Adiabatic isometric mapping algorithm for embedding 2-surfaces in Euclidean 3-space

Shannon Ray\textsuperscript{1}, Warner A Miller\textsuperscript{1,2}, Paul M Alsing\textsuperscript{3} and Shing-Tung Yau\textsuperscript{2}

\textsuperscript{1} Department of Physics, Florida Atlantic University, Boca Raton FL 33431, USA
\textsuperscript{2} Department of Mathematics, Harvard University, Cambridge MA 02138, USA
\textsuperscript{3} Information Directorate, Air Force Research Laboratory, Rome, NY 13441, USA

E-mail: sray11@fau.edu

Received 26 March 2015, revised 16 September 2015
Accepted for publication 21 September 2015
Published 11 November 2015

Abstract

Alexandrov proved that any simplicial complex homeomorphic to a sphere with strictly non-negative Gaussian curvature at each vertex can be isometrically embedded uniquely in $\mathbb{R}^3$ as a convex polyhedron. Due to the nonconstructive nature of his proof, there have yet to be any algorithms, that we know of, that realizes the Alexandrov embedding in polynomial time. Following his proof, we developed the adiabatic isometric mapping (AIM) algorithm. AIM uses a guided adiabatic pull-back procedure on a given polyhedral metric to produce an embedding that approximates the unique Alexandrov polyhedron. Tests of AIM applied to two different polyhedral metrics suggests that its run time is sub cubic with respect to the number of vertices. Although Alexandrov’s theorem specifically addresses the embedding of convex polyhedral metrics, we tested AIM on a broader class of polyhedral metrics that included regions of negative Gaussian curvature. One test was on a surface just outside the ergosphere of a Kerr black hole.

Keywords: quasi-local energy, computational geometry, Weyl embedding, Alexandrov embedding, heuristic optimization

1. Introduction

The problem of embedding surfaces homeomorphic to $S^2$ with a metric of positive Gaussian curvature into $\mathbb{R}^3$ was posed by Weyl [1]. A useful review and analysis of the isometric embedding of Riemannian manifolds in Euclidean spaces can be found in [2]. The first attempt to prove the existence of an embedding was given by Weyl himself. He was not able to complete his proof, but he did make progress in outlining a solution. Following Weyl’s approach, a proof was given by Lewy in 1938 though his solution required the components of
the metric to be infinitely differentiable [3]. Requirements on differentiability of the metric were reduced to continuous fourth derivatives by Nirenberg [4]. They were further reduced to continuous third derivatives by Heinz [5]. In 1941, Alexandrov gave an approach that relied on proving the existence of a convex polyhedron given any convex polyhedral metric. He showed that in the limit as the number of vertices in the polyhedral metric goes to infinity, one recovers the metric in the continuum. This result can be found in a compendium of his work published in 2006 [6].

Our interest in isometric embedding stems from its application in general relativity. Solutions to Einstein’s equations give a four-dimensional spacetime manifold. One can slice these manifolds in such a way to get interesting two surfaces such as the event horizon or ergosphere. To visualize these surfaces it may be best to embed them in three-dimensional Euclidean space. Another application of isometric embeddings include computing quasi-local energy (QLE) in general relativity. Many definitions of QLE, such as the Brown–York and Wang–Yau energies, require isometric embeddings of convex $S^2$ surfaces in $\mathbb{R}^3$ [7, 8]. A detailed review of the development of QLE is given in [9].

Though the mathematics of global isometric embeddings of convex two-surfaces homeomorphic to $S^2$ was studied exhaustively for over 60 years, its numerical implementation has proven formidable and has only been approached practically since the mid-1990s. In 2006, Bondarescu et al introduced a numerical method using the components of the metric in the continuum [10]. They expanded the embedding functions using spherical harmonics and minimized the coefficients by optimizing the squared difference in metric components. Bondarescu et al found that their search would get stuck in local minimum while minimizing in the space of coefficients. To reduce their residuals, they increased the number of coefficients for each embedding function.

In 2011, Jasiulek and Korzyński introduced an algorithm that closely followed the continuity method prescribed by Weyl [11]. First they uniformized their surface using Ricci flow, which gave them conformal relations between their original metric, the round sphere metric and all intermediate metrics. They then embedded the spherical metric into $\mathbb{R}^3$. This allowed them to use the embedded sphere as a starting point for solving the linearized embedding equations which gave them the deformation in coordinates such that they move from the sphere to a sufficiently close conformally related surface. Because they were working in the continuum, they dealt with typical difficulties such as finding suitable coordinate charts, solving nonlinear elliptical PDEs and computing integrals and derivatives of functions on the surface. They were also restricted to only embedding surfaces with strictly positive curvature due to the inversion of the extrinsic curvature tensor in their procedure.

Working in the continuum has the added complication of requiring prior knowledge about the sign of the Gaussian curvature of the surface. Whether or not the embedding equations are elliptical or hyperbolic depends on the sign of the curvature. Additionally one may need to provide numerical techniques to deal with coordinate singularities. Difficulties in the continuum are reasons that one might want to work in the discrete. In 1995, Nollert and Herold developed the wire-frame method which was the first attempt at finding a discrete numerical solution to the embedding problem [12]. Their method used position vectors $\mathbf{r}(\xi(t))$ where $\xi$ are the internal parameters of the surface. Given $\mathbf{r}(\xi(t))$, they made a Taylor series expansion around points on the surface allowing them to write distances in Euclidean space in terms of components of the internal metric. Equating this distance with the actual Euclidean distance gave them a function to minimize where the variables are coordinates in $\mathbb{R}^3$. Because their optimization function only restricted edge lengths, and in general polyhedrons are not uniquely determined by them, their method had no means of choosing what they called smooth solutions.
In 2008, Kane et al wrote a pseudopolynomial time algorithm to give a numerical realization of Alexandrov’s embedding theorem [13]. Instead of following the proof by Alexandrov, they modeled their algorithm using the proof by Bobenko and Izmestiev [14]. Given a convex polyhedral metric, this method uses the variation of three-dimensional curvatures within the interior of the polyhedron to slowly deform their surface to one that is isometric. Alexandrov’s theorem says given a convex polyhedral metric there exists a unique convex polyhedron in \( \mathbb{R}^3 \). Kane et al showed that their algorithm finds an approximate convex polyhedron where no edge has a dihedral angle greater than \( \pi + \epsilon \). The time taken to reach this \( \epsilon \)-convex polyhedron depends on several intrinsic variables of the metric and user specified tolerances. Their algorithm only works for convex polyhedral metrics.

In this paper we present the adiabatic isometric mapping (AIM) algorithm which is a numerical approach for embedding polyhedral metrics. Like Kane et al’s, our algorithm produces approximately convex polyhedrons given convex polyhedral metrics. For metrics that are not convex, AIM produces smooth polyhedrons similar to those mention in [12]. The AIM algorithm borrows techniques from several of the algorithms mentioned above. The first step of AIM uniformizes the initial polyhedral metric under discrete Ricci flow in an affine time parameter \( t \). The second step embeds this constant curvature surface near the surface of a sphere. We use this embedded surface as the starting point for an embedding flow back to the original metric. This is in a similar vein to Jasiulek and Korzyński’s algorithmm. To step from one conformally related polyhedron to the next, we use Newton’s method to minimize an objective function that depends on the edge lengths of the polyhedral metric and the coordinate distances in \( \mathbb{R}^3 \). We use the coordinates of the constant curvature polyhedron as the initial guess for Newton’s method to avoid the problem of local minima found in Bondarescu et al. This puts us close enough to the solution so that we quickly converge to the global minimum. We then use the newly embedded polyhedron as the initial value data for our next step such that \( \Delta t \) is small enough to remain near the global minimum. This is repeated until we reach the original polyhedral metric. Taking \( \Delta t \) small is not enough to ensure that we will not produce non-smooth solutions as seen by Nollert and Herold. Because of this, we introduce the convexity routine at each time step which guides our solution toward smooth embeddings. Using our condition on \( \Delta t \) and the convexity routine, we introduce the guided adiabatic pull-back (GAP) which is unique to our algorithm. We found that AIM is capable of embedding with accuracy in the edge lengths to any desired tolerance above machine precision.

### 2. Adiabatic isometric mapping (AIM)

Following the approach of Alexandrov, we begin by finding a continuous family of polyhedral metrics \( \rho_t \) where \( t \in [0 \leq t \leq t_f] \). A polyhedral metric \( \rho \) of a triangulated surface is a list of its triangles by vertices \( \{v_i, v_j, v_k\} \) together with an assignment of a length to each edge \( \ell_{ij} \). Here, \( \rho_0 \) is the metric we wish to embed and \( \rho_{t_f} \) is a metric with constant Gaussian curvature at each vertex. Each polyhedral metric in the family \( \rho_t \) has \( N_0 \) vertices, \( N_1 = 3N_0 - 6 \) edges and \( N_2 = 2N_0 - 4 \) flat triangular faces. The squared edge lengths of \( \rho_t \) are given as

\[
\ell_{ab}^2(t) = \begin{cases} 
\end{cases}
\]

where the indices \( a \) and \( b \) label the vertices of \( \rho_t \). It should be noted that throughout our algorithm the structure of the triangulation of \( \rho_t \) always remains the same. The structure is defined as a list of triangles by vertices. After we obtain \( \rho_t \), we find a polyhedron \( P_t \) which is...
an isometric embedding of $\rho_{ij}$ in $\mathbb{R}^3$. For $P_{ij}$ to be isometric to $\rho_{ij}$, the coordinates for the vertices of $P_{ij}$,

$$a \rightarrow \{x_a(t_f), y_a(t_f), z_a(t_f)\},$$

must satisfy each of the $N_1$ distance relations

$$\ell_{ab}^2(t_f) = (x_a(t_f) - x_b(t_f))^2 + (y_a(t_f) - y_b(t_f))^2 + (z_a(t_f) - z_b(t_f))^2.$$  \hspace{1cm} (3)

We specify freely six of the $3N_0$ coordinates in order to mod out the translations and rotational degrees of freedom. This is often done by identifying one of the triangles $\Delta_{abc}$ of $P_{ij}$ and fixing (1) the three coordinates of vertex $a$, (2) two of the three coordinates of vertex $b$, and (3) one of the remaining three coordinates of vertex $c$. The isometric embedding problem involves a solution to a quadratic system of $3N_0 - 6$ equations and unknown coordinates. There are many solutions that satisfy the quadratic equations; as a result, one finds oneself in a ‘sea of solutions’ that makes solving this system of equations prohibitively difficult. Not only does one have to solve the nonlinear sparsely coupled system, they also have to find a solution with the desired extrinsic curvature. Constrained optimization problems of this kind are often costly to solve. To navigate through the ‘sea of solutions’ without resorting to constrained optimization, AIM uses a three step procedure: (1) uniformization via Ricci flow that provides a dimensional reduction from three to two dimensions, (2) uniform surface embedding, and (3) a GAP of the coordinates from the uniformed surface $\rho_{ij}$ to the original surface. These three steps ameliorate many difficulties in solving the system of equations and
provide controllable criteria for obtaining a suitable solution. We describe these steps in the
the next three subections which are illustrated in figure 1. From now on when we mention the
metric we are referring to a polyhedral metric.

2.1. Uniformization via Ricci flow

We use a discrete Ricci flow to find a conformal relationship between $\rho_0$ and $\rho_t$. A conformal
factor is assigned to each vertex $a$ of $\rho_0$ and is denoted by the set $\{u_a(t = 0) = 0\}$. The
relationship between edge lengths of $\rho_0$ and $\rho_t$ is given by

$$\ell_{ab}(t) = \ell_{ab}(0) + \frac{u_a^{(t)} + u_b^{(t)}}{2},$$

(4)

where $a$ and $b$ index the bounding vertices of edge $\ell_{ab}$.

The discrete Ricci flow equations we use for the conformal factors are given by

$$\frac{du_a}{dr} = -\left(k_a - \bar{k}\right),$$

(5)

where $k_a$ is the Gaussian curvature at vertex $a$ and $\bar{k}$ is the target curvature of the surface. To
keep the surface area constant, we choose $\bar{k}$ to be the average Gaussian curvature over the
surface

$$\bar{k} = \frac{2\pi \chi}{\mathcal{A}},$$

(6)

where $\chi$ is the Euler characteristic and $\mathcal{A}$ is the area of the simplicial surface. A detailed
description of Gaussian curvature can be seen in the appendix.

It was shown by Chow and Lou that combinatorial Ricci flow and discrete Ricci flow are
essentially equivalent and that they exponentially converge to a surface of constant curvature
[15]. Therefore given some $\epsilon < 1$, it is assured that for sufficiently long times $t_f$ we can Ricci
flow the initial conformal factors such that the Gaussian curvature at each vertex is close to $\bar{k}$,

$$||k_a(t_f) - \bar{k}||_2 < \epsilon.$$  

(7)

Once we have $\{u_a(t_f)\}$, we create a continuous family of conformal factors between $\rho_0$
and $\rho_t$ using the linear relation

$$u_a(t) = u_a(t_f) \left( \frac{t}{t_f} \right).$$

(8)

We also considered using an exponential relationship between conformal factors and the set
of conformal factors produced at each step during Ricci flow. These alternate paths between
conformal factors were shown to be computationally more demanding without any apparent
benefit over the linear path. We discuss this in more detail in section 2.3.

2.2. Uniform surface embedding

The uniform surface embedding is broken into four steps,

1. Embedding on a sphere.
2. Anneal step A.
3. Convexity Routine.
4. Anneal step B.
We will now discuss each step in detail and explain their necessity beginning with embedding on a sphere.

Ideally a constant Gaussian curvature surface will lie on a sphere centered at the origin with radius \( r = \frac{1}{\sqrt{k}} \).

This gives an additional constraint, \( r^2 = x_i^2 + y_i^2 + z_i^2 \), on the coordinates of \( \rho_i \). We are free to specify the initial line segment \( \ell_{ab} \) as \( \{x_a, y_a, z_a\} = \{\frac{r}{\sqrt{2}}, \frac{r}{\sqrt{2}}, \frac{r}{\sqrt{2}}\} \) and

\[
x_b = \frac{r \left( r^2 - L_{ab}^2 \right) + \sqrt{L_{ab}^2 r^2 \left( 2r^2 - L_{ab}^2 \right)}}{2r^2},
\]

\[
y_b = \frac{r \left( r^2 - L_{ab}^2 \right) - \sqrt{L_{ab}^2 r^2 \left( 2r^2 - L_{ab}^2 \right)}}{2r^2},
\]

\[
z_b = \frac{r}{\sqrt{2}}.
\]

Given this embedded line segment, we embed vertex \( c \) by solving

\[
\ell_{ac}^2 = (x_a - x_c)^2 + (y_a - y_c)^2 + (z_a - z_c)^2,
\]

\[
\ell_{bc}^2 = (x_b - x_c)^2 + (y_b - y_c)^2 + (z_b - z_c)^2,
\]

\[
r^2 = x_c^2 + y_c^2 + z_c^2.
\]

There are ordinarily two solutions to these equations which correspond to the reflection of vertex \( c \) about \( \ell_{ab} \). The translational and rotational degrees of freedom are fixed once we embed the three vertices of \( \Delta_{abc} \) on the surface of the sphere.

Starting with this single embedded triangle \( \Delta_{abc} \), we can embed each of the three triangles sharing bounding edges. This procedure is illustrated in figure 2. In fact, given a triangle with a single embedded edge, we embed its opposite vertex by solving (13)–(15). This allows us to generate a growing network of embedded triangles on the surface of the sphere one triangle at a time.
a time. The dimensional reduction afforded to us by the Ricci flow essentially block-diagonalizes the original sparsely-coupled matrix from the quadratic distance equations \( (3) \) into many \( 3 \times 3 \) matrices. By looking at shared edges between embedded and non-embedded triangles, we can successively embed each vertex individually until all vertices \( a \in \rho \), lie on the sphere. We choose the solution to these quadratic systems such that the dihedral angle between edge \( \ell_{ab} \) is maximal and less than \( \pi \). This solution yields two ‘unfolded’ triangles embedded on the surface and is illustrated in figure 3.

Here we explain the necessity and logistics behind anneal step A. The embedding-tree procedure will ordinarily lead to inconsistencies with the isometric embedding equations \( (3) \). The vertices of a constant curvature polyhedron ordinarily will not lie on a sphere of constant radius unless the number of vertices \( N_0 \) is sufficiently large. Inconsistencies occur on the ‘determined edges’, as illustrated in figure 2, whose edge lengths do not agree with those in \( \rho_{ij} \). We correct this disagreement by annealing the coordinates on the sphere to the edge lengths of \( \rho_{ij} \). Annealing is done using Newton’s method to minimize the \( L^2 \) norm of \( (3) \) at time \( t = t_h \).
with the coordinates on the sphere as our initial guess. The surface after annealing is uniform in the intrinsic sense but extrinsically it may have negative curvature in some places. Before we move on to fix this issue, we must discuss the isometric variations of polyhedra.

It is well known that polyhedra are not uniquely determined by their edge lengths since this restriction says nothing about their extrinsic curvature. For example, figure 4 shows two polyhedra with identical edge lengths and Gaussian curvatures but different extrinsic curvatures. If all the edges emanating from a vertex have negative extrinsic curvature, that vertex is identified as an inverted vertex. This is seen in figure 6(a). On average there are six edges emanating from a vertex of an arbitrary triangulated polyhedron. A vertex is identified as partially inverted if more than 70% of the edges emanating from it have negative extrinsic curvature. Identifying these vertices allows us to guide our embedding toward the most convex polyhedron possible during the GAP procedure. The identification of partially inverted vertices was chosen empirically based on experiments with AIM for a variety of surfaces. We expect that this choice is most effective when dealing with triangulations that contain mostly regular triangles. For example, all of our triangulated surfaces had internal angles strictly between $\frac{2}{3}\pi$ and $\frac{2}{3}\pi$

An inverted vertex can be ‘popped out’ such that the edges maintain their lengths but their extrinsic curvatures are no longer negative. This process is local meaning all other vertices maintain their coordinates and all other edge lengths remain the same. We show this in figure 6(b). On the other hand, a partially inverted vertex cannot be removed locally and it is not clear what it means for them to be inverted. That is why we need the convexity routine during the uniform embedding and GAP procedures to apply a sort of outward pressure that makes the embedding as convex as possible. The convexity routine is described as follows: first compute the average vector $v_{\text{avg}}$ of the vertices sharing an edge with the inverted or partially inverted vertex, second compute the difference vector defined as $v_{\text{diff}} = \bar{v} - v_{\text{avg}}$, and third define the new position vector of vertex $v$ by $v_{\text{new}} = \bar{v} + 2v_{\text{diff}}$. For an inverted vertex this will get one close to the ‘popped out’ solution were edge-lengths are invariant. For a partially inverted vertex the edge-lengths are no longer the same, but when annealing, i.e. minimizing in the space of coordinates, our initial guess is now closer to a more convex solution. In both cases annealing after the convexity routine returns the edge lengths to their original values within some set tolerance, this is the purpose of the anneal step B. The convexity routine may not work for an arbitrary triangulation, but it should be sufficient for triangulations where the internal angles of all triangles are greater than $\frac{2}{3}\pi$ and less than $\frac{2}{3}\pi$.

2.3. Guided adiabatic pull-back (GAP)

In the final step of the AIM algorithm we pull-back from $P_f$ to $P_0$ by finding coordinates of $P_t$ for all steps $t_f \in [0, t_f]$. These coordinates are found by minimizing (16) at each $t_f - j$. For $t_f - 1$ we used the uniform embedding found in the previous section as our initial guess. If $j = 2, 3$, we use the coordinates at time $t_{f-1}$ as the initial guess for embedding surface $p_t$. For $j > 3$ we extrapolate the coordinates at $t_{f-j}$ using the previous three coordinate sets. This extrapolation procedure has two purposes: first it brings our initial guess closer to the global minimum which decreases the convergence time for Newton’s method, second it allows us to take larger steps from $t_{f-j}$ to $t_{f-j-1}$ while keeping us within an open ball of the global minimum. This better initial guess further decreases the run time of GAP.
There are several paths that relate conformal factors \( \{ u_{\alpha}(t_f) \} \) to \( \{ u_{\alpha}(0) \} \). We chose the linear path given by (8). This path is optimal because the rate of change of Gaussian curvature is constant at each time step \( t_{f-j} \) to \( t_{f-j-1} \), which allows one’s step size \( \Delta t = \frac{t_f}{T\text{steps}} \) to be constant throughout GAP. Here \( T\text{steps} \) is the number of steps taken. If the path was not linear, we would need adaptive time stepping to account for the variable change in Gaussian curvature. For example, one of the alternative paths that we considered was given by the conformal factors produce at each step of Ricci flow. Since Ricci flow exponentially uniformizes curvature, there is a greater change in curvature near \( t = 0 \). Therefore it is necessary to decrease \( \Delta t \) to maintain adiabaticity near this region of rapid change. We also considered an exponential path given by the time interval, \( t_j = \left( \frac{e^{-\alpha T\text{steps}} - e^{-\alpha 0}}{e^{-\alpha T\text{steps}} - 1} \right) t_f \) where \( \alpha \) is the parameter that controls the rate of convergence. The exponential paths suffers the same problem in adaptive stepping as the Ricci flow path. Neither of these paths yielded improved results over the linear path and both are computationally more demanding.

As we mentioned in the previous section, minimization techniques cannot determine extrinsic curvature. Even after applying annealings and the convexity routine to \( P_j \), it is still possible to begin with an initial surface at \( t_f \) that has many clustered and isolated partial inverted vertices. If nothing was done GAP would produce an undesired crumpled surface surface. We apply the convexity routine at each step to avoid caved in crumpled polyhedra. Within the space of coordinates, this perturbs our initial guess such that we begin near solutions without partially inverted vertices. Although this does not immediately remove the problem vertices, it gradually reduces them at each step until they are eliminated. We illustrate this gradual evolution away from crumpled embeddings in figure 5 keeping in mind that (16) is nonlinear thus having several global minima whose residuals are zero. Using all of these elements, we defined our evolution as adiabatic if \( \Delta t \) is small enough such that we transition to and remain near a global minimum with no inverted or partially inverted vertices.

**Figure 5.** This is a overly simplified visual representation of minimization in coordinate space. Each application of the convexity routine perturbs us away from undesirable global minima. After many time steps we eventually reach a global minimum without any inverted or partially inverted vertices.
This gradual nudging during GAP and its ability to move toward a more convex polyhedra is the main result of this manuscript.

3. Numerical tests

We present three numerical examples of the AIM algorithm. The first two example meshes were made using the distmesh program [16]. As input, distmesh is given $z(x,y)$ to produce coordinates and a list of triangles, by vertices, for surfaces in $\mathbb{R}^3$. We use these coordinates to get edge lengths by calculating the Euclidean distance between vertices. We then use the extracted edge lengths together with the list of triangles as the original polyhedral metric for which we apply AIM. Once AIM is complete, we check how well AIM reproduces intrinsic curvature by comparing edge lengths of the distmesh surface to those produced AIM. We compare extrinsic curvature by looking at the convergence of integrated mean curvature produced by AIM to the continuum value.

Our third example is a surface just outside the ergosphere of a Kerr black hole that is right below maximal rotation. We could not use distmesh to triangulate this surface since its embedding equations are quasi-analytic, which makes them incompatible with distmesh’s input format. Because of this, we triangulated the ergosphere ourselves.

3.1. Distmesh surfaces

The first two surfaces we chose as our test cases are given by

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1, \quad \text{(Ellipsoid),}
\]

(17)

\[
\frac{x^2}{d^2} + \frac{y^2}{e^2} + \frac{z^4}{f^4} = 1, \quad \text{(Drum),}
\]

(18)

where $a = 3$, $b = 2$, $c = 1$, $d = 2$, $e = 1$ and $f = \sqrt{1.5}$. Both surfaces have strictly positive point wise Gaussian and mean curvatures. We will refer to the surfaces described by (17) and (18) as the ellipsoid and drum, respectively.

The following work was programmed using matlab. This preliminary application of AIM does not attempt to optimize our algorithm. The goal is to verify our approach and test how accurately it preserves distances and curvature quantities. Run times can be significantly improved by parallelizing the code and using preconditioning for our optimization routines. When using AIM, one is allowed to choose the accuracy of their embedding by manipulating the tolerances during the annealing and GAP procedures. Tolerances are set using the value of the residuals and the magnitude of the steps in coordinate space for each iteration of the quasi-Newton method. These tolerances are set to $10^{-6}$ and $10^{-8}$, respectively. This means the Newton’s method will stop if the edge lengths are within $10^{-6}$ accuracy or if the distance between points in coordinate space from one iteration to another is on the order of $10^{-8}$. Given these settings, we determine the number of steps necessary for maintaining adiabaticity when performing GAP. We also present our analysis on how each portion of the code scales with increased vertices, as well as the scaling for the number of time steps necessary for adiabaticity. For each surface tested, we recovered an average difference in the edge length between the original metric and AIM within our prescribed tolerance. The standard deviation in the difference between edge lengths is also on the order of our prescribed tolerance. This result was tested with tolerances in accuracy set to $10^{-6}$ and $10^{-9}$. Our reported scaling analysis corresponds to a tolerance of $10^{-6}$. 


We test whether or not the extrinsic curvature is recovered by first looking at a qualitative comparison between continuum and AIM embeddings as seen in figure 6. These surfaces are color coded as a function of mean curvature. The mean curvature of the continuum surfaces were computed using the continuum equations while the triangulated mean curvatures were computed using discrete methods [17]. Figure 7 gives a quantitative comparison, and convergence properties, of the mean curvature between the continuum and AIM by plotting the percent error in integrated mean curvature as a function of resolution. According to our results, the convergence of the integrated mean curvature to the continuum is of first order.

We made log–log plots of run time verses number of vertices for each of the three sub routines and the AIM algorithm as a whole to understand the scaling behavior of our code. The same plots were made for analyzing the number of steps necessary for adiabaticity as a function of resolution. The plots are given in figure 8. It is observed that the highest order contribution to the scaling of AIM comes from the Ricci flow procedure. Our overall results suggest that AIM scales sub cubically.

3.2. Modified ergosphere of Kerr spacetime

In 1973 Smarr analytically embedded an axisymmetric 2-surfaces of a rotating black hole geometry [18]. Following this we use the Boyer–Lindquist coordinates for the metric representation of the rotating black hole spacetime,

$$\text{d}s^2 = g_{tt}\text{d}t^2 + 2g_{t\phi}\text{d}t\text{d}\phi + g_{\phi\phi}\text{d}\phi^2 + g_{\theta\theta}\text{d}\theta^2 + g_{\phi\phi}\text{d}\phi^2,$$

where

$$g_{tt} = -\left(1 - \frac{2Mr + Q^2}{\Sigma}\right),$$

$$g_{t\phi} = -\frac{2Mr - Q^2}{\Sigma}a\sin^2\theta,$$

$$g_{rr} = \frac{\Sigma}{\Delta},$$

$$g_{\theta\theta} = \Sigma,$$

$$g_{\phi\phi} = \left(r^2 + a^2 + \frac{(2Mr - Q^2)a^2\sin^2\theta}{\Sigma}\right)\sin^2\theta.$$

Here we used the usual definitions where

$$\Sigma := r^2 + a^2\cos^2\theta,$$

$$\Delta := r^2 - 2Mr + a^2 + Q^2.$$
Figure 6. On the left we have the embedding using the continuum equations. In the center we have the smooth embeddings produced by AIM. On the right we have a non-smooth embeddings which are produced when we do not pull-back adiabatically and we do not use the convexity routine. The color coding indicates the mean curvature associated to each triangle. The ellipsoid has a resolution of 3226 vertices while the drum’s resolution is 3246.

Figure 7. Convergence of integrated mean curvature as a function of resolution.
The ergosphere is defined by the radius in which $g_{tt} = 0$ in (19). At this point the surface is elongated at the poles which makes it an extreme surface to embed. To make the poles more smooth we distort the surface by adding a small parameter $\epsilon$ that gives us a surface slightly outside the ergosphere. The radius as a function of $\theta$ is given as

$$r = R(\theta) = M \left( 1 + \sqrt{1 - \left( \frac{a}{M} \cos^2 \theta \right)^2} \right) + \epsilon M.$$  

Let the isometric embedding functions be defined as

$$x(\theta, \phi) = \rho(\theta) \cos \phi,$$

$$y(\theta, \phi) = \rho(\theta) \sin \phi,$$

and

$$ds^2 = \sum \left( 1 + \frac{R^2}{\Delta} \right) d\theta^2 + \sum \left( R(\theta)^2 + a^2 + \frac{2a^2MR(\theta) \sin^2 \theta}{\Sigma} \right) \sin^2 \theta d\phi^2.$$  

Figure 8. These are log-log plots of each section of AIM and the algorithm as a whole. The slopes indicate scaling behavior of our algorithm as a function of $N_0$. Our overall results suggest that AIM scales sub cubically with time. Time is in units of seconds.
Equating the line elements between spaces gives the relation

\[ ds^2 = dx^2 + dy^2 + dz^2 = \left( \rho^2 + f^2 \right) \, d\theta^2 + \rho^2 \, d\phi^2. \] (32)

Using (27) and (32) to solve for \( \rho \) and \( f \) we have

\[ \rho(\theta) = \sqrt{h_{\phi\phi}(\theta)}, \] (33)

\[ f(\theta) = \int_0^\theta \sqrt{h_{\theta\theta} - \frac{h_{\phi\phi,\theta}}{4h_{\phi\phi}}} \, d\phi. \] (34)

These equations give us our continuum embedding in \( \mathbb{R}^3 \). The function \( f(\theta) \) is a real valued function on the domain \( \theta \in [0, \pi] \). We construct the polyhedral metric for the ergosphere by first building a list of \( \theta \)s and \( \phi \)s for the vertices. Next we construct a list of triangles by vertices and assign edge lengths by isometrically embedding the vertices of edges and computing the Euclidean distance. After assigning edge lengths, we discard the coordinates and apply AIM to our new polyhedral metric. Figure 9 illustrates the continuum embedding, the smooth embedding given by AIM and the non-smooth embedding. The continuum embedding is a parametric plot, in Euclidean space, using (29)–(31) with \( \rho \) and \( f \) defined by (33) and (34), respectively. The percent error of the integrated mean curvature between the continuum and AIM embeddings is 0.62\%, which is comparable to the percent error for the drum at the same resolution. Although we did not analyze the convergence of integrated mean curvature for this surface, we expect its convergence to be similar to the ellipsoid and drum surfaces. Our main obstacle in completing this convergence analysis is producing consistent high quality triangulations for increasing resolution. The edge lengths are recovered up to the set accuracy of \( 10^{-6} \).
4. Conclusion

The AIM algorithm was developed using several techniques from previous isometric embedding algorithms. Our tests of AIM included non-axial symmetric surfaces with strictly positive Gaussian curvature. We also tested a surface just above the ergosphere of a Kerr black hole, which is axial-symmetric with regions of negative Gaussian curvature.

The main innovation of the AIM algorithm comes from the GAP. Although we used triangulated surfaces, similar to the wire mesh of Nollert and Herold, GAP allows us to distinguish between the global minima of smooth and non-smooth polyhedra. GAP also prevents the problem of getting trapped in local minima, as seen by Bondarescu, Alcubierre and Seidel, during Newton’s method. For the surfaces tested, we were capable of reaching residuals on the order of our prescribed tolerance. AIM also uses an embedding flow similar to Jasiulek and Korzyński’s, but is not restricted to strictly positive scalar curvature surfaces.

Although AIM does not necessarily produce the convex polyhedron given a convex polyhedral metric, we do produce embeddings that recover much of the extrinsic behavior of the continuum embedding. Although there are many approximate polyhedral representations of the convex Alexandrov polyhedron that our algorithm can produce, we observe that they do not vary significantly between each other. We find that this is especially true at higher resolutions. Our future plan is to provide an upper bound on the deviation in extrinsic curvature between these essentially equivalent embeddings.

It was not our goal or intention to optimize the AIM algorithm. Its run time and scaling may be significantly improved by using preconditioning for the optimization routines and parallelization. We would also like to develop an algorithm that produces well posed triangulation, similar to those made by distmesh, given any compact two metric. Since our original interest in isometric embeddings stemmed from computing QLE in general relativity, we plan to use AIM for this purpose. It is also noted that given a suitable triangulation algorithm, AIM should be readily applicable to two surfaces identified within numerical relativity codes that use finite differencing grids. One can interpolate to obtain the proper distances of the edges for the triangulation using the grid’s metric and coordinates. This polyhedral metric can then be embedded into $\mathbb{R}^3$ using AIM.

Acknowledgments

This work was supported from Air Force Research Laboratory (AFRL/RITA) Grant # FA8750-11-2-0089 and # FA5750-15-2-0047. WAM and SR acknowledge support from the Air Force Office of Scientific Research through the American Society for Engineering Educations 2012 Summer Faculty Fellowship Program, and from AFRL/RITA and the Griffiss Institutes 2013 Visiting Faculty Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the AFRL.

Appendix. Intrinsic curvature calculations

In section 2.1 we used simplicial Ricci flow to obtain a triangular mesh with constant Gaussian curvature at each vertex. This procedure relied on curvature values defined on the surface of our lattice. The purpose of this appendix is to inform the reader about the nature of curvature on simplicial geometries and provide the exact equation used to compute Gaussian curvature.
It was established by Alexandrov, and utilized by Regge [19], that the intrinsic curvature of a simplicial geometry, of arbitrary dimensions, is concentrated at co-dimension 2 simplices called hinges. The curvature at these hinges is a conic singularity whose value goes to infinity as the area of the loop of parallel transport enclosing the hinge goes to zero. Later it was determined by Miller et al [20, 21] that given a Daulanay lattice $S$ with a circumcentric Voronoi dual lattice $S^*$, it was possible to define a unique path of parallel transport such that the sectional and Gaussian curvatures are defined. The area $h^*$ of this loop is the Voronoi area perpendicular to a hinge in the Daulanay lattice. The Voronoi area $h^*$ is constructed using dual edges $\lambda \in S^*$ whose vertices are defined as the circumcenters of d-simplices. If one were to parallel transport a vector around this loop, they would find their vector rotated by the deficit angle at the hinge. The deficit angle is defined as $\epsilon_h = 2\pi - \sum_{i} \eta_i$, where the sum is over the interior angles of $d$-simplices that share the hinge $h$. Using the deficit angle and the dual area associated with $h$, the Gaussian curvature is defined as

$$k_h = \frac{\epsilon_h}{|h^*|}$$

(35)

Everything mentioned in this appendix is applicable for arbitrary dimensions. Since we are embedding an $S^2$ surface, we are only concerned with two-dimensions. This means the curvature is concentrated on vertices.

References

[1] Weyl H 1916 über die bestimmheit einer geschlossenen konvex fläche durch ihr linienelement Vierteljahrsschrift Der Nat.-Forschenden Ges., Zürich 61 40–72
[2] Hong J-X and Han Q 2006 Isometric Embedding of Riemannian Manifolds in Euclidean Spaces (Providence, RI: American Mathematical Society)
[3] Lewy H 1938 On the existence of a closed convex surface realizing a given Riemannian metric Proc. Natl Acad. Sci. USA 24 104
[4] Nirenberg L 1953 The Weyl and Minkowski problems in differential geometry in the large Commun. Pure. Appl. Math. 6 337–94
[5] Heinz E 1962 On Weyl’s embedding problem Appl. Math. Mech. 11 421–54
[6] Alexandrov A D 2006 Intrinsic Geometry of Convex Surfaces (Boca Raton, FL: CRC Press)
[7] Brown J D and York J W 1993 Quasilocal energy and conserved charges derived from the gravitational action Phys. Rev. D 47 1407–19
[8] Wang M-T and Yau S-T 2009 Quasilocal mass in general relativity Phys. Rev. Lett. 102 021101
[9] Szabados L B 2009 Quasi-local energy–momentum and angular momentum in general relativity Living Rev. Relativ. 12 88
[10] Bondarescu M, Alcubierre M and Seidel E 2002 Isometric embeddings of black-hole horizons in three-dimensional flat space Class. Quantum Grav. 19 375
[11] Jasiulek M and Korzyński M 2012 Isometric embeddings of 2-spheres by embedding flow for applications in numerical relativity Class. Quantum Grav. 29 14
[12] Nollert H and Herold H 1996 Visualization in curved spacetimes: II. Visualization of surfaces via embedding Relativity and Scientific Computing. Computer Algebra, Numerics, Visualization (Berlin: Springer) pp 330–51
[13] Kane D, Price G N and Demaine E D 2009 A pseudopolynomial algorithm for Alexandrov’s theorem Lecture Notes Comput. Sci. 5664 435–46
[14] Bobenko A I and Izmestiev I 2008 Alexandrov’s theorem, weighted delaunay triangulations, and mixed volumes Ann. Inst. Fourier 58 447–505
[15] Chow B and Luo F 2003 Combinatorial Ricci flows on surfaces J. Differ. Geom. 63 97–129
[16] Persson P-O 2005 Mesh generation for implicit geometries PhD Thesis Massachusetts Institute of Technology
[17] Conboy R, Miller W A and Ray S 2015 Distributed mean curvature on a discrete manifold for Regge calculus Class. Quantum Grav. 32 18
[18] Smarr L 1973 Surface geometry of charged rotating black holes Phys. Rev. D 7 289
[19] Regge T 1961 General relativity without coordinates *Nuovo Cimento* **19** 558–71
[20] Miller W A 1997 The Hilbert action in Regge calculus *Class. Quantum Grav.* **14** 199–204
[21] Miller W A, McDonald R J, Alsing M P, Gu X D and Yau S-T 2014 Simplicial Ricci flow *Commun. Math. Phys.* **329** 579–608