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 $\breve{g}^* \cap X = \{f\}$. 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 will 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 $Orient(f, g) = 0$. On Fig. **??**, the free pair (a, c) and the free pair (d, e) have different directions; the free pairs (a, b) and (d, e) have the same direction, but opposite orientations.

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 (see Fig. **??**c).

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

We say that a d-face of X is a *border face* if it contains a free $(d-1)$-face. Define $\mathit{CBorder}(X)$ as the set of all border faces of X. We are now ready to introduce a directional parallel thinning algorithm (Alg. 1).

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

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 = CBorder(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;
```

Intuitively, we want the algorithm to remove free faces "layer by layer": we don't want, after a single execution ($\ell = 1$), 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: this is why we introduce, on line 4 the set L, and that we remove only faces located in L on line 17. The sets E (line 12) and G (line 15) allows to remove whole sets of free faces in parallel from X, thanks to the direction and orientation of faces previously defined. A detailed view of each step of the algorithm is shown on Fig. **??**.

Different definitions of the orientation and direction can be given, corresponding to different order of removal of free faces in the complex. These changes lead to different results, but arbitrary choices on the order of removal of free pairs must be made in order to obtain, at the end, a thin skeleton (no more n-faces when working in the n-dimension). Once orientation and direction have been defined, the results of the algorithm are uniquely defined.

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 faces of f and the cofaces of g in order to find new free faces, as other faces' status won't change (the implemented algorithm contains these optimizations).

4 Aspect preservation during thinning: a parameter-free method

Algorithm 1 does not necessarily preserves "geometrical information" of 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 of the original object.

4.1 The lifespan of a face

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

Definition 8. *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 9. *Let $f \in X \preceq \mathbb{F}^n$, the* birth date of f in X, *denoted by $Birth_X(f)$, is the minimum between the smallest integer b such that f is a facet of $ParDirCollapse(X, \emptyset, b)$, and $Death_X(f)$.*

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 10. *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 algorithm 1. The lifespan, the death and the birth, as defined here, are dependant of Alg. 1, used for performing the thinning. It is of course possible to use another algorithm for performing the thinning, leading to other values of birth, death and lifespan. It is recommended, in order to have "comparable" values, to compute these three parameters with the same thinning technique.

Algorithm 2 computes the lifespan of all faces of a complex. The algorithm is not linear in time, however, a linear implementation of such algorithm exists and is presented in the appendix. On Fig. **??**, we show a sequence of collapse allowing to compute the lifespan of an edge.

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, also called *saliency*, was used in [**?**] (under 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 will use 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 DT 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.

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 $\mathbb{B}^1_r(x) = \{y \in \mathbb{Z}^n | d_1(x, y) < r\}$. Remark that d_1 is indeed the so-called 4-distance in the 2d DT framework, and the 6-distance in the 3d DT framework. Given $X \subset \mathbb{Z}^n$, the *maximal* 1-*ball of X centered on x* is the set $\hat{\mathbb{B}^1_X}(x) = \mathbb{B}^1_{d_1(x, \overline{X})}(x)$.

Algorithm 2: $Lifespan(X)$

```
Data: A cubical complex X ⪯ 𝔽^n
Result: The map giving the lifespan of all faces of X
for all f ∈ X do
    Death(f) = +∞;
    if f ∈ X^+ then
        Birth(f) = 0;
    else
        Birth(f) = +∞;
Y = X; l = 0;
while there exists free faces for Y do
    Y = ParDirCollapse(Y, ∅, 1);
    l = l + 1;
    for all f ∈ Y^+ such that Birth(f) = +∞ do
        Birth(f) = l;
    for all f ∉ Y such that Death(f) = +∞ and f ∈ X do
        Death(f) = l;
for all f ∈ X do
    Birth(f) = min(Birth(f), Death(f));
    if Death(f) = +∞ then
        Lifespan(f) = +∞;
    else
        Lifespan(f) = Death(f) − Birth(f);
return Lifespan;
```

We set, for all $x \in X$, $\omega_1(x, \overline{X}) = \max_{x \in \mathbb{B}^{\hat{1}}_X(y)} d_1(y, \overline{X})$: 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 openings by balls of various radius, and gives information on the local thickness of an object on each of its points. We show some examples of the opening function on Fig. **??** and **??**.

Given $X \preceq \mathbb{F}^n$, the value of $\omega_1(x, \overline{X})$ of every $x \in X$ can be computed by performing successive dilations of values of the map d_1. The algorithm presented in Alg.**??** is naive, and a more efficient implementation (linear in time depending on the image's size) is discussed in the appendix (see Sec. **??**, p. **??**).

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 Φ 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 $X \preceq \mathbb{F}^n$, we set $X^v = \Phi^{-1}(X \cap \mathbb{F}_n^n)$: the set X^v is a subset of $\mathbb{Z}^n$. 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 $f \in \mathbb{F}^n$,

$$\tilde{D_1}(X)(f) = \begin{cases} d_1(\Phi^{-1}(f), \overline{X^v}) \text{ if } f \text{ is an } n\text{-face} \\ \max\limits_{g \in \check{f}^*} \tilde{D_1}(X)(g) \quad \text{else} \end{cases}$$

The same way, we define $\tilde{\Omega_1}(X) : \mathbb{F}^n \to \mathbb{N}$ as an extension of ω_1 to the cubical complex framework.

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

Thanks to these notions, we can now define sets of faces that will help preserve the visual aspect of an object during thinning. First, let us define the decenterness of a complex as the map $Decenter(X) = \tilde{\Omega_1}(X) - \tilde{D_1}(X)$. For each face of a complex X, the decenterness value of this face gives an information on how well a face is centered inside a visual feature of the object: the lower this value is, the better centered the face is. On Fig. **??**, we give an example of the decenterness map of a shape, where the highest values are represented by the darkest colours.

Faces relevant of the visual aspect of a complex must have a high lifespan ("survive" long to the homotopic thinning process) and a low decenterness (is centered in the object). In a 3-dimensional complex (resp. a 2-dimensional complex), squares (resp. lines) whose lifespan is higher than the decenterness will be chosen as relevant of the surfacic (resp. curvilinear) parts of the complex.

Definition 11. *Given $X \preceq \mathbb{F}^n$, the* k-LC axis *(stands for "Lifespan Centerness") of X is the set*

$$\mathcal{LC}_k(X) = \{f \in X \mid \dim(f) = k \text{ and } Lifespan_X(f) > Decenter(X)(f)\}^-.$$

The LC-axis fails in selecting a good set of curves relevant of the curvilinear parts of a three-dimensional complex. Indeed, given a 3-complex X, the set $\mathcal{LC}_1(X)$ contains generally too many curves. This happens because the algorithm $ParDirCollapse$ "takes more time" to eliminate lines than to eliminate squares in a 3-complex. Consequently, lines of a 3-complex tend to have a high lifespan even though they are not representative of any curvilinear part of the complex. The thicker is the input object, and the more important is the phenomenon.

Lines relevant of the curvilinear parts of a 3-complex have a high lifespan and a low decenterness, especially in the thick parts of the complex. Lines whose lifespan is higher than the decenterness added to the local thickness of the complex will be relevant of the curvilinear parts of the complex.

Definition 12. *Given $X \preceq \mathbb{F}^n$, the* k-LOC axis *(stands for "Lifespan Opening Centerness") of X is the set*

$$\mathcal{LOC}_k(X) = \{f \in X \mid \dim(f) = k \text{ and } Lifespan_X(f) > \tilde{\Omega_1}(X)(f) + Decenter(X)(f)\}^-.$$

The various sets previously defined represent faces which should be kept safe from deletion during homotopic thinning of a complex in order to obtain a pruned skeleton containing visual features from the original object. The set $\mathcal{LC}_1$ of a 2-complex represents curvilinear parts to keep in this complex, the set $\mathcal{LOC}_1$ of a 3-complex represents curvilinear parts to keep in this complex, and the set $\mathcal{LC}_2$ of a 3-complex represents surfacic parts to keep in this complex.

Lemma 13. *Given $X \preceq \mathbb{F}^n$, for every $k < n$, $\mathcal{LC}_k(X) \subseteq CollapseFacet(X, \emptyset)$ and $\mathcal{LOC}_k(X) \subseteq CollapseFacet(X, \emptyset)$.*

Lemma 13 is straightforward once observed that $CollapseFacet(X, \emptyset)$ is indeed the set of k-faces of X ($k < n$) which have a strictly positive lifespan in X. Based on these sets, we propose three algorithms: one for computing a 1d skeleton from a bi-dimensional complex, one for computing a 2d skeleton (which can contain 1d parts) from a three-dimensional complex, and one for computing a curvilinear skeletons from a three-dimensional complex. This last algorithm should produce one-dimensional results (a set of lines) however, the output can be a two-dimensional complex.

Algorithm 3: $CurvilinearSkeleton(X)$

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 $W = \mathcal{LOC}_1(X)$;
2 $ParDirCollapse(X, W, +\infty)$;
3 **return** X;

Algorithm 3 produces as output a 2d or 1d complex, result of the homotopic thinning of a three-dimensional complex. The skeleton produced by this algorithm should be a set of lines describing the shape of the input, however, it may contain squares representative of the topology of the initial object (if the input complex contains a cavity, the skeleton will contain surfacic parts).

Despite the use of constraint sets (sets of faces which should not be removed during thinning) in the three algorithms, they produce thin results. -¿ On peut simplement modifier algo pour y incorporer directement les elements a conserver pdt la skel...

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.

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* $L(x \rightarrow \Theta)$ 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 14. The radiance remains constant along straight lines in the void.

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

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

$r(x, \Phi)$ 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* $L_e(x \rightarrow \Theta)$. 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:

$$L(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Phi \in \Omega_x} f_r(x, \Theta \leftrightarrow \Phi) L(x \leftarrow \Phi) cos(N_x, \Phi) d\omega_\Phi \quad (1)$$

It's actually this equation that a path tracer try to resolve. Let O be the origine 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 Parler rapidement de l'chantillonage et donner l'estimateur de monte carlo

The algorithm Donner l'algorithme en pseudo-code

5.2 Skeleton based importance sampling

Dcrire la stratgie d'chantillonage base sur le squelette

5.3 Multiple importance sampling

Dcrire comment combiner les stratgies brdf + skeleton

5.4 Coarse irradiance estimation using curvilinear skeleton

Dcrire les heuristiques