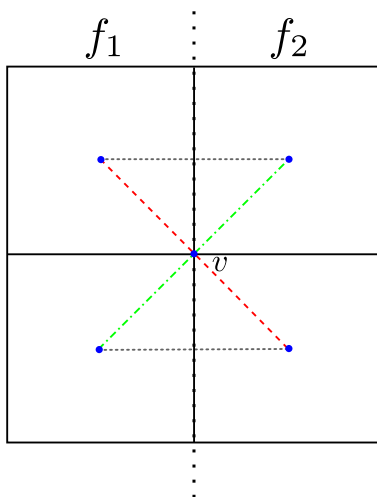


Figure 2.4: An illustration of possible connection arcs between four pixels. The diagonal connection arcs are valid only when an inducing urarc passes through v but because the inclusion rules for v only one of the diagonal connection arcs may exist.

Theorem 2.7. *The original and the rounded arrangement are topologically equivalent up to the collapsing of features.*

Proof. From the properties of planar SR the theorem holds inside a face. The Gnomonic projection does not change topology of features on a single face, thus the projected arrangements of arcs conform as well. The projected arrangements are disjoint. From lemma 2.6 we know that connection arcs are disjoint except at the endpoints and are disjoint from the projected arrangement because the connection arcs are all contained in the forbidden regions. Since the connection arcs do not introduce new intersections, the theorem holds. \square

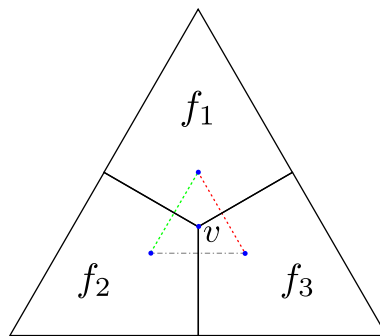


Figure 2.5: An illustration of possible connection arcs between three pixels.

3

The DGGS Approach

We extend the ideas presented in the simple approach by employing an approach used in DGGS. Typically, in practical usage of grids defined on the sphere a *Platonic Solid*¹ is used as an initial representation of the sphere. The best representation of the unit sphere by a Platonic Solid is achieved when an icosahedron is inscribed in it. This is because the icosahedron has the smallest face area of all the Platonic Solids (when inscribed in the unit sphere) yet its vertices are hard to represent using small bit-length numbers. In fact, the next best Platonic Solid to use, in terms of coordinate bit length and face size, after the isocube would be the octahedron (see Figure 3.1). We base our new construction on an octahedron inscribed in a unit sphere. An octahedron inscribed in a unit sphere has all its vertices on the sphere and their coordinates can be represented by small integer values. The tiling that we use employs the triangular faces of the octahedron as the basis for the pixels.

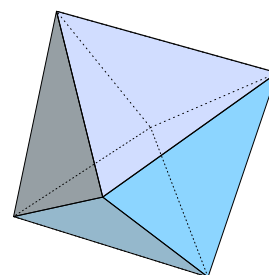


Figure 3.1: The octahedron

The rest of the chapter is organized as follows: First, we describe the subdivision of the octahedron. Next we provide an in-depth analysis of the grid types in the plane that preserve the desirable topological property of the standard planar Snap Rounding process. Finally, we use the conclusions from the analysis to generate a planar grid on the faces of the octahedron which preserves both the topological and the geometric properties of planar SR and describe the SSR process with the octahedron as the base Platonic Solid.

¹A regular convex polyhedron composed of congruent faces.

3.1 The Subdivision Process

We build our tiling using the *Class I Aperture 4 triangle hierarchy* [SWK03] method. The construction is recursive. We start with the faces of the octahedron as our cells. At each step of the recursion we iterate over the cells and create a new vertex at the center of each edge of a triangular cell. Next, in each cell we connect the three new vertices creating four congruent triangular cells.

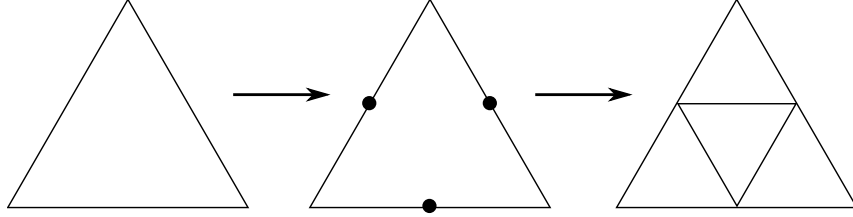


Figure 3.2: Triangular face subdivision

After ρ iterations of the subdivision of the initial faces we are left with $8 \times 4^\rho$ triangular cells overall. We start with an *axis-aligned* octahedron namely, an octahedron with vertices at $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, $(0, 0, \pm 1)$. We define the reference point of a triangular face as the center of its circumcircle. The bit length of the circumcenter equals the bit length of the longest coordinate plus $O(1)$ (see Appendix A.2 for a proof). Therefore, such a subdivision has a similar effect in terms of the bit length of the pixels as in the subdivision of the square faces of an isocube.

We introduce a label for each face of the octahedron: The coordinates of the centroid of that face, namely $(\pm \frac{1}{3}, \pm \frac{1}{3}, \pm \frac{1}{3})$. We assign each face its inclusive and exclusive boundary. The following list is the assignment of the inclusive boundary for each face:

- The face labeled $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ is associated with the open edges $((0, 1, 0), (0, 0, 1))$, $((0, 1, 0), (1, 0, 0))$ and the vertices $(0, 0, 1), (0, 1, 0)$.
- The face labeled $(\frac{1}{3}, -\frac{1}{3}, \frac{1}{3})$ is associated with the open edges $((1, 0, 0), (0, 0, 1))$, $((1, 0, 0), (0, -1, 0))$ and the vertex $(1, 0, 0)$.
- The face labeled $(-\frac{1}{3}, -\frac{1}{3}, \frac{1}{3})$ is associated with the open edges $((0, -1, 0), (0, 0, 1))$, $((0, -1, 0), (-1, 0, 0))$ and the vertex $(0, -1, 0)$.
- The face labeled $(-\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ is associated with the open edges $((-1, 0, 0), (0, 0, 1))$, $((-1, 0, 0), (0, 1, 0))$ and the vertex $(-1, 0, 0)$.
- The face labeled $(\frac{1}{3}, \frac{1}{3}, -\frac{1}{3})$ is associated with the open edge $((0, 1, 0), (0, 0, -1))$ and the vertex $(0, 0, -1)$.
- The face labeled $(\frac{1}{3}, -\frac{1}{3}, -\frac{1}{3})$ is associated with the open edge $((1, 0, 0), (0, 0, -1))$.
- The face labeled $(-\frac{1}{3}, -\frac{1}{3}, -\frac{1}{3})$ is associated with the open edge $((0, -1, 0), (0, 0, -1))$.
- The face labeled $(-\frac{1}{3}, \frac{1}{3}, -\frac{1}{3})$ is associated with the open edges $((-1, 0, 0), (0, 0, -1))$.

There are two limitations for the selection of the associated edges and vertices (the reason for these limitation is explained in the next sections). Each face of the octahedron must be associated with at least one edge and the association must be done in such a way that no degree 6 vertices will be created when we subdivide the faces to create the grid.

3.2 Grid Types for which SR Preserves Topology

At this point we ask a general question (a concrete answer for which is needed here). Which types of planar pixels (p-pixels) would preserve the topological properties of planar SR? (The geometric properties are easier to maintain and prove.) Furthermore, we show that for certain grids the choice of *reference point*, namely the point through which the polysegments will snap to in each pixel is immaterial. It has to be, however, the same point inside each pixel. These properties are needed for our usage of the DGGs scheme. For example, Ben-Moshe *et al.* [BMBS09] show why the standard SR scheme would not work in the case of triangular tessellations of the plane.

The Deformation

Guibas and Marimont [GM98] look at the SR process as a continuous deformation of the segments from the ursegments to the polysegments. First, the ursegments are divided into *fragments* by adding vertices (nodes) at their intersection points with the boundaries of the hot pixels. A fragment is *external* if it does not intersect the interior of a hot pixel and *internal* otherwise. When an ursegment intersects two adjacent hot pixels, a zero-length external fragment is created on the boundary of the two hot pixels at the intersection point. During the first stage, each hot pixel is contracted in the x -direction toward its center dragging the nodes with the boundary of the pixel until the pixel becomes a hot “stick”. After the first stage the internal fragments become nodes on the hot stick and the external fragments connect the hot sticks. In the second stage, each hot stick is contracted in the y -direction to the center of the original pixel. Once the second stage is complete, the external fragments constitute the snap rounded arrangement. Guibas and Marimont show that during the deformation, no external fragment crosses into a contracting hot pixel and thus a vertex never crosses over the interior of a (deforming) segment.

We extend this observation to a more general setting as we aim to use DGGs grids. Consider a single pixel of a general grid as a convex polygon P . We shrink the hot pixels in a single stage from time $t = 0$ to $t = 1$ by moving the vertices of the pixel at a constant speed (per vertex) toward the reference point. During this deformation process, the pixel is always a homothetic copy of the original pixel and at time $t = 1$ it is reduced to the reference point. We start the analysis of this process with two auxiliary lemmas.

Lemma 3.1. *Let Q be a convex polygon in the plane. If p is a point in the plane not contained in Q then for every $t \in [0, 1]$ it holds that $(p \oplus -tQ) \cap (1 - t)Q = \emptyset$.*

Proof. Given two polygons A, B in the plane, where A contains the origin, it is well known that $A \cap B \neq \emptyset$ if and only if the Minkowski sum $B \oplus -A$ contains the origin. (See,

e.g., [dBvKOS00, Chapter 13].)

This, for example, immediately shows that the lemma holds for $t = 1$: As Q does not contain p , $p \oplus -Q$ does not contain the origin.

Assume that for some value $t' \in [0, 1]$ the assertion of the lemma does not hold. Namely the polygons $A := (1 - t')Q$ and $B := p \oplus -t'Q$ intersect. Notice that A contains the origin, and hence $B \oplus -A$ contains the origin. However, $B \oplus -A = p \oplus -t'Q \oplus (t' - 1)Q = p \oplus -Q$, which we already know does not contain the origin. A contradiction, which proves the assertion of the lemma. □

Lemma 3.2. *Let h_1 and h_2 be two shrinking hot pixels and let $s_{\text{frag}}(t)$ be an external fragment with endpoints on the boundaries of h_1 and h_2 at time $0 \leq t \leq 1$. The external fragment $s_{\text{frag}}(t) \subset s_{\text{frag}}(0) \oplus (t \cdot (-P))$, where P is the pixel-polygon with its reference point at the origin.*

Proof. Without loss of generality, let $s_1(t)$ be the endpoint of $s_{\text{frag}}(t)$ on the boundary of h_1 at time t . Consider some time frame $[0, t']$, during which $s_1(t)$ moves with a constant speed toward the reference point of h_1 . Assume that the reference point of h_1 coincides with the origin O . The position of $s_1(t)$ is therefore $t \cdot \overrightarrow{(s_1(0), O)}$. Take the set of such vectors over all possible positions of $s_1(0)$ on the boundary of P and move them such that they emanate from the origin. The result is the boundary of the polygon $(t \cdot (-P))$ (Figure 3.3 illustrates the vectors induced by one edge of P) and thus $s_1(t) \in s_1(0) \oplus (t \cdot (-P))$. During the deformation the endpoints of $s_{\text{frag}}(t)$ move with the greatest speed therefore we can generalize the observation for any internal point of $s_{\text{frag}}(t)$ which means that $s_{\text{frag}}(t) \subset s_{\text{frag}}(0) \oplus (t \cdot (-P))$. □

Theorem 3.3. *Snap rounding applied to an arrangement of segments with a grid which is a tiling of the plane with identical (in terms of shape and orientation) convex polygonal pixels, maintains the topology preserving property of the SR process for any choice of a fixed reference point inside the pixel.*

Proof. To prove that the topology is maintained we show that during our deformation no external fragment crosses over into a shrinking hot pixel. Without loss of generality, let $s_{\text{frag}}(t) = \overline{s_1 s_2}$ be an external fragment with endpoints on the boundary of the shrinking hot pixels $s_1 \in h_1$ and $s_2 \in h_2$, and let h_3 be another (different) shrinking hot pixel. We know that $s_{\text{frag}}(0)$ does not intersect h_3 . Assume to the contrary that $s_{\text{frag}}(t')$ crosses into h_3 and let $t' > 0$ be the first time when $s_{\text{frag}}(t')$ touches its boundary. Assume now, without loss of generality, that $P := h_3$ contains the origin and that the origin coincides with the reference point. Since $s_{\text{frag}}(0)$ did not cross h_3 , it follows from Lemmas 3.1 and 3.2 that $s_{\text{frag}}(0) \oplus (t' \cdot (-P))$ is disjoint from $(1 - t') \cdot P$, and therefore it is impossible for $s_{\text{frag}}(t')$ to touch the boundary of the shrunk h_3 . □

Corollary 3.4. *Every tiling of the plane with either (i) identical parallelograms or (ii) identical hexagons having parallel opposite edges preserves the topological property of SR, for any selection of a reference point in the tile.*

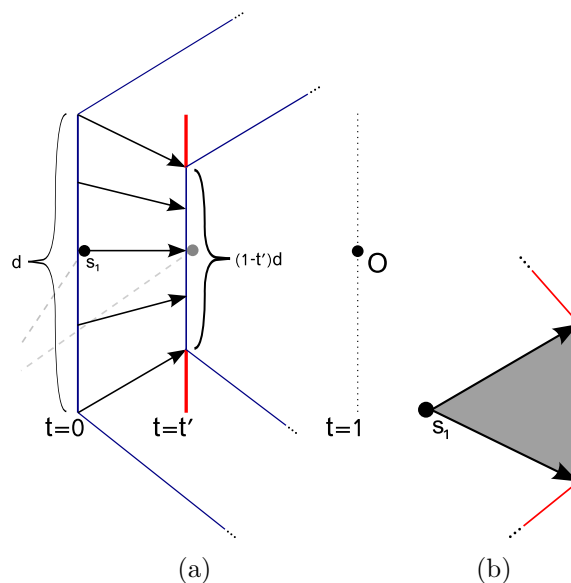


Figure 3.3: (a) An illustration of an edge of a polygon P during the deformation to the origin O . The point s_1 is an endpoint of an external fragment (dashed line). The sum of the lengths of the thick (red) sections is equivalent to the length of the corresponding edge in (b). (b) The union of the vectors induced by considering all the possible initial position of s_1 on one edge of P at time t' when they originate at s_1 .

It is well known that it is impossible to tile the plane with a convex n -gon for $n \geq 7$ therefore we consider only tilings of triangles, quadrilaterals, pentagons and hexagons. The plane can be tiled with any triangle or convex quadrilateral but our tiling must consist of polygons with identical orientation. Clearly such a tiling is impossible for triangles or pentagons even if we allow a vertex to touch the edge of another polygon as they cannot be symmetric on both axes. In the case of quadrilaterals and hexagons such a tiling would work only when the opposite angles and sides are equal.

3.3 Defining Pixels

Back to SSR with a grid induced by the octahedron, we use Corollary 3.4 to define a new type of pixels on a triangular face. On every face of the octahedron we define new x and y axes with respect to the axes of our setup, the octahedron and the sphere. The new axes of each face are presented in Figure 3.4 and summarized in Table 3.1.

We now define the pixels on each face of the octahedron as follows (see Figure 3.5 for an illustration). We start from a boundary edge of the triangular face which is included in that face. There are triangles with two possible orientations o_s — similar to the orientation of the face and o_o — opposite orientation. We merge each triangular pixel with orientation o_s with its left-hand side neighbor (with orientation o_o) thus creating a parallelogram pixel. For triangles in the last column that do not have a neighbor we create a *phantom* triangle (outside the boundary of the face) with orientation o_o and merge the two triangles. We define

Face	X	Y
$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$X - Y$	$-X - Y + Z$
$(\frac{1}{3}, \frac{1}{3}, -\frac{1}{3})$	$X - Y$	$-X - Y - Z$
$(-\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	$X + Y$	$X - Y + Z$
$(-\frac{1}{3}, \frac{1}{3}, -\frac{1}{3})$	$X + Y$	$X - Y - Z$
$(\frac{1}{3}, -\frac{1}{3}, \frac{1}{3})$	$X - Y$	$X + Y + Z$
$(\frac{1}{3}, -\frac{1}{3}, -\frac{1}{3})$	$X - Y$	$X + Y + Z$
$(-\frac{1}{3}, -\frac{1}{3}, \frac{1}{3})$	$X + Y$	$X + Y + Z$
$(-\frac{1}{3}, -\frac{1}{3}, -\frac{1}{3})$	$-X + Y$	$X + Y - Z$

Table 3.1: The definition of 2D coordinate axes on each face of the octahedron.

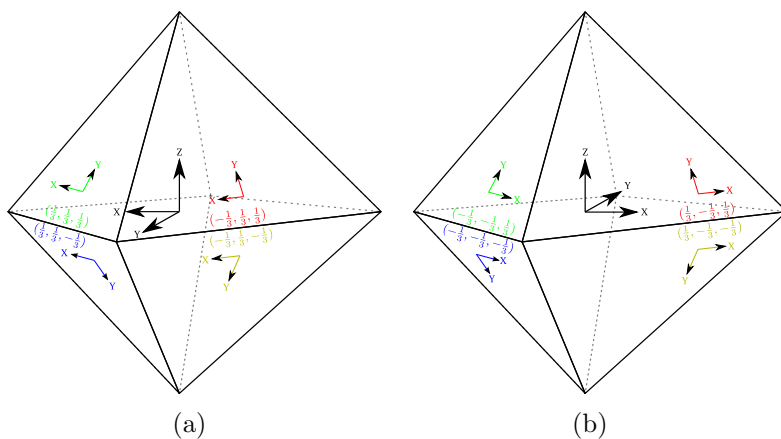


Figure 3.4: An illustration of the 2D coordinate axes on each face relative to the 3D coordinate axes of the octahedron.

the reference point of each parallelogram pixel to be the circumcenter of the triangle with orientation o_s . This selection guarantees that all the reference points of the parallelogram pixels are in the interior of the face of the octahedron and not in the phantom regions. The inclusive region of such a pixel is the part of the parallelogram boundary that was the boundary of the triangle with orientation o_s . Consider the vertices which are in the closure of two adjacent p-pixels (marked with a dashed circle in Figure 3.5). In the planar SR scheme they are in the inclusive boundary of the p-pixels left to the to the leftmost p-pixels of the face, which do not exist in our setting. We map these vertices to be in the inclusive boundary of the top p-pixel with phantom region which is incident to the vertex. We map the topmost vertex to its only incident p-pixel. This selection is arbitrary as no segment extends outside the original face. We define the s-pixels by projecting the non phantom parts of the p-pixels onto the sphere using inverse Gnomonic projection.

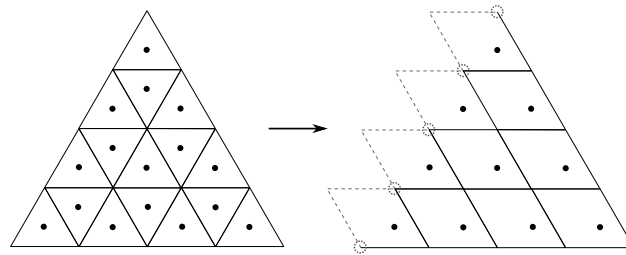


Figure 3.5: Merging the triangular cells into parallelogram p-pixels on a face. The phantom boundary parts are denoted in a dashed gray line. The vertices marked with a dashed circle are in the closure of two adjacent p-pixels with phantom region.

3.4 Additional Notation

To show that the properties of SSR still hold for our new selection of pixels and their reference point we use additional notation. Some previous notation is adapted to suit the new scheme.

- P - The axis-aligned octahedron.
- f_i - A face of P .
- G_P - The central Gnomonic projection from the unit sphere onto the faces of P .
- G_P^{-1} - The inverse transformation of G_P : the central Gnomonic projection from the faces of P onto the sphere.
- $C(p_i)$ - The smallest circle bounding a p-pixel p_i whose center coincides with the reference point of the p-pixel.
- $SR(s)$ - The polysegment induced by the ursegment s after applying the augmented version of the planar snap-rounding algorithm.
- $SSR(\sigma)$ - The polyarc induced by the urarc σ after applying the DGGS version of spherical snap-rounding.

3.5 The Spherical SR Process

The spherical snap rounding process in the DGGs approach is quite similar to the isocube case presented in Chapter 2. As before, the input is an arrangement $\mathcal{A}(\mathcal{C})$ of geodesic arcs on a unit sphere centered at the origin and an integer ρ . We inscribe an axis-aligned octahedron inside the sphere. On every face of the octahedron we perform ρ steps of Class I Aperture 4 subdivision and merge each triangle with its left neighbor as described in section 3.3 creating p-pixels. We project the arrangement $\mathcal{A}(\mathcal{C})$ onto the faces of the octahedron using central projection. A single arc may create a segment over several faces, in this case we consider the segment to be *broken* and the point of the intersection between a segment and an edge of the octahedron as the *break-point* of the segment (a segment can have more than one break-point). Note that if a projected segment ends exactly on the boundary we use the inclusion rules that were previously defined to determine whether this is a break-point. We define a p-pixel to be hot if it contains a vertex in the projected arrangement or a break-point. For every break-point we create two vertices infinitesimally close to it, one in each p-pixel on a different face, and register the two p-pixels as *boundary-connected* and **hot**. Each vertex of the projected arrangement is replaced by a reference point of the hot p-pixel containing it and each projected segment s is replaced by a polygonal chain going through the hot p-pixels in the same order they are met by s on a single face. After the rounding process, all the rounded segments are projected back onto the sphere using G_P^{-1} . For each registered pair of connected p-pixels, their corresponding s-pixels reference points are connected with a small geodesic arc.

3.6 The Topological and Geometric Properties of DGGs SSR

Lemma 3.5. *Let r be the radius of the circle $C(p_i)$ on a face of P and s a segment contained in f_i . Let D be a disc of radius r centered at the origin. After SR on the face f_i the polysegment \hat{s} corresponding to the ursegment s is contained within the Minkowski sum of s and D .*

Proof. We consider the hot pixels met by s in the order they are met by it. Now consider the segment part between two consecutive hot p-pixels. The reference points of the two p-pixels are within the Minkowski sum of s and a disc with radius r because this disc envelopes any p-pixel on f_i . The Minkowski sum is convex and therefore it contains the entire segment part between the two p-pixels. Repeating on all fragments of the segment shows that $\hat{s} \in s \oplus D$ \square

Observation 3.6. *Lemma 2.2 and Lemma 2.3 hold for the DGGs version of SSR.*

Theorem 3.7. *Let σ be an urarc and $\hat{\sigma} = SSR(\sigma)$ (the spherically snap rounded version of σ) and let ξ_i be the s-pixel with the largest circumcircle crossed by σ . The directed Hausdorff distance $d_H(\hat{\sigma}, \sigma)$ is no larger than the diameter D_i of the circumcircle of ξ_i .*

Proof. We show this for an urarc that is contained entirely inside a face and a connection arc separately. A snap rounded arc can be described as a concatenation of polyarcs that

are contained entirely within a face and connection arcs. Therefore for every point $\alpha \in \hat{\sigma}$, $d_H(\alpha, \sigma) \leq D_i$ thus $d_H(\hat{\sigma}, \sigma) \leq D_i$.

□

It remains to show that this scheme preserves the topological property as it is defined in the Introduction. The next theorems show that: (a) During the SR process on a face no vertex crosses over an edge and no new vertices are created. (b) The connection arcs do not intersect the rounded arcs or themselves (except at the endpoints).

Theorem 3.8. *Let f_i be a triangular face tiled with parallelograms and let $\mathcal{A}(\mathcal{S})$ be an arrangement of segments contained completely in f_i . Then during the SR process on f_i no vertex crosses over an edge and no new vertices are created.*

Proof. We define the pixels on the face f_i as parallelograms (as described above) and select the reference point of the pixel to be within each such pixel. Because the arrangement $\mathcal{A}(\mathcal{S})$ is contained within f_i we can apply Theorem 3.3 directly to the planar SR scheme. □

Let f_i be a triangular face of the octahedron and g_i the triangle whose corners are the reference points of the corner pixels (pixels that touch at least two edges) of that face. We define the *forbidden region* of the face as $f_i^{\text{forb}} = f_i \setminus g_i$ (see Figure 3.6).

Observation 3.9. *After SR on the face f_i the forbidden region f_i^{forb} does not contain vertices or segments in the rounded arrangement on f_i .*

The proof is identical to the proof of Lemma 2.5

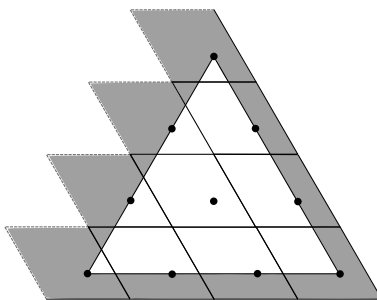


Figure 3.6: The forbidden region of a triangular face (shaded).

Lemma 3.10. *No two connection arcs intersect except at the endpoints*

Proof. By contradiction, assume that there are two connection arcs that intersect at the interior. We examine all the possible configurations of the boundary pixels. There are three types of vertices on the face boundary: A vertex on the boundary of two faces with the same sign in z -coordinate are adjacent, a vertex on the boundary of two faces with different sign in the z -coordinate and a vertex incident to four different faces. A connecting arc created between s-pixels that share an edge cannot intersect another connection arc because connection arcs are always extended between the reference points of the pixels in the interior

of the connected s-pixels. The remaining case is when two connection arcs extend between pixels that share only the vertex. In this case an original urarc that caused the registration had to pass through the vertex but the vertex belongs to only one s-pixel therefore any connection arc that is created in this case will end up in a single s-pixel and will not intersect any other connection arc.

□

Theorem 3.11. *The original and the rounded arrangement are topologically equivalent up to the collapsing of features.*

Proof. The proof of Theorem 2.7 applies here verbatim.

□

4

SSR With Labeled Pixels

In the previous sections we have concentrated on rounding the coordinates of the vertices of the arrangement to coordinates of small bit length. In this section we explore a different approach of rounding to *labeled pixels*. More formally, we assign a unique label to every pixel and represent the rounded polyarc as a series of labels — the labels of the hot pixels the polyarc intersects in the order it intersects them. By taking this approach we are no longer restricted to pixels with center coordinate of small bit length. The goal here is to overcome the problem that the initial faces are quite large and therefore the distance between the face and the sphere varies a lot. In the isocubical SSR the faces would touch the sphere in the center while the distance of the face corners to the sphere was $\sqrt{2} - 1 \approx 0.414$. In the DGGs SSR the corners of the face touch the sphere but the distance between the center of the face and the sphere is $1 - \sqrt{\frac{1}{3}} \approx 0.423$. The platonic solid inscribed in a unit sphere with the smallest faces (in terms of area and perimeter) is the icosahedron. The main reason not to use the icosahedron in the DGGs approach is that the centers of the faces cannot be represented using small bit length coordinates. Once we free ourselves from the restriction on the bit length we can adapt the algorithm to work with the icosahedron. A similar approach was presented by Ben-Moshe *et al.* [BMBS09] by enumerating the hexagonal pixels instead of using the coordinates of their centers. The disadvantage of this approach is that the output of the algorithm is no longer an arrangement of rounded arcs but also a map of the enumerated pixel centers to their coordinates. This output is non-standard and it cannot be passed directly to subsequent algorithms.

4.1 Labeling the Pixels

We use the quaternary base to label the pixels. Each initial face of the polytope is assigned a value with a fixed number of digits (sufficient to be unique among all the faces). For

the icosahedron we label the initial faces with unique three-digit labels (i.e 000, 001, ..., 103). When we subdivide a triangular face, the labels of the four newly created subfaces are a concatenation of the digits 0,1,2,3 to the end of the label of the subdivided face (see Figure 4.1). The decision which subface gets which label does not affect the algorithm, therefore any order is sufficient. This labeling scheme generates a unique label for each triangular subface.

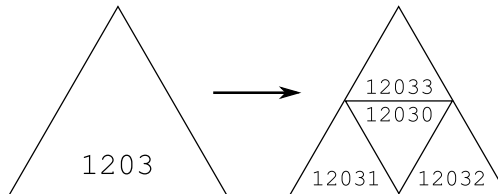


Figure 4.1: An example of assigning labels to new pixels after a subdivision step.

4.2 Labeled Spherical SR Process

The SSR process is slightly different from the SSR in the DGGS case. The input to the new algorithm is the set of arcs C , the subdivision parameter ρ , and η — the orientation of the icosahedron with respect to the Cartesian axes. The output is a list of polyarcs described as a series of labels. We assign each edge of the icosahedron to a face such that each face contains at least one edge on its inclusive boundary. This can be done in various ways, an example is presented in Figure 4.2. The rest of the process remains the same as in the DGGS case, the arcs are projected onto the faces, augmented SR is run on each face and the result is projected back onto the sphere. Next, the connection arcs are created and the final result is reported as mentioned above.

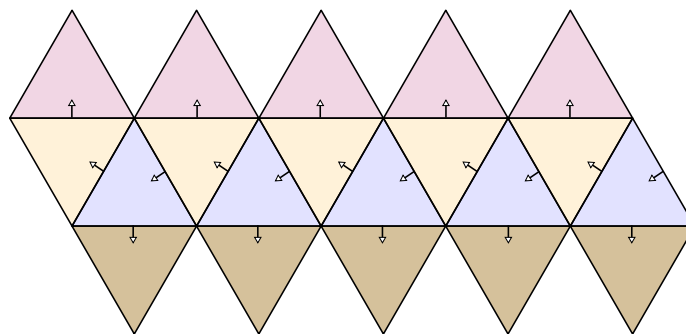


Figure 4.2: An illustration of an icosahedral net with a partial assignment of edges to faces such that each face includes at least one edge on its boundary. An arrow from an edge points toward the face that includes this edge on the boundary. The unassigned edges can be arbitrarily assigned.

We omit the proofs of the topological and geometric properties of the snap rounded arrangement of arcs, which hold in this case with minor adjustments.

4.3 Conclusion

The labeling approach allows the use of wide variety of polytopes with triangular faces as the base polytope for the SSR algorithm. During our discussion we did not assume anything on the base polytope except that it has triangular faces and that in the resulting spherical grid there are no degree 6 vertices. Using a base polytope inscribed in a unit sphere which has more faces means that it is possible to move the faces closer to the unit sphere. This allows a better initial approximation of the unit sphere thus reducing the inherent distortion of the spherical pixels caused by the projection which in turn creates better rounding results (in terms of distortion).

The aforementioned disadvantage of the labeling approach (the lack of specific numerical coordinates) makes it unpractical for real world uses. In many cases the rounded arrangement is an input for another algorithm. With the labeling approach, any algorithm using the rounded arrangement would have to make adjustments to be able to process the output (and the additional mapping between pixel centers and coordinates). This in mind, the labeling approach can still be used in algorithms where the coordinates of the pixel reference point are not important and the required output is the “hot” pixels and the edges between them (e.g., for graphical display). In these cases the labeling approach can be quite useful.

5

Implementation

In this chapter we present our implementation of both the isocube and the DGGSSR schemes. We have implemented both algorithms using CGAL with emphasis on using the 2D Arrangement [WFZH07], the 2D Snap Rounding [Pac09], and the prototypical Arrangement-on-Sphere traits class. Before we dive into the details of the implementation itself we describe the aforementioned building blocks in some detail.

5.1 CGAL Packages Used by the Implementation

We use several packages that come out-of-the-box when installing CGAL. The following sections give an overview of the packages and their contribution to our implementation.

5.1.1 2D Arrangement

The 2D Arrangement package (`Arrangement_on_surface_2`) supports the construction, manipulation, querying, and traversal of an arrangement of curves. The package includes generic implementations of two algorithmic frameworks, which give the ability to compute the zone of an arrangement, and line-sweep the plane, the arrangement is embedded on. The main construct of the package is the `Arrangement_2` class, which is parameterized by a traits class (modeling the *ArrangementBasicTraits_2* concept) and the DCEL (modeling the *Arrangement-Dcel* concept). The arrangement supports various types of curves such as line segments, rays, lines, Bézier curves, algebraic curves, and more. The DCEL can be extended to contain additional data within the vertices, edges and face records.

5.1.2 2D Snap Rounding

The 2D Snap Rounding package provides the function `snap_rounding_2` that is templated by the `SnapRoundingTraits_2` and input and output iterator types. This function is an implementation of the ISR algorithm described in [HP02] for snap rounding planar arrangement of segments (but it essentially allows us to run the standard SR by setting the number of iterations to 1). The `snap_rounding_2` function accepts the pixel size as an argument allowing the user to specify non-integer pixel sizes and a list of segments as the main input. It outputs a list containing a polygonal chain for each input segment. Each chain is represented by a list of points; each point is the center of a hot pixel, retaining the incidence order of the original segment with the hot pixels.

5.1.3 Arrangement of Geodesic Arcs on the Sphere

The prototypical Arrangement-on-Sphere traits class [FSH08a,FSH08b]¹ enables using geodesic arcs on a sphere as the inducing curves for the `Arrangement_2` class. The traits class `Arr_geodesic_arc_on_sphere_traits_2` models the `ArrangementBasicTraits_2` concept as follows: The sphere is parameterized as $\Phi = [-\pi + \alpha, \pi + \alpha] \times [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\phi_S(u, v) = (\cos u \cos v, \sin u \cos v, \sin v)$. The contraction points are: $p_s = (0, 0, -1) = \phi_S(u, -\frac{\pi}{2})$ is the south pole and $p_n = (0, 0, 1) = \phi_S(u, \frac{\pi}{2})$ is the north pole, where α is set such that the, *identification* curve (the curve that represents both the smallest and the largest u value in the parameter space) passes through the poles and the point $(-0.8, 0.6, 0)$, which is both on the unit sphere and has rational coordinates. An x -monotone curve is defined to be a geodesic arc that does not intersect the identification curve.

The geometry-traits class defines the `Point_2` type as an unnormalized vector in \mathbb{R}^3 , representing the intersection point of a ray emanating from the origin in the relevant direction with the sphere. An arc is represented by its two endpoints, and by the plane that contains the endpoint directions and the origin. The orientation of the plane and the source and target points determine which one of the two possible arcs is the represented one. This representation enables an exact and efficient implementation of all the required geometric operations, defined by the geometry-traits concept, using exact rational arithmetic. Efficiency is achieved by avoiding normalizing vectors and plane normals.

5.2 Implementation Details

We implement both SSR algorithms using GMP, the GNU Multi-Precision bignum library [5] that provides us with an implementation of rational numbers with unlimited precision. Throughout our implementation all the traits classes are parameterized using the CGAL Cartesian geometric kernel [BFG⁺09,HHK⁺07]. The implemented algorithm is divided into several phases. The following sections provide details on the implementation of each phase.

¹Arrangements embedded on parametric surfaces are supported as of version 3.4 albeit only partially documented.

5.2.1 Input

The input to the algorithm is a set of initial arcs, the parameter ρ , and a description of the underlying polytope. The initial arcs are provided via a text file that contains the coordinates of the initial input arcs. The full path to the input files and ρ is provided in the command line arguments. Throughout our implementation, the pixels are not actually constructed, instead the pixel size is calculated and later passed to the `snap_rounding_2` function.

There are two types of input files. The type is denoted by the first line in the file and can be either “0” or “1”. When the file type is “0” the coordinates of the arcs are given in spherical form as a four-tuple of real numbers (in floating-point representation) separated by comma (i.e. $\theta_1, \phi_1, \theta_2, \phi_2$). Each subsequent line must contain a single four-tuple and an end of line character. When a tuple is read, the coordinates are converted into the corresponding direction by using ϕ_S . For this conversion we use inexact arithmetic (this is the only place in the implementation where we use inexact arithmetics; all the following computation are carried out with exact rational numbers). When the file type is “1” the coordinates are given as two directions separated by a comma. Each direction contains the x, y , and z coordinates separated by a single space. Each line must contain only a single pair of such directions representing an arc. When type “1” input is processed the directions are read directly into `Point_2` and the smaller geodesic arc is created between the two directions. We will refer to such a direction simply as a point on the unit sphere. An example of each input file type can be found in Appendix A.3

The algorithm reads the underlying polytope from the provided OFF file [8]. For the isocube approach we use an isocube with side length 2 centered at the origin and for the DGGS approach we use an octahedron with vertices at $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, $(0, 0, \pm 1)$. The polytope is read into the `Polyhedron_3` class [Ket09] directly from the input OFF file using a specialized reader.

5.2.2 Distribution to Faces

In this phase we distribute the input arcs among the faces of the polytope and register the connection points using tools provided by the `Arrangement_on_surface_2` package and two additional classes.

- The `face_data` class — Represents a single face of the polytope and holds all the data and functionality needed to perform SSR within this face and its spherical equivalent. More precisely, the `face_data` class stores the arc parts that fall within the represented face, the supporting plane and two functions that can transform a point in 3D to and from its 2D representation on the supporting plane.
- The `face_distribution` class — Represents the array of `face_data` instances associated with a specific polytope. The size of the array is the number of faces in the polytope (denoted below by $\#Faces$). This class provides the functionality of answering queries regarding the inclusion rules of the edges and vertices of the polytope with respect to the faces.

After reading the coordinates of the arcs from the input file, we insert the arcs into a spherical arrangement $\mathcal{A}(\mathcal{C})$ and create an additional spherical arrangement $\mathcal{A}_{\text{Polytope}}$ from the input polytope. $\mathcal{A}_{\text{Polytope}}$ is created by projecting each edge of the polytope onto the unit sphere using G_P and inserting the resulting arc into the arrangement. Next, we assign a unique ID to every face of $\mathcal{A}_{\text{Polytope}}$ from the range $0 \dots \#Faces - 1$, we use these IDs as an index for the `face_distribution` class and for any future reference to the faces. We overlay $\mathcal{A}(\mathcal{C})$ (red) with $\mathcal{A}_{\text{Polytope}}$ (blue) and provide a custom traits class to the overlay function. This custom traits class tracks the intersections created during the overlay process and stores the vertices created due to an intersections between red and blue features as a list of connection points. Special care is needed when a connection point is incident to a vertex of $\mathcal{A}_{\text{Polytope}}$, as there is some ambiguity regarding which connection arc it represents. To avoid this ambiguity we record all the red edges that induce this connection point and use this data when we construct the connection arcs. More details on building connection arcs are given in section 5.2.4.

Finally, we remove all the edges and vertices that were created from a red face and a blue edge or vertex. We are left with an arrangement where each edge (arc) can be projected onto a single face of the polytope using G_P . We add the arcs from this arrangement to the `face_distribution` class. Each arc is added to the `face_data` that represents the face that this arc projects to. At the end of this phase we have a list containing all the connection points and a `face_distribution` data structure containing all the arcs split into their respective faces.

5.2.3 Snap Rounding on Faces

In this phase we run the planar Snap Rounding algorithm on each face of the underlying polytope by using the associated `face_data` structure. The `snap_rounding_2` function works on the xy -plane with square pixels therefore we adapt the data to this requirement. There are several differences between the two approaches. Therefore, we discuss the isocube and the octahedron separately.

Isocube: In the isocube algorithm the adaptation of the input is trivial as each face of the isocube is orthogonal to one of the axes. The 3D coordinates on the faces of the isocube are generated by intersecting the spherical direction with the respective plane. We project each coordinate on to the xy -plane by eliminating one dimension as described in Table 2.1. The `snap_rounding_2` function accepts the pixel size as one of its arguments. We supply the number $\frac{2}{2^\rho}$ to reflect the subdivision of the grid. Note that when $\rho = 0$ we do not subdivide the faces and the pixel is a square with side length of 2.

Octahedron: In the DGGs approach some additional preprocessing is needed to be able to use the `snap_rounding_2` function as the underlying polytope is an octahedron. We generate the initial 3D coordinates as in the isocube approach. Next, we project each coordinate to the xy -plane by removing the z -coordinate (and sometimes changing the sign of the x or y -coordinates as described in Table 3.1). This projection projects a face of the octahedron (which is an equilateral triangle in 3D) to a 2D right triangle in the xy -plane, with legs of unit length, such that the vertex between them coincides with the origin. In this projection the parallelogram pixels become squares in the xy -plane. We run the

`snap_rounding_2` function on the projected arrangement with some care, as the vertices and segments are snapped to the square pixels centers. The adjustment is performed after the completion of the `snap_rounding_2` function with pixel size of $\frac{1}{2^p}$ by translating the vertices in the resulting arrangement. Each vertex is translated by $\frac{1}{6 \cdot 2^p}$ along both axes toward the origin.

The snap rounded arrangements (from each face) are projected back onto the unit sphere and inserted into a single output arrangement, which is passed to the next phase.

5.2.4 Adding Connection Arcs

To complete the spherical snap rounding process we insert the connection arcs into the output arrangement from the previous phase. We do this by going over the connection points we collected during the distribution phase described in section 5.2.2 and creating connection arcs as needed. Note that when all the arcs that induced a connection point are contained in a single face, and this connection point was created on an edge included in the boundary of that face, there is no need to create a connection arc. A connection point can be mapped to a single connection arc by examining the s-pixels incident to it except when it coincides with a vertex that is on the boundary of more than two faces. When such a connection point is encountered the inducing arcs are examined and the correct connection arcs are constructed. Table 6.2 illustrates this special case. The created connection arcs are added to the output arrangement. At the end of this phase the output arrangement contains the spherically snap rounded version of $\mathcal{A}(\mathcal{C})$.

5.2.5 Output

The input of this phase is the spherically snap rounded version of $\mathcal{A}(\mathcal{C})$, which we obtain from the previous phase and can be used in subsequent calculations. We provide two methods of visualizing the resulting array. By default the result is exported to a VRML file². The resulting VRML contains a representation of the unit sphere and three arrangements. The default settings are: Red for the input arrangement $\mathcal{A}(\mathcal{C})$, blue for the arrangement representing the partitioning of the sphere into spherical faces by the underlying polytope, and green for the spherically snap rounded version of $\mathcal{A}(\mathcal{C})$. It is also possible to export both the input and the output to a KML [7] file by providing the full path for the respective files as a command line argument.

²We are using an extended VRML format that contains a special element, which represents an arrangement on the sphere. A standard VRML reader cannot display this file. Currently, the only program that can open the output file is the “Player” developed by Efi Fogel in the Applied Computational Geometry Lab [1]. The Player is based on a Scene Graph Algorithm Library (SGAL).

5.3 Measuring the Directed Hausdorff Distance on the Sphere

We have implemented additional helper functions to help us gather statistics on the quality of the output. The most notable helper function measures the directed Hausdorff distance between two arcs on the sphere. Some of the distance calculations are done using floating-point arithmetic, this is sufficiently accurate for our purposes as it is not part of the actual result for the algorithm. The distance between two points on the sphere can be easily measured in spherical-coordinates representation by calculating the length of the arc between them on a unit sphere. Our method of measuring the directed Hausdorff distance between two arcs assumes that one of the arcs lies on the equator (it lies on the xy -plane). We start with two general arcs, a source arc σ_s and a target arc σ_t . We measure the directed Hausdorff distance $d_H(\sigma_s, \sigma_t)$. We use the affine transformation class `Aff_transformation_3` provided by CGAL and rotate the sphere to make σ_t coincide with the xy -plane.

With this setting, the directed Hausdorff distance can be one of the following:

- The distance between an endpoint of σ_s to the equator, if this endpoint is in $VL(\sigma_t)$ (see Definition 1.1).
- The distance between an endpoint of σ_s and an endpoint of σ_t , if the respective endpoint of σ_s is not in $VL(\sigma_t)$.
- The distance between the point with the largest absolute z value on the underlying great circle of σ_s if this point is on σ_s and in $VL(\sigma_t)$.

We subdivide σ_s into (possibly three) parts by intersecting it with the two geodesics that make $VL(\sigma_t)$. For each part we perform the required calculation (as described above) and obtain the directed spherical Hausdorff distance between the part and σ_t . Next, we compare the distances and output the longest one.

6

Experimental Results

We present experimental results to show how both methods perform for various inputs. For each example we give a statistical summary of the result in a table and provide some drawings to illustrate the result. In general, an unrounded coordinate on the sphere is represented by two double precision [IEEE08] values (for example, in the Latitude-Longitude grid these are two angles) of 64 bits each with a total of 128 bits per coordinate. In our implementation each coordinate is composed of three rational values which allows some flexibility in the bit length as both the numerator and the denominator are integers and can be represented using the exact number of bits needed for the rational number. The maximum and average bit length of the coordinates in the arrangement are presented in all the examples.

We present Table 6.1 to give a feeling for how the spherical grids look on the sphere. Note how irregular the grids become near the vertices of the original projected faces.

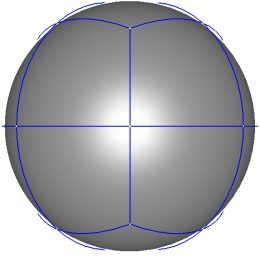
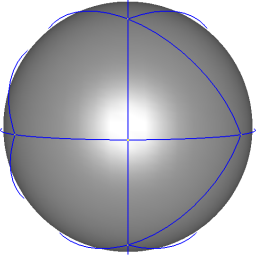
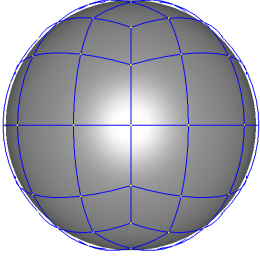
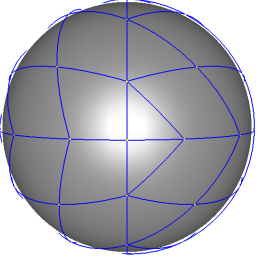
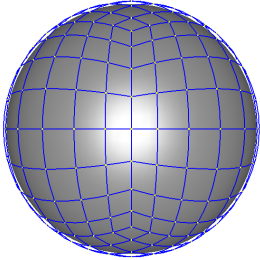
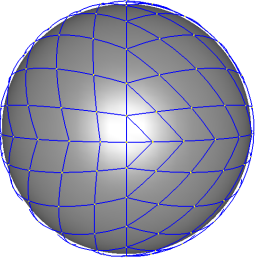
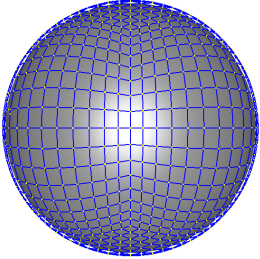
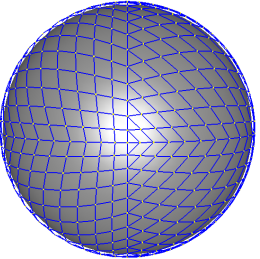
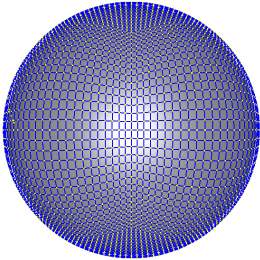
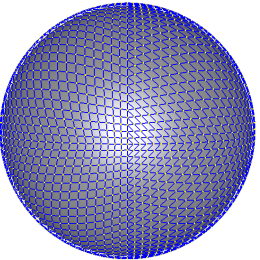
ρ	Isocube	DGGS
1		
2		
3		
4		
5		

Table 6.1: Examples of spherical grids.

6.1 Robustness

Before we give the details of the example inputs we remark about the robustness of our implementation. One of the biggest problems in implementing algorithms in computational geometry is their robustness. Many implementations cannot be robust because they use inexact number types and non-rigorous approximations. Our implementation uses generic components from CGAL, which are parameterized with exact numbers. This parametrization together with special care of degenerate cases makes our implementation robust. Table 6.2 illustrates some cases of degenerate inputs and the way they are handled in both our rounding algorithms.

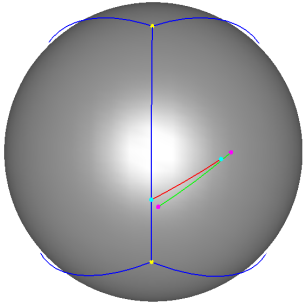
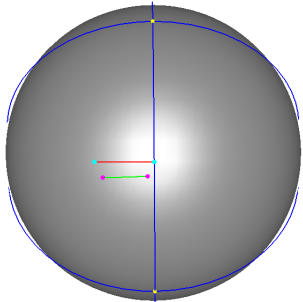
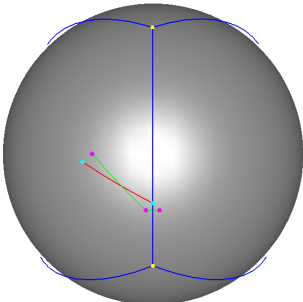
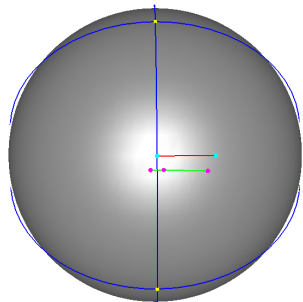
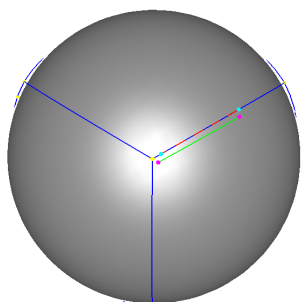
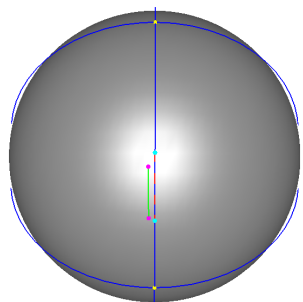
Description	Isocube	DGGS
An arc ending on an edge of a face that owns that edge. The resulting connection point does not create a connection arc because the arc is contained entirely in the face.		
An arc ending on an edge of a face that does not own that edge. The resulting connection point creates a connection arc into the owning face.		
An arc overlapping an edge of a face is rounded toward the owning face.		

Table 6.2: Examples of how degeneracies are handled.

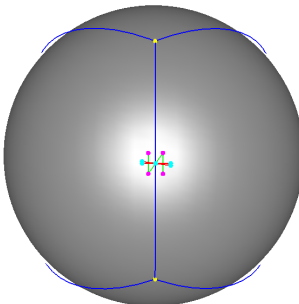
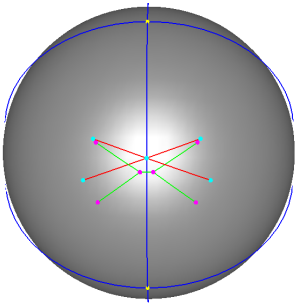
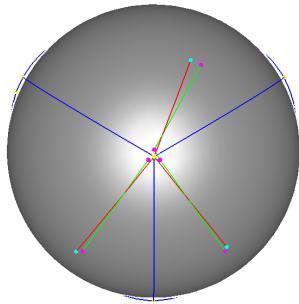
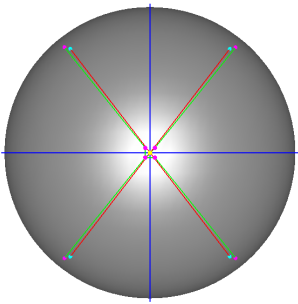
Description	Isocube	DGGs
Two arcs intersecting on a connection point. The resulting connection arc is between the pixels that include the connection point on their boundary.		
Several arcs intersecting on a vertex of several (three for the isocube and four for the DGGs) spherical faces. All the connection arcs originate in the s-pixel that includes the vertex on its boundary.		

Table 6.2: Examples of how degeneracies are handled (continued).

Abbreviation	Explanation
ρ	The number of subdivision iterations to perform on a face of the underlying polytope.
mhd	The maximum directed spherical Hausdorff distance between a rounded polyarc to the original urarc.
ahd	The average directed spherical Hausdorff distance between a rounded polyarc to the original urarc.
nsv	The number of urarcs that were contained entirely within a single hot pixels and reduced to a vertex in the rounded arrangement.
nca	The number of connection arcs created in the rounded arrangement.
mcl	The maximum number of bits needed to represent a single coordinate of a vertex in the rounded arrangement.
acl	The average number of bits needed to represent a single coordinate of a vertex in the rounded arrangement.
rtf	The running time in seconds of the full rounding algorithm.
rtsr	The relative part of the planar snap rounding algorithm within the full algorithm.
nvi	The number of vertices in the input arrangement.
nei	The number of edges in the input arrangement.
rdd	The relative difference between the mhd and the minimum mhd . $(mhd - \text{Min}(mhd)) / mhd$.

Table 6.3: Abbreviations.

6.2 Rounding Distance

In SSR the spherical pixels are not identical, which causes some distortion when rounding identical arcs located at different positions on a face. To illustrate this fact, we ran the following examples: On each run, we take a single input arc and place it in such a way that when projected onto the face of the underlying polytope its rounded version moves a constant distance from the projected arc. We do this separately for the two algorithms using $\rho = 4$. In the isocube algorithm, in each run, we move the arc further from the boundary toward the center of the face. As expected, the directed spherical Hausdorff distance is increasing as the input arc moves toward the center of the face (see Table 6.4). The maximum distortion in the distance is around 35% at the edge of the face. In the DGGs algorithm we move the input arc across the face such that the projected segment is parallel to an edge of the face. The result in this case is opposite to the result of the isocube algorithm. When the arc is closer to the center of the face (Recall that the isocube contains the unit sphere while the icosahedron is inscribed in the unit sphere) the distortion is greater (see Table 6.5). This result is expected because a face of the isocube is closest to the sphere at its center while a

ahd	rdd
0.0215396	0.0000000 %
0.0234622	8.1944575 %
0.0254236	15.2771441 %
0.0273571	21.2650464 %
0.029176	26.1735673 %
0.0307783	30.0169275 %
0.0320546	32.8034042 %
0.0329017	34.5334740 %

Table 6.4: Rounding results of a single arc with respect to its distance from the spherical face boundary on a face of the isocube

ahd	rdd
0.0180329	74.6123474%
0.0204952	77.6624283%
0.0188096	75.6606733%
0.0098083	53.3237768%
0.0045781	0.0000000%
0.0165166	72.2816439%
0.0210585	78.2599425%
0.0199949	77.1035114%

Table 6.5: Rounding results of a single arc with respect to its distance from the spherical face boundary on a face of the octahedron.

face of the octahedron closest at the vertices.

6.3 Random Input

We tested both algorithms with a batch of random inputs in two configurations. In the first configuration we created an input of 200 random arcs on the sphere. Inserting the arcs into an arrangement results in 2636 vertices and 4672 edges. We ran the program with an increasing value of ρ . Tables 6.6 and 6.7 show the results. We started from $\rho = 5$, lower values of ρ incur a phenomenon where all (or most) of the pixels become hot and a grid is created on the sphere between the reference points of the hot s-pixels. This phenomenon can be observed by looking at the `nsv` parameter which is quite high for lower values of ρ but completely diminishes at $\rho \geq 15$. We see that the average bit length of the coordinates remains lower than in the input even when $\rho = 25$ and the average distance in radians is around 10^{-7} . In the second configuration we ran the algorithms on inputs with increasing number of random arcs while keeping $\rho = 15$. The inputs contain 20,50,100,200 and 300 arcs. The results for this experiment are presented in Table 6.8. The results show that as the number of arcs grows the relative time it takes to run the planar version of SR increases

ρ	Isocube				DGGS			
	mhd	ahd	nsv	nca	mhd	ahd	nsv	nca
5	0.0399914	0.0165321	1543	149	0.0788381	0.0283228	1704	145
7	0.0107987	0.00387748	514	194	0.0199101	0.006216	550	192
9	0.00256069	0.000913888	157	208	0.00488705	0.00150039	177	220
12	0.000326569	0.000117932	11	212	0.000638699	0.000182145	16	225
15	3.97099e-005	1.43818e-005	2	213	7.93889e-005	2.36877e-005	4	225
18	6.32156e-006	1.83132e-006	0	213	2.18426e-005	3.09808e-006	0	225
25	4.90098e-006	9.97513e-008	0	213	1.80024e-005	3.54824e-007	0	227

Table 6.6: The results of running both algorithms on an input of 200 random arcs.

ρ	Isocube				DGGS			
	mcl	acl	rtf	rtsr	mcl	acl	rtf	rtsr
5	29	22.3793	106	63.2075%	45	32.083	97	72.1649%
7	37	27.6663	144	72.2222%	57	39.9651	139	80.5755%
9	45	33.1271	158	75.3165%	69	48.0634	154	82.4675%
12	57	41.3777	164	76.2195%	87	60.0023	159	83.0189%
15	69	49.7107	168	76.1905%	105	72.2336	162	83.3333%
18	81	58.0562	169	76.3314%	123	84.6977	164	83.5366%
25	109	77.5248	169	76.9231%	364	113.672	162	83.3333%

Table 6.7: The results of running both algorithms on an input of 200 random arcs (continued).

(this is also evident in further experiments where the number of input arcs is larger).

6.4 Real World Input

Perhaps the most interesting experiment is the one we carried out on real-world input. We ran the algorithms on the border of USA (including Alaska and Hawaii). The results are summarized in Tables 6.9 and 6.10. The results show that rounding with $10 \leq \rho \leq 15$ produce excellent results. When $\rho = 15$ there are no visible problems in the rounded arrangements. An example can be seen in Figure 6.1¹. The results show that the average bit length in the isocube algorithm is about 60 bits which is less than half of the bit length of a coordinate in the original arrangement. When reducing ρ to 10, the results are still good at most areas but areas with many small faces resemble a grid on the reference points of the hot pixels. An example of this phenomenon can be seen in the north west of mainland USA (see Figure 6.2). The next input was the map of major roads in North America. The results were similar to those obtained by rounding the border. With $\rho = 15$ there were no significant distortions in the rounded output while the average bit length was as low as 52.7734 (see Tables 6.11, 6.12 and Figure 6.3). Furthermore, the results show that the planar SR takes more

¹Images in this section are produced using Google Earth.

			Isocube		DGGS	
#Arcs	nvi	nei	rtf	rtsr	rtf	rtsr
20	48	36	0.344	59.0116%	0.328	61.8902%
50	196	241	2.953	67.7277%	3.281	70.4968%
100	637	974	19.188	73.2906%	19.359	71.7547%
200	2696	4792	178.781	76.6826%	159.500	83.758%
300	4973	9045	466.110	78.7466%	442.844	85.1986%

Table 6.8: The results of running both algorithms on inputs with increasing number of arcs.

	Isocube				DGGS			
ρ	mhd	ahd	nsv	nca	mhd	ahd	nsv	nca
5	0.0365355	0.0244899	10690	4	0.076651	0.0465627	10738	2
8	0.00510799	0.00267337	9082	4	0.00989853	0.00513007	9356	4
10	0.00125666	0.000564112	5240	4	0.00248137	0.0010695	5792	4
15	3.86343e-005	1.48827e-005	2	4	7.96059e-005	2.61095e-005	7	4
20	2.23644e-006	4.6937e-007	0	4	1.27244e-005	8.04328e-007	0	4

Table 6.9: The results of spherically snap rounding the borders of USA.

than 99% of the processing time which makes by far the most time-consuming component in the algorithm and hence the natural candidate for further improvement work.

	Isocube				DGGS			
ρ	mcl	acl	rtf	rtsr	mcl	acl	rtf	rtsr
5	29	24.0811	972	97.5309%	40	34.3837	1150	98.087%
8	41	34.131	177	85.8757%	55	47.8935	187	88.2353%
10	49	41.464	571	95.6217%	65	57.4998	313	92.6518%
15	69	59.936	2189	98.5381%	90	81.8034	1280	97.6563%
20	89	77.4457	2127	98.4955%	115	105.724	1289	97.6726%

Table 6.10: The results of spherically snap rounding the borders of USA (continued).

ρ	Isocube				DGGS			
	mhd	ahd	nsv	nca	mhd	ahd	nsv	nca
10	0.00126452	0.000596916	92180	45	0.00252517	0.00109385	98903	38
15	3.98691e-005	1.33338e-005	786	47	8.09463e-005	2.33477e-005	1437	39

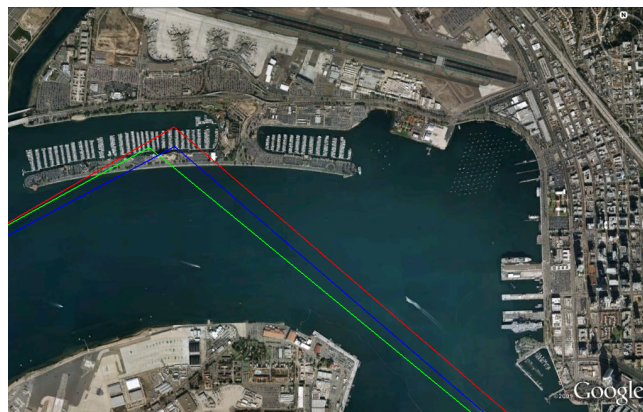
Table 6.11: The results of spherically snap rounding the map of major roads in North America.

ρ	Isocube				DGGS			
	mcl	acl	rtf	rtsr	mcl	acl	rtf	rtsr
10	49	37.9866	12331	93.8529%	65	56.5948	9075	94.0716%
15	69	52.7734	108348	99.2543%	90	79.9858	103038	99.473%

Table 6.12: The results of spherically snap rounding the map of major roads in North America (continued).



Snap rounded arrangement with $\rho = 15$.

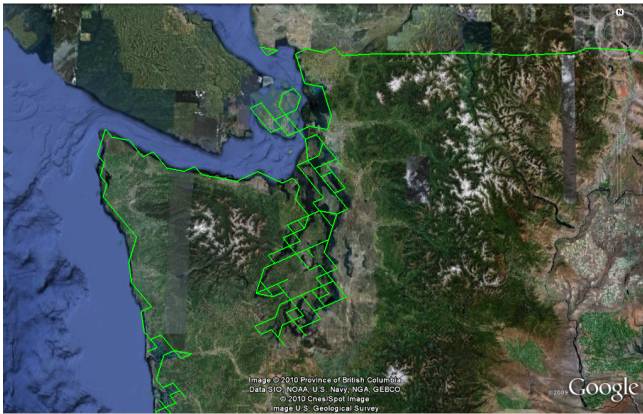


Zoom in on the San Diego bay area. The input, isocube rounded and DGGs rounded arcs (red, green and blue respectively) are very close such that there is no substantial difference between the three arrangements. On the bottom right, the USS Midway aircraft carrier.



The input arrangement.

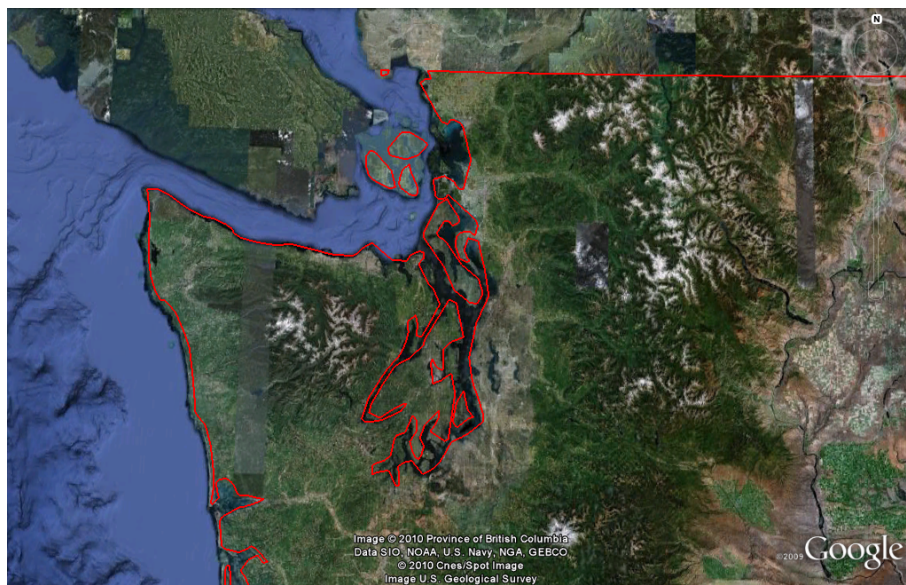
Figure 6.1: Rounding the map of USA with $\rho = 15$.



Isocube algorithm.

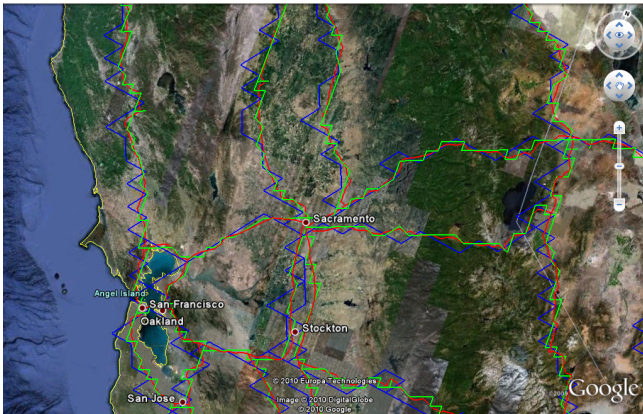
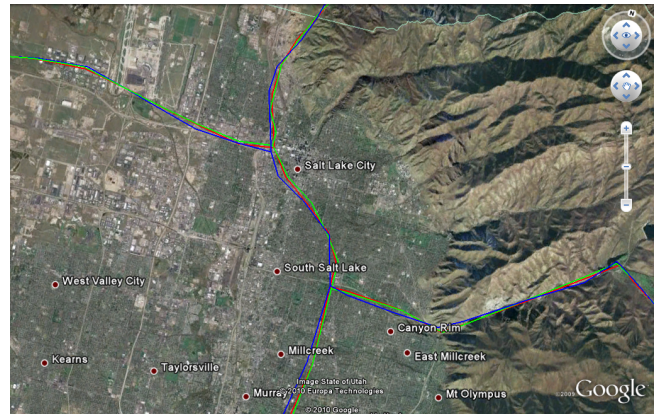


DGGS algorithm.



The input arrangement.

Figure 6.2: The north west of continental USA rounded with $\rho = 10$.

Zoom in on the Sacramento area with $\rho = 10$.Zoom in on the Salt Lake City area with $\rho = 15$.

The input arrangement.

Figure 6.3: Examples from the spherically snap rounded map of North American roads.

7

Conclusion and Future Work

We have devised and implemented two efficient and practical algorithms for snap rounding an arrangement of geodesic arcs on the sphere. The experimental results show that the isocube approach produces better results than the DGGs algorithm for the same resolution values of ρ , yet it cannot be easily extended to be used with different underlying polytopes. The results demonstrate that even for small values of ρ (namely, coarse rounding) the snap-rounded arrangement can be made topologically and geometrically similar to the input arrangement while achieving a reduction of more than 50% in the bit length of the coordinates representing the vertices of the arrangement. We found out that it is prudent to run the algorithm with several values of ρ and tune its value until the results fit the required needs in both the bit length of the coordinates and the quality of the output.

In our implementation it is evident that the most significant part of the algorithm, in terms of running time, is the `snap_rounding_2` function which is an implementation of the algorithm in [HP02]. A more efficient version of the SR algorithm was presented by Hersberger [Her08]; we anticipate that its implementation and incorporation within our SSR algorithms will improve the running time greatly for large inputs. Furthermore, the SSR algorithms can be easily parallelized. The planar SR is run on each face of the underlying polytope separately as there are no dependencies on the rounding results from other faces. After the rounding phase the results are disjoint therefore combining them does not require complicated overlay procedure but only list concatenation (of faces, edges and vertices) which can be done in linear time. The connection arcs generation process can be run immediately after the distribution of the arcs in parallel to running SR on each face.

The DGGs approach is extensible to other underlying polytopes with triangular faces as we show a general method of building p-pixels that preserve the topological consistency required by snap rounding. Our insights on the grids that preserve the topological consistency of the planar snap rounding may be the basis for future work in this area.

Further work could be carried out to improve the DGGS method by building a polytope that models the sphere better than a Platonic solid. This can be done by constructing a polytope that is inscribed in a unit sphere and has many small triangular faces. Some examples of how such polytopes can be constructed are described in [SWK03, WKSS98] where a Platonic solid is subdivided and the new vertices are moved to the sphere after each subdivision phase. A similar approach is also presented in [SGF⁺07] where the authors build a triangular mesh to approximate the sphere.

Our observations on the grids that can be used in the planar SR method can be used to create a new snap rounding scheme. In this new scheme the plane is not tiled completely by the pixels but instead a set of hot pixels are distributed in the plane with respect to some criteria. Such a scheme may reduce the number of hot pixels and new intersections while providing good rounding results. It is also possible to tile the plane with specific pixels in terms of shape and pattern of the pixels (e.g., a tiling which looks like a brick wall where the pixels are rectangles and all the vertices are of degree 3). Such tilings may be more appropriate in some cases and improve the overall rounding results compared to running the SR algorithm on the standard tiling.

A

Appendix

A.1 Convex Sets on a Sphere

A set \mathcal{C} on the sphere is convex iff for any two points in \mathcal{C} the small geodesic arc between the two points is contained in \mathcal{C} , e.g., (see [CDGM97]).

Lemma A.1. *Let S be the unit sphere and C a small circle¹ on S . Let P be the plane whose intersection with S contains C . Let C_p be a convex set on P that is the interior of C . The spherical cap \hat{C} induced by P is a convex set on S .*

Proof. C_p is a convex set on P and $\hat{C} = G_P^{-1}(C_p)$. From the definition of a convex set, every segment in C_p is contained entirely in C_p . From the definition of the Gnomonic projection we know that a segment in C_p is transformed to a geodesic arc on \hat{C} and vice versa. Therefore $G_P^{-1}(C_p)$ is a convex set on S . \square

A.2 Circumcenter Bit Length

Lemma A.2. *Let L_c be the bit length of the circumcenter of a triangular cell during the subdivision of the axis-aligned octahedron (as described in Chapter 3) and let L_v be the maximum bit length of the coordinates of the vertices of that cell. Then $L_c \leq L_v + 6$.*

Proof. We show this for one initial face of the octahedron; the same arguments apply to any subdivided triangular cell. A face of the octahedron creates a 45° angle with the xy -plane. Figure A.1 illustrates one such face denoted by the triangle $\triangle KMN$. The point D is the

¹A small circle on a sphere is the intersection of a plane and the sphere such that the center of the sphere is not on the plane.

midpoint of \overline{MN} . The point C is the circumcenter of $\triangle KMN$ and therefore $\overline{DC} = \frac{1}{3}\overline{DK}$. The point T is the projection of C on the xy -plane and it lies on $\overline{DO} \Rightarrow \overline{TO} = \frac{2}{3}\overline{DO}$. $|\overline{MN}| = M_x\sqrt{2} = N_y\sqrt{2} \Rightarrow |\overline{MD}| = |\overline{DO}| = |\overline{DN}| = \frac{|\overline{MN}|}{2}$ (since $\triangle MNO$ is a right triangle with hypotenuse MN) $\Rightarrow |\overline{TO}| = \frac{2}{3}|\overline{DO}| = \frac{M_x\sqrt{2}}{3} = \frac{N_y\sqrt{2}}{3}$. We look at the square on the xy -plane formed between the origin O and T , with the diagonal \overline{TO} . By examining side of this square we get the coordinates of the point T on the xy -plane. The coordinates of T are $(\frac{M_x}{3}, \frac{N_y}{3})$ hence the coordinates of C are $(\frac{M_x}{3}, \frac{N_y}{3}, \frac{K_z}{3})$ which means $L_c \leq L_v + 6$ as needed. \square

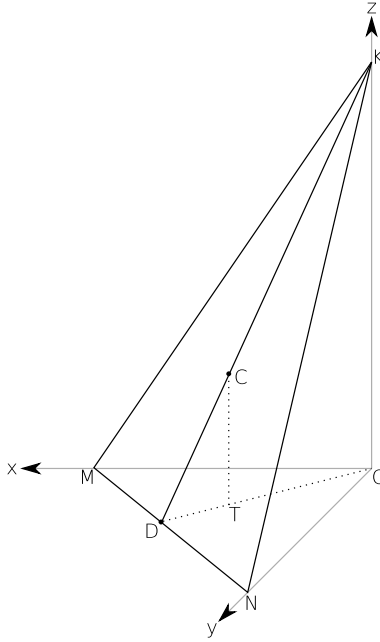


Figure A.1: The bit length of the circumcenter C is the the maximum bit length of the coordinates of the vertices in $\triangle KMN + 6$.

A.3 Input Files Example

An example of an input file given in spherical coordinates. The first line defines the file type as “0”. Next, three arcs are defined using spherical coordinates given in radians.

```
0
0,1.8,2.1,1.2
1,1.1,0.5,2
0.55,2.01,0.67,1.98
```

An example of an input file given as rational vectors. The line numbers are added to indicate lines in the file. They do not exist in the actual file. The first line defines the file type as “1”. Each subsequent line contains two three-tuples separated by a comma. Within each three-tuple the numbers are separated by space and the numerator is separated from the denominator by a forward slash.

```
1) 1
2) 4184953696439088806209662791635/2535301200456458802993406410752
   4184953696439088806209662791635/2535301200456458802993406410752
   2928923127738069817745860668949/20282409603651670423947251286016
   ,548227478442335/562949953421312 0/1
   -4092909075990333/18014398509481984
3) -3493578394255567/9007199254740992
   7633608056877581/9007199254740992 1631914248035957/4503599627370496,
   4184953696439088806209662791635/2535301200456458802993406410752
   4184953696439088806209662791634/2535301200456458802993406410752
   2928923127738069817745860668949/20282409603651670423947251286016
4) 4184953696439088806209662791635/2535301200456458802993406410752
   4184953696439088806209662791635/2535301200456458802993406410752
   2928923127738069817745860668949/20282409603651670423947251286016,
   548227478442335/562949953421312 0/1
   -4292909075990333/18014398509481984
```


Bibliography

- [AK00] Franz Aurenhammer and Rolf Klein. Voronoi diagrams. In Jörg-Rüdiger Sack and Jorge B. Urrutia, editors, *Handbook of Computational Geometry*, chapter 5, pages 201–290. Elsevier Science Publishers, B.V. North-Holland, 2000.
- [BFG⁺09] Herv Brnnimann, Andreas Fabri, Geert-Jan Giezeman, Susan Hert, Michael Hoffmann, Lutz Kettner, Stefan Schirra, and Sylvain Pion. 2d and 3d geometry kernel. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.
- [BMBS09] Boaz Ben-Moshe, Binay K. Bhattacharya, and Jeff Sember. Efficient snap rounding in square and hexagonal grids using integer arithmetic. Technical Report TR-2009-04, The University of British Columbia, 2009.
- [BO79] Jon Louis Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, 28(9):643–647, 1979.
- [BS07] Binay K. Bhattacharya and Jeff Sember. Efficient snap rounding with integer arithmetic. In *In Proceedings of the 19th Annual Canadian Conference on Computational Geometry, CCCG 2007, August 20-22, Carleton University, Ottawa, Canada*, pages 145–148, 2007.
- [CDGM97] Francisco Javier Cobos, Juan Carlos Dana, Clara I. Grima, and Alberto Márquez. The width of a convex set on the sphere. In *In Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG'97)*, 1997.
- [Cox69] H. S. M. Coxeter. *Introduction to geometry*. John Wiley & sons, inc., second edition, 1969.
- [dBHO07] Mark de Berg, Dan Halperin, and Mark Overmars. An intersection-sensitive algorithm for snap rounding. *Computational Geometry: Theory and Applications*, 36(3):159–165, 2007.
- [dBvKOS00] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

- [DJ05] Qiang Du and Lili Ju. Finite volume methods on spheres and spherical centroidal Voronoi meshes. *SIAM Journal on Numerical Analysis*, 43(4):1673–1692, 2005.
- [FSH08a] Efi Fogel, Ophir Setter, and Dan Halperin. Exact implementation of arrangements of geodesic arcs on the sphere with applications. In *Abstracts of 24th European Workshop on Computational Geometry*, pages 83–86, 2008.
- [FSH08b] Efi Fogel, Ophir Setter, and Dan Halperin. Movie: Arrangements of geodesic arcs on the sphere. In *Abstracts of 24th European Workshop on Computational Geometry*, pages 218–219. Association for Computing Machine, 2008.
- [GGHT97] Michael T. Goodrich, Leonidas J. Guibas, John Hershberger, and Paul J. Tanenbaum. Snap rounding line segments efficiently in two and three dimensions. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 284–293, New York, NY, USA, 1997. Association for Computing Machinery (ACM) Press.
- [GHB⁺05] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *Astrophysics Journal*, 622:759–771, April 2005.
- [GM98] Leonidas J. Guibas and David H. Marimont. Rounding arrangements dynamically. *The International Journal of Computational Geometry and Applications*, 8(2):157–176, 1998.
- [GY86] Daniel H. Greene and F. Frances Yao. Finite-resolution computational geometry. In *SFCS '86: Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 143–152, Washington, DC, USA, 1986. IEEE Computer Society.
- [Her08] John Hershberger. Improved output-sensitive snap rounding. *Discrete & Computational Geometry*, 39(1-3):298–318, 2008.
- [HHK⁺07] Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion, and Michael Seel. An adaptable and extensible geometry kernel. *Computational Geometry: Theory and Applications*, 38(1-2):16–36, 2007.
- [Hob99] John D. Hobby. Practical segment intersection with finite precision output. *Computational Geometry: Theory and Applications*, 13(4):199–214, 1999.
- [HP02] Dan Halperin and Eli Packer. Iterated snap rounding. *Computational Geometry: Theory and Applications*, 23(2):209–225, 2002.
- [IEE08] IEEE Task P754. *IEEE 754-2008, Standard for Floating-Point Arithmetic*. IEEE Computer Society Press, aug 2008.

- [JDG02] Lili Ju, Qiang Du, and Max Gunzburger. Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations. *Parallel Computing*, 28(10):1477–1500, 2002.
- [Ket09] Lutz Kettner. 3d polyhedral surfaces. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.
- [Mil89] Victor Milenkovic. Double precision geometry: A general technique for calculating line and segment intersections using rounded arithmetic. In *In Proceedings of the 30th Annual IEEE Annual Symposium on Foundations of Computer Science*, pages 500–505, 1989.
- [Mil90] Victor Milenkovic. Rounding face lattices in d dimensions. In *In Proceedings of the 2nd Canadian Conference on Computational Geometry*, pages 40–45, 1990.
- [Mil00] Victor Milenkovic. Shortest path geometric rounding. *Algorithmica*, 27(1):57–86, 2000.
- [OBSC00] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, NYC, 2nd edition, 2000.
- [Pac08] Eli Packer. Iterated snap rounding with bounded drift. *Comput. Geom.*, 40(3):231–251, 2008.
- [Pac09] Eli Packer. 2d snap rounding. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.
- [PS85] Franco P. Preparata and Michael I. Shamos. *Computational geometry: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.
- [SGF⁺07] Alexander S. Szalay, Jim Gray, George Fekete, Peter Z. Kunszt, Peter Kukol, and Ani Thakar. Indexing the sphere with the hierarchical triangular mesh. *The ACM Computing Research Repository*, abs/cs/0701164, 2007.
- [SKS02] Lian Song, A. Jon Kimerling, and Kevin Sahr. Developing an equal area global grid by small circle subdivision. In M. Goodchild and A. J. Kimerling, editors, *Discrete Global Grids*, chapter 1. National Center for Geographic Information and Analysis, 2002.
- [Sny92] John P. Snyder. An equal-area map projection for polyhedral globes. *Cartographica*, 29(1):10–21, 1992.
- [SWK03] K. Sahr, D. White, and A.J. Kimerling. Geodesic discrete global grid systems. *cartography and geographic information science.*, 2003.
- [WFZH07] Ron Wein, Efi Fogel, Baruch Zukerman, and Dan Halperin. 2D arrangements. In CGAL Editorial Board, editor, *CGAL User and Reference*. 3.3 edition, 2007.

- [WKSS98] Denis White, A. Jon Kimerling, Kevin Sahr, and Lian Song. Comparing area and shape distortion on polyhedral-based recursive partitions of the sphere. *International Journal of Geographical Information Science.*, 12(8):805–827, 1998.

Links

- [1] Applied Computational Geometry Lab.
<http://acg.cs.tau.ac.il/>.
- [2] CGAL — Computational Geometry Algorithms Library.
<http://www.cgal.org>.
- [3] ESRI Virtual Globe product page.
<http://www.esri.com/software/arcgis/explorer/index.html>.
- [4] Geographic Coordinate System.
http://geology.isu.edu/geostac/Field_Exercise/topomaps/grid_assign.htm.
- [5] GMP — GNU Multiple Precision Arithmetic Library.
<http://gmplib.org>.
- [6] Google earth.
<http://earth.google.com>.
- [7] KML Reference at Google.
<http://code.google.com/apis/kml/documentation/kmlreference.html>.
- [8] Geomview object file format.
<http://people.sc.fsu.edu/~jburkardt/data/off/off.html>.
- [9] NASA Wold Wind download page.
<http://worldwind.arc.nasa.gov/download.html>.