A 3d curvilinear skeletonization algorithm with application to bidirectional path tracing

John Chaussard, Venceslas Biri, and Michel Couprie

Université Paris Est, LABINFO-IGM, A2SI-ESIEE
2, boulevard Blaise Pascal, Cité DESCARTES
BP 99 93162 Noisy le Grand CEDEX, France
chaussaj@esiee.fr, bertrang@esiee.fr, coupriem@esiee.fr

Abstract. What we propose is really great...

1 Introduction

Blah blah blah...

2 The cubical complex framework

2.1 Basic definitions

In the 3d voxel framework, objects are mae of voxels. In the 3d cubical complex framework, objects made of voxel, but of cubes, squares, lines and vertices. Let $\mathbb{Z}$ be the set of integers, we consider the family of sets $\mathbb{F}_0^1$ and $\mathbb{F}_1^1$, such that $\mathbb{F}_0^1 = \{\{a\} \mid a \in \mathbb{Z}\}$ and $\mathbb{F}_1^1 = \{\{a, a+1\} \mid a \in \mathbb{Z}\}$. Any subset f of $\mathbb{Z}^n$ such that f is the cartesian product of m elements of $\mathbb{F}_1^1$ and $(n-m)$ elements of $\mathbb{F}_0^1$ is called a face or an *m-face* of $\mathbb{Z}^n$, m is the dimension of f, we write $dim(f) = m$. A 0-face is called a *vertex*, a 1-face is an *edge*, a 2-face is a *square*, and a 3-face is a *cube*.

We denote by $\mathbb{F}^n$ the set composed of all faces in $\mathbb{Z}^n$. Given $m \in \{0, \dots, n\}$, we denote by $\mathbb{F}_m^n$ the set composed of all m-faces in $\mathbb{Z}^n$.

Let $f \in \mathbb{F}^n$. We set $\hat{f} = \{g \in \mathbb{F}^n | g \subseteq f\}$, and $\hat{f}^* = \hat{f} \setminus \{f\}$. Any element of $\hat{f}$ is *a face of* f, and any element of $\hat{f}^*$ is *a proper face of* f. We call *star of* f the set $\check{f} = \{g \in \mathbb{F}^n | f \subseteq g\}$, and we write $\check{f}^* = \check{f} \setminus \{f\}$: any element of $\check{f}$ is a *coface of* f. It is plain that $g \in \hat{f}$ iff $f \in \check{g}$.

A set X of faces in $\mathbb{F}^n$ is a *cell*, or *m-cell*, if there exists an m-face $f \in X$ such that $X = \hat{f}$. The *closure* of a set of faces X is the set $X^- = \cup\{\hat{f} | f \in X\}$. The set $\overline{X}$ is $\mathbb{F}^n \setminus X$.

Definition 1. *A finite set X of faces in $\mathbb{F}^n$ is a* cubical complex *if $X = X^-$, and we write $X \preceq \mathbb{F}^n$.*

Any subset Y of X which is also a complex is a subcomplex of X*, and we write $Y \preceq X$.*

Informally, in 3d, a set of faces X is each side (square) of a cube belonging to X also belong to X, each edge of a square belonging to X also belong to X, and each vertex of an edge belonging to X also belong to X.

A face $f \in X$ is *a facet of* X if f is not a proper face of any face of X. We denote by X^+ the set composed of all facets of X . A complex X is *pure* if all its facets have the same dimension. The *dimension of* X is $\dim(X) = \max\{\dim(f) \mid f \in X\}$. If $\dim(X) = d$, then we say that X is a d-complex. In $\mathbb{F}^n$, a complex X is *thin* if $\dim(X) < n$; in $\mathbb{F}^3$, a complex is thin if it contains no cube.

2.2 From binary images to cubical complex

Traditionally, a voxel image is defined as a finite subset of $\mathbb{Z}^n$. This kind of image is the most common one in the field of image processing and we give now a simple way to transpose a binary image to the cubical complex framework.

Informally, to do so, we associate to each element of $S \subseteq \mathbb{Z}^n$ an n-face of $\mathbb{F}^n$ (to a pixel we associate a square, to a voxel we associate a cube). More precisely, let $x = (x_1, ..., x_n) \in S$, we define the n-face $\Phi(x) = \{x_1, x_1+1\} \times \ldots \times \{x_n, x_n+1\}$. We can extend the map Φ to sets: $\Phi(S) = \{\Phi(x) | x \in S\}$. Given a binary image S, we associate to it the cubical complex $\Phi(S)^-$.

2.3 Thinning: the collapse operation

The collapse operation is the basic operation for performing homotopic thinning of a complex. It consists of removing two distinct elements (f, g) from a complex X under the condition that g is contained in f and is not contained in any other element of X. This operation may be repeated several times.

Definition 2. *Let $X \preceq \mathbb{F}^n$, and let f, g be two faces of X. The face g is* free for X*, and the pair (f, g) is* a free pair for X *if f is the only face of X such that g is a proper face of f.*

In other terms, (f, g) is a free pair for X whenever $\check{g}^* \cap X = \{f\}$. It can be easily seen that if (f, g) is a free pair for a complex X and $\dim(f) = m$, then f is a facet and $\dim(g) = m - 1$.

Definition 3. *Let $X \preceq \mathbb{F}^n$, and let (f, g) be a free pair for X. The complex $X \setminus \{f, g\}$ is an* elementary collapse of X.

Let $Y \preceq \mathbb{F}^n$, the complex X collapses onto Y *if there exists a sequence of complexes $(X_0, ..., X_\ell)$ of $\mathbb{F}^n$ such that $X = X_0$, $Y = X_\ell$ and for all $i \in \{1, \ldots, \ell\}$, X_i is an elementary collapse of X_{i-1}. We also say, in this case, that Y* is a collapse of X.

If Y is a collapse of a complex X, then Y is a complex. We now introduce some elements that will serve for proving the thinness of our skeletons. Let f_0, f_ℓ be two n-faces of $\mathbb{F}^n$ (with ℓ being even). An $(n-1)$*-path from f_0 to f_ℓ* is a sequence $\pi = (f_0, ..., f_\ell)$ of faces of $\mathbb{F}^n$ such that for all $i \in \{0, \ldots, \ell\}$, either i is even and

f_i is an n-face, or i is odd and f_i is an $(n-1)$-face with $\check{f_i}^* = \{f_{i-1}, f_{i+1}\}$ (such path always exists).

Proposition 4. *Let $X \preceq \mathbb{F}^n$ be an n-complex, with $n > 0$. Then X has at least one free $(n-1)$-face.*

Proof. Since X is an n-complex (hence X is finite) there exists an n-face a in X and an n-face b in $\overline{X}$. Obviously, there exists an $(n-1)$-path from a to b. Let h be the first n-face of π that is not in X, let k be the last n-face of π before h (thus k is in X), and let $e = k \cap h$ be the $(n-1)$-face of π between k and h. Since k and h are the only two n-faces of $\mathbb{F}^n$ that contain e, we see that the pair (k, e) is free for X. □

In conclusion, in $\mathbb{F}^3$, as long as a complex still contains 3-faces, it has a free 2-face and more collapse operations can be performed. Therefore, it is possible to perform collapse on a complex until no more 3-faces (volumes) can be found (until it is thin).

3 A parallel directional thinning based on cubical complex

3.1 Removing free pairs in parallel

In the cubical complex framework, parallel removal of simple pairs can be easily achieved when following simple rules that we give now. First, we need to define the *direction* and the *orientation* of a free face.

Let $f \in \mathbb{F}^n$, the *center of f* is the center of mass of the points in f, that is, $c_f = \frac{1}{|f|} \sum_{a \in f} a$. The center of f is an element of $[\frac{\mathbb{Z}}{2}]^n$, where $\frac{\mathbb{Z}}{2}$ denotes the set of half integers. Let $X \preceq \mathbb{F}^n$, let (f, g) be a free pair for X, and let c_f and c_g be the respective centers of the faces f and g. We denote by $V(f, g)$ the vector $(c_f - c_g)$ of $[\frac{\mathbb{Z}}{2}]^n$.

We define a surjective function $Dir() : \mathbb{F}^n \times \mathbb{F}^n \to \{0, \ldots, n-1\}$ such that, for all free pairs (f, g) and (i, j) for X, $Dir(f, g) = Dir(i, j)$ if and only if $V(f, g)$ and $V(i, j)$ are collinear (we don't bother defining $Dir()$ for non free pairs as it won't be useful in this case). The number $Dir(f, g)$ is called the *direction* of the free pair (f, g). Let (f, g) be a free pair, the vector $V(f, g)$ has only one non-null coordinate: the pair (f, g) has a *positive orientation*, and we write $Orient(f, g) = 1$, if the non-null coordinate of $V(f, g)$ is positive; otherwise (f, g) has a *negative orientation*, and we write $Orient(f, g) = 0$.

Now, we give a property of collapse which brings a necessary and sufficient condition for removing two free pairs of faces in parallel from a complex, while preserving topology.

Proposition 5. *Let $X \preceq \mathbb{F}^n$, and let (f, g) and (k, ℓ) be two distinct free pairs for X. The complex X collapses onto $X \setminus \{f, g, k, \ell\}$ if and only if $f \neq k$.*

Proof. If $f = k$, then it is plain that (k, ℓ) is not a free pair for $Y = X \setminus \{f, g\}$ as $k = f \notin Y$. Also, (f, g) is not free for $X \setminus \{k, \ell\}$. If $f \neq k$, then we have $g \neq \ell$, $\check{g}^* \cap X = \{f\}$ (g is free for X) and $\check{\ell}^* \cap X = \{k\}$ (ℓ is free for X). Thus, we have $\check{\ell}^* \cap Y = \{k\}$ as $\ell \neq g$ and $k \neq f$. Therefore, (k, ℓ) is a free pair for Y. □

From Prop. 5, the following corollary is immediate.

Corollary 6. *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X such that, for all $a, b \in \{1, \dots, m\}$ (with $a \neq b$), $f_a \neq f_b$. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

Considering two distinct free pairs (f, g) and (i, j) for $X \preceq \mathbb{F}^n$ such that $\mathit{Dir}(f, g) = \mathit{Dir}(i, j)$ and $\mathit{Orient}(f, g) = \mathit{Orient}(i, j)$, we have $f \neq i$. From this observation and Cor. 6, we deduce the following property.

Corollary 7. *Let $X \preceq \mathbb{F}^n$, and let $(f_1, g_1) \dots (f_m, g_m)$ be m distinct free pairs for X having all the same direction and the same orientation. The complex X collapses onto $X \setminus \{f_1, g_1 \dots f_m, g_m\}$.*

3.2 A directional parallel thinning algorithm

Intuitively, we want the algorithm to remove free faces of a complex "layer by layer": we don't want to have unequal thinning of the input complex. Therefore, we want each execution of the algorithm to remove free faces located on the border of the input complex. We say that a d-face of X is a *border face* if it contains a free $(d-1)$-face. We define $\mathit{Border}(X)$ as the set of all border faces of X. We now introduce a directional parallel thinning algorithm (Alg. 1).

On a single execution of the main loop of Alg. 1, only faces located on the border of the complex are removed (l. 15). Thanks to corollary 7, we also remove faces with same direction and orientation in parallel (l. **??**).

Different definitions of the orientation and direction can be given, corresponding to different order of removal of free faces in the complex and leading to different results. Algorithm 1 may be easily implemented to run in linear time complexity (proportionally to the number of faces of the complex). Indeed, checking if a face is free or not may be easily done in constant time. Moreover, when a free pair (f, g) is removed from the input complex, it is sufficient to scan the elements of $\hat{f}^* \cup \check{g}^*$ in order to find new free faces, as other faces' status won't change.

The following property is a direct consequence of of Prop. 4 and the fact that Alg. 1 stops when no more free faces can be found in the complex :

Proposition 8. *Let $X \preceq \mathbb{F}^n$, the result of $\mathit{ParDirCollapse}(X, \emptyset, +\infty)$ is a thin complex.*

Algorithm 1: $ParDirCollapse(X, W, \ell)$

Data: A cubical complex $X \preceq \mathbb{F}^n$, a subcomplex $W \preceq X$ which represents faces of X which should not be removed, and $\ell \in \mathbb{N}$, the number of layers of free faces which should be removed from X

Result: A cubical complex

```
while there exists free faces in X \ W and ℓ > 0 do
    L = Border(X)^-;
    for t = 0 → n − 1 do
        for s = 0 → 1 do
            for d = n → 1 do
                E = {(f, g) free for X | g ∉ W,
                     Dir(f, g) = t, Orient(f, g) = s, dim(f) = d};
                G = {(f, g) ∈ E | f ∈ L and g ∈ L};
                X = X \ G;
    l = l − 1;
return X;
```

4 Aspect preservation during thinning: a parameter-free method

Algorithm 1 does not necessarily preserves "geometrical information" from the original object in the resulting skeleton. For example, if the input was a filled human shape (one connected component, no tunnel nor cavity) and that $W = \emptyset$ and $l = +\inf$, then the result of Alg. 1 would be a simple vertex. However, when one wants to preserve "geometrical information" from the original shape in the skeleton, then one expects to obtain a "stickman" from a human shape. However, it is important to not retain "too much information" during thinning in order to avoid noisy branches in the skeleton.

In the following, we will see a new method in the cubical complex, requiring no user input, for obtaining a curvilinear skeleton yielding satisfactory visual properties. This will be achieved by adding, in the set W of Alg. 1, some faces from the original object during the thinning process.

4.1 The lifespan of a face

In the following, we define more notions in the cubical complex. The first one we present is the *death date* of a face.

Definition 9. *Let $f \in X \preceq \mathbb{F}^n$, the* death date of f in X*, denoted by $Death_X(f)$, is the smallest integer d such that $f \notin ParDirCollapse(X, \emptyset, d)$.*

The death date of a face indicates how many layers of free faces should be removed from a complex X, using Alg. 1, before removing completely the face from X. We now define the *birth date* of a face:

Definition 10. *Let $f \in X \preceq \mathbb{F}^n$, the* birth date of f in X*, denoted by $Birth_X(f)$, is the smallest integer b such that either f is a facet of $ParDirCollapse(X, \emptyset, b)$, or either $f \notin ParDirCollapse(X, \emptyset, b)$.*

The birth date indicates how many layers of free faces must be removed from X with Alg.1 before transforming f into a facet of X (we consider a face "lives" when it is a facet). Finally, we can define the *lifespan* of a face :

Definition 11. *Let $f \in X \preceq \mathbb{F}^n$, the lifespan of f in X is the integer*
$$Lifespan_X(f) = \begin{cases} +\infty & \text{if } Death_X(f) = +\infty \\ Death_X(f) - Birth_X(f) & \text{otherwise} \end{cases}$$

The three parameters previously defined are dependant on the order of direction and orientation chosen for algorithm $ParDirCollapse$.

The lifespan of a face f of X indicates how many "rounds" this face "survives" as a facet in X, when removing free faces with Alg. 1.

The lifespan is a good indicator of how important a face can be in an object. Typically, higher the lifespan is, and more representative of an object's visual feature the face is. The lifespan, sometimes called *saliency*, was used in [**?**] (with the name "medial persistence") in order to propose a thinning algorithm in cubical complexes based on two parameters.

4.2 Distance map and opening function

In addition to the lifespan of a face, the proposed homotopic thinning method uses information on distance between faces in order to decide if a face should be kept safe from deletion. We define hereafter the various notions needed for this, based on distance in the voxel framework.

Two points $x, y \in \mathbb{Z}^n$ are 1-*neighbours* if the Euclidean distance between x and y is equal or inferior to 1 (also called direct neighbours). A 1-*path from x to y* is a sequence $\mathcal{C} = (z_0, ..., z_k)$ of points of $\mathbb{Z}^n$ such that $z_0 = x$, $z_k = y$, and for all $j \in [1; k]$, z_j and z_{j-1} are 1-neighbours. The length of $\mathcal{C}$ is k. Remark that d_1 is indeed the 6-distance in the 3d voxel framework.

We set $d_1(x, y)$ as the length of the shortest 1-path from x to y. Let $S \subset \mathbb{Z}^n$, we set $d_1(x, S) = \min_{y \in S} d_1(x, y)$. The 1-*ball of radius r centered on x* is the set $\mathbb{B}^1_r(x) = \{y \in \mathbb{Z}^n | d_1(x, y) < r\}$. Given $X \subset \mathbb{Z}^n$, the *maximal* 1-*ball of X centered on x* is the set $\mathbb{MB}^1_X(x) = \mathbb{B}^1_{d_1(x, \overline{X})}(x)$.

We set, for all $x \in X$, $\omega_1(x, \overline{X}) = \max_{x \in \mathbb{MB}^1_X(y)} d_1(y, \overline{X})$: this value indicates the radius of the largest maximal 1-ball contained in X and containing x. If $x \in \overline{X}$, we set $\omega_1(x, \overline{X}) = 0$. The map ω_1 is known as the opening function (based on the 1-distance): it allows to compute efficiently results of morphological openings by balls of various radius, and gives information on the local thickness of an object.

Given $X \subset \mathbb{Z}^n$, the value of $\omega_1(x, \overline{X})$ of every $x \in X$ can be naively computed by performing successive morphological dilations of values of the map d_1.

In order to extend d_1 and ω_1 to the cubical complex framework, let us introduce the map Φ^{-1}, inverse of the bijective map $\Phi : \mathbb{F}^n_n \to \mathbb{Z}^n$ defined in Sec. 2.2. It is used to project any n-face of $\mathbb{F}^n$ into $\mathbb{Z}^n$. We indifferently use Φ^{-1} as a map from $\mathbb{F}^n_n$ to $\mathbb{Z}^n$, and as a map from $\mathcal{P}(\mathbb{F}^n_n)$ to $\mathcal{P}(\mathbb{Z}^n)$.

Given $Y \subset \mathbb{F}^n$, we define the map $\tilde{D}_1(X) : \mathbb{F}^n \to \mathbb{N}$ as an extension of d_1 to the cubical complex framework: for all $f \in \mathbb{F}^n$,

$$\tilde{D}_1(Y)(f) = \begin{cases} d_1(\Phi^{-1}(f), \Phi^{-1}(Y \cap \mathbb{F}^n_n)) \text{ if } f \text{ is an } n\text{-face} \\ \max\limits_{g \in \tilde{f}^* \cap \mathbb{F}^n_n} \tilde{D}_1(Y)(g) \qquad \text{else} \end{cases}$$

Informally, if f is a 3-face, then $\tilde{D}_1(Y)(f)$ is the length of the shortest 1-path between the voxel "corresponding" to f and the set of voxels corresponding to Y. The same way, we define $\tilde{\Omega}_1(Y) : \mathbb{F}^n \to \mathbb{N}$ as an extension of ω_1 to the cubical complex framework.

Finally, we define the decenterness map:

Definition 12. *The* decenterness map*, denoted by Decenter*(Y)*, is equal to* $\tilde{\Omega}_1(Y) - \tilde{D}_1(Y)$.

4.3 Parameter-free thinning based on the lifespan, opening function and decenterness

As previously said, we will add edges to the set W of Alg. 1 in order to retain, in the resulting curvilinear skeleton, important edges from the original object. If an edge has a high decenterness value, then it is probably located too close to the object's border and does not represent an interesting visual feature to preserve. On the other hand, if an edge has a high lifespan, then it means it was not removed quickly, after becoming a cell, by the thinning algorithm and might represent some precious visual information on the original object. An idea would be to keep, during thinning, all edges whose lifespan is superior to the decenterness value. Unfortunately, this strategy produces skeletons with many spurious branches in surfacic areas of the original object.

We can identify surfacic areas of a complex as zones where squares have a high lifespan. Therefore, in order to avoid spurious branches in surfacic areas, we need to make it harder for edges to appear in these zones. It can be achieved by deciding that an edge will be kept safe from deletion by the thinning algorithm if its lifespan is superior to the decenterness value plus the lifespan of squares "around" this edge. This leads us to

5 Application to path tracing

5.1 The path tracing algorithm

Path tracing is a global illumination algorithm that is able to render photo-realistic images of a scene viewed by a camera. The idea behind the algorithm is very simple: for each pixel of the image, throw rays of light through the pixel that will bounce on the scene to accumulate lighting exchanges. It is based on physical laws of radiometry.

Algorithm 2: $CurvilinearSkeleton(X)$

Data: A cubical complex $X \preceq \mathbb{F}^3$
Result: A cubical complex $Y \preceq \mathbb{F}^3$ such that $\dim Y \leq 2$

```
1 W = LOC_1(X);
2 ParDirCollapse(X, W, +∞);
3 return X;
```

Basic definitions of radiometry Let S be a set of surfaces of $\mathbb{R}^3$ that don't overlap ($\forall x \in \mathbb{R}^3$ there exists at most one surface of S that contains x) and $S_L \subset S$ a set of surface lights. S is called the scene.

The basic radiometric quantity is the *radiance* $L(x \to \Theta)$ that express the quantity of light energy produce by the surface point $x \in \mathbb{R}^3$ towards direction $\Theta \in \Omega_x$ per unit of time per unit area per unit solid angle (expressed in $W.m^{-2}.sr^{-1}$). This is the quantity that the eye actually "sees" and that must be computed.

We can define another quantity related to radiance called *incoming radiance* $L(x \leftarrow \Phi)$ that represents the radiance that reach x from direction Φ.

Property 13. The radiance remains constant along straight lines in the void.

That property allows us to write the incoming radiance in terms of radiance.

$$L(x \leftarrow \Phi) = L(r(x, \Phi) \to -\Phi)$$

$r(x, \Phi)$ is the surface point seen by x in the direction Φ. It can be defined formally by:

$$\begin{aligned} r(x, \Phi) &= x + t_{min}\Phi \\ t_{min} &= min\{t \in (0, +\infty] \mid x + t\Phi \in S\} \end{aligned}$$

If $t_{min} = \infty$ then $r(x, \Phi)$ is undefined and $L(x \leftarrow \Phi) = 0$. We call r the *raytrace* function. In practice it is implemented by throwing a ray in the scene and looking for the first intersection.

A third quantity related to radiance is *emitted radiance* $L_e(x \to \Theta)$. It is not null if x is a point of a light source ($x \in S_L$). In practice the emitted radiance is given with the set S_L as an input of the algorithm (in can be constant across each of the lights or given by a texture).

The radiance is the solution of an equation called the Rendering Equation:

$$\begin{aligned} L(x \to \Theta) &= L_e(x \to \Theta) + \int_{\Phi \in \Omega_x} f_r(x, \Theta \leftrightarrow \Phi) L(x \leftarrow \Phi) cos(N_x, \Phi) d\omega_\Phi \\ &= L_e(x \to \Theta) + L_r(x \to \Theta) \end{aligned}$$

This equation express an intuitive idea: the radiance produced by the point x in the direction Θ is the sum of the emitted radiance and the reflected radiance.

The reflected radiance is computed by taking all the possible directions of the hemisphere around x and integrate the incoming radiance along those directions scaled by a factor $f_r(x, \Theta \leftrightarrow \Phi)cos(N_x, \Phi)$.

The cosine is used to attenuate the radiance that arrive by a "direction rasante". N_x is the surface's normal at point x.

The f_r function is called the *bidirectional reflectance function* (BRDF) and express the exchange of radiance between the incoming direction Φ and the output direction Θ. Intuitively it represents the material properties of the surface at point x. A mirror doesn't reflect light the same manners a wall do, the BRDF encodes that. The BRDF has to respect three properties to be physically coherent:

- Helmotz reciprocity: $f_r(x, \Theta \leftrightarrow \Phi) = f_r(x, \Phi \leftrightarrow \Theta)$
- Positivity: $f_r(x, \Theta \leftrightarrow \Phi) \geq 0$
- Conservation of energy: $\int_{\Omega_x} f_r(x, \Theta \leftrightarrow \Phi)cos(N_x, \Phi)d\omega_\Phi \leq 1$

It's actually the rendering equation that a path tracer try to resolve. Let O be the origin of the camera and I the image we want to compute (a rectangle embedded in $\mathbb{R}^3$). Then the path tracing algorithm compute an approximation of the radiances $L(O \leftarrow \boldsymbol{OP})$, $\forall P \in I$.

Solving the rendering equation The rendering equation is a recursive integral equation. The integrand of the equation contains the radiance function that we must compute:

$$\begin{aligned} L(x \to \Theta) &= L_e(x \to \Theta) + \int_{\Phi \in \Omega_x} f_r(x, \Theta \leftrightarrow \Phi) L(x \leftarrow \Phi) cos(N_x, \Phi) d\omega_\Phi \\ &= L_e(x \to \Theta) + \int_{\Phi \in \Omega_x} f_r(x, \Theta \leftrightarrow \Phi) L(r(x, \Phi) \to -\Phi) cos(N_x, \Phi) d\omega_\Phi \end{aligned}$$

In practive it is impossible to compute anatically the integral, it has to be estimated. The most commonly used method is the Monte-Carlo integration that provide the following estimator for the integral:

$$\langle L_r(x \to \Theta) \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{f_r(x, \Theta \leftrightarrow \Phi_i) L(r(x, \Phi_i) \to -\Phi_i) cos(N_x, \Phi_i)}{p(\Phi_i)}$$

p is a probability density function (pdf) that is used to sample N samples $(\Phi_i)_{i=1...N}$ of the hemisphere Ω_x.

This estimator is unbiased, it means that the esperance $E[\langle L_r(x \to \Theta) \rangle]$ is equal to $L_r(x \to \Theta)$.

The variance of the estimator express its quality. The lower the variance is, the better is the estimator. We have:

$$\sigma^2[\langle L_r(x \to \Theta)\rangle] = \frac{1}{N}\int_{\Omega_x}(\frac{f(\Phi)}{p(\Phi)} - L_r(x \to \Theta))^2 d\omega_\Phi$$

Then if we increase the number of samples we decrease the variance and the estimator gives better results. Another way to decrease the variance is to choose a good pdf. Actually the best is to choose a pdf that match the shape of the function to integrate (ie. give high density to samples that have high values for the function and low density to samples that have low values).

This str ategy of choosing a good pdf is called *importance sampling* and is over-used in global illumination to improve the convergence of the algorithms.

In the path tracing algorithm we cannot use more than one sample. Actually the evaluation of the integrand implies the evaluation of L and thus a new application of the Monte-Carlo estimator. So if we use more than one sample the complexity of the algorithm becomes exponential ($O(d^N)$ with d the length of a path and N the number of samples).

The algorithm We are now able to construct the algorithm for computing radiance:

Algorithm 3: Incoming radiance $L(x \leftarrow \Phi)$

Data: An incoming direction Φ, a point x
Result: $L(x \leftarrow \Phi)$
1 **return** $L(r(x, \Phi) \to -\Phi)$;

Algorithm 4: Radiance $L(x \to \Theta)$

Data: An outcoming direction Θ, a point x
Result: $L(x \to \Theta)$
1 $L \leftarrow L_e(x \to \Theta)$;
2 **if** *continue the recursion* **then**
3 $\quad \Phi \leftarrow sample(\Omega_x, p)$
4 $\quad L \leftarrow L + \frac{f_r(x,\Phi,\Theta)L(x \leftarrow \Phi)cos(N_x,\Phi)}{p(\Phi)}$;
5 **return** L;

This algorithm is called for each pixel of the final image to compute. To improve the quality of the result we can throw multiple rays accross each pixel and computing the mean of the results.

In the radiance algorithm the Φ direction is sampled on Ω_x based on the pdf p. We will discuss what pdf we use in the next section.

The stopping condition of the recursion is not specified here. In our implementation we simply fix a maximum depth for the path but it's a bad solution. Actually it biased the method because some light paths have no chance to be evaluated. A better solution is to use a method called russian roulette.

5.2 Skeleton based importance sampling

We have presented the path tracing algorithm without describe the pdf p used to sample the hemisphere around a point. The choice of the pdf is very important because the convergence speed of the algorithm depends on it. Here is the integrand of the integral estimated with Monte-Carlo:

$$f_r(x, \Phi, \Theta)L(x \leftarrow \Phi)cos(N_x, \Phi)$$

To improve the quality we have to choose a pdf that match the shape of this function. There is a cosine factor in the function. It takes higher values when Φ is close to N_x. That means we have to give better chances to be samples to directions that are close to the normal. But it's not enough: when the environment is not totally diffuse, the brdf take higher values for some pair of directions. A perfect mirror, for exemple, reflect light only in a single direction: the symetric of Θ by the normal N_x. In that case it's totally useless to sample Ω_x, we juste have to take the correct direction without taking into account the cosine.

The most common strategy used to sample Ω_x is to use the BRDF because we know it: it is given as an input of the algorithm for exemple with textures.

L is rarelly used because we cannot evaluate it in constant time. Remember that it represents the distribution of light in the scene. It takes high values when Φ carry a lot of energy. Out method is based an a topological skeleton of the void (because light travels in the void) to sample Ω_x according to L.

Computation of the skeleton The skeleton is computed in two steps:

- Voxelization of the void
- Skeletonization in the cubical complex framework

The voxelization is done with a software called Binvox. It voxelizes the surfaces of the scene and we invert it to get the voxelization of the void. The voxelization is then converted to a cubical complex and we apply our algorithm to skeletonize it. We get a curvilinear skeleton that has the same topology than the void.

The skeleton is represented by a graph of 3D positions with edges encoding the topology.

Construction of the importance points Given the skeleton we compute a set of points called importance points. This points will be used to sample Ω_x in the path tracing algorithm.

Let L be a light of the scene ($L \in S_L$) and n_L the shortest visible node by L. We first compute a tree of shortest paths along the skeleton with n_L as root. For that we use the Dijkstra algorithm with a distance based on visibility from n_L: We put $d(n, m) = 1$ if m is visible from n_L and $d(n, m) = 10$ if not. It results that the paths that are enlighted by L will have a shorter distance.

Let n be a node of the skeleton and V_n the set of visible nodes from n along the shortest path toward n_L. The importance point imp_n associated to n is computed by taking the barycenter of V_n.

Sampling according to L Given a point x and a direction Θ we want to compute $L(x \rightarrow \Theta)$ and then sample the hemisphere Ω_x. We search for the nearest skeleton node n and its importance point imp_n. We sample the hemisphere with a power-cosine pdf center on $\boldsymbol{ximp_n}$:

$$p_{skel}(\Phi) = \frac{s+1}{2\pi} cos^s \alpha$$

with α the angle between $\boldsymbol{ximp_n}$ and Φ and s a parameter called skeleton strength. The more s is high, the more we sample close $\boldsymbol{ximp_n}$.

Results

Problems A few problems arise from this sampling strategy:

- $\boldsymbol{ximp_n}$ may point backward the surface (ie. $cos(\boldsymbol{ximp_n}, N_x) < 0$)
- The importance point imp_n may be not visible from x

In those case we don't use the skeleton and switch to a basic BRDF sampling.

Another problem from the method is the apparition of speckles. Speckles are anormally white pixels on the computed image that tend to vanish when we augment the number of rays thrown by pixel. It comes from the Monte-Carlo estimator when we sample a direction Φ with a low pdf $p(\Phi)$ but a high value $f_r(x, \Theta \leftrightarrow \Phi)L(x \leftarrow \Phi)cos(N_x, \Phi)$: the ratio becomes very high and speckles appears. In the next section we discuss a method to attenuate the speckles.

5.3 Skeleton based multiple importance sampling

Our strategy is good, especially in dark areas. It reduces noise by taking into account the distribution of energy in the scene. The problem is that we take into account only one importance direction for sampling, we totally ignore the BRDF and the cosine factor (unless we fall in a bad case like the two describe above). We implemented a solution called multiple importance sampling (MIS) that is able to combine multiple sampling strategies into one.

Multiple importance sampling Let $f(x) = f_1(x)f_2(x)...f_n(x)$ a function that we want to integrate over a domain D:

$$F = \int_D f_1(x)f_2(x)...f_n(x)dx$$

Monte-Carlo suggests us to use the estimator (with one sample x in that case, like in path tracing):

$$\langle F \rangle = \frac{f_1(x)f_2(x)...f_n(x)}{p(x)}$$

But how to choose efficiently p ? Suppose we have n sampling strategy $p_1, p_2, ..., p_n$, with p_i efficient for the function f_i. Let $c_1, ..., c_n$ be n scalar from $[0, 1]$ such that $\sum_{i=1} nc_i = 1$. c_i is the probability to select the strategy p_i for sampling. Then the combined pdf of a sample x is:

$$p(x) = \sum_{i=1}^{n} c_i p_i(x)$$

We use this pdf for Monte-Carlo estimation. First we choose the strategy to use, based on the probabilities $(c_i)_{i=1...n}$, then we sample with p_i and then we estimate the integral by $\frac{f_1(x)f_2(x)...f_n(x)}{p(x)}$. This method is called multiple importance sampling.

MIS for skeleton based path tracing In our case we have the following function to integrate:

$$f_r(x, \Theta \leftrightarrow \Phi)L(x \leftarrow \Phi)cos(N_x, \Phi)$$

We suppose that we have a pdf p_{brdf} that efficiently sample Ω_x according to the product $f_r(x, \Theta \leftrightarrow \Phi)cos(N_x, \Phi)$ and our strategy p_{skel} that efficiently sample according to $L(x \leftarrow \Phi)$.

Let c_{skel} be the probability to sample with the strategy p_{skel} and $1 - c_{skel}$ the probability to sample with p_{brdf} (c_{skel} could be a parameter of the program). That gives us the following algorithm to sample:

Results

Algorithm 5: Multiple importance sampling

Data: An outcoming direction Θ, a point x
Result: A incoming direction Φ and its pdf $p(\Phi)$

$r \leftarrow$ a random number between 0 and 1;
if $r \leq c_{skel}$ **then**
    return $\Phi \leftarrow sample(\Omega_x, p_{skel})$
else
    return $\Phi \leftarrow sample(\Omega_x, p_{brdf})$
return $\Phi, p(\Phi) \leftarrow c_{skel}p_{skel}(\Phi) + (1 - c_{skel})p_{brdf}(\Phi)$