An image-space algorithm for immersive views in 3-manifolds and orbifolds

P. Berger [*], A. Laier [†], L. Velho [‡]

January 15, 2013

Abstract

We propose a new image-space algorithm to generate an immersive view in flat or hyperbolic 3-manifolds, orbifolds and more generally polyhedral complexes. The complexity of the algorithm is linear instead of being exponential in depth, as all the previous algorithms developed for this application.

1 Introduction

With the Geometry Center [15], William Thurston initiated a program to disseminate modern geometry using interactive visualization. This initiative resulted in software which provided immersive views of 3-manifolds or 3-orbifolds. The goal of such visualization algorithms is to render what one would see if he/she would lived inside that space. In particular, hyperbolic manifolds, which will be formally defined further, are fundamental: according to Thurston [17], most of the manifolds are hyperbolic — i.e., there is a very rich variety of them.

In a view of a compact manifold, a single object appears many times, and in general its images fill up all the horizon. Indeed a ray of light might loop and such paths can represent every element of the manifold's fundamental group. In hyperbolic manifolds, the sizes of the object's copies are exponentially small w.r.t. to their distance to the camera, whereas the number of such copies is exponentially large (since the fundamental group grows exponentially fast).

We present here a new image-space visualization algorithm, based on ray tracing, to render immersive views in flat or hyperbolic manifolds, orbifolds and more generally polyhedral complexes. As a consequence, the proposed algorithm is much more efficient than the previous algorithms based on object-space visualization.

For typical example, it allows interactive visualization of high-resolution images of scenes with full optical complexity, and can render a scene formed by up to 9^{100} copies of the same object

[*]berger@math.univ-paris13.fr, CNRS-LAGA Université Paris 13, partially supported by the Balzan Research Project of J. Palis

[†]alex.laier@gmail.com, Universidade Federal Fluminense

[‡]lvelho@impa.br, Instituto de Matematica Pura e Aplicada

instead of 9^4 copies, as in previous algorithms [1]. Amazingly, although this may seem unreachable, such a great number of objects reaches subpixel precision and changes dramatically the pattern of the landscape with respect to the one formed by only 9^4 objects as is illustrated in Figure 1.

The previous algorithms dealt only with flat or hyperbolic *developable orbifolds*, the proposed algorithm deals with the more general category of flat or hyperbolic *polyhedral complex* which includes the stratified spaces of constant non positive curvature [3]. These "geometric" polyhedral complexes will be defined in section 5. Such general objects appear in cosmology [5] and the study of their geodesic flows is an active area of research in geometry.

The latter algorithm are also generalizable (as the previous algorithms) to geometries which are: spherical $\mathbb{S}^3$, the product $\mathbb{H}^2 \times \mathbb{E}^1$ of the Hyperbolic plane with Euclidean line, the product $\mathbb{S}^2 \times \mathbb{E}^1$ of the 2-sphere with Euclidean line, and the universal covering $\widetilde{SL}_2$ of the modular group.

2 Related Work

As previously mentioned the seminal work in mathematics visualization was developed at the Geometry Center, during the period of 1994 to 1998. For that purpose, the main software platform of the center was Geomview, an interactive 3D graphics viewer [1]. This software featured a powerful plugin architecture, which allowed it to be extended through new modules, to a variety of uses ranging from animation to video production and scientific simulation. Geomview was based on a generic object-oriented graphics library, OOGL. It was initially developed for the Silicon Graphics workstation using the GL interactive low-level graphics. It also allowed output in the Renderman format for off-line high quality rendering. Subsequently, Geomview was ported to the X11 Linux platform using OpenGL, and it is still in use today by many practitioners and researchers, even after the termination of the Geometry Center activities in 1999.

By design, Geomview supported viewing in Euclidean, spherical and hyperbolic geometries exploiting the fact that the graphics pipeline using projective transformations allowed an elegant unification of the modeling and camera transformations for these three geometries.

The plugin architecture of Geomview, made possible, among other things the development of *Maniview*, a module that works as a Manifold Viewer [8]. Maniview implements the discrete group structure to generate the elements of $PGL(\mathbb{R}, 4)$ required to allow the visualization of the insider's view of a manifold.

Maniview has also been used in conjunction with *Snappea*, a program developed at the Geometry Center by Jeffrey Weeks for computing hyperbolic structures on three-dimensional manifolds [19]. Other software for modeling and investigation in 3-manifolds includes Regina [4], implemented in C++ and used in various related projects.

Subsequently, Weeks, created a software for real-time rendering of curved spaces using OpenGL graphics [20]. It is a standalone program that exploits features of the programmable graphics pipeline, such as shaders and textures.

[1] In comparison with previous work, our algorithm can render ray paths 25 times deeper than the ones in object-space algorithms. This is a significant improvement even if we consider the recent development in graphics hardware.

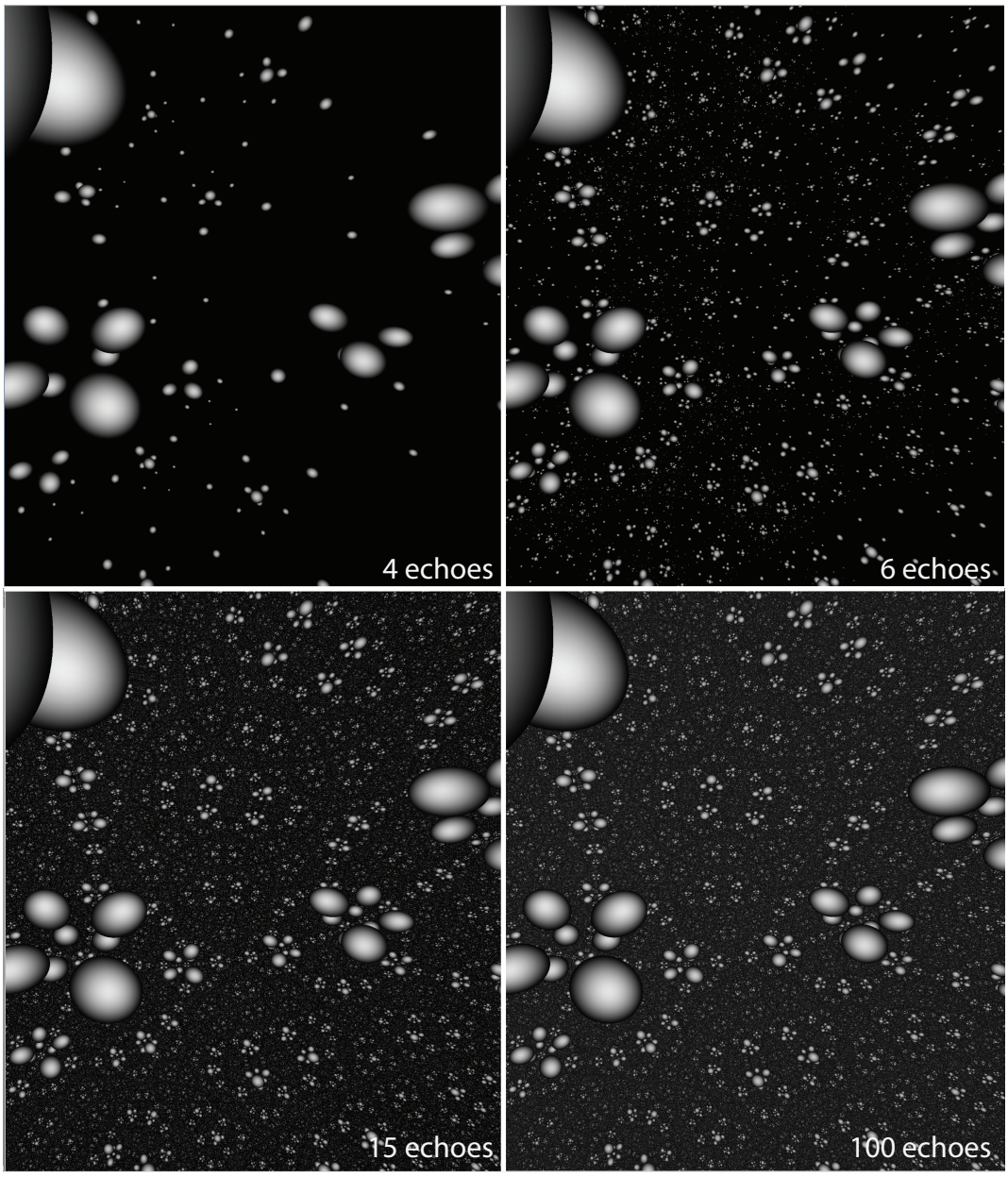

Figure 1: Landscape given by a single sphere in the mirrored hyperbolic dodecahedron rendered with different levels of echoes.

Similarly to Geomview, jReality is a Java based, 3D scene graph package designed for mathematical visualization at TU-Berlin and associated research centers [23]. Among other features, it provides JOGL, a backend for interactive OpenGL rendering. JReality has intrinsic support for Euclidean, hyperbolic and spherical geometries, which can be used for creating immersive views of 3-manifolds.

One of the main goals of using computer graphics methods for mathematics visualization is to provide insights into the world of geometric structures, both for research and education purposes [10]. In that respect, the Geometry Center produced highly successful videos, such as "Not-Knot" [7] and "The Shape of Space" [18], that were complemented by illustrated books [22].

At another level, Virtual Reality installations allows the user, not only to have a glimpse at the visual landscape inside a 3-manifold, but also to experiment the sensation of being completely immersed in such enviroment. Two projects following this directions are Mathenautics [12] and Alice [6].

In addition to direct visualization and interaction, the properties of such geometric structures can be exploited to investigate other types of data, such as the World Wide Web connections [14].

A common characteristic of all visualization software developed in previous work for creating immersive views of 3-manifolds is that they employ object-space rendering algorithms. This fact, in practice, imposes a limit on the scene complexity which can be depicted by these programs. In contrast, the new visualization program proposed in this paper uses an image-space algorithm which can efficiently render scenes with several orders of magnitude more complexity than possible in previous work. For a comparison, see Figure 1.

3 Three-Manifolds and Orbifolds

A 3-*manifold* is a paracompact, metric space locally modeled by *charts* on open sets of $\mathbb{R}^3$, such that the coordinate changes are diffeomophisms of open sets of $\mathbb{R}^3$. The unit sphere of $\mathbb{R}^4$ and the torus $\mathbb{R}^3/\mathbb{Z}^3$ are examples of 3-manifolds.

A 3-*orbifold* is a paracompact, metric space locally modeled by *charts* on open sets of quotients of $\mathbb{R}^3$ by finite groups, such that the coordinate changes are homeomorphisms of the quotient which lift to diffeomorphisms of open sets of $\mathbb{R}^3$. We remark that a manifold is an orbifold.

Also, if the finite group acts by a single symmetry with respect to a plane, then the quotient is $\mathbb{R}^2 \times \mathbb{R}_+$. If it is the unique action involved (with the trivial one), then the orbifold is a manifold with boundary. If the action of the finite group is done by symmetries with respect to a plane in general position, then the group is $(\mathbb{Z}/2\mathbb{Z})^3$ and the quotient is diffeomorphic to $\mathbb{R}^3_+$. If it is the only action involved, then the orbifold is a manifold with corners. Other kinds are possible, for example see [2] or [17].

A *Riemannian metric* on an open set U of $\mathbb{R}^3$ is a smooth family of inner products of $\mathbb{R}^3$, indexed by the points of U. A *Riemannian metric* on a manifold is the data of a Riemannian metric on the image for every chart, being pairwise compatible by the coordinate changes. A *Riemannian metric* on an orbifold is the data of an equivariant Riemannian metric on the lifted image of every chart,

being pairwise compatible by the lifted coordinate changes.

An orbifold endowed with a Riemannian metric is a *Riemannian orbifold*.

Orbifolds are canonically stratified. An orbifold is a manifold if its stratification is trivial, that is, it has no singularity.

Remark 3.1. For every Riemannian 3-orbifold, the dihedral angle around any stratum of dimension 1 is a divisor of 2π. In particular, if a Riemannian 3-orbifold is a manifold with corner then the dihedral angle between any two faces is a divisor of 2π.

A parametrized curve in a manifold is *differentiable* if its composition with any chart is differentiable (on its domain of definition). A parametrized curve in an orbifold M is *differentiable* if its composition with the chart lift to a differentiable curve on its domain.

A *geodesic* is a smooth curve between two points such that its length is locally minimal. The *length* of a smooth curve is the integral of the norm of a unit vector tangent to the curve, with respect to the Riemannian metric. Whenever the space is connected, given two points there exists a geodesic passing by both of them. Among all geodesics passing by them, the minimal length of the geodesic segment between these points defines a distance.

Geodesics are the paths taken by rays of light. From a unit vector at a point of the manifold, there is a unique geodesic in that direction.

These definitions are crucial since they define the immersive view in a manifold. Given an object embedded in a manifold, a viewer in the manifold will see in a given direction u, the first intersection between the object and the half geodesic starting at the viewer position and in the direction u.

Many different Riemannian metrics can be employed.

If the metric is with curvature constant and equal to 0, then the Riemannian orbifold is *Euclidean* or *flat*. For instance the 3-torus is a flat manifold.

Example 3.2 (Euclidean space $\mathbb{E}^3$)**.** Let N be the Euclidean norm: $N(x_1, x_2, x_3) = \sqrt{x_1^2 + x_2^2 + x_3^2}$ on $\mathbb{R}^3$. The *3-Euclidean space* $\mathbb{E}^3$ is $\mathbb{R}^3$, with a Riemannian metric the inner product $\sum_i x_i \cdot x'_i$ induced by N. The Euclidean space has zero curvature. The geodesics of $\mathbb{E}^3$ are affine lines. The (global) metric on $\mathbb{E}^3$ is $d_{\mathbb{E}^3}(\underline{x}, \underline{x}') := N(\underline{x} - \underline{x}')$.

The group of isometries is the semi-product $O(3) \ltimes \mathbb{R}^3$ of the orthogonal group and translation groups of $\mathbb{E}^3$.

If the metric is with curvature constant and negative, then the Riemannian orbifold is *hyperbolic*. A basic example of hyperbolic manifold is the hyperbolic space.

Example 3.3 (Klein model $\mathbb{H}^3$ of the hyperbolic space)**.** Let $q\colon \mathbb{R}^4 \to \mathbb{R}$ be the (Lorenzian) quadratic form defined by $q(x_1, x_2, x_3, x_0) = x_0^2 - x_1^2 - x_2^2 - x_3^2$. The *3-hyperbolic space* $\mathbb{H}^3$ is

$$\{\underline{x} \in \mathbb{R}^4 : \ q(\underline{x}) = 1\} \cap \mathbb{R}^3 \times \mathbb{R}^+$$

endowed with the metric g equal to the restriction of $-q$ to the tangent space of $\mathbb{H}^3$. The space $\mathbb{H}^3$ is a Riemannian 3-manifold of constant negative curvature.

The geodesics $\mathbb{H}^3$ cannot be straight, since $\mathbb{H}^3$ is not an affine subspace of $\mathbb{R}^4$. However, the geodesic passing trough $x \in \mathbb{H}^3$ in the direction u tangent to $\mathbb{H}^3$ at x is the intersection of $\mathbb{H}^3$ with the 2-plane spanned by the vectors $0\vec{x}$ and u of $\mathbb{R}^4$.

The bilinear symmetric form associated to q is $B(\underline{x},\underline{x}') = x_0 \cdot x'_0 - x_1 \cdot x'_1 - x_2 \cdot x'_2 - x_3 \cdot x'_3$, with $\underline{x} = (x_i)_i$ and $\underline{x}' = (x'_i)_i$. The (global) metric on $\mathbb{H}^3$ is $d_{\mathbb{H}^3}(\underline{x},\underline{x}') := \operatorname{arcosh} B(\underline{x},\underline{x}')$.

The group of isometries of $\mathbb{H}^3$ is equal to the group $O(3,1)$ of 4×4-matrices which leaves invariant the form q. Every matrix of this group acts on $\mathbb{R}^4$ by matrix product and leaves invariant $\mathbb{H}^3$. By definition of g, every matrix of $O(3,1)$ acts as an isometry of $(\mathbb{H}^3, g)$.

Example 3.4 (3-Sphere $\mathbb{S}^3$)**.** Let us denote also by N the Euclidean norm of $\mathbb{R}^4$: $N(x_1,x_2,x_3,x_4) = \sqrt{x_1^2+x_2^2+x_3^2+x_4^2}$. The *3-sphere space* is

$$\mathbb{S}^3 := \{\underline{x} \in \mathbb{R}^4 :\ N(\underline{x}) = 1\}$$

endowed with the metric g_1 equal to the restriction of the Euclidean inner product $\sum_{i=1}^4 x_i \cdot x'_i$ to the tangent space of $\mathbb{S}^3$. The space $(\mathbb{S}^3, g_1)$ is a Riemannian 3-manifold of constant positive curvature.

The geodesic from $x \in \mathbb{S}^3$ in the direction u tangent to $\mathbb{S}^3$ at x is the intersection of $\mathbb{S}^3$ with the 2-plane spanned by the vectors x and u of $\mathbb{R}^4$.

The group of isometries of $\mathbb{S}^3$ is equal to the orthogonal group $O(4)$ of 4×4-matrices which leaves invariant the norm N.

A fundamental way to construct 3-orbifolds (and manifolds in particular) is to take a discrete subgroup Γ of $O(3) \rtimes \mathbb{R}^3$ (resp. $O(3,1)$, resp. $O(4)$) and then regard the orbit space $M := \Gamma\backslash\mathbb{E}^3$ (resp. $\Gamma\backslash\mathbb{H}^3$) of the Γ-action. It is easy to show that M is an orbifold. Such flat (resp. hyperbolic, spherical) orbifolds are called *developable*.

Example 3.5. The 3-torus is the quotient $\mathbb{Z}^3\backslash\mathbb{E}^3$ where the (free) action of $\mathbb{Z}^3$ on $\mathbb{E}^3$ is done by translation.

More generally there is the following result (see Thm 8, [9]):

Theorem 3.6. *Every orbifold (and so manifold) with constant non-positive curvature is developable.*

Proposition 6 of [9] shows that there are orbifolds of constant positive curvature which are not developable.

In Section §7 we will recall the scheme of all previous algorithms for rendering flat and hyperbolic orbifolds. They correspond to object space algorithms.

Here we introduce a rendering algorithm for immersive views in the more general class of polyhedral complexes, in which polyhedrons are either flat or hyperbolic. It is an image space algorithm.

4 Visualization of 3-dimensional spaces

A 3D visualization algorithm renders an image of a three-dimensional scene according to a view specification. The input of the algorithm is a scene description composed of: an ambient 3d space; 3d shapes placed in this ambient space and a viewpoint, among other parameters. The output is a two dimensional view. In that sense, the rendering process transforms geometric three-dimensional information into visual two-dimensional information.

4.1 The Viewing Transformation Pipeline

In order to understand better this process, let us recall the *viewing transformation pipeline*, which relates the different spaces and coordinate systems involved in the computation of a rendered image.

Each shape of an object $o \in S$ is naturally defined in it's own local *object coordinate system*. The object can be described in parametric or implicit form. All the objects are then placed in the global *world coordinate system* of the scene S by a modeling transformation that embeds the object into the ambient space (which can be a manifold or orbifold). The view is specified by a viewing transformation V, which defines the *camera coordinate system* relative to the world (e.g., its position and orientation). Finally, the objects that are visible from the camera are mapped to the *image coordinate system* (which implements the viewing window). This pipeline is shown in Figure 2.

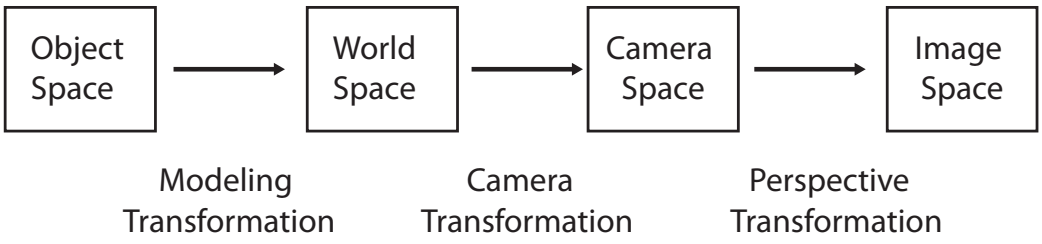

Figure 2: Viewing Transformation Pipeline .

The transformations of the viewing pipeline can be defined by real projective mappings in homogeneous coordinates in $\mathbb{RP}^3$. This scheme has been adopted as a standard in most graphics systems because it unifies all the transformations involved using 4×4 projective matrices. In that way, the 3D objects are immersed in projective space, transformed and then, projected to a 2D image space.

This particular way in which the viewing pipeline is defined opens up the possibility to render views of ambient spaces different than the Euclidean space $\mathbb{E}^3$, in particular, with a flat or hyperbolic geometry, as it will be seen below.

4.2 Types of Algorithms

There are two main types of 3D visualization algorithms. They can be classified into:

- object space;
- image space.

Object space algorithms apply the direct viewing transformation to points of the objects, while image space algorithms apply the inverse viewing transformation to rays originating from the camera and corresponding to image pixels.

Algorithm 1 Object-Space Visualization

```
for each o ∈ S do
    Map o from scene to camera space
    if o is visible then
        Project o to image space
    end if
end for
```

The structure of these two types of algorithms is described by the pseudo-codes below:

The object-space algorithm is widely adopted in Computer Graphics and is the one used in the OpenGL standard.

Algorithm 2 Image-Space Visualization

```
for each pixel p ∈ I do
    Generate a ray r in camera space
    Transform r to scene space
    Find the intersection i(r) with visible object o ∈ S
    if i(r) ≠ ∅ then
        Paint pixel
    end if
end for
```

The image-space algorithm is the basis of *ray tracing* rendering methods.

Note that object-space algorithms work at geometric precision and, in principle, must perform full evaluation when transforming objects in the scene, while image-space algorithms work at image resolution and may perform a lazy evaluation of transformations required by each ray.

Furthermore, the opposite nature of these two algorithms have a determinant impact on the complexity of the visualization process and on the strategies used to make them more efficient. This depends on various factors, such as scene and depth complexity, among others.

The bottom line is that rendering acceleration methods exploit some kind of coherence in different classes of scenes. We will return to this point further in the paper, regarding the immersive views of 3-manifolds.

4.3 Inside Views in Non-Euclidean spaces

With a few exceptions, mentioned in Section 2, most 3D visualization software support only rendering of scenes in the Euclidean space $\mathbb{E}^3$. However, exploiting the fact that the visualization process resorts to projective transformations in order to create images of 3D spaces, it is possible to render different model geometries without structural changes to the visualization algorithm, by using the appropriate transformations in the viewing pipeline.

Thus, beside the Euclidean space $\mathbb{E}^3$, other isotropic model geometries, such as spherical and hyperbolic can be rendered by the above visualization algorithms. The specialization required in

the pipeline amounts to taking care of the correct transformations that respect the intrinsic metric of the model geometry, as discussed in the previous section.

The isometries for a given space (i.e., flat, hyperbolic or spherical) define the set of model transformations in the viewing pipeline. They are combined with the camera and perspective transformations, which are defined by the view specification

These transformations are subsumed as elements of $PGL(\mathbb{R}, 4)$, the projective general linear group, and represented as 4×4 transformations matrices in the visualization algorithm.

To render inside views of 3-dimensional spaces, we generate images that would be seen by an observer (e.g. a camera) placed in that space. Light paths follow geodesics and reveal the topology of the space. In the next section we will discuss in greater detail the construction of manifolds/orbifolds, and implications to generate inside views of these spaces.

5 Constructions of 3-dimensional spaces by polyhedral complexes

A simple way to construct a 2-torus is to glue pairwise the opposite edges of a filled square. The same applies to construct a 3-torus: we merely glue pairwise the opposite faces of a filled cube.

An observer living in this 3-torus, looking towards above would see himself from below. Similarly, looking towards the right he would see himself from the left, and so on for other directions.

The algorithm below is an image space algorithm to view an object inside this 3-torus from a camera placed at the observer's position.

(Ray tracing inside a 3-torus)

for each pixel $p \in I$ **do**
  Generate a ray r from the camera
  for $i \leq level$ **do**
    Find the first intersection $i(r)$ with object
    if $i(r) \neq \emptyset$ **then**
      Paint pixel **break**
    else The ray r intersects a face F of the cube
      Translate r so that it continues it's path from the opposite face of the cube.
    end if
  end for
end for

Let us generalize the above mathematical construction for the 3-torus, to get the intuition of the new algorithm for immersive views in hyperbolic or flat orbifolds, and more generally polyhedral complexes.

5.1 Polyhedral complex

We can extend the concept of polyhedral complex to Riemannian manifolds (M, g) which are not necessarily the Euclidean space, see [3]. For the sake of simplicity, we shall deal only with dimension 3: in particular the dimension of M is 3. Let us recall a few classical definitions on which polyhedral complex are based.

A surface S of M is *totally geodesic* if every geodesic of M tangent to S is included in it. A compact subset C of M is *convex* if every pair of points in C can be joined by a geodesic segment included in C. A *polygon of* (M, g) is a convex topological disk included in a totally geodesic surface, whose boundary is formed by a finite union of geodesic segments. The geodesic segments are the *edges* of the polygon, and their end points are the *vertices*.

A *(convex) polyhedron* D is a convex topological closed ball embedded in (M, g), the boundary of which is a finite union of polygons $(P_i)_{i \in I_D}$ with disjoint interiors and such that every non-empty intersection $P_i \cap P_j$ is equal to a common edge or a vertex of both P_i and P_j.

For any finite family $(D_\alpha)_{\alpha \in J}$ of polyhedrons of (M, g), the compact subset $\sqcup_J D_\alpha$ of the J-copies $\sqcup_J M$ of M is called the *disjoint union* of $(D_\alpha)_{\alpha \in J}$. Let us denote by $\hat{I} := \sqcup_J I_{D_\alpha}$ the set which indexes the polygons of forming the boundary of $\hat{D}$.

Definition 5.1. A (M, g)*-polyhedral complex structure* is:

- a finite disjoint union of polyhedrons $\hat{D} := \sqcup_J D_\alpha$,
- an involution σ of the set of faces indexes $\hat{I}$ of $\hat{D}$,
- an isometry g_i from P_i onto $P_{\sigma(i)}$, such that $g_i^{-1} = g_{\sigma(i)}$, for every $i \in \hat{I}$.

such that the relation

$$x \sim y \text{ iff } (x = y) \text{ or } (\exists i \in I : x \in P_i \text{ and } g_i(x) = y)$$

spans an equivalence relation on D. The equivalence classes form a topological space $S := D/\sim$ called a (M, g)-*polyhedral complex*.

5.2 Flat polyhedral complex

If the Riemannian manifold (M, g) is the Euclidean space $\mathbb{E}^3$, then the polyhedral complex S is called *flat* (or Euclidean).

For instance the 3-torus has a structure of flat polyhedral complex, where $\hat{D}$ is formed by a unique polyhedron equal to a filled cube $[0, 1]^3$, and the isometries are the translations $(0, 0, \pm 1)$, $(0, \pm 1, 0)$ and $(\pm 1, 0, 0)$.

We can obtain other spaces by changing the translations $(0, 0, \pm 1)$ by composing them with the symmetry about the axis $\{0\}^2 \times \mathbb{R}$.

Example 5.2 (A flat polyhedral complex which is not an orbifold)**.** Let P denote the regular planar octagon centered at 0. The angle of P are all equal to $3\pi/4$. The product of P with $[0, 1]$ is a polyhedron of the Euclidean space $\mathbb{E}^3$. As before we glue the opposite faces with translations.

The polyhedral complex obtained is topologically the product of torus of genus 2 with a circle. However it is not anymore diffeomorphic to it. For the quotient identifies all the vertical edges and immerses them to a circle; in the polyhedral complex, around this circle the angle is 6π. To be an orbifold, the angle must be a divisor of 2π.

5.3 Hyperbolic polyhedral complex

If the Riemannian manifold (M, g) is the hyperbolic space $\mathbb{H}^3$, then the polyhedral complex S is called *hyperbolic*.

Observe that every hyperbolic polyhedron D is sent by the map:

$$p : \mathbb{H}^3 \ni (x_1, x_2, x_3, x_0) \mapsto \left(\frac{x_1}{x_0}, \frac{x_2}{x_0}, \frac{x_3}{x_0}\right) \in \mathbb{R}^3$$

to an Euclidean polyhedron.

Let us consider a regular dodecahedron of the Euclidean space centered at zero. For a certain diameter of it, it is included in the unit ball, and its preimage by p is a regular dodecahedron of the hyperbolic space, such that all the dihedral angles between its faces are $\pi/2$. We can glue the faces by the identity of each of them. The hyperbolic polyhedral complex obtained is an orbifold, and more precisely it is a manifold with corner, called *hyperbolic mirrored dodecahedron*.

The same construction can be done for any hyperbolic polyhedron. Then the dihedral angle between the faces is not necessarily a divisor of 2π and so the polyhedral complex is not necessarily a differentiable orbifold. Such a generality occurs for instance in [5], where the Einstein equation near a singularity are modeled by the geodesic flow on a hyperbolic polyhedron, the faces of which are mirrors.

5.4 Others homogeneous polyhedral complexes

We can certainly take for (M, g) the sphere $\mathbb{S}^3$, the universal covering $\widetilde{SL}_2$ of the modular group, or the products $\mathbb{E}^1 \times \mathbb{H}^2$ and $\mathbb{E}^1 \times \mathbb{S}^2$ of the Euclidean line with the 2-hyperbolic space and the 2-sphere respectively.

Those are four other Thurston geometries. We believe that for paradigmatic examples of orbifolds of these geometries, the image space algorithm described in the next section works as well, after a few tricks. The tricks are to deal only with geodesics which project to real lines of the Euclidean space. Such tricks were performed in [20].

6 An image-space algorithm for immersive views in 3-non Euclidean spaces.

We are now ready to describe the new image-space algorithm for visualization in 3-orbifolds and more generally polyhedral complexes M which are flat or hyperbolic.

We recall that we want to render the view from a camera in M of an object O inside M.

6.1 Ray tracing in flat polyhedral complexes

Let $M = \hat{D}/\sim$, where $\hat{D}$ is an union of Euclidean polyhedrons (in $\mathbb{E}^3$) with faces $(F_i)_i$. Let $O_0 \subset \hat{D}$ be the preimage of O by $\pi\colon \hat{D} \to M$. Remember that $\hat{D}$ is a disjoint union of closed topological balls. We take the same orientation for each component of the boundary ∂D. Observe that there exists a unique way to extend each isometry $g_i\colon F_i \to F_{\sigma(i)}$ to an isometry $\tilde{g}_i$ of $\mathbb{R}^3$ which push forward the orientations of F_i to the opposite orientation of $F_{\sigma(i)}$. Observe that the differential dg_i of g_i is constantly equal to an element of $O(3)$.

In contrast to Algorithm 5 in the 3-torus example, the union of polyhedrons is not necessarily connected.

A *ray* R from a point $c \in D$ in the direction $u \in \mathbb{R}^3 \setminus \{0\}$ is the segment:

$$\cup_{t\geq 0}\{c + t \cdot u :\ \forall s \in [0, t]\ c + s \cdot u \in D\}.$$

which is parametrized by t.

To trace rays from the camera, each pixel of the screen is canonically associated to a ray R, such that c is the center of projection and u is the view direction for the pixel:

There are two possibilities:

(i) If R intersects O_0, then the pixel color is computed from the intersection of R with O_0.

(ii) If R does not intersect O_0, then we compute the intersection point x of R with the boundary of D – The case where x belongs to two different faces F_i is of probability 0 and so is not considered. In other words, we assume that x belongs to a unique face F_i. Let u be the

direction of R. We continue the ray path using the new ray R' starting at c' and in the direction u' given by:

$$c' := g_i(x) \quad u' = d_x g_i(u). \tag{6.1}$$

Note that when R does not intersect O_0, the above procedure is repeated with R' instead of R, until a maximum number of echoes is reached, then a background color is painted.

6.2 Ray tracing Hyperbolic polyhedral complexes

We let $M = \hat{\Delta}/\sim$, where $\hat{\Delta}$ is a disjoint union of hyperbolic polyhedrons (in $\mathbb{H}^3$) with faces $(\Pi_i)_i$. Let $O' \subset \hat{\Delta}$ be the preimage of the object O by $\pi\colon \hat{\Delta} \to M$. We chose as before a particular orientation on the boundary $\partial\hat{\Delta}$. Again there exists a unique way to extend each isometry $g_i\colon \Pi_i \to \Pi_{\sigma(i)}$ to an isometry $\tilde{g}_i$ of $\mathbb{H}^3$ which push forward the orientations of Π_i to the opposite orientation of $\Pi_{\sigma(i)}$.

Let O_0 be the image of O' by the diffeomorphism

$$p:\ \mathbb{H}^3 \ni (x_1, x_2, x_3, x_0) \mapsto \left(\frac{x_1}{x_0}, \frac{x_2}{x_0}, \frac{x_3}{x_0}\right) \in \mathbb{R}^3$$

onto the unit ball of $\mathbb{R}^3$. Remember that $\hat{D} := p(\hat{\Delta})$ is a disjoint union of Euclidean polyhedrons. Observe that $O_0 = p(O')$ is included in $D = p(\Delta)$.

As the image of geodesics of $\mathbb{H}^3$ by p are lines, we are going to deal with rays of D, as defined in section 3.

However there are two significant differences in the algorithm for hyperbolic polyhedral complexes.

The first difference is due the fact that in general the differential of p does not preserve the angles of $\mathbb{H}^3$ to those of $\mathbb{E}^3$. However, the angles at the point $c_0 = (0, 0, 0, 1) \in \mathbb{H}^3$ are preserved.

Therefore, we suppose that the camera is at c_0 which is itself in the interior of Δ, even if it means moving the scene by an (orientation preserving) isometry of $\mathbb{H}^3$.

The second difference is from the fact that g_i are maps of $\mathbb{H}^3$ and not of $\mathbb{R}^3$. Thus when a ray intersects a face F_i of $\hat{D}$, we shall continue the ray path by using the new ray R' starting at c' and in the direction u' given by:

$$c' := p \circ g_i \circ p^{-1}(x) \quad u' = d_x(p \circ g_i \circ p^{-1})(u). \tag{6.2}$$

Analytical expression of $\psi_i = p \circ g_i \circ p^{-1}$. We recall that:

$$p(x_j)_j = \left(\frac{x_j}{x_0}\right)_j$$

Observe that p is homogeneous: on every vectorial line it takes a unique value. On the other hand g_j is linear. The following inclusion is helpful:

$$\mathrm{inc}\colon \mathbb{R}^3 \ni (y_j)_j \mapsto (y_1, y_2, y_3, 1) \in \mathbb{R}^4.$$

We remark that $\mathrm{inc}(y_j)_j$ belongs to the same line as $p^{-1}(y_j)_j$, and so $g_i \circ \mathrm{inc}(y_j)_j$ and $g_i \circ p^{-1}(y_j)_j$ belongs to the same line, as well. Thus:

$$\psi_i = p \circ g_i \circ \mathrm{inc}$$

Observe that

$$dg_j = g_j, \quad d\,\mathrm{inc}(u_j)_{j=1}^3 = (u_1, u_2, u_3, 0), \quad d_{(x_j)_j} p(u_j)_j = \left(\frac{u_j x_0 - x_j u_0}{x_0^2} \right)_j$$

This provides an easy way to compute the expression for the differential of ψ_i:

$$d_{\underline{y}} \psi_i = (d_{g_i \circ \mathrm{inc}(\underline{y})} p) \circ g_i \circ d\,\mathrm{inc}$$

6.3 General Algorithm

In summary, the general image-space algorithm for immersive views in flat or hyperbolic polyhedral complexes is as follows:

Algorithm 3 Image-Space visualization in a polyhedral complex

```
for each pixel p ∈ I do
    Let c := 0 and let u be the direction associated to p
    Generate a ray r from (c, u).
    for i ≤ level do
        Find the intersection i(r) with visible object O_0
        if i(r) ≠ ∅ then
            Paint pixel break
        else The ray r intersects a face F_i of the disjoint union polyhedrons D̂
            (⋆) Compute the new origin c' and direction u' for the continuation of the ray path.
        end if
    end for
end for
```

The step $(\star)$ is given by expressions (6.1) and (6.2) for polyhedral complex which are flat and hyperbolic respectively.

We remark that this algorithm has complexity linear with the length of the ray path (asymptotic to *level*), that is the number of the echoes. Also, this algorithm uses ray tracing and enables us to represent analytical surfaces easily and perform computations in closed form.

7 Object-space algorithms for immersive views in 3-orbifolds

Up to our knowledge all the previous rendering methods for immersive views in non-Euclidean spaces were based on image-space algorithms. In order to compare the performance of these two

approaches we shall recall their principles. As there is a large literature on this subject we will be brief. For more details see [16, 8, 21].

Object-space algorithms were implemented to render immersive views in manifolds or orbifolds which are developable in the geometry G equal[2] to $\mathbb{H}^3$ or $\mathbb{E}^3$.

Let $\hat{\Delta}$ be a G-polyhedral complex with isometries $\{g_i\}_{i\in I}$ of G which "glue" the faces of $\hat{\Delta}$.

Let Γ be the group spanned by the isometries $\{g_i\}_{i\in I}$:

$$\Gamma = \{g_{i_n} \times \cdots \times g_{i_1} : \ (i_j)_j \in I^{(\mathbb{N}}\}.$$

The object-space algorithms works for less general spaces. This restricts our generality by supposing moreover the two following conditions:

(a) the disjoint union of polyhedrons $\hat{\Delta}$ consists of a unique polyhedron Δ, and so it is connected and convex,

(b) $\forall g \in \Gamma, \quad \forall x \in int(\Delta), \quad g(x) = x \Leftrightarrow g = identity$, where $int(\Delta)$ denotes the interior of the polyhedron Δ.

A stronger condition than (b) is the following:

(b') $\forall g \in \Gamma, \quad \forall x \in \Delta, \quad g(x) = x \Leftrightarrow g = identity.$

The polyhedron Δ is called *a fundamental domain*, Γ is called the *fundamental group* (of the orbifold) and G is called the *universal covering* (of the orbifold).

By condition (b), we can extend to G the equivalence relation $\sim$ defining the polyhedral complex $\Delta/\sim$ by the following for $x, y \in G$:

$$x \sim y \Leftrightarrow \exists g \in \Gamma : y = g(x).$$

Also the polyhedral complex $\Delta/\sim$ is equal to the space of equivalence classes $G/\sim$. The latter space is usually denoted $\Gamma\backslash G$.

The following is well known:

Proposition 7.1. *If conditions (a) and (b) hold, the space $\Gamma\backslash G$ is an orbifold. If conditions (a) and (b') hold, the space $\Gamma\backslash G$ is a manifold. Every compact, connected orbifold (and so manifold) of a non positive curvature is homotopic to one of the form $\Gamma\backslash G$, with Γ satisfying (a) and (b).*

One can see that the images of the fundamental domains $(g(\Delta))_{g\in\Gamma}$ tile the universal covering G:

$$\bigcup_{g\in\Gamma} g(\Delta) = G$$

From condition (b), the union $\cup_{g\in\Gamma} g(int(\Delta))$ is disjoint.

[2]Actually Weeks also implemented this object space algorithm for the geometries $\mathbb{S}^3$, $\mathbb{S}^2\times\mathbb{E}^1$, $\mathbb{H}^2\times\mathbb{E}^1$ and $\widetilde{SL}_2(\mathbb{R})$ but the basic idea is the same and our presented algorithm seems to be generalizable without substantial change.

Let us explain the main difference between our image-space algorithm and the object-space algorithm. Suppose that we want to render an object O embedded in $\Gamma\backslash G$. Let $O_0 \subset \Delta$ be the preimage of O by p.

With our image-space algorithm when a ray R gets out of the polyhedron Δ to enter to the polyhedron $g(\Delta)$, we merely continue the ray path to $g^{-1}(R)$.

For the object-space algorithm when a ray R gets out of the polyhedral Δ to enter to the polyhedron $g(\Delta)$, we must make a copy of the object $g(O_0)$ and render it on the screen.

Abstractly both constructions are the same, but the way they perform the computations is dramatically different. Indeed the object-space algorithm is based on the idea that the immersive view in the quotient $\Gamma\backslash G$ is the same as the immersive view in G of the object:

$$\widetilde{O} := \cup_{g\in\Gamma} g(O_0).$$

In general Γ has infinitely many elements.

The first step of the algorithm is to approximate, given $d \geq 0$, the set $\tilde{O}$ by the set:

$$O_d := O_0 \cup \bigcup_{j\leq d,\ (a_i)_i\in\{1,\dots N\}^j} p \circ g_{a_1} \circ \cdots \circ g_{a_j}(O_0).$$

Observe that when d is large, the set O_d is "close" to $\tilde{O}$.

The second step is to project the object O_d to the Euclidean space. This step is trivial when we deal with $G = \mathbb{E}^3$. When $G = \mathbb{H}$ it is the projection $p := (x, y, z, w) \mapsto (x/w, y/w, z/w) \in \mathbb{E}^3$.

The last step is to project the object $p(\check{O}_d)$ to the screen space. This is then done by a perspective projection.

The operation of duplication, and of rendering are projective transformations, and done by 4×4-matrix. Those operations are done very efficiently by the graphics processing unit (GPU) using OpenGL. Therefore this algorithm is not only the one used by Geomview, but also the one behind the software "Curved space" which produces real-time immersive visualization.

However, the latter works in real time, for hyperbolic mirrored dodecahedron, with level equal to 4. The reason is the following. For the simplest hyperbolic manifold, N is equal to 12. Therefore, the cardinality of $\{1,\dots N\}^d$, with $d = 4$, is already 24349. It is already a large number of objects. Remark that some elements of $g_{a_1} \times \cdots \times g_{a_j}$ can be equal to the identity even if these elements are not the identity. Therefore one can simplify the list in order to remove the copies. But this does not change the performance significantly, although the level can be pushed to $d = 6$ for a simple object with the current high performance computers.

The main reason is that the isometry group of hyperbolic spaces has its cardinality which increases exponentially fast. This means that the cardinality of

$$\Gamma_d := \{g_{a_1} \times \cdots \times g_{a_j} :\ (a_i)_i \in \{1,\dots N\}^j,\ j \leq d\}$$

is bounded from below by an exponential function of d. For instance the cardinality of Γ_5 is 128762, the one of Γ_6 is 1276425 and the one of Γ_7 is 12257733.

As a consequence:

Claim 7.2. *Object-space Algorithms are exponential with respect to the level of depth d of the image.*

Moreover, the way on which the objects $(g(Ob)_{g\in\Gamma_d}$ is distributed tend to be equi-distributed on the horizon, when d goes to infinity. However for low values, such as $d = 6$, the objects $(g(Ob)_{g\in\Gamma_d}$ are not equi-distributed : the image presents many "holes" although the missing objects are bigger than a pixel (see figure 1).

A fundamental problem of object-space algorithms, is that they compute many objects which do not need to be computed, since they are occluded by visible objects.

8 Conclusions and Future Work

In this paper we introduced a new image space algorithm for immersive visualization of flat and hyperblic 3-manifolds and orbifolds. The algorithm is based on ray tracing and can efficiently render inside views of flat and hyperbolic polyhedral complexes.

Our work improves the state-of-the-art in two main aspects. First, we can generate views of scenes several orders of magnitude more complex that are possible with previous methods. Second, we can handle a more general type of spaces that could not be visualized up to now.

The quality of the image is also improved by the performing stochastic anti-aliasing, i.e. we send randomly the ray associated to each pixel and then we do the mean of the different colors obtained.

Ongoing and future work, goes in many directions.

The above anti-aliasing is investigated to render radial and topological limit set of finitely generated Kleinian group. We are also currently extending the algorithm to other homogeneous polyhedral complexes, such as solvable and nilpotent manifolds.

References

[1] Nina Amenta, Stuart Levy, Tamara Munzner, and Mark Phillips. Geomview: a system for geometric visualization. In *Proceedings of the eleventh annual symposium on Computational geometry*, SCG '95, pages 412–413, New York, NY, USA, 1995. ACM.

[2] Michel Boileau, Sylvain Maillot, and Joan Porti. *Three-dimensional orbifolds and their geometric structures*, volume 15 of *Panoramas et Synthèses [Panoramas and Syntheses]*. Société Mathématique de France, Paris, 2003.

[3] Martin R. Bridson and André Haefliger. *Metric spaces of non-positive curvature*, volume 319 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1999.

[4] Benjamin Burton. Introducing regina, the 3-manifold topology software. *Experimental Mathematics*, 13(3):267–272, 2004.

[5] T. Damour, M. Henneaux, and H. Nicolai. Cosmological billiards. *Class.Quant.Grav.*, 20:R145–R200, 2003.

[6] George K. Francis, Camille M. A. Goudeseune, Henry J. Kaczmarski, Benjamin J. Schaeffer, and John M. Sullivan. Alice on the eightfold way: Exploring curved spaces in an enclosed virtual reality theatre. In *Visualization and Mathematics III*, pages 305–315, 2003.

[7] Charles Gunn and Delle Maxwell. Not knot, 1991. http://www.geom.uiuc.edu/video/NotKnot/.

[8] Charlie Gunn. Discrete groups and visualization of three-dimensional manifolds. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 255–262, New York, NY, USA, 1993. ACM.

[9] André Haefliger. Orbi-espaces. In *Sur les groupes hyperboliques d'après Mikhael Gromov (Bern, 1988)*, volume 83 of *Progr. Math.*, pages 203–213. Birkhäuser Boston, Boston, MA, 1990.

[10] Andrew J. Hanson, Tamara Munzner, and George Francis. Interactive methods for visualizable geometry. *Computer*, 27(7):73–83, July 1994.

[11] J. F. P. Hudson. *Piecewise linear topology.* University of Chicago Lecture Notes prepared with the assistance of J. L. Shaneson and J. Lees. W. A. Benjamin, Inc., New York-Amsterdam, 1969.

[12] Randy Hudson, Charlie Gunn, George K. Francis, Daniel J. Sandin, and Thomas A. DeFanti. Mathenautics: using vr to visit 3-d manifolds. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, I3D '95, pages 167–170, New York, NY, USA, 1995. ACM.

[13] Michael J. Laszlo. Techniques for visualizing 3-dimensional manifolds. In *Proceedings of the 1st conference on Visualization '90*, VIS '90, pages 342–352, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[14] Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *Proceedings of the first symposium on Virtual reality modeling language*, VRML '95, pages 33–38, New York, NY, USA, 1995. ACM.

[15] University of Minnesota. The geometry center, 1994. http://www.geom.uiuc.edu/.

[16] Mark Phillips and Charlie Gunn. Visualizing hyperbolic space: unusual uses of 4x4 matrices. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, I3D '92, pages 209–214, New York, NY, USA, 1992. ACM.

[17] W.P. Thurston. *The geometry and topology of three-manifolds.* Princeton University, 1979.

[18] Jeffrey Weeks. The shape of space, 1995. http://www.geom.uiuc.edu/video/sos/.

[19] Jeffrey Weeks. Snappea, 1999. http://www.geometrygames.org/SnapPea/.

[20] Jeffrey Weeks. Real-time rendering in curved spaces. *IEEE Comput. Graph. Appl.*, 22(6):90–99, November 2002.

[21] Jeffrey Weeks. Real-time animation in hyperbolic, spherical, and product geometries. In András Prékopa and Emil Molnár, editors, *Non-Euclidean Geometries*, volume 581 of *Mathematics and Its Applications*, pages 287–305. Springer US, 2006.

[22] J.R. Weeks. *The shape of space*. Pure and Applied Mathematics. Marcel Dekker, 2002.

[23] Steffen Weismann, Charles Gunn, Peter Brinkmann, Tim Hoffmann, and Ulrich Pinkall. jreality: a java library for real-time interactive 3d graphics and audio. In *ACM Multimedia'09*, pages 927–928, 2009.

[24] Wikipedia. 4k resolution standard, 2010. "http://en.wikipedia.org/wiki/4K_resolution".