Master Thesis

Analysis of Light Transport

Author:
Baptiste Angles

Supervisor:
Dr. Mathias Paulin

September 9, 2017

Abstract

The photorealistic rendering algorithms such as *path tracing* sample the light path space to evaluate the light transport. The distribution of these samples is crucial to obtain a good convergence rate. However the global structure of the path space is not well-known and the rendering algorithms only use local informations to determine this distribution.

In litterature methods exploiting the properties of some paths to sample them efficiently exist. However, these are limited to particular classes of paths and are difficult to generalize. In this thesis, we explore the possibility of decomposing the path space in visually coherent sub-spaces. We make use of unsupervised learning and more specifically *clustering* methods, to partition the path space.

The results show that, in order to guarantee a partionning corresponding to visual features, it is necessary to bring domain expertise to guide the choice of the similarity measure.

Résumé

Les algorithmes de rendu photoréalistes comme le *path tracing* échantillonnent l'espace des chemins lumineux pour évaluer le transport lumineux. La distribution de ces échantillons est cruciale pour obtenir une bonne vitesse de convergence. Cependant la structure globale de l'espace des chemins est mal connue et les algorithmes de rendu n'utilisent que des informations locales pour déterminer cette distribution.

Dans la littérature, il existe des méthodes qui exploitent les propriétés de certains chemins pour les échantillonner efficacement. Il s'avère qu'elles sont limitées à des classes de chemins particulières et sont difficilement généralisables. Lors de ce stage, nous explorons la possibilité de décomposer l'espace des chemins en sous-espaces visuellement cohérents. Nous utilisons des méthodes d'apprentissage non-supervisé et plus particulièrement de *clustering*, afin de partitionner l'espace des chemins.

Les résultats obtenus montrent qu'afin de garantir un partitionnement correspondant à des effets visuels, il est nécessaire d'apporter de l'expertise du domaine pour guider le choix de la mesure de similarité.

Contents

List of Figures

1 | Introduction

In Computer Graphics, Rendering consists in computing a 2D image based on the description of 3D scene. This description includes the objects present in the scene, their materials which define their light interaction properties, the light sources, and the camera. Rendering is more and more present in our every day life, and its applications are very diverse. The most obvious example is video games. *Interactive rendering*, often associated with video games, focuses on producing images with a very small time budget at the cost of quality. On the other hand, *photorealistic rendering* aims at producing high quality images without time constraint. One obvious application of photorealistic rendering is the movie industry. Nowadays almost every movie includes rendered objects or even entire scenes. Photorealistic rendering is also used extensively in architecture, advertising and industrial design.

Figure 1.1: The movements and expressions of the actor are captured to generate a 3D model of the ape. A photorealistic rendering method is then used to include it in the film.

The models and algorithms for computing an accurate image have been known for decades [**?**] but, despite the improvement of computer technology, their computational time stays prohibitive for their adoption in interactive rendering applications. The computational time can vary from a couple of minutes to several days depending on the complexity of the scene.

In order to compute the value of a pixel, the rendering algorithms need to solve an integral. The integral is estimated by a stochastic Monte Carlo method. We are interested in the reduction of the variance of this estimator. In fact, the reason these rendering methods are so slow is because of the variance of the Monte

Carlo estimator. The estimator randomly draws light path samples and compute their contribution. A lot of samples are required in order to obtain a converged image. However, it is possible to drastically reduce the variance without adding new samples, by carefully choosing the sample distribution. The current methods, in addition to being expensive, only make use of local informations to choose this distribution, resulting in a non-optimal choice.

The set of all possible light paths form the *path space*. This space is sparse since most paths have a null contribution and its dimension is infinite. Contrary to the current methods, we are interested in its global structure with, in the long term, the goal of reducing the variance. We apply unsupervised learning and more specifically *cluster analysis* in order to decompose the path space in visually coherent sub-spaces. In addition to being a lead to variance reduction, a path space decomposition has applications in artistic control since it could allow a variety of new tools.

2 | Physics of Light

Computing a realistic image requires simulating the behavior of light according to the laws of physics. This chapter discusses the physical foundations of computer graphics.

Light can be described as a collection of small particles called *photons*, but also as an electromagnetic wave. In fact, light exhibits properties consistent with both of these models. This is called the wave-particle duality.

In computer graphics however, a simpler model called *geometric optics* that ignores the wave behavior of light is often used because it handles most of the humanly perceptible properties of light. In this model, light travels instantaneously in a straight line until it hits a surface. Upon contact, light interacts with the surface by reflection, refraction or absorption. For the sake of simplicity, the effects of participating media like smoke are often ignored and light is considered as traveling in free space.

2.1 Radiometry

In order to compute an image with a physically based method, it is necessary to manipulate a few physical quantities. This section gives a definition of the most important ones.

2.1.1 Wavelength

As explained earlier, light can be thought as an electromagnetic wave. A basic property of a wave is its wavelength. The wavelength of a light ray is responsible for its perceived color. The human eye can perceive electromagnetic waves whose wavelengths belong to the 400-700 nm range.

2.1.2 Radiant Flux

Radiant flux (also called radiant power) Φ describes the energy that passes through a surface per unit time. It is expressed in watt (W) or joule per second (J/s).

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t} \tag{2.1}$$

2.1.3 Irradiance

Irradiance E is the energy that passes through a unit area per unit time. Its unit is the watt per square meter ($\mathrm{W} \cdot \mathrm{m}^{-2}$).

$$E = \frac{\mathrm{d}\Phi}{\mathrm{d}A} \tag{2.2}$$

2.1.4 Solid Angle

The solid angle is an extension of the concept of one dimensional angle. It is used to measure the area of projection of a three dimensional object on a unit sphere (i.e. how large the object appears from a point of view). The maximum solid angle is 4π (the area of the unit sphere). The measure unit is the steradian (sr). It is defined as:

$$\omega = \frac{A}{r^2} \tag{2.3}$$

where A is the area on the sphere of radius r.

In computer graphics, the differential solid angle is used to represent an infinitely thin cone centered around a direction.

$$\mathrm{d}\omega = \frac{\mathrm{d}A}{r^2} \tag{2.4}$$

2.1.5 Radiance

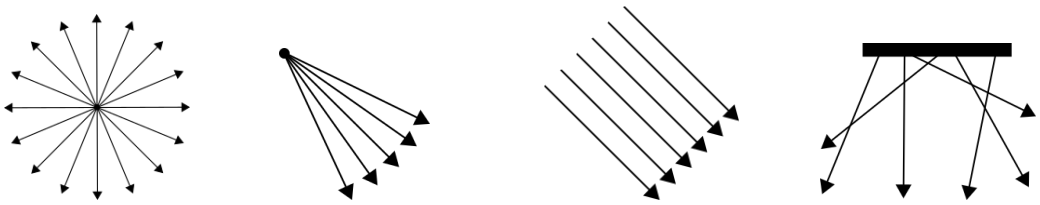
Radiance L expresses the energy that passes through a surface per unit projected area, per unit solid angle, and per unit time. Its unit is the watt per steradian per square meter ($\mathrm{W} \cdot \mathrm{sr}^{-1} \cdot \mathrm{m}^{-2}$).

$$L = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\omega \cdot \mathrm{d}A^{\perp}} = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\omega \cdot \mathrm{d}A \cos\theta} \tag{2.5}$$

Where $\mathrm{d}\omega$ is the differential solid angle, θ is the angle between the surface normal and the direction of interest, and $A^{\perp}$ is the projected area.

2.2 Light Interactions

Apart from being emitted by a source, light can only get absorbed, reflected or refracted when it strikes an object. All of these interactions can occur in the same time to the same light ray but the fundamental rule is the conservation conservation of energy. This is formally defined in 2.3.2.



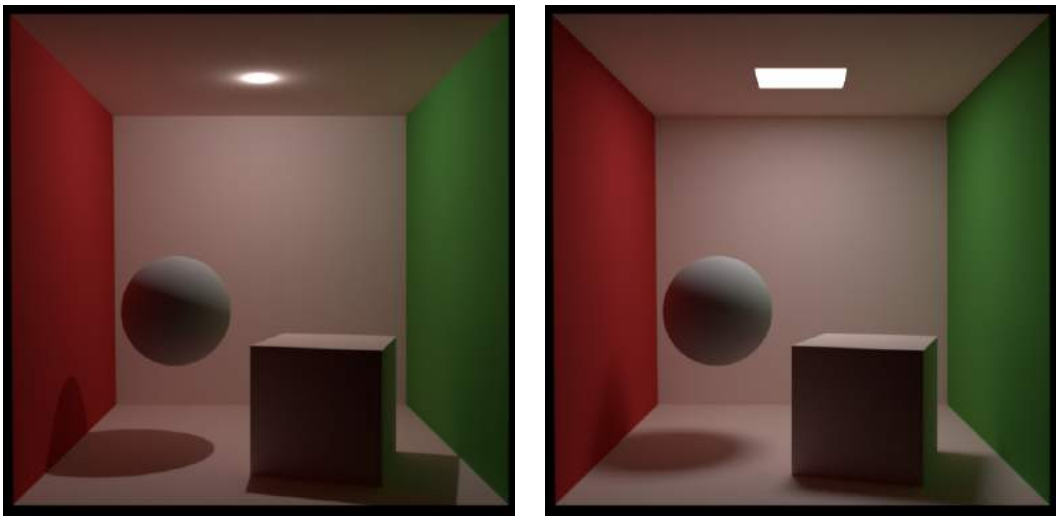
(a) Point light (b) Spot light (c) Directional light (d) Area light

Figure 2.1: The main light source types

2.2.1 Emission

In the real world, the main source of light emission is heat and it is possible to compute the radiance of light source, knowing its temperature, from the black body model and the Planck's law. However, in computer graphics we are rarely interested in simulating the causes of light emission and the light source characteristics are usually directly specified in the scene description.

The most common light source models are the point light, the spot light, the directional light, and the area light. The directional light is mainly used to simulate the light coming from a virtually infinite distance (e.g. the sun). The area light is the most realistic light source because every real world light source has an non-zero area. The area light model is the only one able to produce soft shadows (see figure 2.2).



(a) Point light (b) Area light

Figure 2.2: Hard and soft shadows

2.2.2 Absorption

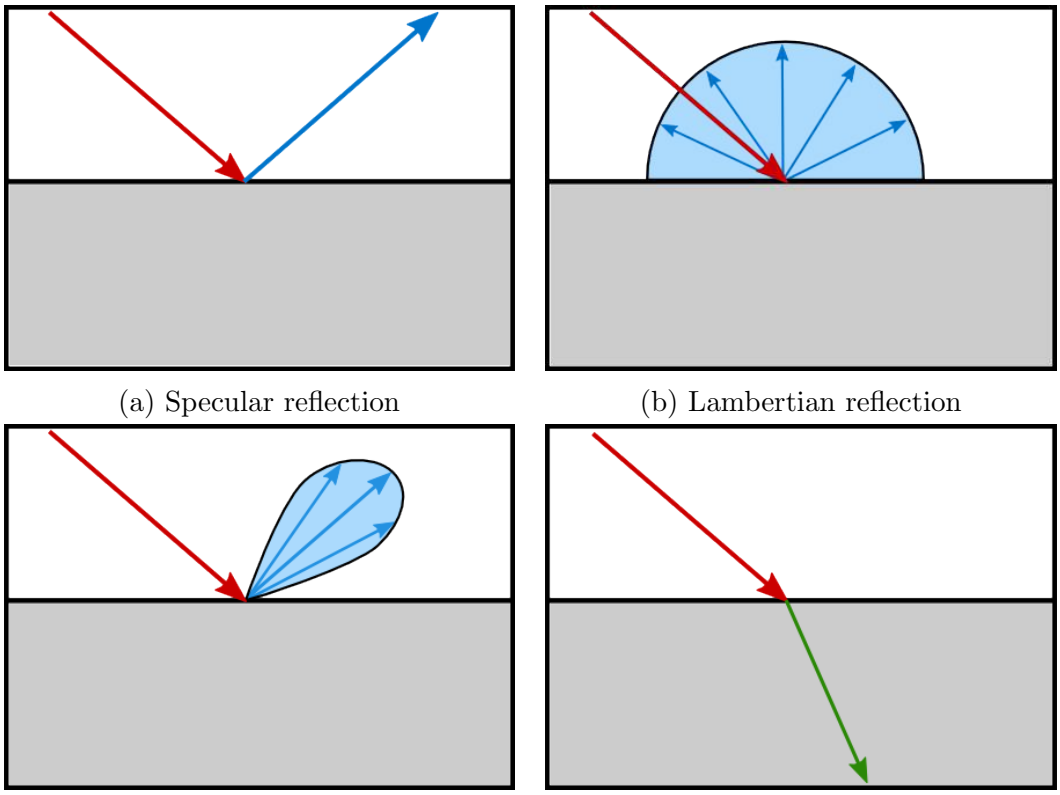
Upon contact with a surface, light can be converted in thermal energy. In computer graphics, thermal phenomena are generally ignored and absorption is accounted by leaving the incoming radiance superior or equal to the sum of reflected and refracted radiance.

2.2.3 Reflection

Reflection is probably the most noticeable interaction of light with matter. In the framework of geometric optics, we consider the surfaces to be completely smooths. When a light ray hits a surface, it can get reflected in its "mirror" direction given by the relation:

$$\theta_1 = \theta_2 \tag{2.6}$$

Where θ_1 is the incident angle and θ_2 is the exitant angle.



(a) Specular reflection (b) Lambertian reflection

(c) Glossy reflection (d) Specular refraction

Figure 2.3: Different BSDFs

2.2.4 Refraction

Refraction occurs when a light ray reaches a non-opaque medium like glass or water. The direction of the light ray is modified depending on the refraction indices of the two media. In geometric optics the new direction is given by the Snell-Descartes law.

$$\sin\theta_1 \cdot n_1 = \sin\theta_2 \cdot n_2 \tag{2.7}$$

Where n_1 and n_2 are the refraction indices of the original and new media. The refraction index of a given medium varies with the ray wavelength.

2.3 Materials

2.3.1 Color

Color is not an intrinsic property of a material. Instead, the perceived color depends on the spectrum of the incident light and on which part of this spectrum is reflected. For example, an object is seen as red under a monochromatic (e.g. white) light because its material filters the frequencies that do not correspond to red. These frequencies get absorbed and thus converted in thermal energy. This is the reason why black objects absorb more heat under the sun light than lighter objects.

2.3.2 Bidirectional Reflectance Distribution Function

In the real world, a photon that hits a surface at a precise location from a specific direction can get reflected in only one particular direction. However, the geometry of realistic surfaces varies at a microscopic scale which is impossible to explicitly represent in computer graphics due to the huge computational cost this implies. Thus, materials reflectance is modeled as a *bidirectional reflectance distribution function* (BRDF) which defines the distribution of the reflected incoming radiance in the outgoing directions. This was first defined by Nicodemus [**?**].

$$f_r(x, \omega_i, \omega_o) = \frac{\mathrm{d}L_r(x, \omega_o)}{L_i(x, \omega_i) \cos\theta_i \, \mathrm{d}\omega_i}$$

In order to be physically plausible, a BRDF has to respect some properties. First of all, the law of conservation of energy must be respected.

$$\int_{S^2} f_r(x, \omega_i, \omega_o) \cos\theta_i \, \mathrm{d}\omega_o \leq 1 \quad \forall \omega_i \tag{2.8}$$

The Helmholtz reciprocity principle ensures the BRDF is symetrical.

$$f_r(x, \omega_i, \omega_o) = f_r(x, \omega_o, \omega_i) \tag{2.9}$$

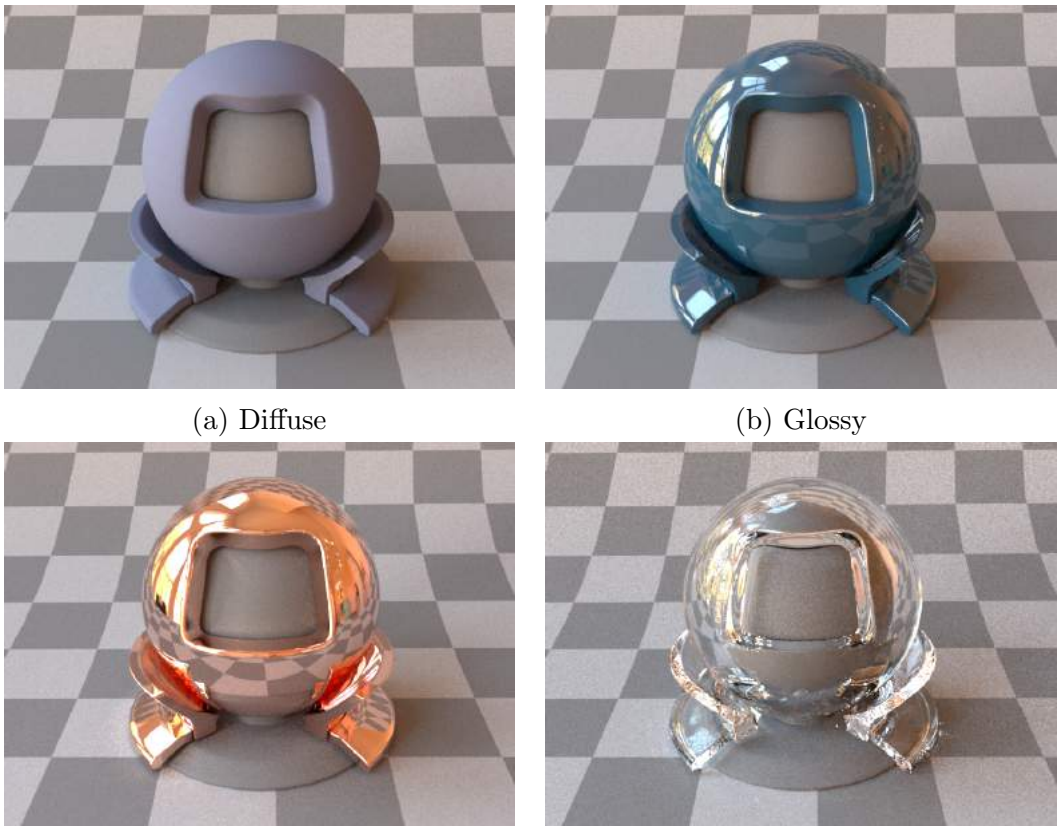
And finally, the BDRF must be possible for any value of ω_i and ω_o.

$$f_r(x, \omega_i, \omega_o) \geq 0 \quad \forall \omega_i \, \omega_o \tag{2.10}$$

The concept of BRDF can be extended to describe other types of scattering. Similarly to a BRDF, A *bidirectional transmittance distribution function* (BTDF) models the distribution of the refracted radiance. A *bidirectional scattering distribution function* (BSDF) combines a BRDF and a BTDF to model both scattering events.

2.3.3 Classification

Materials are often classified based on their reflection profile. Those that reflect light equally in all directions are refered as *lambertian* or purely *diffuse*. There exist no perfect diffuse reflector in the real world but paper or concrete are very close examples. Materials that reflect light in only one direction are called *specular*. The obvious example is a mirror. Most real world materials lie between these two extremes and can be refered as *glossy*.



(a) Diffuse (b) Glossy

(c) Conductor (d) Dielectric

Figure 2.4: Different materials can be rendered by using different BSDFs

3 | Rendering

This chapter defines the rendering task and describes some of the most popular rendering algorithms. An object can receive light directly from a source (*direct illumination*) or after one or several scattering events (*indirect illumination*). In order to compute a realistic image, it is neccessary to compute both direct and indirect lighting. This is refered as *global illumination*.

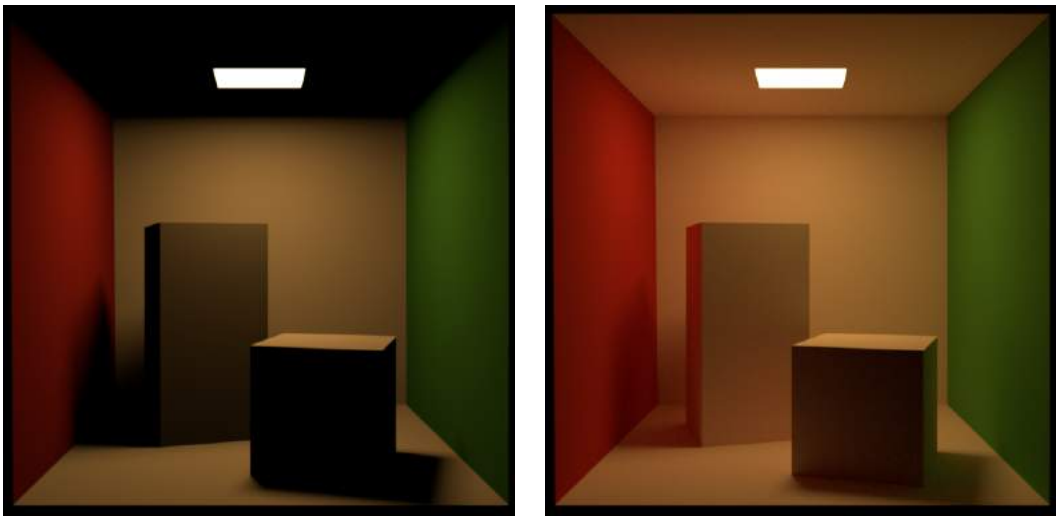


Figure 3.1: Direct and global illumination

3.1 Rendering

3.1.1 The Measurement Equation

A renderer computes an image by evaluating the value of each of its pixels. The value I_j of the j^{th} pixel is given by the *measurement equation*. A rendering method that converges toward the exact solution of the measurement equation is called an *unbiased* method.

$$I_j = \int_{S^2} \int_M W_e(x, \omega_i) L_i(x, \omega_i) \cos(\theta_i) \, \mathrm{d}A(x) \, \mathrm{d}\sigma(\omega_i) \tag{3.1}$$

Where S^2 is the direction domain, M is the camera lens surface, $W_e(x, \omega_i)$ is the sensibility of the sensor to the radiance at the position x and coming from the direction ω_i, θ_i is the angle between the incoming direction and the surface normal, $A(x)$ is the area measure and $\sigma(\omega_i)$ is the solid angle measure.

3.1.2 The Rendering Equation

In order to evaluate the measurement equation, it is necessary to compute the incident radiance $L_i(x, \omega_i)$, which can be expressed as the excitant radiance with a change of variable.

$$L_i(x, \omega_i) = L_o(\text{tr}(x, \omega_i), -\omega_i) \tag{3.2}$$

Where $\text{tr}(x, \omega_i)$ is the *ray-casting function* which returns the first point visible from x in the direction ω_i.

In 1986, Kajiya introduced the *rendering equation* [**?**]. The main task of every renderer is to solve this equation.

$$\begin{aligned} L_o(x, \omega_o) &= L_e(x, \omega_o) + L_{o,s}(x, \omega_o) \\ &= L_e(x, \omega_o) + \int_{S^2} L_i(x, \omega_i) f_s(x, \omega_i, \omega_o) \cos(\theta_i) \, \mathrm{d}\sigma(\omega_i) \\ &= L_e(x, \omega_o) + \int_{S^2} L_o(\text{tr}(x, \omega_i), -\omega_i) f_s(x, \omega_i, \omega_o) \cos(\theta_i) \, \mathrm{d}\sigma(\omega_i) \end{aligned} \tag{3.3}$$

Where $L_{o,s}(x, \omega_o)$ is the scattered radiance and $f_s(x, \omega_i, \omega_o)$ is the BSDF.

3.2 Monte Carlo Integration

For most scenes, it is not possible to solve the rendering equation analytically. Instead, rendering algorithms have to rely on a stochastic integration method. Given the integral J:

$$J = \int_D f(x) \, \mathrm{d}x$$

The expected value of a random variable Y distributed according to the probability distribution function p can be defined as:

$$\mathbb{E}[Y] = \int_D y p(y) \, \mathrm{d}y \tag{3.4}$$

We can use this equation by introducing the random variable X distributed according to p and the random variable $Y = f(X)$. The integral can now be expressed as the expected value of Y:

$$J = \int_D f(x) \, \mathrm{d}x = \int_D \frac{f(x)}{p(x)} p(x) \, \mathrm{d}x = \mathbb{E}[f(X)] = \mathbb{E}[Y] \tag{3.5}$$

We can now define J_N a stochastic estimator of J called the Monte Carlo estimator:

$$J \approx < J >_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{3.6}$$

This estimator is unbiaised but its convergence rate is slow due to its variance which is proportional with $\frac{1}{\sqrt{n}}$.

$$\sigma(< f(X) >_N) = \frac{1}{\sqrt{N}}\sqrt{< f(X)^2 >_N - < f(X) >_N^2} \tag{3.7}$$

3.2.1 Importance Sampling

The variance of the Monte Carlo estimator is highly dependent on the choice of the sampling distribution. Choosing a sampling distribution approximately proportional to the integrand is a very popular variance reduction technique known as *importance sampling*. In the ideal case we want to define $p(x)$ as:

$$p(x) = \frac{|f(x)|}{\int_D |f(x)| \, \mathrm{d}x} \tag{3.8}$$

In this case, the variance would be equal to zero. However this is not possible because this would require to know the solution of the integral that we are already trying to solve. In practice, we choose a sampling distribution proportional to some factors of the integrand.

3.3 Path Space

The rendering equation cannot be solved analytically in most cases. Thus, we rely on the Monte Carlo integration method to solve it. However, when we do so, we have to face a combinatorial explosion. Eric Veach [**?**] reformulated the measurement equation as an integral over the light path space Ω as

$$I_j = \int_\Omega f_j(\bar{x}) \, \mathrm{d}\mu(\bar{x}),$$

where $\bar{x}$ is the light path of length $k \in [2, \infty]$ and $f_j(\bar{x})$ is the contribution of the path $\bar{x}$ to the j^{th} pixel

$$\bar{x} = \mathbf{x}_0 \mathbf{x}_1 \ldots \mathbf{x}_k$$

$$f_j(\bar{x}) = L_e(x_0, x_1) G(x_0, x_1) \prod_{i=1}^{k} f_s(x_{i-1}, x_i, x_{i+1}) G(x_i, x_{i+1}) W_e(x_{k-1}, x_k). \tag{3.9}$$

3.4 Path Tracing

Path tracing is an unbiaised rendering method that applies Monte Carlo integration to solve the measurement equation. The path space is sampled by building paths starting from the camera or a light source (or from both [**?**]). At each vertex, the next one is determined by choosing a random direction and tracing a ray to find the first intersection with a surface.

The contribution $f_j(\bar{x})$ of each path is computed according to equation 3.9 and weighted by its sampling probability density function $p(\bar{x})$.

$$I_j \approx \frac{1}{n} \sum_{i=1}^{n} \frac{f_j(\bar{x})}{p(\bar{x})}$$

Due to the variance of the Monte Carlo estimator, a low number of samples produces a noisy image (see figure 3.2). However, since the algorithm is unbiaised, the image converges toward the exact solution as more samples are added.

(a) 4 samples per pixel (b) 128 samples per pixel

Figure 3.2: Monte Carlo noise

3.5 Multiple Importance Sampling

In rendering many integration problems are difficult and finding a low variance sampling strategy remains a challenge. Instead of trying to find one optimal sampling strategy, it is possible to combine several potentially good strategies in a low variance estimator. This is called *multiple importance sampling* [**?**]. The Monte Carlo estimator is then defined as:

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(x_{ij}) \frac{f(x_{ij})}{p_i(x_{ij})} \tag{3.10}$$

and where n is the number of sampling strategies, n_i is the number of samples allocated to each strategy, $w_i(x)$ is the weight given by each strategy to the sample. The weighting functions have to respect two constraints in order to keep the estimator unbiaised:

$$\sum_{i=1}^{n} w_i(x) = 1 \quad \forall x \in \{\, x : f(x) \neq 0 \,\}$$

$$p_i(x) = 0 \Rightarrow w_i(x) = 0$$

In the context of path tracing, when building a path, the next vertex can be determined by sampling the BSDF or by sampling the light sources for instance. None of these strategies work well in every case but both can be combined with multiple importance sampling to define a more robust estimator.

3.6 Metropolis Light Transport

The Metropolis-Hastings algorithm (see algorithm 1) is used to draw samples from a probability distribution $P(x)$ that is hard to sample directly from.

Algorithm 1: The Metropolis-Hastings algorithm

initialize x randomly;
while *more samples are required* **do**
  randomly pick x' according to $g(x \to x')$;
  $a(x \to x') := \min\left(1, \dfrac{P(x')g(x' \to x)}{P(x)g(x \to x')}\right)$;
  $x = \begin{cases} x' & \text{with probability } a(x \to x') \\ x & \text{with probability } 1 - a(x \to x') \end{cases}$
  register x;
end

Usually, the first samples are discarded in order to remove the correlation with the initial sample x. Metropolis Light Transport (MLT) [**?**] is a rendering method inspired by this algorithm. The goal is to sample the path space with a distribution probability proportional to the path contribution $f(\bar{x})$.

Each new path $\bar{x}'$ is generated by applying a mutation on the current one $\bar{x}$ according to an arbitrary *transition probability* $g(\bar{x} \to \bar{x}')$. Each mutated path can get accepted or rejected according to the *acceptance probability* $a(\bar{x} \to \bar{x}')$. The possible mutations include adding, moving, or removing some vertices. Since the probability distribution $P(\bar{x})$ should ideally be proportional to $f(\bar{x})$, we can rewrite the acceptance probability as:

$$a(\bar{x} \to \bar{x}') = \min\left(1, \frac{f(\bar{x}')g(\bar{x}' \to \bar{x})}{f(\bar{x})g(\bar{x} \to \bar{x}')}\right)$$

A strength of the MLT algorithm is that it is able to sample more efficiently difficult light transports. Once a path has been found, the similar paths can be explored locally by applying small mutations. The local exploration of the path space leads to a slow convergence rate.

4 | Analysis of Light Transport

This chapter presents our work. Sampling the path space efficiently is not an easy thing to do. The dimension of this space is infinite (a light path can be of any length) and the vast majority of paths do not contribute to the image. Furthermore, its global structure is not well known.

The current rendering methods only use local informations to sample the path space. For instance, a path tracing algorithm usually chooses each vertex one-by-one, the choice of the next vertex depending only on the two previous ones. Metropolis Light Transport stochastically applies a mutation to the current path to find the next one, thus exploring the path space locally.

However, we are interested in building a better understanding of the global structure of the path space. To this end, we want to explore the possibility of decomposing the path space in subspaces visually coherent in the image space. The applications of a path space decomposition are diverse: improved sampling, layer composing and geometric deformations to cite only a few of them.

4.1 Previous Work

4.1.1 Heckbert's Notation

The path grammar introduced by Paul Heckert [**?**] represents each vertex with one symbol, S or D for specular or diffuse scattering. The light source and camera vertices are respectively represented by L and E. For instance, a wall directly lit by the sun and then seen through a mirror is represented by $LDSE$.

This notation combined with regular expressions can be used to select larger families of light paths. For instance, all the possible paths match the regular expression $L(S|D)^*E$. More concretely, the paths contributing to a caustic (such as the one visible on figure 4.1) can be separated from the rest by the regex LSS^+DS^*E independently of the scene.

4.1.2 Path Manifold Exploration

The specular paths when they are present in a scene, transport a high amount of radiance and thus contribute highly to the image because they are mostly made of specular interactions. However they are hard to sample efficiently because they are rare in the path space since they obey specific constraints. Wenzel Jakob introduced the *Path Manifold Exploration* [**?**] method to alleviate this problem.

A surface interaction is specular when the surface normal direction n is equal to the *half vector* direction ω_h, where the half vector is defined as $\omega_h = (\omega_i + \omega_o)/2$

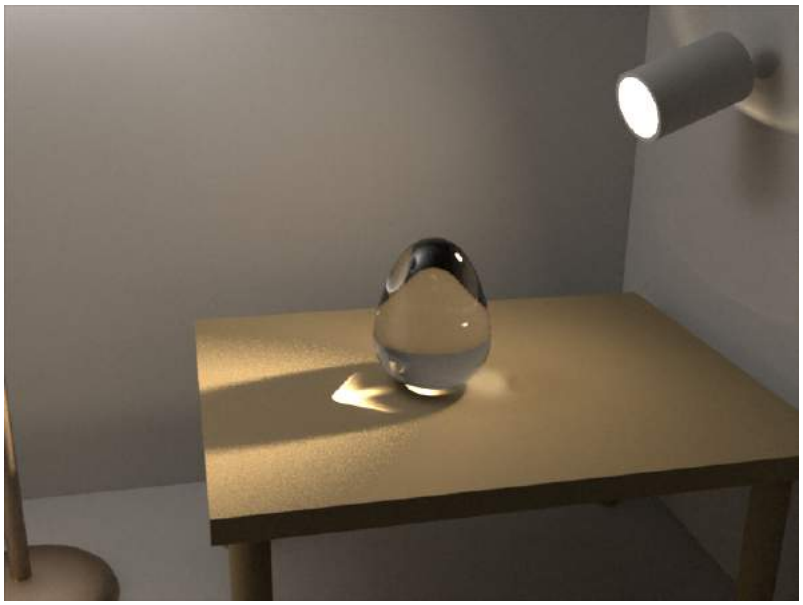


Figure 4.1: Several caustics

with ω_i and ω_o the incident and exitant directions. In order to take into account the refraction interactions, the *generalized half vector* is defined as:

$$\omega_h = \frac{\eta_i \omega_i + \eta_o \omega_o}{||\eta_i \omega_i + \eta_o \omega_o||}$$

where η_i and η_o are the indices of refraction of the incident and excitant media.

The Path Manifold Exploration method is based on the fact that the specular light paths form a manifold in the path space and that this manifold can be explored locally. By randomly moving a diffuse vertex of the path and exploiting the constraint gradient at each specular vertex it is able to generate a viable specular path.

4.1.3 Discussion

The Heckbert's notation has been used successfully to isolate the paths contributing to some particular visual effects [**?**], leading to useful tools for content creation. However it lacks the expessiveness necessary to take into account some characterics of paths like the transported radiance and then does not allow to classify many visual effects.

The Manifold Exploration method, in addition to being computationally expensive, is restricted to specular or glossy paths and does not generalize well to other classes of paths.

4.2 Path Space Clustering

Clustering is the task of grouping similar data points together. Our goal is to find a visually coherent partition of the path space. Thus it is obvious that clustering is well-suited for this.

4.2.1 Approach

Our approach consists of generating a very large number (usually billions) of light paths in order to render a converged image. We can then use the data of the generated paths to create our dataset.

In order to have all paths represented by a data point of equal dimension, we characterize a path by various features: position of some vertices, BRDF values, etc. In total, we end up with more than 60 features to characterize each path. In order to give an equal importance to each feature, we use the standard score normalization method defined as:

$$\mathbf{z} = \frac{\mathbf{x} - \mu}{\sigma} \tag{4.1}$$

where $\mathbf{x}$ is the raw data, μ is the mean vector, and σ is the standard deviation vector. This normalized dataset is passed to the k-means algorithm that returns the clusters.

4.2.2 The *k*-Means Algorithm

The k-means algorithm [**?**] is the most popular clustering algorithm. Its goal is to find the centroids that minimize the *sum of square error* (SSE) given a dataset of M-dimensional points and a specified number of clusters k. The SSE is defined as:

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mu_j) \tag{4.2}$$

where μ_j is the centroid of the j^{th} cluster and $d(\mathbf{x}, \mu_j)$ is the distance function. The default distance function is the squared euclidean distance.

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{M} (x_i - y_i)^2 \tag{4.3}$$

4.2.3 Results

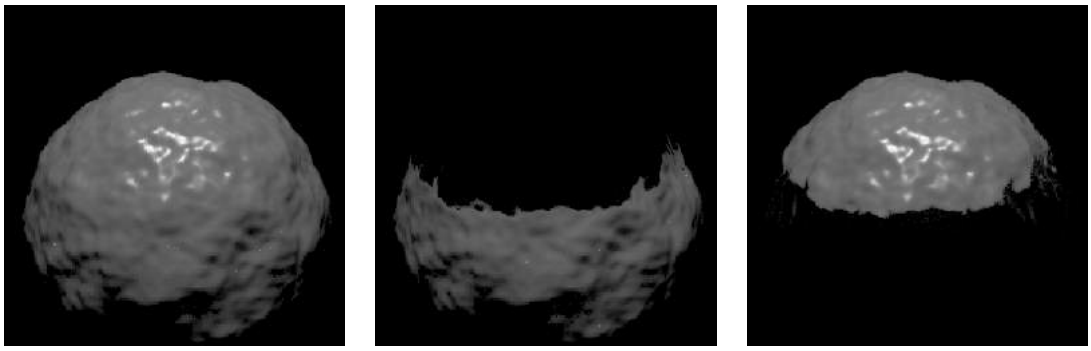
The first results obtained are visible on figure 4.6. The paths are classified in two clusters. The images obtained for each cluster show the high impact of geometric features. This is caused by the high correlation and redundancy present among the feature set. These results cannot be satisfying since we are looking for a visually coherent classification.

However, we can guide the choice of the similarity measure by expertise. In order to separate specular and non-specular paths, we can give a high weight to the path contribution function and a low weight to the geometrical features. Concretely, we redefine the distance function as:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{M} w_i (x_i - y_i)^2 \tag{4.4}$$

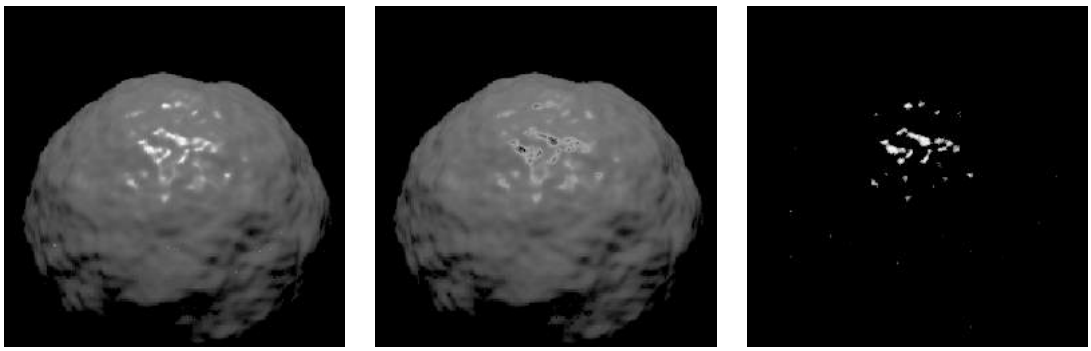
where $\mathbf{x}$ and $\mathbf{y}$ are M-dimensional data points and w_i is the weight of the i^{th} feature.

As it is visible on figure 4.3, by choosing the appropriate weights, we successfully separated the specular paths (cluster 2) from the others (cluster 1). However, using the same weights on a different scene gives different results. Thus, we couldn't reach our goal of scene independence and the problem now resides in the choice of the weights. These results motivate the need of domain expertise or human interaction to guide the clustering algorithm toward a visually coherent solution.



(a) Original image (b) Cluster 1 (c) Cluster 2

Figure 4.2: Clustering with k-means



(a) Original image (b) Cluster 1 (c) Cluster 2

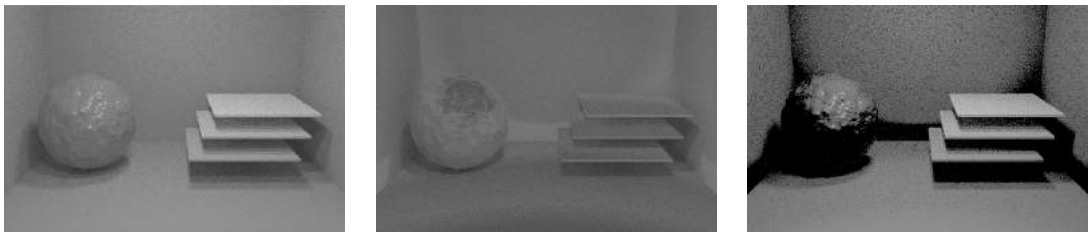
Figure 4.3: Clustering with k-means and a weighted similarity measure

(a) Original image (b) Cluster 1 (c) Cluster 2

Figure 4.4: Clustering with k-means

4.3 ICA for Visual Feature Separation

The *Independent Component Analysis* [**?**] (ICA) is a statistical tool that allows to recover linearly mixed signals. It has a wide range of applications such as the cocktail



(a) Original image (b) Cluster 1 (c) Cluster 2

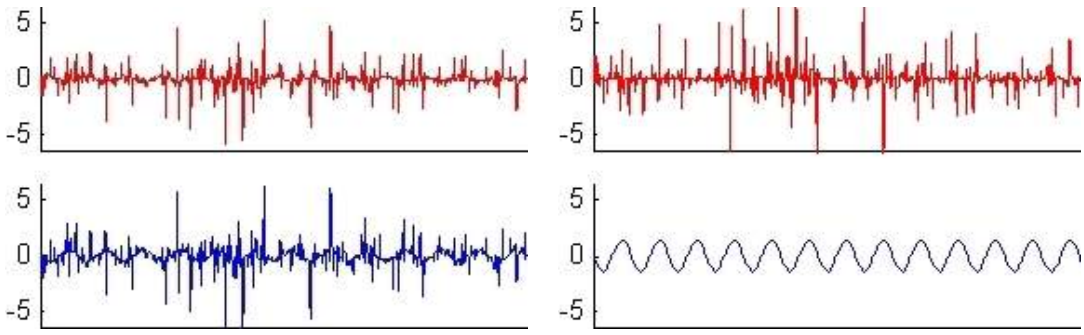
Figure 4.5: Clustering with k-means

party problem. Concretely, it estimates the linear transformation from the observed space to the original space where each dimension is statistically independent from the others (according to the sample set). This is formulated as:

$$\mathbf{s} = \mathbf{W}\mathbf{x}$$

where $\mathbf{x}$ is a d-dimensional vector defining a data point in the original space, $\mathbf{W}$ is a $d \times d$ matrix, and is $\mathbf{s}$ a d-dimensional vector defining a data point in the transformed space.

In order to estimate the matrix $\mathbf{W}$, it is necessary to provide n samples which means n d-dimensional vectors (in the original space). In other words, only a $n \times d$ matrix is required as input. As the output, we get the matrix $\mathbf{W}$ and then we can compute the position of any sample in the new space.



(a) Observed signals (b) Separated signals

Figure 4.6: The observed signals are a linear combination of the separated signals

4.3.1 Approach

We apply ICA to the problem of visual feature separation in the context of photorealistic rendering. We model visual effects as abstract light sources. We make the hypothesis that these sources are linearly mixed in some variables that we can observe. These observed variables are those that characterize the light paths.

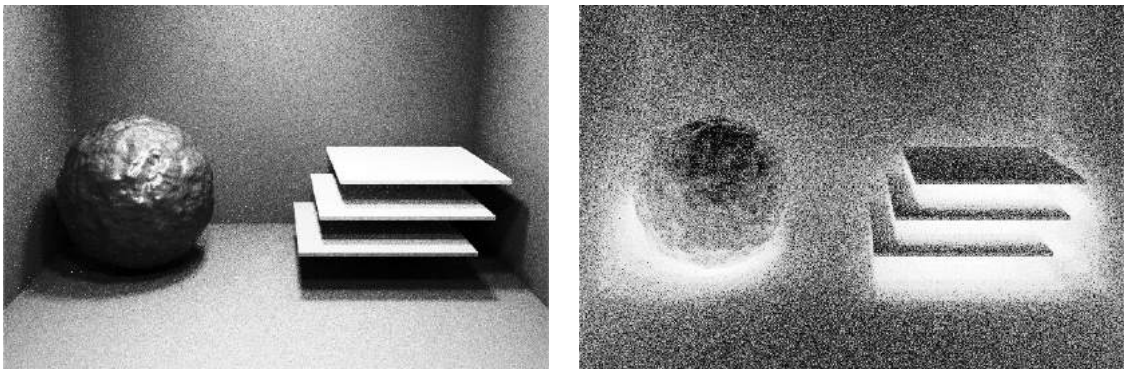
We choose the path length k and the path contribution to the image v as our observed variables. Each sample $\mathbf{x}$ corresponds to a light path.

$$\mathbf{x} = \begin{pmatrix} k \\ v \end{pmatrix}$$

In the ideal case, the dimensions of the new space correspond to the degree of belonging of the path to the different visual features.

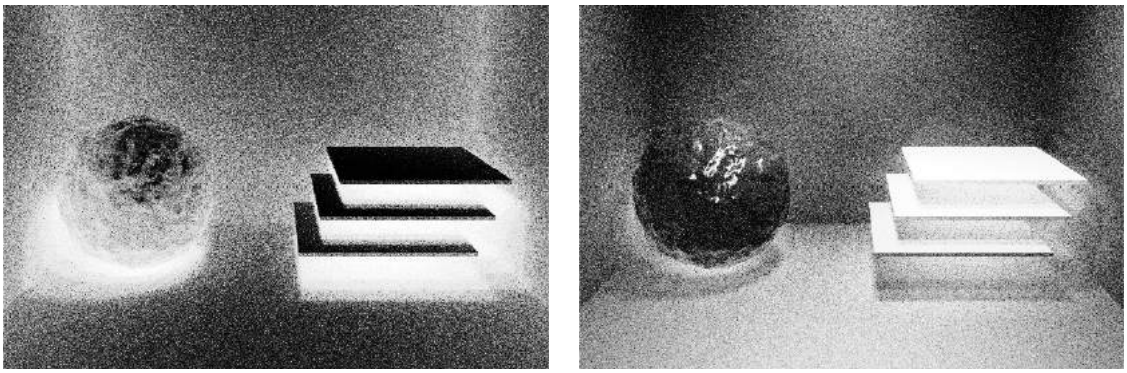
4.3.2 Results

We render a 400x600 image by generating 7.10^7 path using the *bidirectional path tracing* algorithm [**?**]. The input and transformed features are visible on figures 4.7 and 4.8. As we can see, the obtained features are not visually interesting. This is an indication that our abstract light sources are not linearly mixed in the chosen variables k and v. However these independent features can be used in clustering in addition to some other ones.



(a) The average contribution (b) The average path length

Figure 4.7: The input signals. Correlation: -0.43



(a) The average s_0 (b) The average s_1

Figure 4.8: The output signals. Correlation: 0

4.4 Notation Clustering

4.4.1 Approach

A major flaw of our previous approaches comes from the fact that we only take into account the very last interactions of the paths. This is due to the necessity to represent each path by a fixed length vector when using the k-means clustering

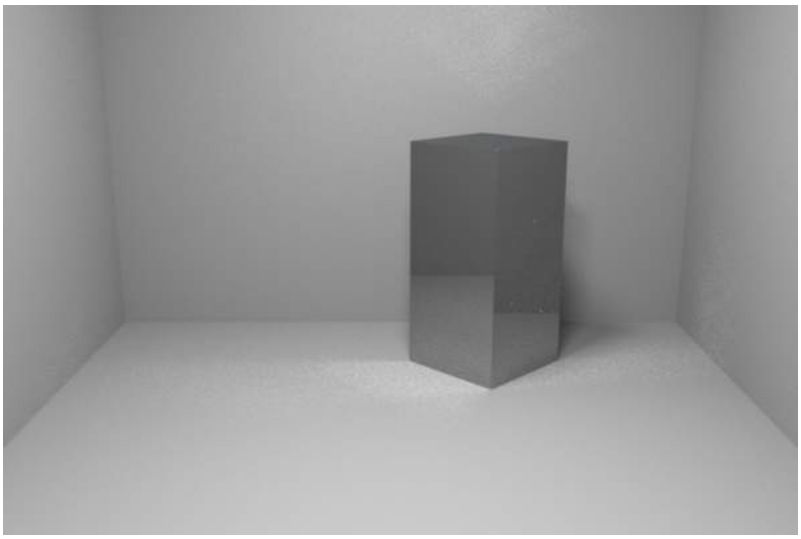


Figure 4.9: The complete image of the test scene

algorithm or the ICA method. The k-medoids clustering algorithm removes this constraint by allowing any kind of distance function to be used. This algorithm is very similar to the k-medoids but it uses only existing points (*medoids*) instead mean points (*centroids*) to represent each cluster. Its main drawback is a higher complexity because of the medoids update step.

The Levenshtein distance [**?**], also known as the *edit distance*, defines the distance between two strings of characters as the smallest number of operations necessary to make them equal. The set of operations typically includes insertion, deletion, and substitution and each operation cost can vary depending on the characters involved. The Levenshtein distance can be computed by the Wagner-Fischer algorithm [**?**].

We use the Heckbert's notation to represent light paths by strings of characters and apply the k-medoids algorithm with the levenshtein distance to find visually coherent clusters.

4.4.2 Results

Similarly to our previous approaches, we use a simple scene (figure 4.9) and generate a large number of light paths. Each clusters obtained is displayed on an image. The results of the Levenshtein distance are displayed in figure 4.10. All path notations are given in the forward order, which means that the complete notation of DS is LDSE.

It appears clearly that the Levenshtein distance does not fit our task very well. The clustering seem to be mainly based on the length of the paths. It must be kept in mind that each cluster is made of a large number of different path notations, even though there is only one medoid for each of them. We identify several issues with the Levenshtein distance.

First of all, it does not take into account the position of the characters within the string. We believe that the interactions closest to the eye should have a greater impact and thus the cost of operations at these locations should be high.

Also, it does not take into account the context of the operations, i.e. the characters around the position of the modification. We think that the operations that have

(a) DD (b) DDD (c) DDDD

(d) DDDDD (e) DDDDDD (f) DDDDDDDD

Figure 4.10: The clusters obtained with the Levenshtein distance

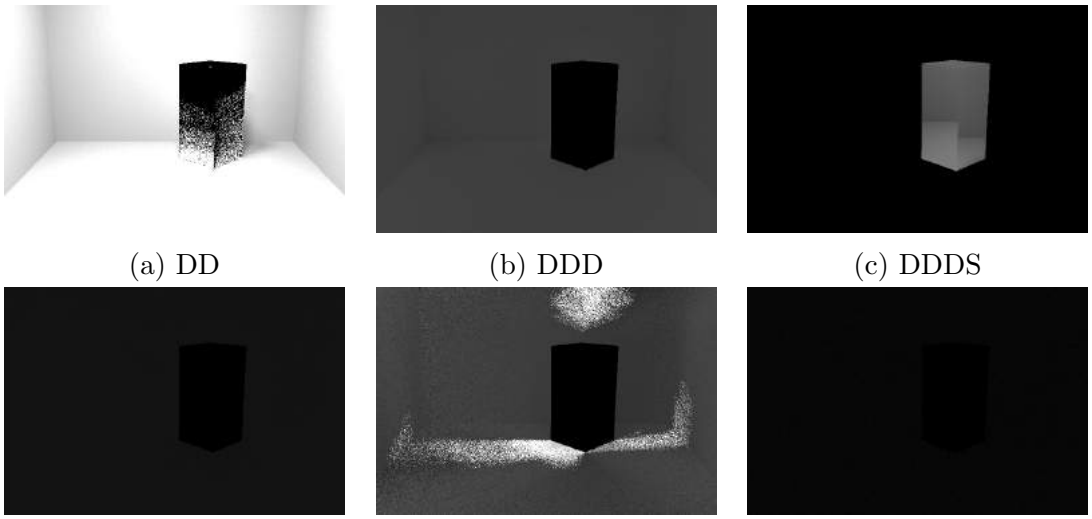
a low impact on the shape of the notation should have a lower cost. Concretely, the cost of inserting a specular interaction in the middle of a specular chain should be lower than the cost of inserting a diffuse one.

In order to reach these goals we defined several distance functions. The simpliest of all is the alignement distance. In order to compute it, the notations are parsed in the backward (eye to light) direction and the number of different characters is counted. Once the end of the smallest notation has been reached, the distance is obtained by adding the number of remaining characters to the number of different characters multiplied by a cost factor $\alpha > 1$.

$$\text{alignementDistance}(A, B) = \alpha \cdot \text{numberOfDifferent}(A, B) + \text{sizeDifference}(A, B)$$

This distance function has the property of being highly dependent on the similarity of the closest interactions from the eye. It means that adding a sequence of characers to a notation will not generate a very distant notation from the original one.

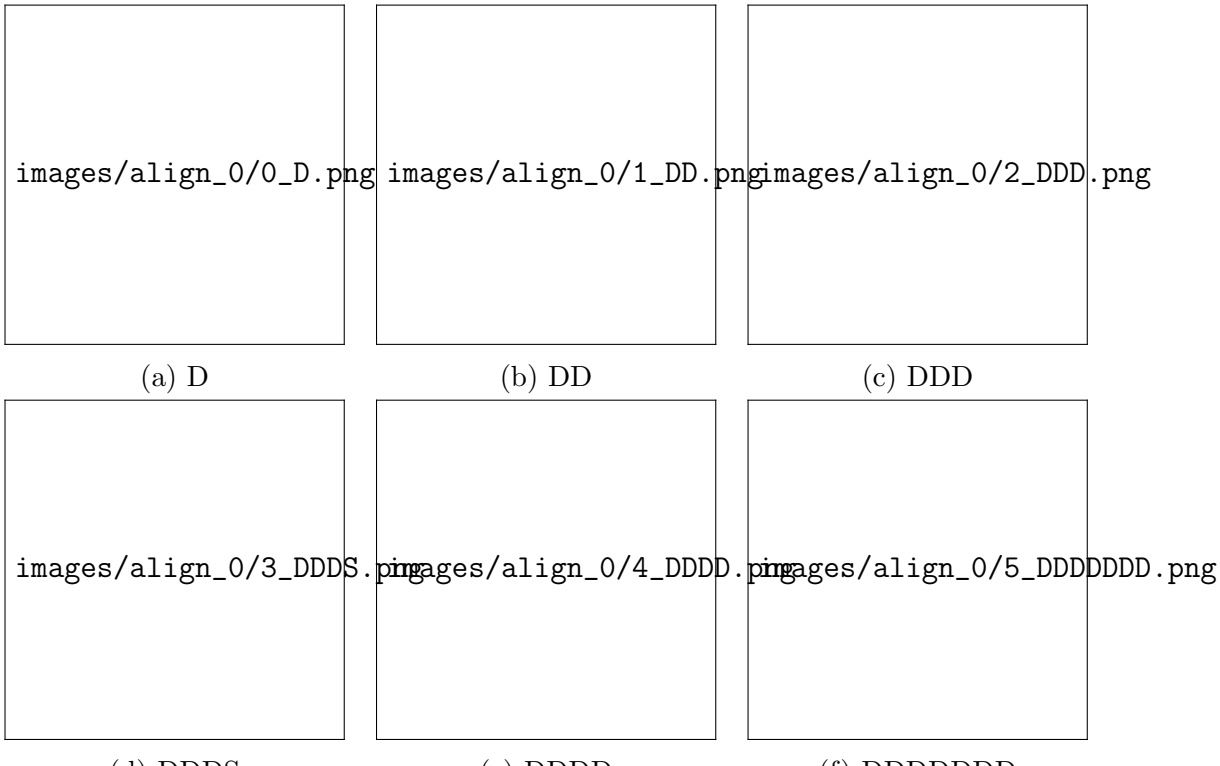
The results obtained with the alignement distance look more promising than the previous ones.



(a) DD (b) DDD (c) DDDS

(d) DDDDD (e) DDDSD (f) DDDDDDD

Figure 4.11: The clusters obtained with the alignement distance



(a) D (b) DD (c) DDD

(d) DDDS (e) DDDD (f) DDDDDDD

Figure 4.12: The clusters obtained with the modified Levenshtein distance

5 | Conclusion & Future Work

We started this thesis by defining the physical fundations of computer graphics in chapter 2. In chapter 3, we introduced the rendering task and some popular photorealistic rendering algorithms. These methods are slow to converge due to the high number of samples of the path space required to reduce the variance of the Monte Carlo estimator. The variance is highly dependent on the probability distribution of these samples. Without any knowledge on the global structure of this space, the current algorithms choose their sampling distribution with computationally expensive methods which only give local optima.

We described our work in chapter 4. Our main goal was to get a better understanding of the global structure of the light path space. By using of a photorealistic rendering algorithm such as path tracing, we were able to generate large, unlabeled and high dimensional dataset representing the lights paths of any scene. We applied clustering methods to decompose this dataset and, by extension, the light path space itself. In another approach, but with the same goal, we applied independent component analysis to find independent signals in the path space.

Our results show that, in order to decompose the path space in visually coherent features, it is necessary to provide domain expertise to guide the clustering methods.

5.1 Future Work

Cluster analysis

Despite his popularity, the k-means algorithm is very limited. It is not suited to high dimensional data because of the *curse of dimensionality* and it has not been designed to handle large datasets. Also, it requires to specify the number of clusters k as a parameter which is something hard to provide since when cannot visualize a high dimensional dataset. But maybe its biggest flaw is its inability to find arbitrarily shaped clusters.

To our knowledge, few clustering algorithms do not suffer from any of these flaws but we believe that the *subspace clustering* methods [**?**] or the recent advances in *spectral clustering* [**?**] could provide the robustness required by our task.

In order to guide the clustering algorithm toward a visually coherent solution, *interactive clustering* [**?**] seems to be a promising approach.

Supervised learning

We decided to use unsupervised learning methods because of high volume of data we deal with and the prohibitive cost of labeling each path by hand. However, we believe that a viable approach could consist of labeling large amounts of light paths by selecting them either by their Heckbert's notation or by using interactive tools like those developed by Schmidt et al. [**?**].

After this, a supervised learning algorithm could learn on a large set of features and be used to classify any new path. Although the cost of creating a sufficiently large dataset is not clear and could possibly remain prohibitive.

Bibliography

[1] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[2] F.E. Nicodemus, J.C. Richmond, J.J. Hsia, W.I. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. *Applied Optics*, 9:1474–1475, 1977.

[3] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[4] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[5] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of the 3rd International Conference on Computational Graphics and Visualization Techniques*, COMPUGRAPHICS '93, pages 145–153, 1993.

[6] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[7] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 145–154, New York, NY, USA, 1990. ACM.

[8] Wenzel Jakob and Steve Marschner. Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.*, 31(4):58:1–58:13, July 2012.

[9] Thorsten-Walther Schmidt, Jan Novak, Johannes Meng, Anton S. Kaplanyan, Tim Reiner, Derek Nowrouzezahrai, and Carsten Dachsbacher. Path-space manipulation of physically-based light transport. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2013)*, 32(4), August 2013.

[10] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[11] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000.

[12] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[13] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.

[14] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 94–105, New York, NY, USA, 1998. ACM.

[15] Donghui Yan, Ling Huang, and Michael I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 907–916, New York, NY, USA, 2009. ACM.

[16] Marie desJardins, James MacGlashan, and Julia Ferraioli. Interactive visual clustering. In *Proceedings of the 12th International Conference on Intelligent User Interfaces*, IUI '07, pages 361–364, New York, NY, USA, 2007. ACM.