Modeling and Rendering Non-Euclidean Spaces approximated with Concatenated Polytopes

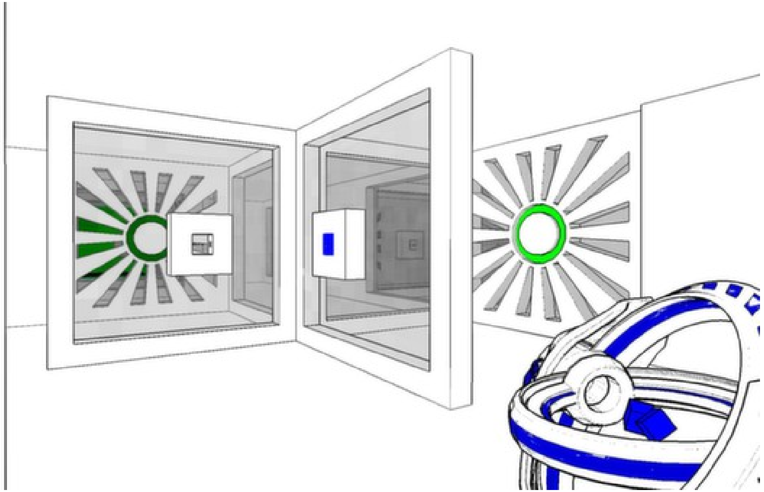
SIGGRAPH 2022

张弛 2022.11

Introduction

... the general methods for rendering m-manifolds embedded in n-D Euclidean spaces.

portal 2



antichamber

Concepts

... mathematical backgrounds and methodologies ...

manifold

A manifold is a topological space which looks locally like a Cartesian space, commonly a finite-dimensional Cartesian space $\mathbb{R}^n$.[1]

polytope

The notion of polytope is the generalization of the notion of polygon to arbitrary dimensions.[2]

---

1. nLab/manifold
2. nLab/polytope

Navigation and rendering in 2D-manifolds

a loop involving ray marching

camera.pose.update()

// get a normal vector using cross operation

step += delta

// the marching step size is sufficiently small

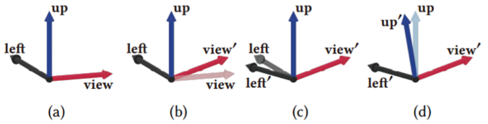
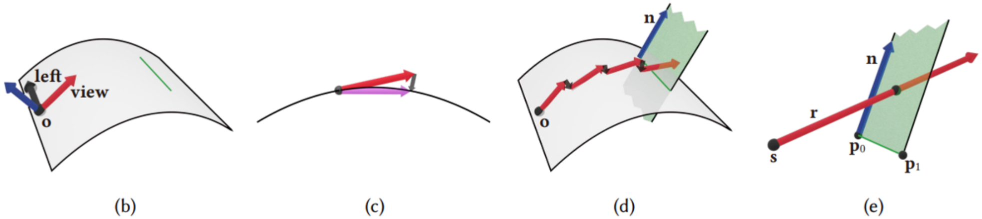


Fig. 3. Steps for updating the camera-space basis: (a) The original camera-space basis is composed of three orthonormal vectors, **left**, **up** and **view**. (b) Small fractions of **left** and **up** are added to **view** to define **view′**. (c) Then, **left** is changed to **left′**: $\mathbf{left}' = \mathit{normalize}(\mathbf{up} \times \mathbf{view}')$. (d) Finally, **up** is changed to **up′**: $\mathbf{up}' = \mathbf{view}' \times \mathbf{left}'$.

if ($\mathbf{p}_0 + \alpha(\mathbf{p}_1 - \mathbf{p}_0) + \beta\mathbf{n} = \mathbf{s} + \gamma\mathbf{r}$)

// If the ray of the ‘current’ marching step hits the infinite rectangle in the 3D Euclidean space, it is determined to hit the line segment in the 2-manifold.

render()



else

the ray is projected onto the manifold

cross operation

$$cross(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{n-1}) := \bigstar(\mathbf{v}_1 \wedge \mathbf{v}_2 \wedge \cdots \wedge \mathbf{v}_{n-1})$$

exterior product

$$\wedge : \mathbb{R}^{n-1} \times \cdots \times \mathbb{R}^{n-1} \to \mathbb{R}^{(n-1)\times(n-1)}$$

> Example
>
> in 3D space with orthonormal basis vectors, $(a\mathbf{e}_1 + b\mathbf{e}_2 + c\mathbf{e}_3) \wedge (d\mathbf{e}_1 + e\mathbf{e}_2 + f\mathbf{e}_3) = (ae - bf)\mathbf{e}_1\mathbf{e}_2 + (bf - ce)\mathbf{e}_2\mathbf{e}_3 + (cd - af)\mathbf{e}_3\mathbf{e}_1$ (a bi-vector)

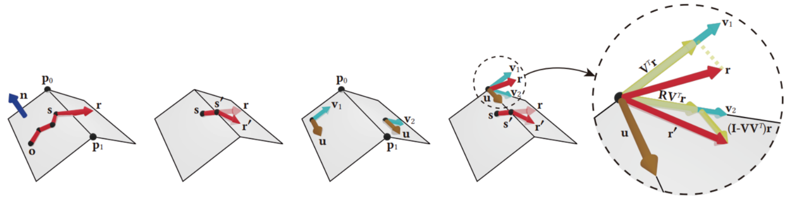
Hodge star operator

$$\bigstar : \mathbb{R}^{(n-m)\times(n-m)} \to \mathbb{R}^{(n-m)\times m}$$

> Example
>
> in 3D space with orthonormal basis vectors, $\bigstar((ae - bf)\mathbf{e}_1\mathbf{e}_2 + (bf - ce)\mathbf{e}_2\mathbf{e}_3 + (cd - af)\mathbf{e}_3\mathbf{e}_1) = ((ae - bf)\mathbf{e}_3 + (bf - ce)\mathbf{e}_1 + (cd - af)\mathbf{e}_2)$

Navigation in Polytopes

In order to reduce the computational cost, we approximate an m-manifold with a concatenation of m-D polytopes (henceforth, m-polytopes).



(a) (b) (c) (d)

The dotted circle of Fig. 6-(d) shows that $\mathbf{r}$ is projected onto $\mathbf{v}_1$ by $\mathbf{V}^T$. The projected, $\mathbf{V}^T\mathbf{r}$, is rotated by $\mathbf{R}$ presented below so that it is aligned with $\mathbf{v}_2$:

$$\mathbf{R} = \begin{pmatrix} \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} \\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} \end{pmatrix} \tag{5}$$

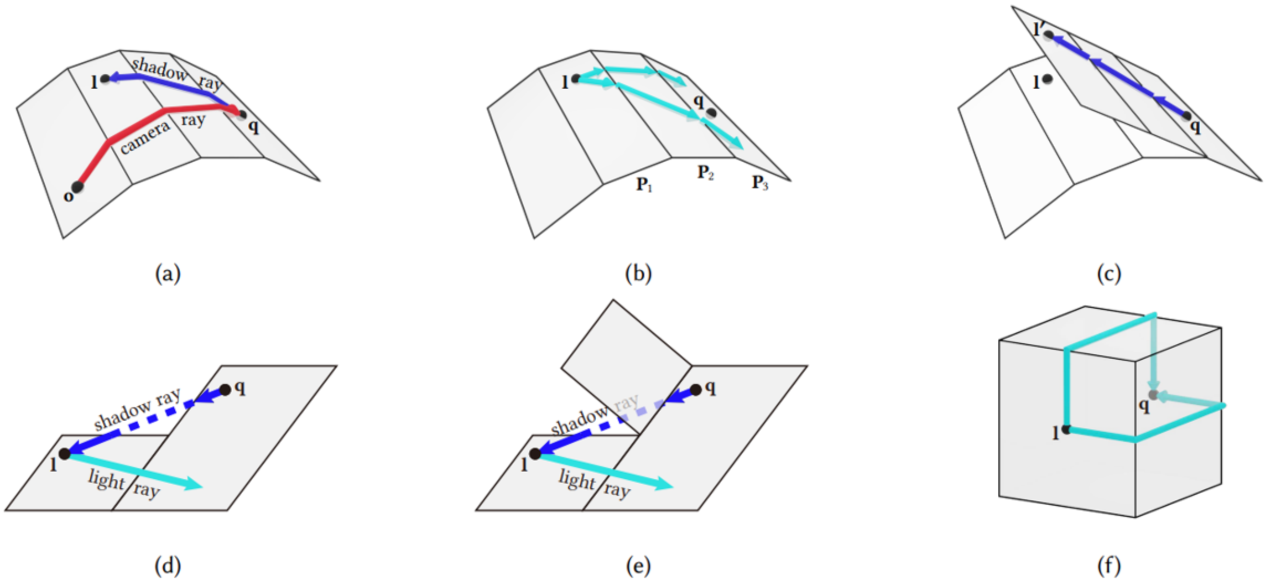
Finally, the rotated, $\mathbf{RV}^T\mathbf{r}$, is given the "3D representation" by $\mathbf{V}$, i.e., $\mathbf{VRV}^T\mathbf{r}$ represents the 3D vector aligned with $\mathbf{v}_2$. Let us call it $\mathbf{r}'_0$.

On the other hand, note that $\mathbf{VV}^T\mathbf{r}$ is the "3D representation" of $\mathbf{r}$'s projection onto $\mathbf{v}_1$. Then, the vector connecting it and the original $\mathbf{r}$ is represented as $(\mathbf{I}-\mathbf{VV}^T)\mathbf{r}$, where $\mathbf{I}$ is the identity matrix. Adding it to $\mathbf{r}'_0$, we obtain $\mathbf{r}'$:

$$\mathbf{r}' = \mathbf{r}'_0 + (\mathbf{I} - \mathbf{VV}^T)\mathbf{r} = \mathbf{VRV}^T\mathbf{r} + (\mathbf{I} - \mathbf{VV}^T)\mathbf{r}$$

Rendering in Polytopes

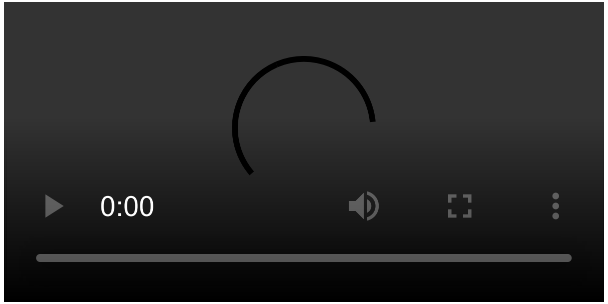
In order to reduce the computational cost, we approximate an m-manifold with a concatenation of m-D polytopes (henceforth, m-polytopes).



multiple light-ray paths from a single light source to a single point

Scene modeling: polytope concatenation

minimize the lengths between mating facets (portals)



Spring forces and collision resolution

start: while (!timeout) { init_simulator(dimension); sf(); cr(); }

if(good) return;

// Alternating between spring forces and collision resolution is repeated until either the polytopes are concatenated with no interpenetration or the predefined count of iterations is reached.

Adding joint polytopes

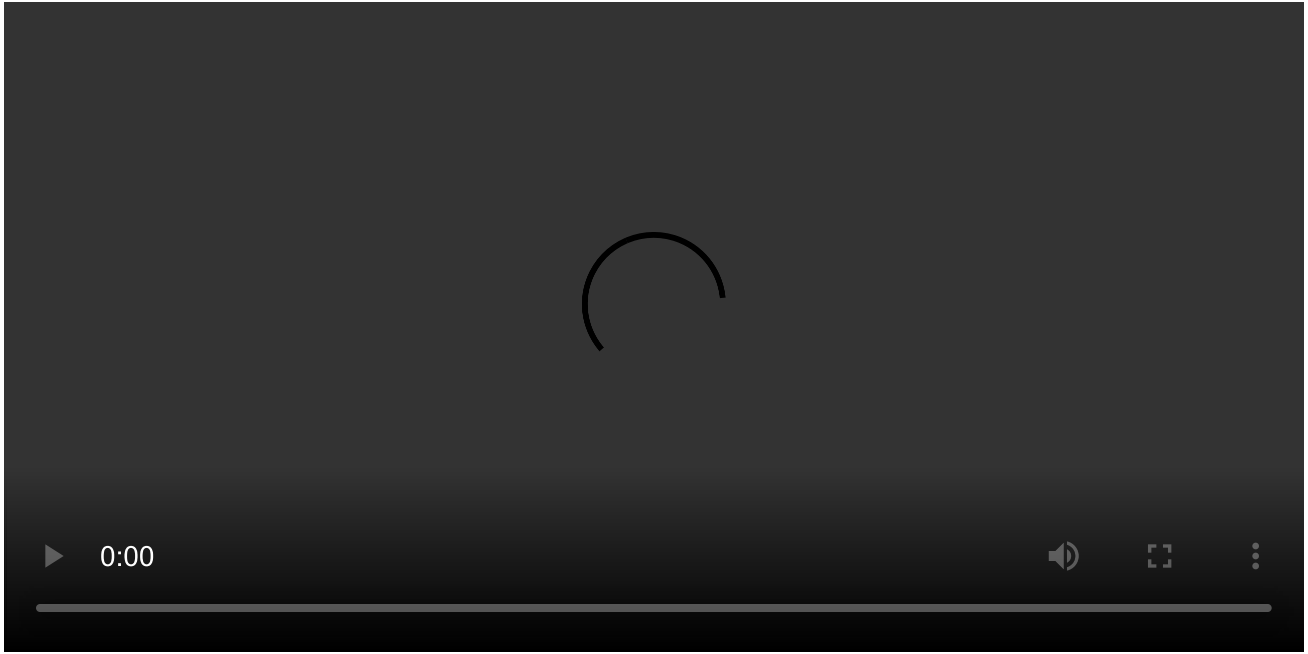
try { addjoint(); }

// The corresponding vertices of two mating facets are connectedafter exceeding the predefined iteration count.

Embedding in higher-dimensional spaces

catch(e) { dimension++; goto start; }

Effects



ray tracing effect

NEXT TIME …

Generalized Resampled Importance Sampling: Foundations of ReSTIR