PEDECIBA INFORMÁTICA

Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

TESIS DE MAESTRÍA EN INFORMÁTICA

Efficient Representations of Large Radiosity Matrices

José Pedro Aguerre
jpaguerre@fing.edu.uy

19 de diciembre de 2016

Dr. Eduardo Fernández	Director académico y tutor de tesis
Dr. Benoit Beckers	Co-tutor de tesis
Dr. Gustavo Patow	Revisor
Dr. Javier Baliosián	
Dr. Álvaro Pardo	

Resumen

La ecuación de radiosidad tiene por objetivo el cálculo de la interacción de la luz con los elementos de la escena. Ésta se puede expresar como un sistema lineal, cuya resolución ha derivado en el desarrollo de diversos métodos gráficos para satisfacer propósitos específicos. Por ejemplo, en problemas inversos de iluminación para geometrías estáticas, se debe resolver la ecuación de radiosidad miles de veces. Además, este cálculo debe considerar muchos (infinitos) rebotes de luz, si se quieren obtener resultados precisos de iluminación global. Entre los métodos desarrollados, se destacan aquellos que generan aproximaciones u otras representaciones de la matriz de radiosidad, debido a que su almacenamiento requiere grandes cantidades de memoria. Algunos ejemplos de estas técnicas son la radiosidad jerárquica, el refinamiento progresivo y la radiosidad basada en wavelets. Si bien estos métodos son eficientes en cuanto a memoria, pueden ser lentos cuando se requiere el cálculo de muchos rebotes de luz, debido a su naturaleza iterativa. Recientemente se han desarrollado métodos eficientes para la resolución directa de la ecuación de radiosidad, basados en el pre-cómputo de la inversa de la matriz de radiosidad. En estos casos, el desafío consiste en reducir los requerimientos de memoria y tiempo de ejecución para el cálculo de la matriz y de su inversa.

El principal objetivo de la tesis consiste en explotar propiedades específicas de ciertos problemas de iluminación para reducir los requerimientos de memoria de la ecuación de radiosidad. En este contexto, se analizan dos casos diferentes. El primero consiste en hallar la radiosidad para escenas con alta coherencia espacial, tal como ocurre en algunos modelos arquitectónicos. El segundo involucra escenas con un elevado factor de oclusión entre parches.

Para el caso de alta coherencia espacial, se presenta un nuevo método de factorización de matrices que es computacionalmente eficiente, y que genera aproximaciones cuyo error es configurable. Está basado en el uso de múltiples descomposiciones en valores singulares (SVD) junto a una curva de recubrimiento espacial, lo que permite explotar la coherencia espacial. Esta técnica acelera la factorización de matrices que entran en memoria, y permite trabajar con matrices que no entran en memoria, recorriéndolas una única vez. En el análisis experimental, el método presentado es aplicado a escenas de hasta 163 mil parches. Luego de una étapa de precómputo, se logra resolver la ecuación de radiosidad en tiempos interactivos, para geometrías estáticas e infinitos rebotes.

Para el problema de alta oclusión, se utilizan modelos de ciudades. En este caso, se aprovecha la baja densidad de la matriz de radiosidad, y se propone una técnica para el cálculo aproximado de su inversa. En este cálculo, los elementos cercanos a cero son eliminados. La técnica es aplicada a la simulación de la luz natural en ambientes urbanos compuestos por hasta 140 mil parches.

Palabras clave: radiosidad, iluminación global en tiempo real, factorización de matrices, matrices dispersas, intercambio de radiación en ciudades, luz natural.

Abstract

The radiosity equation can be expressed as a linear system, where light interactions between patches of the scene are considered. Its resolution has been one of the main subjects in computer graphics, which has lead to the development of methods focused on different goals. For instance, in inverse lighting problems, it is convenient to solve the radiosity equation thousands of times for static geometries. Also, this calculation needs to consider many (or infinite) light bounces to achieve accurate global illumination results. Several methods have been developed to solve the linear system by finding approximations or other representations of the radiosity matrix, because the full storage of this matrix is memory demanding. Some examples are hierarchical radiosity, progressive refinement approaches, or wavelet radiosity. Even though these methods are memory efficient, they may become slow for many light bounces, due to their iterative nature. Recently, efficient methods have been developed for the direct resolution of the radiosity equation. In this case, the challenge is to reduce the memory requirements of the radiosity matrix, and its inverse.

The main objective of this thesis is exploiting the properties of specific problems to reduce the memory requirements of the radiosity problem. Hereby, two types of problems are analyzed. The first problem is to solve radiosity for scenes with a high spatial coherence, such as it happens to some architectural models. The second involves scenes with a high occlusion factor between patches.

For the high spatial coherence case, a novel and efficient error-bounded factorization method is presented. It is based on the use of multiple singular value decompositions along with a space filling curve, which allows to exploit spatial coherence. This technique accelerates the factorization of in-core matrices, and allows to work with out-of-core matrices passing only one time over them. In the experimental analysis, the presented method is applied to scenes up to 163K patches. After a precomputation stage, it is used to solve the radiosity equation for fixed geometries and infinite bounces, at interactive times.

For the high occlusion problem, city models are used. In this case, the sparsity of the radiosity matrix is exploited. An approach for radiative exchange computation is proposed, where the inverse of the radiosity matrix is approximated. In this calculation, near-zero elements are removed, leading to a highly sparse result. This technique is applied to simulate daylight in urban environments composed by up to 140k patches

Keywords: radiosity, real-time global illumination, matrix factorization, sparse matrices, urban radiation exchange, daylighting.

Contents

List of Figures

List of Tables

Agradecimientos

En primer lugar quisiera agradecer a Eduardo Fernández por su incansable dedicación para que mi trabajo saliera adelante. Por poner su tiempo y esfuerzo a disposición para mi bienestar, que se ha visto reflejado en mis aprendizajes. Ha estado directamente involucrado en todo lo relacionado a mi tesis: los preparativos iniciales, las horas de trabajo en conjunto, los detalles técnicos cuidadosamente explicados, la redacción de artículos y del documento final, siempre sin dejar el humor y la amistad de lado.

En segundo lugar agradezco a mis colegas del Instituto de Computación por haberme acompañado en estos dos años de trabajo. En especial quisiera mencionar a mis amigos del MuDi y del Centro de Cálculo por estar siempre cerca en los momentos importantes de mi tesis.

Agradezco también a la Universidad de la República y a la Agencia Nacional de Investigación e Innovación por haber financiado mi trabajo. También considero importante el apoyo económico paralelo recibido por otras dos instituciones en las que estuve involucrado laboralmente estos años: la Universidad Católica del Uruguay y NVIDIA.

Por otro lado, quisiera realizar algunos agradecimientos a nivel personal. A Dani, Mamá, Papá y Javier por haber caminado conmigo no solo estos dos años, sino también durante mi carrera universitaria. Por haber aceptado y apoyado mi idea de seguir en formación, y por haber colaborado en descubrir mi vocación académica. Agradezco también a mis amigos y amigas que tanto aprecio, en especial a mi comunidad de vida y a las Hermanas Esclavas, por querer siempre lo mejor para mi y para los míos.

Por último, una mencion especial para aquel que es mi guía en la lucha por un mundo mas justo, de paz y amor, con los ojos fijos en el Reino: Jesús.

José Pedro Aguerre

Chapter 1

Introduction

Computers have helped simulating many physical phenomena since their beginning. In fact, most of the advances on computer systems have been driven by the goal of modeling and approximating different types of nature's events [Freed and Ishida, 1995]. In this matter, computer graphics techniques have focused on simulating the human vision and light interaction with objects, which behave according to their physical properties [Foley et al., 1994]. Global illumination (GI) methods try to consider the light indirectly reflected and transmitted by components of the scene, with the objective of generating photo-realistic images, or physically-realistic data results [Ritschel et al., 2012]. Moreover, the computational efficiency of GI algorithms is an important subject of study, since the models to handle are often large, and light-object interaction is costly to simulate.

Historically [Ritschel et al., 2012], there have been two main approaches to solve the GI problem: ray-based techniques and radiosity methods. Both are designed to solve the indirect illumination by modeling light interaction in the scene, but they are well-suited for different types of problems. For example, ray tracing engines can simulate glossy reflections efficiently, while radiosity methods are designed for diffuse surfaces. Computing diffuse reflections using ray tracing is computationally expensive, as well as calculating glossy illumination using radiosity becomes inefficient.

In recent years, several techniques have been developed based on both approaches, even mixing them to accomplish more general solutions [Bouatouch and Bouville, 2013]. Nevertheless, it is still a good strategy to analyze every specific problem before choosing the technique to be used, in order to obtain realistic results in short execution times. In addition, developing new GI algorithms, or implementing existing schemes with a specific problem in mind, can be a good option if the problem shows properties that can be exploited. In this thesis, we study two variants for the classic formulation of the radiosity algorithm that take advantage of specific features of the problems to solve.

The radiosity method [Goral et al., 1984] is a GI technique for scenes with lambertian surfaces. It has been applied in many areas such as computer animation, architectural design and heat transfer. The term "radiosity" stands for the emitted and reflected energy leaving a surface. Its mathematical formulation leads to an integral equation that can be solved through the use of a finite element methodol-

ogy. The discretization of the linear problem brings new obstacles to the strategy, since the error of the representation must be minimized. Usually, several thousand polygons (n "patches") are needed to represent the geometry of a scene precisely. In the classic formulation, the linear system is represented by a $n \times n$ matrix called the radiosity matrix. This leads to a computational complexity of $O(n^3)$ and memory requirement of $O(n^2)$, which becomes an important restriction for medium-to-large problems. Therefore, other methods have been proposed to approximate radiosity solutions using less memory and operations [Cohen et al., 1986, 1988, Hanrahan et al., 1991].

1.1 Challenges and objectives

Despite the previous fact, the computation of the classical radiosity linear system has advantages for some types of problems. For example, in problems where the geometry of the scene is static and only the emission changes, the illumination can be computed quiet efficiently after a pre-computation stage. In this case, the challenge is to reduce the memory requirements of the radiosity matrix by finding an alternative representation. This challenge becomes the main objective of this thesis: exploiting the properties of specific problems to reduce the memory requirements of the radiosity matrix.

After the radiosity matrix is approximated, the illumination of the scene is calculated by solving the linear system of equations. Among other techniques, this can be achieved using iterative methods (such as Jacobi or Gauss-Seidel [Cohen et al., 1993]), or by inverting the matrix. In the iterative approach, each iteration represents a new bounce of light in the scene, which allows to control the precision of the final solution. More iterations lead to more realistic approximations, but require larger execution times. On the other hand, the inverse of the radiosity matrix allows to find solutions with infinite bounces with just one matrix-vector product. Inverting this matrix can be another challenging pre-computation operation, but it may be worth if the problem allows to do it efficiently. This brings the second objective: finding reduced representations of the radiosity matrix that allow to approximate its inverse efficiently.

1.2 Specific problems to solve

Given the previous motivations, two different problems are analyzed and solved in this thesis. In both cases, the radiosity matrix is represented in a compressed manner, and its inverse is approximated to compute real-time radiosity solutions for static scenes and dynamic emissions. The first problem is to solve radiosity for scenes with a high spatial coherence, such as architectural models. The second involves scenes with a high occlusion factor between patches. In particular, urban environments are studied and daylight simulation is performed.

Scenes with a high spatial coherence. In this kind of scenes, matrices involved in radiosity calculations have a low numerical rank [Baranoski et al., 1997], because close patches have a high probability of being affected similarly by the rest of

the scene. This fact enables the application of factorization techniques to compute low rank approximations that can be stored in main memory. For this purpose, a hierarchical factorization (HF) technique for radiosity calculations is proposed. This technique implements a divide-and-conquer strategy based on the calculation of multiple singular value decomposition (SVD factorization [Golub and Van Loan, 1996]). In addition, the sorting of rows/columns of the matrix by similarity is exploited to speed-up the technique. For this purpose, we use the Z-order curve [Morton, 1966] over the patches of the scene, because each patch is associated to a row/column in the matrix. This method is intended to accelerate the factorization of in-core and out-of-core matrices, requiring just one pass. Using this method, the inverse of the radiosity matrix can be efficiently approximated, reducing the memory and time resources needed to compute radiosity with infinite bounces. In the experimental analysis, the method is applied to scenes up to 163K patches.

Scenes with a high occlusion factor between patches. Two patches completely occluded do not exchange energy directly. If this property is satisfied for most pair of patches on a scene, then the radiosity matrix is a sparse matrix. Since this property is present in City models [Wimmer and Bittner, 2015], an approach for radiative exchange computation that can approximate the inverse of the radiosity matrix is proposed. The problem is formulated as a Neumann series [Golub and Van Loan, 1996] and the inverse is approximated by eliminating unimportant terms. The application of this technique on different kinds of urban models shows that an accurate approximation of the inverse of the radiosity matrix can be computed. This allows to calculate radiosity for models composed of 140k patches efficiently.

1.3 Thesis structure

The rest of the thesis is organized as follows. Chapter 2 introduces the state of the art and related works. First, the global illumination problem is defined, and the radiosity method is presented thoroughly. The most used techniques for radiosity calculations are described here, along with some aspects about the low-rank properties of the radiosity matrix, and the use of sparse matrices. Then, the matrix factorization problem and some decomposition strategies are shown. Finally, the field of radiative methods for urban environments is presented. Chapter 3 defines a technique for reducing and inverting the radiosity matrix in scenes with a high spatial coherence. A factorization algorithm is described, and several studies, such as error bound and computational complexity, are developed. Next, the experimental analysis is reported, including studies on the performance of the algorithm and a comparison with other factorization techniques. In this section, radiosity results are computed using the proposed method. Chapter 4 proposes a technique for radiosity calculations in scenes where there is a high occlusion factor between patches. In particular, urban models are used because they satisfy this condition. A daylight simulation method is used to define a realistic illumination problem for the experimental analysis section. The sparsity results for the radiosity matrix and its inverse, associated to several cities are shown, and skylighting is computed to test the algorithm. Finally, Chapter 5 is dedicated to the conclusions and main lines of future work.

1.4 Table of symbols

symbol	definition
HF	Hierarchical Factorization method
n	number of patches
k, r	matrix rank
σ_{max}	largest singular value (σ_{max}=σ_1)
σ_i	i^{th} singular value ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_i \geq ... \geq 0$)
ε	expected error
ε_p	expected error of parent node
ε_{ch}	expected error of child node
μ, σ	mean and standard deviation
u, v	left and right singular vectors
v_{max}	right singular vector associated to σ_{max}
B	radiosity vector
$\widetilde{B}$	approximation to the radiosity vector
E	emission vector
$\mathbf{I}$	$n \times n$ identity matrix
$\mathbf{I}_k$	$k \times k$ identity matrix
$\mathbf{R}$	diagonal matrix with reflectivity indexes
$\mathbf{F}$	form factors matrix
$\mathbf{RF}$	matrix result of the product $\mathbf{R} \cdot \mathbf{F}$
$(\mathbf{I} - \mathbf{RF})$	radiosity matrix
$\mathbf{M}$	$(\mathbf{I} - \mathbf{RF})^{-1}$
$\widetilde{\mathbf{M}}$	approximation of $\mathbf{M}$
$\mathbf{U}_k$, $\mathbf{V}_k$	$n \times k$ matrices, $\mathbf{U}_k \mathbf{V}_k^T \approx \mathbf{RF}$
$\mathbf{Q}_k$	$\mathbf{U}_k \mathbf{D}_k$
$\mathbf{D}_k$	diagonal matrix with $\sigma_1 \ldots \sigma_k$
$(\mathbf{A}_1 \vert \mathbf{A}_2)$	matrix partition in two column blocks

Table 1.1: Symbol notation and meaning.

1.5 Table of quantities

quantity	meaning	unit
radiant flux	Radiant energy emitted, reflected, transmitted or received, per unit time.	W
radiosity	Radiant flux emitted, reflected and transmitted by a surface per unit area.	Wm^{-2}
irradiance	Radiant flux received by a surface per unit area.	Wm^{-2}
radiant intensity	Radiant flux emitted, reflected, transmitted or received, per unit solid angle.	Wsr^{-1}
radiance	Radiant flux emitted, reflected, transmitted or received by a surface, per unit solid angle per unit projected area.	$Wm^{-2}sr^{-1}$
luminance	Luminous power per unit solid angle per unit projected source area.	$cd\,m^{-2}$
illuminance	Luminous power incident on a surface.	$cd\,sr\,m^{-2}$

Table 1.2: Quantities used in the thesis, along with the correspondent units [Vollmer and Möllmann, 2010]. W is watts, and m is meters. sr is steradian, which is the unit of solid angle (square radian), and cd is candela.

Chapter 2

State of the Art

This chapter introduces an overview of the most important areas and works related to this thesis, which includes defining the Global Illumination problem, the radiosity method, studying some techniques for matrix factorization, and urban radiative methods.

2.1 Global Illumination

In computer graphics, an illumination model calculates the color of a point in space by studying its light emission and the incoming light after reflection from and/or transmission through other surfaces in the scene [Foley et al., 1994]. When the model takes only direct lighting into account, it is called a *local illumination* model. On the other hand, *global illumination* (GI) is the light indirectly reflected and transmitted by other components on the scene.

In general, light arriving at a point A from some other point B is considered as "direct" if it is generated in B. If the arriving light was reflected or transmitted before arriving at A, then it is considered as "indirect" illumination [Ritschel et al., 2012].

Light in real-world environments is composed of both direct and indirect sources. This creates visual phenomena such as color bleeding, reflections, crepuscular rays, and caustics [Ritschel et al., 2012], whose computational modeling is a challenging problem. It is often useful to develop simplified models, customized algorithms and data structures that help in the simulation of global illumination.

It is important to classify global illumination algorithms into view-dependent and view-independent. The former discretize the view window into points where the illumination is computed, usually by an independent (and hence parallel-friendly) calculation. Examples of view-dependent GI algorithms are *Ray Tracing* [Kuchkuda, 1988] and *Instant Radiosity* [Keller, 1997]. On the other hand, view-independent GI techniques discretize the scene and process it in such way that the illumination can be computed at any point, without considering the viewing direction. Examples of this are *Radiosity* [Cohen et al., 1986] and *Photon Tracing* [Jensen, 2001]. View-dependent algorithms are well suited for specular surfaces, while view-independent are specifically built for working with diffuse materials.

According to Glassner [1994], the most general equation that relates the global interaction between light and matter is the "full radiance equation", which takes into account phenomena such as photon transport in scattering media, polarization, phosphorescence and fluorescence.

Kajiya [1986] proposed a simple approach to model the GI problem, called the "rendering equation" (Eq. 2.1). This equation is time invariant, where the media has a homogeneous refractive index, and light wavelength or polarization properties have no influence.

$$I(x, x') = g(x, x') \left[\varepsilon(x, x') + \int_S \rho(x, x', x'') I(x', x'') dx'' \right] \tag{2.1}$$

$I(x, x')$ represents the radiance (measured in $W\, m^{-2}\, sr^{-1}$, see Table 1.2) going from x' to x (Figure 2.1). The geometric term $g(x, x')$ is equal to 0 when x and x' are occluded from each other, and $1/r^2$ otherwise, being r the distance between them. $\varepsilon(x, x')$ and $\rho(x, x', x'')$ are related to the intensity of light, emitted from x' to x, and scattered from x'' to x by a patch of surface at x', respectively.

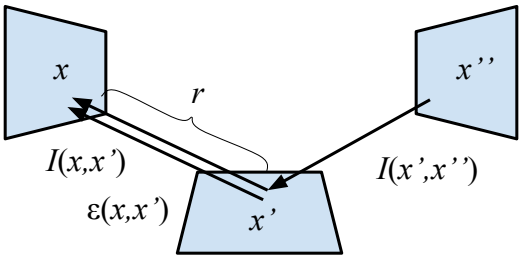


Figure 2.1: Some elements of the rendering equation.

The illumination problem involves a wide range of physical phenomena: direct and indirect lighting, ambient occlusion, natural illumination, single and multiple bounces, caustics, diffuse and glossy bounces, and scattering. Simulating all these effects using one single global illumination algorithm becomes a difficult task, where time rates are always a restriction. Therefore, several methods were proposed to solve the rendering equation, each one addressing different problems and phenomena. Despite this fact, there are some concepts which are usually solved by the most common GI algorithms, such as multiple light bounces, multiple light sources, and different materials. These variables affect directly on the execution times.

Ritschel et al. [2012] establishes that the classical methods to compute global illumination are: Finite elements (e.g., radiosity) [Goral et al., 1984], Monte Carlo ray tracing [Kajiya, 1986], photon mapping [Jensen, 1996], instant radiosity [Keller, 1997], many-light-based global illumination [Hašan et al., 2007, Walter et al., 2005], point-based global illumination [Christensen, 2008, Ritschel et al., 2009], discrete ordinate methods [Chandrasekar, 1950], and precomputed radiance transfer [Sloan et al., 2002]. Before getting a deeper insight into the radiosity algorithm, some state-of-the-art GI algorithms are briefly described next.

Ray tracing is one of the most used techniques for GI. It has been utilized

since the beginnings of computer graphics [Appel, 1968]. It offers a solution for computing visibility in a scene, applying very simple geometry concepts to simplify the rendering equation. The main idea of this algorithm consists on tracing rays (which are represented as lines in the 3D space) in order to follow light or view-directions, which combined with some local illumination techniques (such as the Phong reflection model) allow to simulate a wide range of light phenomena. For example, the Whitted ray tracing algorithm [Whitted, 1979] is a view-dependent technique that consists on tracing rays from the camera point through the pixels of the view plane. Each ray is intersected with the scene, in order to compute its corresponding color. First, shadow rays are traced from the intersected point to the light sources, in order to determine if the area is in shadow. Then, the local illumination (ambient+diffuse+specular) is computed, and other rays are recursively casted symmetrically to simulate reflections. The final color is composed of the local illumination and the reflections. Alternatively, refracted rays can be added to the algorithm in a very simple manner, following Snell law to calculate the direction changes in rays. The principal components of the algorithm are shown in Fig. 2.2.

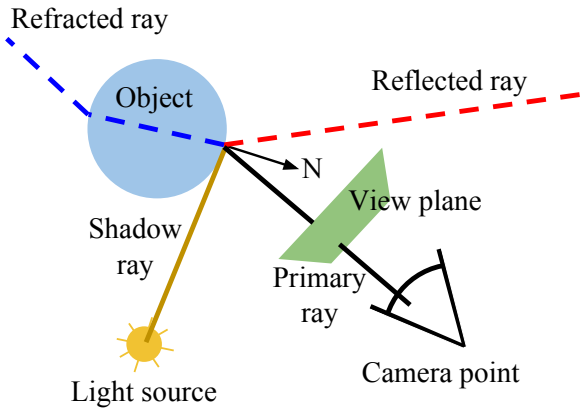


Figure 2.2: Ray tracing components.

The previous scheme is very simple, but it is not able to compute other phenomena such as indirect illumination coming from diffuse surfaces, glossy reflections or caustics. For this purpose, Monte Carlo methods were combined with ray tracing to stochastically consider every possible light path. This technique, called Monte Carlo Ray Tracing [Jensen et al., 2003], is based on casting random rays from the intersected points to accumulate the radiance contribution.

Another ray-based method that has been subject of development in recent years is photon mapping, introduced by Jensen [1996]. It is a two pass GI method based on the concept of photon, which are packets of energy emitted from light sources to the scene. These photons are treated as infinitesimal points that travel in straight lines. In this way, the technique starts by casting rays in different directions from the light sources, simulating reflections and other light phenomena, and storing the hit points in a map structure, called photon map. Then, at each visible point, the illumination is computed by operating with the stored photons that are near. The photon map is used to accelerate these operations. This first pass is view-independent, while the

second is view-dependent.

On a different path than previous algorithms, instant radiosity [Keller, 1997] is a very efficient GI technique to generate photo-realistic image synthesis. In spite of its name, this method is neither based on radiosity nor a finite element methodology. It takes this name because it is able to simulate the radiosity algorithm using fewer resources. The technique works in two main steps. First, a small number of photons are traced from the light sources into the scene. The hit points are called Virtual Point Lights (VPL), and are used to generate a first diffuse radiance approximation using a Quasi-random walk (similar to Monte Carlo). Secondly, the scene is rendered several times using a hardware accelerated z-buffer technique with shadow maps, each one with a different VPL as the unique light source. The output image is generated by the accumulation of rendering results. The superposition of the primary shadows (generated by the shadow maps) simulates the effect of light bounces on lambertian surfaces, in such way that the result looks similar to radiosity results.

2.2 The Radiosity Method

The radiosity method [Goral et al., 1984] is a technique which allows to compute global illumination on scenes with Lambertian surfaces. It has been applied in many areas of design and computer animation [Dutre et al., 2006]. The term "radiosity" stands for energy of light leaving a surface per unit area (W/m^2), composed of emitted and reflected light. A lambertian surface is an ideally diffuse surface, whose apparent brightness is the same regardless of the observer's angle of view (Figure 2.3). This property is also known as isotropic luminance, where the light intensity follows Lambert's cosine law.

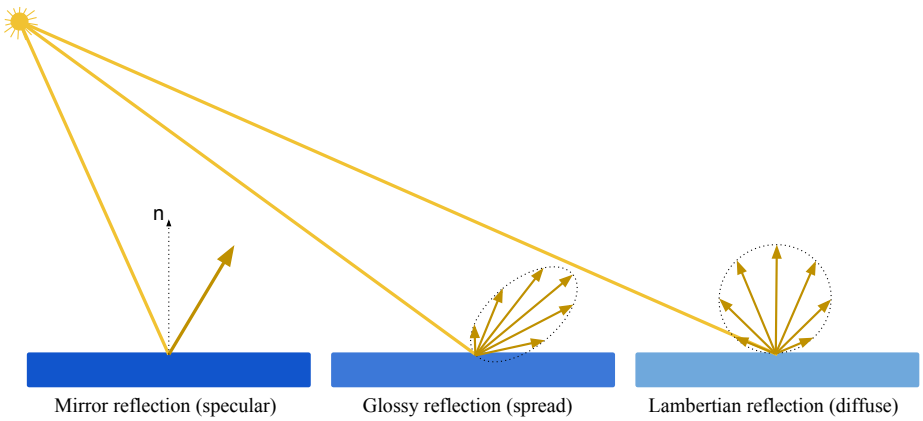


Figure 2.3: Different kinds of reflections.

Lambertian surfaces simplify the ρ term in the rendering equation (Eq. 2.1), which is transformed into the "radiosity equation" (Eq. 2.2).

$$B(x) = E(x) + \rho(x) \int_S B(x')G(x,x')dA' \tag{2.2}$$

$B(x)$ and $E(x)$ are the radiosity and emission values at point x, $\rho(x)$ is the reflectivity factor of the material in x, and $G(x, x')$ is a geometric factor that depends on the normal vectors at x' and x, and on the correspondent distance between the points [Cohen et al., 1993].

The continuous radiosity equation can be discretized through the use of a finite element methodology. The scene is discretized into a set of patches, leading to express the problem using the following set of linear equations:

$$B_i = E_i + R_i \sum_{j=1\ldots n} B_j \mathbf{F}(i,j) \quad , \quad \forall i \in \{1 \ldots n\}$$

This set of linear equations is expressed algebraically in Equation 2.3 (known as the Discrete Radiosity Equation).

$$(\mathbf{I} - \mathbf{RF})B = E, \tag{2.3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{R}$ is a diagonal matrix containing the reflectivity index of each patch, B is the radiosity vector to be found (W/m^2), and E is the emission vector. $\mathbf{F}(i,j)$ is a number between 0 and 1 expressing the form factor between patch i and j. This value indicates the fraction of the light power emitted by i going to j. Therefore, the form factor matrix is a $n \times n$ matrix, where n is the number of patches in the scene.

2.2.1 The form factors between patches of a scene

$\mathbf{F}(i,j)$ is called form or view factor, and it is calculated using the formula in Eq. 2.4, being A_i and A_j two sufaces. $G(x_{A_i}, x_{A_j})$ incorporates the geometric and visibility relationship between the two regions as it affects radiant energy transport [Cohen and Wallace, 2012].

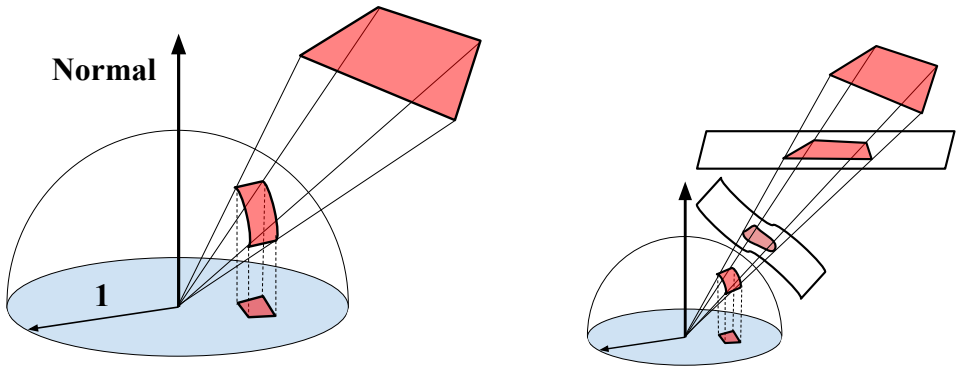
$$\mathbf{F}(i,j) = \frac{1}{A_i} \int_{A_i} \int_{A_j} G(x_{A_i}, x_{A_j}) dx_{A_j} dx_{A_i} \tag{2.4}$$

The concept of form factor was formulated in the field of thermodynamics (heat transfer). Its calculation was initially performed analytically, even before the use of computers. For convex surfaces, energy leaving a surface will not hit it again later, because radiation travels in straight lines. Therefore, $\mathbf{F}(i,i) = 0$. This property is not satisfied for concave surfaces, which leads to $\mathbf{F}(i,i) > 0$. In this thesis, we assume every surface to be convex.

Computing form factors is relatively complex due to the presence of a double integral (Eq. 2.4), and because the visibility between each patch needs to be calculated. This determines the obstruction between points in the scene, which is fundamental to predict their energy exchange. Also, given n patches, the amount of form factors to process is n^2, leading to computational and memory restrictions.

The form factors compose the $\mathbf{F}$ matrix. Depending on the scene, this can be dense or sparse. A scene where each patch sees few others, generates sparse $\mathbf{F}$. On the other hand, when each patch sees a big portion of the scene, a dense $\mathbf{F}$ matrix is generated.

The geometric argument used to compute form factors is called Nusselt analog, represented at Fig. 2.4. This expresses the relation between a differential area and a surface, without considering visibility. The Nusselt analog proposes the geometrical factor to be the same as projecting the area of element j onto the surface of a unit hemisphere centered at element i, and then projecting the result onto a unit circle in the plane of element i. The form factor is equal to the fraction of the unit circle covered by this projection [Cohen and Greenberg, 1985]. This construction leads to the following property: the sum of the elements of each row of **F** is less or equal than 1. Also, it is shown how different shaped surfaces can have the same form factor with the differential area.



(a) Nusselt analog.

(b) Areas with the same form factor.

Figure 2.4: The form factor from the differential area to an element, calculated using a projection onto the hemishpere and another onto the disk.

Computing F

The most expensive part of calculating form factors is determining visibility between two given patches in a scene, which, if implemented naively, has a complexity of $O(n)$ for each pair. Teller and Hanrahan [1993] present several algorithms for the visibility problem. The other necessary operation is to compute Eq. 2.4, which depends highly on obstruction: an element on the scene "sees" only part of other element. This fact makes the calculation even more complex.

According to Cohen et al. [1993], the numerical solutions to the form factor problem are categorized in two approaches: differential-area to area solutions, or straight area-to-area methods. In the former, the area of the source element that receives energy from the scene is considered infinitesimal, while in the latter depends on the shape and size of it. Most of the related works focus on the differential-area approach, since it is simpler to implement and provides a general solution, independent of the polygonal shape of the patches.

One of the most utilized area-to-area approaches is based on Monte Carlo integration. Given two patches i and j, k pair of points $[x_i, x_j]$ are randomly selected, where x_i belongs to patch i and x_j to patch j. For each pair, the form factor is calculated using Eq. 2.4, where the visibility part of the geometrical term is evaluated

by tracing k rays. Each ray is traced between the pair $[x_i, x_j]$, and the intersection with the scene is computed. This method becomes more accurate as k grows, but this also means a performance decay. Other techniques are based on hierarchical subdivisions or contour integrals [Goral et al., 1984].

On the differential-area to area side, several methods have been proposed. Simpler techniques use area sampling strategies, where several rays are casted from the source point into a recipient polygon and the resulting intersections are used to solve Eq. 2.4. The casted rays can be uniformly distributed into the element [Wallace et al., 1989], or can follow a random Monte Carlo distribution [Wang, 1992].

Following the Nusselt analog, the most used techniques are based on sampling the hemisphere around the source point. This sampling can be implemented in several ways. For example, casting rays in every direction and retrieving the intersections. Another approach is to project the scene surfaces into planes instead of a hemisphere, which ensures the same form factor value (as seen in Fig. 2.4b), but is simpler to compute using specialized rasterization hardware.

Cohen and Greenberg [1985] propose the hemi-cube algorithm to compute the form factors of any element in a scene. This method uses the Z-buffer algorithm as a simple solution to the visibility problem, which is the most commonly used strategy for rendering [Catmull, 1974]. This adds the information of the distance between the center of projection and the projected polygon to each rendered pixel. If more than one element of the scene is to be drawn at the same region, only the nearest is rendered and the depth is updated for that pixel. The memory array used to store this information is called the depth buffer.

The hemi-cube algorithm computes the form factors between patch i and the rest of the elements of the scene (this is i-th row of $\mathbf{F}$). This is accomplished by rendering five projections of every patch onto the five surfaces of a half cube, using the Z-buffer technique (see Fig. 2.5). At the end of the process, each pixel of the hemi-cube contains the information of visible patches. In particular, the color of each pixel denotes the corresponding rendered patch. The final form factor $\mathbf{F}_{ij}$ is equal to the proportion of pixels covered by polygon j in the hemi-cube centered at patch i.

This algorithm has become very successful and it is widely used in radiosity engines. It is usually implemented using graphics hardware, which runs the Z-buffer technique natively.

The main problem of the hemi-cube is the aliasing that can be generated when working with pixels. For example, if an element whose size is smaller than a pixel needs to be rendered, it will not be taken into account. This element could be emitting great amounts of light, making it important to the illumination calculation. For this purpose, it is fundamental to use an image resolution that corresponds to the problem. Another problem is the need to generate five independent projections, which leads to computational restrictions.

In this way, another approach is the single-plane form factor algorithm developed by Sillion and Puech [1989]. It projects elements onto a single plane above the differential area using an adaptive hidden surface algorithm. As shown in Fig. 2.6, this technique can adaptively subdivide the view-plane for large or small elements, and thus avoid some of the sampling problems of the hemi-cube. The main disad-

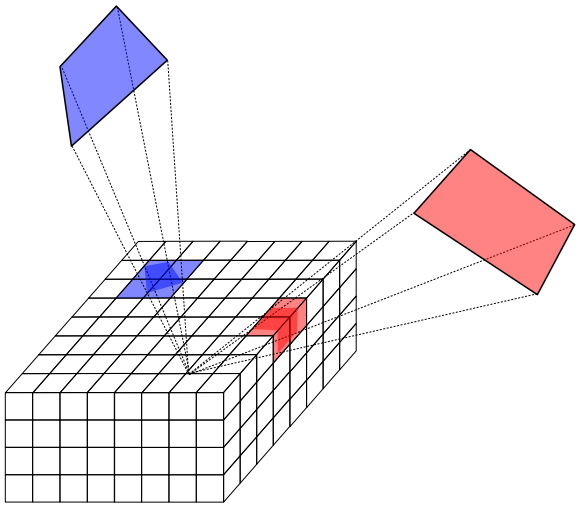


Figure 2.5: The hemi-cube algorithm.

vantage of this method is that it misses elements near the horizon. However, it is not strange that these elements contribute very little to the overall radiosity, due to the cosine dependence of the form factor.

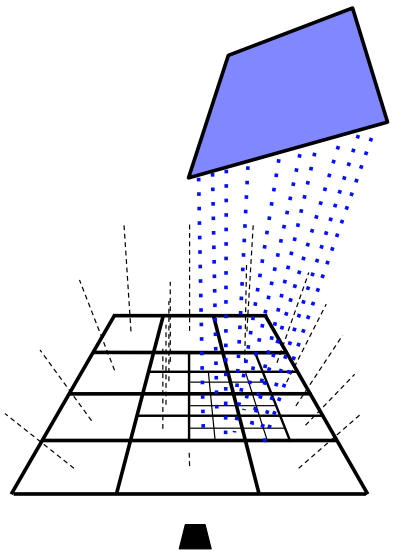


Figure 2.6: The single-plane algorithm.

2.2.2 Solving the Discrete Radiosity Equation

There are several strategies used to solve the radiosity linear system. As any linear system resolution methodology, they can be classified into direct and iterative methods.

In radiosity problems, direct methods focus on finding the inverse of the radiosity matrix: $\mathbf{M} = (\mathbf{I} - \mathbf{RF})^{-1}$, which represents a global transport operator relating the emitted light with the final radiosity of the scene, $B = \mathbf{M}E$. Originally, these were believed to be prohibitively expensive, except when applied to small problems, or when the system of equations is sparse. Recently, factorization strategies have been applied to the computation of the inverse, which have led to good results for scenes with big spatial coherence (meaning low numerical rank factorizations) [Fernández, 2009, Fernández and Besuievsky, 2012]. Refer to Sec. 2.2.4 for more information. Although these methods are still slower than the iterative methods for dynamic geometries (which implies the re-computation of $\mathbf{F}$), there are some problems where only the emission varies and, then, computing the inverse of the radiosity matrix is a good strategy. After a pre-computation step, thousands of emissions can be evaluated relatively simply, because only a matrix-vector operation is required per emission. On the other hand, $\mathbf{M}$ can also be approximated using iterative methods such as Neumann series (Sec. 2.2.5).

Another approach is to compute B by solving the linear system of equations iteratively, using methods such as Jacobi or Gauss-Seidel ([Cohen and Wallace, 2012]). Eq. 2.5 presents the radiosity iterative resolution. Each iteration adds the radiosity of a new light bounce to the global radiosity result. This iteration is repeated until $\|B^{(i+1)} - B^{(i)}\|$ is less than a given threshold.

$$B^{(i+1)} = \mathbf{RF}B^{(i)} + E \ , \ \text{where } B^{(0)} = E \tag{2.5}$$

The convergence of the iterative methods is ensured because the radiosity matrix $(\mathbf{I}-\mathbf{RF})$ is diagonally dominant [Cohen and Wallace, 2012]. This property is satisfied if the absolute value of the sum of the terms in each row (excepting the diagonal term) is less than or equal to the absolute value of the diagonal term [Golub and Van Loan, 2012]. In the radiosity matrix, the values in the diagonal are all ones, because $\mathbf{F}_{ii} = 0$. Since the sum of each row in $\mathbf{F}$ is less than or equal to one, and a realistic scheme implies each reflectivity index to be less than one, the sum of each row of $\mathbf{RF}$ is less than one. Thus, the matrix is diagonally dominant.

Other iterative techniques include the stochastic relaxation methods, which are basically a combination of traditional iterative methods and Monte Carlo. This strategy avoids the computation of form factors by casting rays from each surface, in order to retrieve the radiosity values of visible elements. For this, Malley's method is used [Malley, 1988]. More information about stochastic relaxation techniques for radiosity can be found at Cohen et al. [1993].

2.2.3 Alternative methods for the radiosity problem

Methods requiring the calculation of the $\mathbf{F}$ matrix can be computationally prohibitive for large problems. In particular, computing the form factors matrix is a heavy memory process. Stochastic relaxation methods are an example of radiosity resolution without computing $\mathbf{F}$. In this section, other methods that avoid this issue are presented.

Progressive refinement [Cohen et al., 1988] is one of the classic formulations,

where a selection of form factors is computed to shoot energy from significant surfaces. The elements are selected by its area and partial radiosity result. At each step, the radiosity of every element is updated, calculating how much energy arrives from the selected surface. This is repeated until the radiosity does not vary more than a given threshold (same stopping criteria than previous methods). This technique is very useful when partial results are needed, because it makes most of the progress at the beginning steps.

Another important contribution is the hierarchical radiosity (HR) method [Hanrahan et al., 1991], which subdivides each surface into a hierarchy of patches (as shown in Fig. 2.7) and links different sample points to patches at different levels in the hierarchy. This strategy substitutes the radiosity matrix by a hierarchical tree structure, avoiding the computation of parts of $\mathbf{F}$. HR then allows the surfaces to interact at a level in the hierarchy where all the interactions have an error less than some bound.

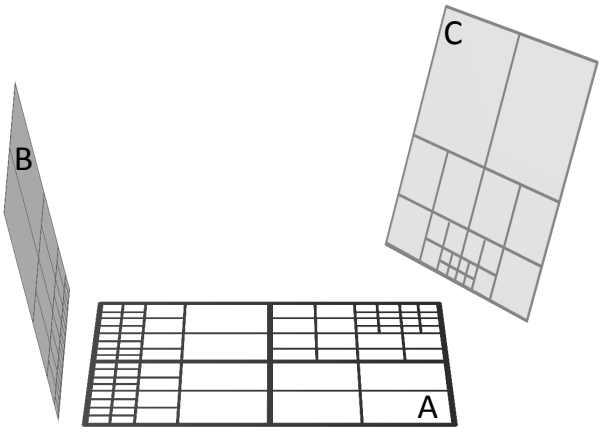


Figure 2.7: Subdivision of patches in HR. Patch A is affected by patch B and C.

HR constructs hierarchies by subdividing surfaces, but does not exploit a hierarchical grouping on existing surfaces. On this behalf, hierarchical radiosity with clustering [Garland et al., 2001, Smits et al., 1994] proposes a clustering algorithm that allows estimating energy transfers between collections of objects. This avoids the initial cost of linking patches on traditional HR.

The arrival of the GPU architecture helped to accelerate previous techniques and led to new hardware-specific approaches. Dong et al. [2007] propose an HR-based method for dynamic scenes. To speed-up the technique, they make several simplifying assumptions that reduce the radiosity equation. One of them is related to visibility, where partial occlusion is not taken into account. They group elements into “bins”, facilitating the creation of new links. Besides the pre-computation cost, where a geometric hierarchy is built and then reused at run-time, this algorithm is faster but less accurate than other methods. In other related works, Dachsbacher et al. [2007] also used links for antiradiance calculations, avoiding the visibility computation. Meyer et al. [2009] introduced the updating of the link structure at runtime. It is important to highlight that these techniques are well suited for

low-complexity scenes with moderate transformations.

2.2.4 Factorization of the RF matrix

Despite the obstacles and solutions described in previous sections, there are certain problems in which computing the inverse of the radiosity matrix is necessary. This matrix, $\mathbf{M} = (\mathbf{I}-\mathbf{RF})^{-1}$, provides important information about the global radiosity of the scene. Element $\mathbf{M}(i,j)$ contains the contribution of energy emitted by j to the final radiosity value in i. The product between a row $\mathbf{M}(i,:)$ and the emission vector E results in the final radiosity value for patch i.

For instance, when the geometry is fixed and only the emission varies, $\mathbf{M}$ is constant. Thus, computing an approximation of this matrix leads to the possibility of calculating several radiosity results per second with infinite light bounces. Even taking pre-computational costs into account, this strategy is completely valid. In this section, we study the possibility of inverting the radiosity matrix by obtaining a low-rank factorization of $\mathbf{RF}$.

There is a clear relation between the spatial coherence of a scene and the low numerical rank of the matrix $\mathbf{RF}$. References about low-rank properties of radiosity and radiance matrices can be found in Baranoski et al. [1997], Ashdown [2001], Hasan et al. [2007], and Fernández [2009]. This property allows to approximate $\mathbf{RF}$ by the product of two matrices $\mathbf{U}_k\mathbf{V}_k^T$, both with dimension $n\times k$ ($n\gg k$), without losing relevant information.

The memory required to store $\mathbf{U}_k$ and $\mathbf{V}_k^T$ is $O(nk)$, while for $\mathbf{RF}$ it is $O(n^2)$. When $n\gg k$, the memory savings can be significant, even allowing to store them in system memory for large scenes.

If $\mathbf{RF}$ is replaced by its approximation $\mathbf{U}_k\mathbf{V}_k^T$ in Eq. 2.3, Eq. 2.6 is obtained, where the radiosity B is now transformed into its approximation $\widetilde{B}$.

$$\left(\mathbf{I} - \mathbf{U}_k\mathbf{V}_k^T\right)\widetilde{B} = E \tag{2.6}$$

Using the Sherman-Morrison-Woodbury formula [Golub and Van Loan, 2012], the matrix $\left(\mathbf{I} - \mathbf{U}_k\mathbf{V}_k^T\right)$ is inverted:

$$\widetilde{B} = E + \mathbf{U}_k\left(\left(\mathbf{I}_k - \mathbf{V}_k^T\mathbf{U}_k\right)^{-1}\left(\mathbf{V}_k^T E\right)\right) \tag{2.7}$$

$\left(\mathbf{I}_k - \mathbf{V}_k^T\mathbf{U}_k\right)^{-1}$ is a $k\times k$ matrix, while $\mathbf{U}_k$ and $\mathbf{V}_k$ are $k\times n$. The product of a $k\times n$ matrix times a $n\times k$ matrix is $O(nk^2)$, which is what happens at $\left(\mathbf{V}_k^T\mathbf{U}_k\right)$. Hence, $O(nk^2)$ operations and $O(nk)$ memory are required to find $\widetilde{B}$ using Eq. 2.7.

Eq. 2.7 can be transformed as follows:

$$\begin{aligned} \widetilde{B} = &\quad E - \mathbf{Y}_k\left(\mathbf{V}_k^T E\right) \\ \text{where} \quad \mathbf{Y}_k = &\quad -\mathbf{U}_k\left(\mathbf{I}_k - \mathbf{V}_k^T\mathbf{U}_k\right)^{-1} \end{aligned} \tag{2.8}$$

$\mathbf{Y}_k$ is a $n\times k$ matrix. When the geometry and the reflectivity of the scene are static ($\mathbf{F}$ and $\mathbf{R}$ are static), $\mathbf{U}_k$, $\mathbf{V}_k$, and $\mathbf{Y}_k$ are computed only once in a pre-computation stage. After $\mathbf{Y}_k$ is found, the calculation of $\widetilde{B}$ is performed by two matrix $\times$ vector

multiplications and one vector addition. Both $\left(\mathbf{V}_k^T E\right)$, and its result times $\mathbf{Y}_k$, are $O(nk)$, and the addition is $O(n)$. Then, the computation of the radiosity solution has complexity $O(nk)$. This result is useful when the radiosity equation needs to be solved many times, and E is the only element modified in Eq. 2.6.

One example of a factorization technique applied to radiosity problems is the Low Rank Radiosity (LRR) method [Fernández and Besuievsky, 2012]. This method allows to factorize the **RF** matrix into one full matrix and one sparse matrix. This technique is relatively fast and can be computed by passing just one time over the whole matrix. The main problem of this technique is that two different scene meshes are needed, with two different granularity levels (*coarse* and *fine meshes*). The number of patches of these meshes define the dimensions k and n of the factorization matrices. For many scenes, it is not possible to define a coarse mesh with a small enough number of patches, useful for radiosity purposes. This can lead to large $\mathbf{Y}_k$ and $\mathbf{V}_k$ matrices.

Relation between the error in the factorization of RF and the radiosity error

Following Fernández [2009], when **RF** is substituted by **RF**+$\mathbf{\Delta RF}$, the relative error of the radiosity B (Eq. 2.3) is upper bounded:

$$\frac{\|\widetilde{B} - B\|}{\|B\|} \leq \frac{\|\mathbf{\Delta RF}\|}{1 - \|\mathbf{RF}\|}$$

Taking into consideration that $\|\mathbf{\Delta RF}\| \leq \varepsilon$, where ε is the expected error of the factorization, and using the 2-norm, another bound can be formulated:

$$\frac{\|\widetilde{B} - B\|_2}{\|B\|_2} \leq \frac{\varepsilon}{1 - \sigma_{max}} \tag{2.9}$$

where $0 < \sigma_{max} \leq 1$ is the largest singular value of **RF**. This bound is larger than ε. Experimental results show that in common situations the relative error of the radiosity results can be orders of magnitude smaller than ε.

2.2.5 Inverting the radiosity matrix via Neumann Series

Given an Operator $\mathbf{T}$, its Neumann series [Stewart, 1998] is a series of the form

$$\sum_{k=0}^{\infty} \mathbf{T}^k$$

The expression $\mathbf{T}^k$ is a mathematical notation that means applying the operator $\mathbf{T}$, k consecutive times. Suppose that $\mathbf{T}$ is an operator such that $\mathbf{T}^k$ converges to zero, and $\mathbf{I}$ is the identity operator. If the Neumann series converges, then $(\mathbf{I} - \mathbf{T})$ is invertible and its inverse is the series:

$$(\mathbf{I} - \mathbf{T})^{-1} = \sum_{k=0}^{\infty} \mathbf{T}^k = \mathbf{I} + \mathbf{T} + \mathbf{T}^2 + \mathbf{T}^3 + ...$$

This property can be used to calculate radiosity [Cohen and Wallace, 2012], by computing an approximate to the inverse of $(\mathbf{I} - \mathbf{RF})$ through l iterations:

$$\tilde{\mathbf{M}} = (\mathbf{I} - \mathbf{RF})^{-1} \approx \mathbf{I} + \mathbf{RF} + (\mathbf{RF})^2 + ... + (\mathbf{RF})^l$$

In this series, $(\mathbf{RF})^i$ contains the information of the i^{th} bounce of light between the surfaces in the scene. The main computational cost of this approach is the multiplication of matrices. Thus, if $\mathbf{RF}$ is sufficiently big, the method can be prohibitively expensive.

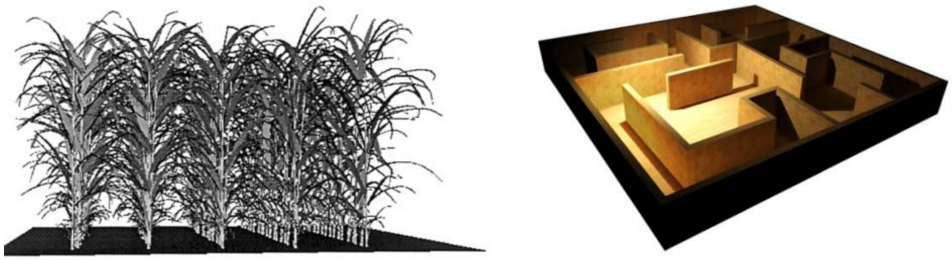
Kontkanen et al. [2006] use a variant of this method to compute a global transport operator for radiance calculations. This operator expresses the relationship between the converged and incoming incident lighting. In this process, the matrices are compressed using the following strategy: at each step, all the coefficients below a certain threshold are removed. This results in sparse matrices, which allow to reduce the memory requirements, and to speed up the calculations. The computation is stopped when all the coefficients in $(\mathbf{RF})^i$ are smaller than the threshold.

2.2.6 Using sparse matrices

A *sparse matrix* is any matrix with enough zeros that it pays to take advantage of them [Wilkinson, 1971]. Generally, using sparse representations allows to save time or memory (usually both) by exploiting the number of zeros. Furthermore, these kind of matrices are applied in problems where the use of full matrices is not possible due to memory restrictions.

The computer graphics area is closely related to geometry, which at the same time is linked to linear algebra. This relation forces the use of matrices for many graphics computation. In this way, sparse matrix techniques have been applied to computer graphics problems since the first non-full matrix techniques came out (for example [Chace, 1984, Light and Gossard, 1983]). Also, sparse representations have been used in the field of computer vision [Wright et al., 2010] and image processing [Cichocki and Amari, 2002]. Due to the intrinsic computational parallelism related to computer graphics and sparse matrices, the latter have taken advantage of graphics hardware [Bolz et al., 2003, Choi et al., 2010, Krüger and Westermann, 2003], which is one of the most important relations between both fields.

Beyond these facts, the use of sparse matrices in radiosity calculations is still a subject of study. Gortler et al. [1993] present the Wavelet Radiosity method, which is based on wavelet theory. Expressing the kernel operating in a radiosity function in a wavelet basis leads to a sparse approximation of it. On the other side, other works [Borel et al., 1991, Chelle and Andrieu, 1998, Goel et al., 1991] solve the radiosity problem using iterative methods (like Gauss-Seidel) taking advantage of the sparsity of the form factors matrix. This property is present in the tested scenes (plant canopies, see Fig. 2.8a), where there is a high occlusion level between distant polygons. On the other hand, as stated in the previous section, Kontkanen et al. [2006] also use a sparse representation of the form factors matrix, associated to the maze scene presented in Fig. 2.8b. This property is exploited to invert the radiosity matrix through the use of an iterative process.



(a) Plant canopy scene [Chelle and Andrieu, 1998]

(b) Maze scene [Kontkanen et al., 2006]

Figure 2.8: Two scenes that derive into sparse form factors matrices.

2.3 Spatial coherence

Coherence denotes similarity between entities. It is based on the principle of locality [Groller and Purgathofer, 1993], where "nearby" items have similar characteristics or attributes. Coherence properties have been exploited in the field of computer graphics [Badt, 1988, Crocker, 1984, Sutherland et al., 1974]. There are several types of data coherence, such as spatial, temporal or object coherence. These concepts have been studied since the beginnings of computer graphics, and are mainly inspired in the coherence notion used in physics [Beran and Parrent, 1974].

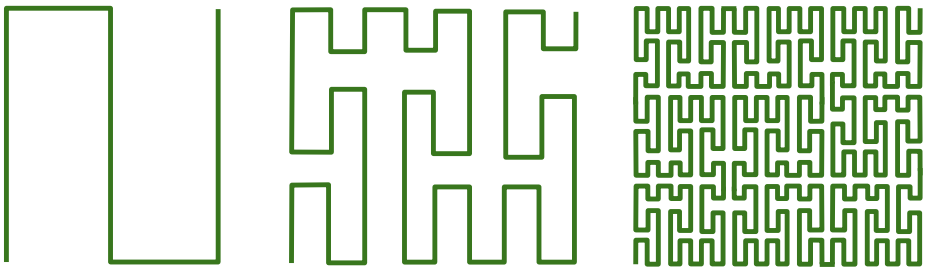
Spatial coherence denotes spatial homogeneity, and it is a direct consequence of a constant or slow variation in the relation between the disposition of objects or data [Groller and Purgathofer, 1993]. A model is normally composed of a list of objects or data. An object or data is said to be coherent if its own properties, such as shape, color, or disposition in the scene, vary smoothly in space. The coherence of a scene depends on the coherence and disposition of its components. These facts determine if the graphical or physical information derived from the scene will contain smooth or abrupt changes, and the redundancy of nearby data. Spatial coherence is then a degree that describes spatial homogeneities of the scene, which may be exploited to avoid the reiteration of calculations that are very similar for nearby sections.

For example, if we are processing or calculating data for a scene composed of a house, we can expect all the calculations related to a flat roof to be similar, as well as within the walls or the floor. On the other hand, parts of the house such as hips, ridges, rakes or wall edges, should be considered closely, forcing an independent focus. Luckily, these parts are usually smaller than the first, which may allow most of the calculations to be reusable, because of its coherence.

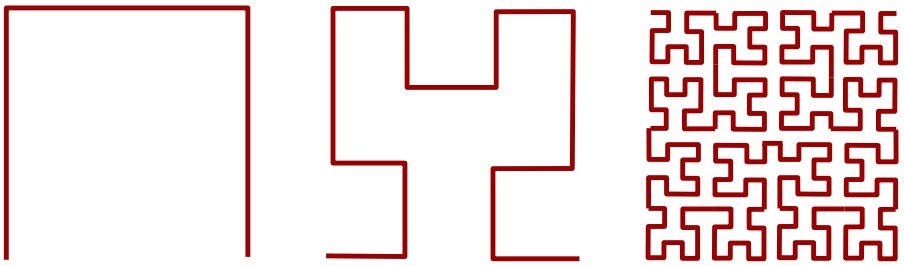
Global Illumination techniques take advantage of many types of coherence. For example, view-dependent GI methods use object and ray coherence to accelerate the calculations [Gonzalez and Gisbert, 1998]. View-independent methods, such as Radiosity, use the spatial coherence resulting from discretizing the scene into elements [Cohen et al., 1993].

Space-filling curves

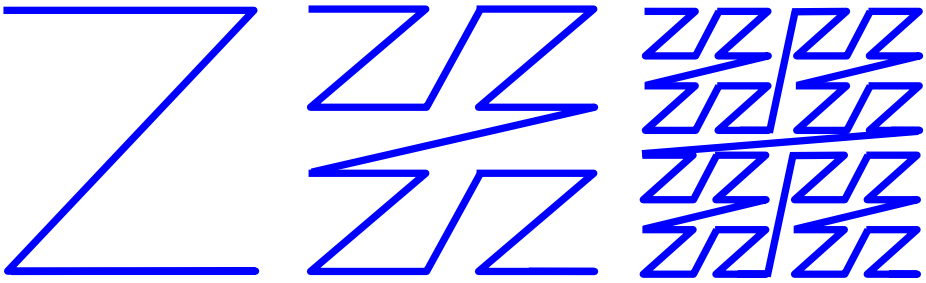
One of the techniques used to exploit spatial coherence in GI algorithms is to sort the involved elements by spatial proximity. In this way, the use of space-filling curves can be an efficient solution. A space-filling curve is a function which maps multidimensional data to one dimension while preserving locality of the data points. They define an indexing scheme that assigns spatially local elements to local indexes. The



(a) Peano curve



(b) Hilbert curve



(c) Z-order curve.

Figure 2.9: Three space-filling curves examples.

curves can be interpreted as fractals, because they are self-similar at different view levels. This property allows to exploit the coherence at lower and higher resolution levels, which in a 3D space is related to smaller and larger areas respectively [Pharr

and Fernando, 2005]. Some examples are the Peano, Hilbert [Voorhies, 1991], and Z-order [Morton, 1966] curves (see Fig. 2.9). In this figure, the 2D examples are shown. Nevertheless, this three curves can be extended to any N-Dimensional space.

Peano was the first to develop a space filling curve in 1890 [Peano, 1890]. His original idea was to obtain a continuous function to traverse the space in a specific order. This resulted in a simple (but effective) curve, named after its inventor. Later works, performed by David Hilbert in 1891 [Hilbert, 1891], extended these ideas to a develop curve that maintains a better locality, which is more suitable for coherence purposes. This resulted in the Hilbert Curve. Many years later, in 1966, Morton introduced a new function that can be implemented with very simple bitwise operations (see next section). This curve, named the Z-order curve, is less complex than the rest, but still effective for coherence purposes. Both the Hilbert and the Z-order curves have been widely used in the computer graphics field [Hughes et al., 2014].

The Z-order curve

The Z-order curve, named after its "Z" shape, can be implemented with a few bitwise operations per index computation (see Fig. 2.10). Each dimension of the space is divided into 2^N intervals. Therefore, each interval can be tagged with N bits, generating 2^{2N} quadrants in a 2D space. In this case, a quadrant is identified by $2N$ bits, computed by interleaving the bits of its dimensions. These identifiers follow the Z-order curve, and are assigned to the geometrical elements that belong to each quadrant. In this way, the elements are sorted by Z-order.

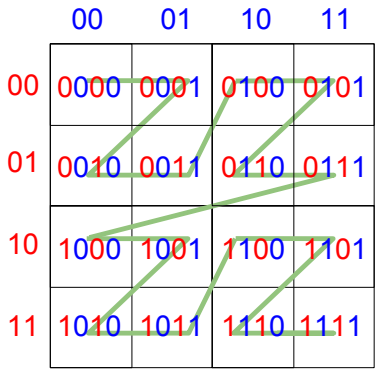


Figure 2.10: Implementation of the Z-order curve by bit interleaving.

Fig. 2.11 shows an example of sorting triangles in a 2D space by nearness using this strategy. The technique can be used in any N-Dimensional space, though in this thesis its 3D variant is used.

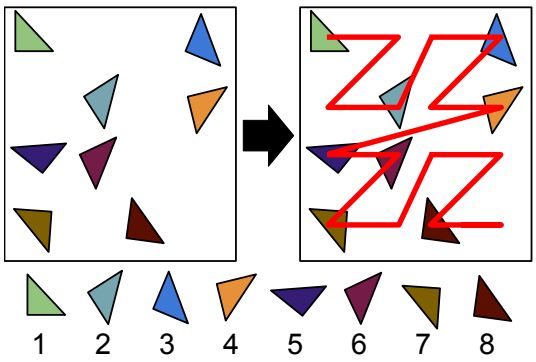


Figure 2.11: Sorting 2D triangles using the z-order curve.

2.4 Matrix factorization

Matrix factorization [Golub and Van Loan, 2012] is applied in several subjects related to mathematics and computer sciences. It can be useful when low numerical rank matrices are present in the problem. Nevertheless, it is not always simple to compute the factorization due to memory and execution time reasons. Hence, it is important to continue developing techniques to exploit the different characteristics offered by every specific problem. Moreover, it is a good contribution to find solutions that can be later applied to a wide range of problems where similar possibilities are present.

A low-rank factorization can be generalized as the decomposition of a matrix into two matrices $\mathbf{U}_{m\times k}$ and $\mathbf{V}_{m\times k}$, such that

$$\|\mathbf{A} - \mathbf{U}\mathbf{V}^T\| \leq \varepsilon$$

In this equation, ε represents the difference between the original matrix and the factorization measured using the norm $\|\ \|$. For a given ε, it is usually necessary to find the minimum k such that the previous equation is fulfilled. On the other hand, it is necessary to compute the factorization in lower execution times, memory and precision orders, when compared to traditional techniques like the singular value decomposition (SVD).

2.4.1 SVD decomposition

The singular value decomposition (SVD) of a matrix $\mathbf{A}_{m\times n}$ is a factorization of the form $\mathbf{A}$=$\mathbf{U}\mathbf{D}\mathbf{V}^T$ where $\mathbf{U}_{m\times m}$ and $\mathbf{V}_{n\times n}$ are unitary matrices, and $\mathbf{D}_{m\times n}$ is a diagonal matrix with non-negative values. The values in $\mathbf{D}$ are known as the *singular values* of $\mathbf{A}$ (represented as σ_i, where σ_i=$\mathbf{D}(i,i)$). The columns of $\mathbf{U}$ and $\mathbf{V}$ are its left and right *singular vectors* respectively.

The singular values in $\mathbf{D}$ are high-to-low ordered ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$). Following Eckart-Young Theorem [Eckart and Young, 1936], if we only consider the r largest singular values of $\mathbf{D}$, then $\mathbf{A}_r$=$\mathbf{U}_r\mathbf{D}_r\mathbf{V}_r^T$(rank($\mathbf{A}_r$)$\leq r$) minimizes $\|\mathbf{A} - \hat{\mathbf{A}}\|_2$ for all $\hat{\mathbf{A}}$ with rank($\hat{\mathbf{A}}$)$\leq r$. Here, the matrices $\mathbf{U}_r$ and $\mathbf{V}_r$ are the left and right singular vectors associated to the r singular values selected, and $\mathbf{D}_r$ is a diagonal matrix containing these values. Also, it is worth mentioning that, given the matrix $\mathbf{A}$, $\|\mathbf{A} - \mathbf{A}_r\|_2$=$\sigma_{r+1}$.

In Chapter 3, the SVD decomposition is used with this notation: $[\mathbf{Q}, \mathbf{V}]$=SVD($\mathbf{A}$), where $\mathbf{Q}$=$\mathbf{UD}$. In particular, the truncated SVD [Golub and Van Loan, 2012] is applied:

$$[\mathbf{Q}_r, \mathbf{V}_r] = \text{TSVD}(\mathbf{A}, \varepsilon) \tag{2.10}$$

where ε is a given "error", r is the index of the smallest *singular value* σ_r that satisfies $\sigma_r \geq \varepsilon$, and $\mathbf{Q}_r$ and $\mathbf{V}_r$ are the first r columns, from left to right, of the SVD resulting matrices.

2.4.2 Other factorizations

Many low-rank matrices approximation studies have been developed using factorizations. Not all of them use the singular value decomposition as the factorization technique. This section summarizes some previous works on low-rank matrices approximation techniques using several factorization strategies and some existent SVD applications on the topic.

Finding a low rank approximation of a matrix is a fundamental subject in applied mathematics, scientific computing and numerical analysis. If the Frobenius or 2-norm is used in the error calculation, the truncated singular value decomposition (TSVD) method can be used. However, in cases where the matrix is large, this technique may become too expensive. Therefore, some less expensive alternatives have been implemented. These solutions can be classified into deterministic and non-deterministic algorithms. For example, the former could use the Lanczos bidiagonalization process [Millhauser et al., 1989, Simon and Zha, 2000]), while the latter a Monte-Carlo based algorithm [Drineas et al., 2006, Frieze et al., 2004]. Also, the algorithms may or may not support out-of-core matrices.

Non-deterministic methods include several approaches. Libery et al. [2007] describe randomized algorithms for the construction of low-rank approximations using a selection of the columns of the original matrix. Besides, an algorithm to transform the low-rank matrix into the SVD decomposition is presented. On a similar path, the works by Drineas [2004, 2003] also presents a column/row sampling strategy. A subset of columns from the original matrix A is sampled, selecting each column with a probability proportional to its squared 2-norm. Then, the optimal k-dimensional subspace for the sample is determined, and A is projected onto this subspace to get a rank k matrix D.

Drineas [2006] presents algorithms based on Monte-Carlo, which are faster than the SVD. Given an $m \times n$ matrix A, a description of a low-rank approximation D^* to A is computed. Those algorithms have bounds for the error matrix $(A - D^*)$ using the Frobenius and spectral norm and can be computed by passing two or three times over the entire matrix stored in external memory. This gives the advantage of not having to store the whole original matrix in RAM. The algorithms take time linear in $\max(m, n)$ or independent of m and n.

One factorization technique used to get low-rank approximations is the QR decomposition [Golub and Van Loan, 2012], and in particular the rank revealing QR decomposition (RRQR) [Bischof and Quintana-Ortí, 1998, Gu and Eisenstat, 1996]. Halko et al. [2011] describe a set of randomized algorithms, for instance the Randomized Subspace Iteration (RSI). Given a matrix $\mathbf{A}$, these algorithms first use

random sampling to construct a low-dimensional subspace that approximate the column space of **A**. Then, the matrix is restricted to this subspace and a decomposition like QR or SVD is applied to the reduced matrix. In an experimental stage, these algorithms have shown to be simple but highly efficient. However, the need to pass over the whole matrix several times makes them inefficient for matrices that do not fit into fast memory.

As an example on deterministic low-rank approximations using matrix decomposition, Harbrecht [2012] presents an algorithm based on the pivoted Cholesky decomposition for dense, positive semi-definite matrices, where the resulting truncation error is controlled in terms of the trace norm. In this work, the properties of the matrix derive to choose the largest diagonal entry as pivot element, which allows to compute only the main diagonal and the entries of the selected columns of A. Libery [2013] also describes a deterministic strategy for matrix sketching (called Frequent-Directions) in a context of streaming algorithms, where the matrix A is processed by a server one row at a time using successive $SVDs$. It is proven that it is possible to use the algorithm to produce low rank approximations.

Other works using SVD variants include the "Quantized Iterative Cosine tree" algorithm developed by Holmes et al. [2009], which specifies a way to leverage cosine trees in the construction of an approximate SVD while providing a probabilistic error guarantee. They use a Monte Carlo technique to estimate the squared error of a matrix projection onto a subspace. This algorithm works well with middle sized matrices, but does not scale for bigger matrices. Finally, Aizenbud and Averbuch [2016] describe the Sub-Gaussian-based Randomized SVD Decomposition (SGRSVD). This method uses sub-gaussian random matrices as sparse projections and the QR decomposition to find an orthonormal basis. The pseudoinverse algorithm is applied to compute an approximation to the SVD.

2.4.3 Description of two factorization strategies used

This section presents two different techniques that are used in Chapter 3. Two stochastic methods are studied: one described by Halko [2011] and one presented by Aizenbud and Averbuch [2016].

Halko [2011] presents several methods to solve what he calls the fixed-rank problem:

Given an $m \times n$ matrix $\mathbf{A}$, a target rank k, and an oversampling parameter p, this procedure computes an $m \times (k+p)$ matrix $\mathbf{Q}$ whose columns are orthonormal and whose column space approximates the column space of $\mathbf{A}$. Thus:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\| \approx \min_{rank(X)\leq k} \|\mathbf{A} - \mathbf{X}\| \tag{2.11}$$

Then, using the matrix $\mathbf{Q}$, it is easy to obtain other standard factorizations, like the SVD or the Eigenvalue Decomposition. In this way, two matrices $\mathbf{U}_{m\times k} = \mathbf{Q}$ and $\mathbf{V}^T_{m\times k} = \mathbf{Q}^*\mathbf{A}$ can be computed, such that $\mathbf{A} \approx \mathbf{U}\mathbf{V}^T$.

The reason why an oversampling parameter is used is because the new factorization needs to have a very similar error compared to the optimum factorization with

rank k. Besides, using k or $k+p$ in the fixed-rank problem consumes almost the same computational time when p is small enough. Nevertheless, this is not the best strategy when the problem is sensible to an increment on the size of the factorization matrices. For example, if the result is going to be used to perform hundreds of thousands matrix-vector or matrix-matrix multiplications, the oversampling parameter can be prohibitive.

Given a matrix $A_{m\times n}$ and an integer l, Halko presents five variants to solve the problem of finding $\mathbf{Q}$ satisfying Eq. 2.11. These techniques are based on the "proto-algorithm", which is presented in Algorithm 1.

Algorithm 1 Proto-algorithm

1: Draw an $n \times l$ Gaussian random matrix Ω.
2: Form the $m \times l$ matrix $\mathbf{Y} = \mathbf{A\Omega}$, which is a random sample of the column space of $\mathbf{A}$.
3: Construct an $m \times l$ matrix $\mathbf{Q}$ whose columns form an orthonormal basis for the column space of $\mathbf{Y}$, e.g., using the QR factorization $\mathbf{Y} = \mathbf{QR}$.

In this thesis, the randomized subspace iteration is analyzed and used for comparison purposes, which is one of the five algorithms presented by Halko.

Randomized subspace iteration (RSI)

The **Randomized subspace iteration** algorithm is presented as a modification to the previous scheme for matrices whose singular values decay slowly. It offers an iterative solution to find a basis that approximates most of the action of the input matrix to factorize. The scheme of this algorithm is presented in Algorithm 2, where q is an integer parameter indicating the number of iterations (typically with values no bigger than 5).

Algorithm 2 Randomized subspace iteration

1: Draw an $n \times l$ Gaussian random matrix Ω.
2: Form the $\mathbf{Y}_0 = \mathbf{A\Omega}$ and compute its QR factorization $\mathbf{Y}_0 = \mathbf{Q}_0\mathbf{R}_0$.
3: **for** j=1,2,...,q
4: Form $\tilde{\mathbf{Y}}_j = \mathbf{A}\mathbf{Q}_{j-1}$ and compute its QR factorization $\tilde{\mathbf{Q}}_j$ and $\tilde{\mathbf{R}}_j$
5: Form $\mathbf{Y}_j = \mathbf{A}^*\tilde{\mathbf{Q}}_\mathbf{j}$ and compute its QR factorization $\mathbf{Q}_j$ and $\mathbf{R}_j$
6: **end**
7: $\mathbf{Q} = \mathbf{Q}_q$

The iterations allow to find a more precise approximation than in the proto-algorithm, but not without an increment in the computational cost. Therefore, depending on the problem, q can affect both the quality of the solution and the execution times. Taking a closer look into the algorithm, it can be seen that the iterations are equal to performing $\mathbf{Y} = (\mathbf{AA}^*)^q\mathbf{A}\Omega$. The matrix $(\mathbf{AA}^*)^q\mathbf{A}$ has the same singular vectors as $\mathbf{A}$, but its singular values decay more quickly. This is because $(\mathbf{AA}^*)^q\mathbf{A} = \mathbf{UD}^{q+1}\mathbf{V}$, being $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ the results of applying SVD to $\mathbf{A}$ ($\sigma_1^{q+1} \geq \sigma_2^{q+1} \geq ... \geq \sigma_n^{q+1} \geq 0$). This technique aims to reduce the weight in $\mathbf{Y}$ of the singular vectors associated with small singular values, because they interfere in the calculations. The dominant singular vectors take more relative weight by using

the powers of the matrix to be analyzed. Calculating $\mathbf{Y}$ with the previous operation is vulnerable to round-off errors, which is why an iterative scheme is used.

Note that this algorithm, as every of the other five proposed by Halko, does not need to have the whole input matrix in memory. This is because the only operations performed over $\mathbf{A}$ are multiplications, which can be executed reading $\mathbf{A}$ only one time. Nevertheless, the need to perform more than one multiplication can be prohibitive in some problems. The number of passes over the matrix depends on the number of iterations.

Sub-Gaussian-based Randomized SVD Decomposition

Aizenbud and Averbuch [2016] present a method that outputs an approximated truncated SVD decomposition, using sub-Gaussian random matrices. These matrices are used because they have strong tail decay properties. Many non-asymptotic results on a sub-Gaussian matrix distribution have recently appeared [Rudelson, 2014, Vershynin, 2010]. The authors select a subclass of these matrices called Sparse-Gaussian, to reduce the computational complexity of the algorithm. Sparse-Gaussian matrices are sparse matrices, where each entry is independently distributed with a certain probability to be zero or Gaussian otherwise. They call the algorithm **Sub-Gaussian-based Randomized SVD Decomposition (SGRSVD)**.

Algorithm 3 Sub-Gaussian-based Randomized SVD

1: Create a $k_1 \times n$ sub-Gaussian random matrix $\mathbf{\Omega}_1$.
2: Create a $l_2 \times k_1$ Gaussian random matrix $\mathbf{\Omega}'_1$.
3: Compute $\mathbf{B} = \mathbf{A}\mathbf{\Omega}_1^*\mathbf{\Omega}_1'^*$.
4: Compute the QR decomposition $\mathbf{B} = \mathbf{QR}$.
5: Create a $k_2 \times m$ sub-Gaussian random matrix $\mathbf{\Omega}_2$.
6: Compute $\mathbf{C} = \mathbf{\Omega}_2\mathbf{Q}$, $\mathbf{D} = \mathbf{\Omega}_2\mathbf{A}$ and $\mathbf{E} = \mathbf{C}^\dagger$.
7: Compute the SVD of $\mathbf{E} * \mathbf{D} = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^*$, truncated at l.
8: Output $\mathbf{U} = \mathbf{Q}\mathbf{U}_1\mathbf{\Sigma}_1$ and $\mathbf{V} = \mathbf{V}_1$.

Given a matrix $\mathbf{A}_{m\times n}$, the desire rank l and l_2,k_1, k_2 the number of columns to use, Algorithm 3 describes the method. The scheme of the algorithm is presented with a modification on the output variables, in order to preserve the previous $\mathbf{U}$ and $\mathbf{V}$ notation (instead of the SVD notation $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$). The symbol † represents the pseudo-inverse operation.

Lines 1-4 follow a similar procedure than Algorithm 1, but using two Gaussian random matrices instead of one. The advantage of this approach is that both matrices are smaller than the previous $\mathbf{\Omega}$, and that $\mathbf{\Omega}_1$ is sparse. This allows to reduce the number of operations.

Lines 5-8 are dedicated to obtain an approximate SVD. A new sub-Gaussian random matrix $\mathbf{\Omega}_2$ is computed. Theorem 3.5 in [Aizenbud and Averbuch, 2016] establishes that $\mathbf{C} = \mathbf{\Omega}_2\mathbf{Q}$ is invertible from the left. Thus, $(\mathbf{\Omega}_2\mathbf{Q})^\dagger\mathbf{\Omega}_2\mathbf{Q} = \mathbf{I}_{k_1\times k_1}$, and therefore:

$$\|\mathbf{Q}\mathbf{Q}^*\mathbf{A} - \mathbf{A}\| = \|\mathbf{Q}(\mathbf{\Omega}_2\mathbf{Q})^\dagger(\mathbf{\Omega}_2\mathbf{Q})\mathbf{Q}^*\mathbf{A} - \mathbf{A}\| \tag{2.12}$$

Now, take $\mathbf{U} = \mathbf{Q}\mathbf{U}_1\mathbf{\Sigma}_1$ and $\mathbf{V} = \mathbf{V}_1$, the outputs of the algorithm. Then, $\|\mathbf{U}\mathbf{V}^* - \mathbf{A}\| = \|\mathbf{Q}\mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^* - \mathbf{A}\| \approx \|\mathbf{Q}\mathbf{E}\mathbf{D} - \mathbf{A}\| = \|\mathbf{Q}(\mathbf{\Omega}_2\mathbf{Q})^\dagger\mathbf{\Omega}_2\mathbf{A} - \mathbf{A}\|$. Using the Eq. 2.12, Aizenbud and Averbuch use their Theorem 4.5 [Aizenbud and Averbuch, 2016] to prove that $\|\mathbf{Q}(\mathbf{\Omega}_2\mathbf{Q})^\dagger\mathbf{\Omega}_2\mathbf{A} - \mathbf{A}\| \leq K\|\mathbf{Q}\mathbf{Q}^*\mathbf{A} - \mathbf{A}\|$, where K is a constant between 1 and 2. Note that $\|\mathbf{Q}\mathbf{Q}^*\mathbf{A} - \mathbf{A}\|$ was already bounded by Halko (take, for example, $\mathbf{Q}$ output from Algorithm 1).

This algorithm preserves the same memory properties than the algorithm by Halko, since the only operations to compute over the input matrix $\mathbf{A}$ are multiplications. However, as there are no power iterations, the method only requires two passes over the input matrix. As can be seen, the algorithm executes a QR decomposition, a SVD decomposition and a pseudo-inverse operation.

2.5 Urban environments and radiative methods

Factorization techniques are well-suited for reducing the computation associated with dense low-rank matrices, typically present when computing radiosity in architectural scenarios like house models or small buildings. On bigger environments, like cities, the associated matrices may neither be full nor low-rank. In Chapter 4, a technique is proposed for computing radiosity in urban environments, taking advantage of the sparsity of the related form factors matrices.

Simulating radiation for urban environments is a challenging task. The process is composed of several steps, among which three stand out: 1) modeling cities, 2) estimating daylight, and 3) using efficient methods for radiative calculation at large scale. These steps are very related with the computer graphics field, where the radiosity method takes a big importance in step 3. This section introduces the field, and describes related works for these three steps.

2.5.1 Modeling urban scenarios

Modeling cities and urban environments requires the application of techniques related to computer graphics, computer vision, and visualization. These kind of scenarios are composed of buildings, parcels, streets and other structures that are positioned over a certain orography. The goal in geometrically modeling a city is to generate models quickly, based on a set of specifications or taken from some input data (like aerial and terrestrial pictures) [Parish and Müller, 2001]. Fig. 2.12 shows an example of a city model, composed of several thousand buildings.

There are several research works that tackle the problem of city modeling. Aliaga [2012] divides the approaches into three groups:

1. *Reconstruction*: The main purpose of these methods is to automatically reconstruct the geometry of existing urban environments. The input data is usually aerial and ground-level images, and other sensor related information. Musialski et al. [2013] provide a summary of reconstruction techniques.

2. *Interactive modeling*: Using human input and guidance, these methods intend to create models manually. They are closely related to the computer aided design (CAD) field. Yin et al. [2009] summarize interactive modeling methods.

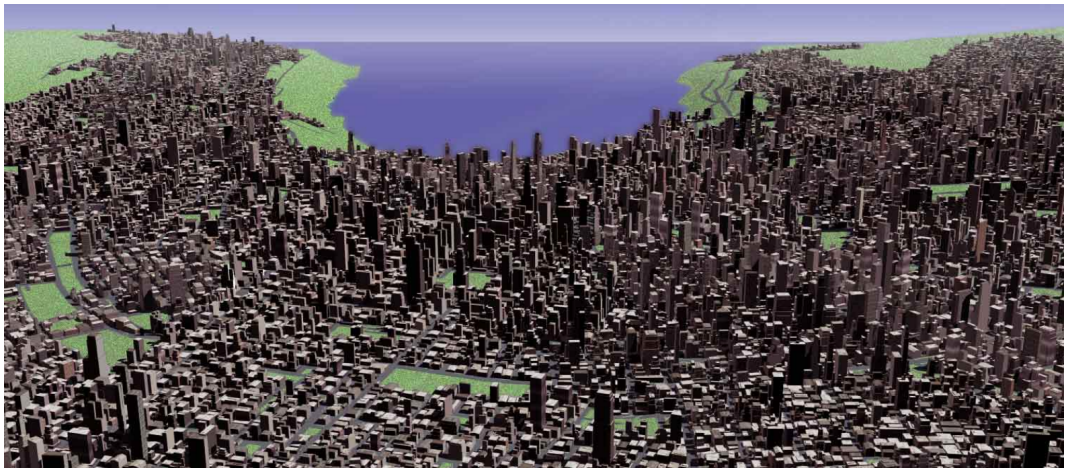


Figure 2.12: Example city model composed of approximately 26000 buildings, taken from [Parish and Müller, 2001].

3. *Procedural modeling*: There is a big redundancy degree in city models, which can be exploited in urban geometrical modeling. These methods allow to produce models using the global and local repetition of structures, which is interesting due to computational reasons. A small number of settings can produce a very detailed model of an existing or created city. From large scale designs like urban plants [Alexander et al., 1977, Prusinkiewicz and Lindenmayer, 2012], to middle sized schemes such as buildings [Parish and Müller, 2001, Wonka et al., 2003], they provide an easy way of modeling cities. Streets and parcels are other aspects addressed by procedural modeling [Galin et al., 2010, Lipp et al., 2011].

2.5.2 Daylight simulation

The second step is to define a daylight scheme that approximates the sunlight incidence in the city throughout a given time frame. This includes calculating the sun position, and the sky view from the goal city point.

The calculation of the sun position can be solved by adopting a geocentric point of view and using celestial mechanics. The sun relative orbit around the earth can be described using some well-known parameters such as eccentricity, inclination, mean anomaly, etc [Danby, 1992]. This calculation helps to determine the earth's revolution, which, along with the earth's rotation, derives into the sun position in the sky. A detailed description of this is presented by Pierre and Benoit Beckers [2012].

Another variable that affects the daylight is climate, which depends on the geographical position. To represent this in the model, an hourly test reference weather data [EERE, 2016] can be used, which contains direct normal and diffuse horizontal irradiation data for each hour of the year.

The sky section observed from a point in the city can be accurately computed through stereographic spherical projections [Beckers et al., 2011], as shown in Fig.

2.13. Another approach is to use a form factors calculation algorithm (see Sec. 2.2.1), where the hemisphere representing the sky is divided into tiles. Then, the meteorological data is mapped onto this discretization using specialized software (such as *gendaymtx*, a Radiance package [Ward, 1994] extension), where the emittance of each tile represents the light contribution of its correspondent section of the dome. There are several ways to mesh the sky, among which the works by Tregenza (see Fig. 2.14) and Mardaljevic [1999] stand out.

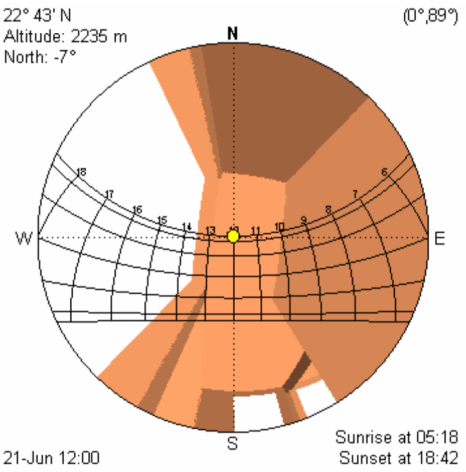


Figure 2.13: Example of a stereographic projection of sky and building, taken from [Beckers and Rodríguez, 2009]. The yellow circle in the middle represents the position of the sun at the simulated moment, while the black lines show its trajectory.

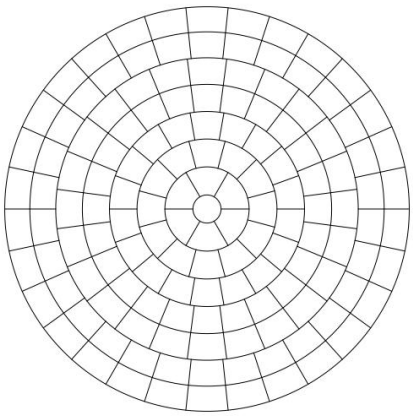


Figure 2.14: Sky dome discretized into 145 tiles (Tregenza [1987]).

After the daylighting model has been selected, a simulation software is executed to compute the effect of the sky light in the global illumination. There are many daylight simulation programs that allow to analyze this, each one with its own advantages and disadvantages, and each one capable of modeling different kind of

problems. For example, Radiance [Ward, 1994] based methods use a backward ray-tracer + radiosity engine, which are suitable for computing the illumination at a small number of sensors [Reinhart, 2011]. On the other hand, Autodesk Ecotect [Marsh, 2003] uses a Split Flux Method which is only able to model diffuse sky conditions by splitting the light influence into three components: a sky component (SC), an externally reflected component (ERC) and an internally reflected component (IRC). The choice between these two options or any other alternative (such as Velux [Labayrade et al., 2009], Dialux [Mangkuto, 2016], or Heliodon [Beckers and Masset, 2008]), depends directly on the specifications of the problem to solve.

2.5.3 Computing radiation

Before describing radiative methods, it is important to state the difference between shortwave and longwave radiation, main components of the thermal radiation (see Fig. 2.15). Shortwave radiation is radiant energy with wavelengths between 0.1μm and 5.0μm [Felsen and Marcuvitz, 1994]. The sun emits this type of radiation, which is absorbed by the earth and re-emitted as longwave radiation in the form of infrared rays (non-visible spectra). Radiative methods for urban environments take care of both types of waves, either separately or together.

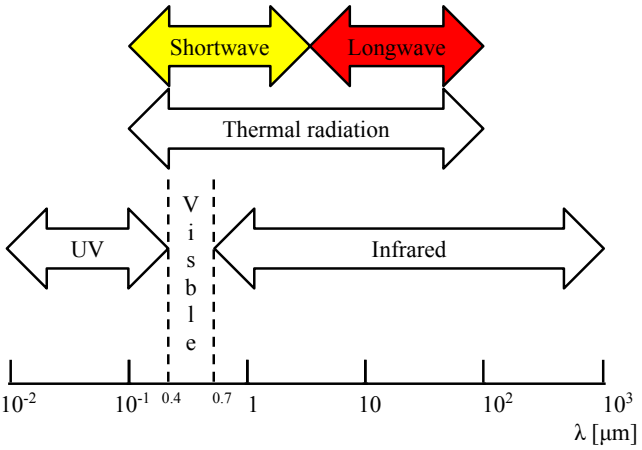


Figure 2.15: Part of the electromagnetic spectrum, highlighting shortwave and longwave radiation.

Radiative methods for urban environments

The two main methodologies for solving urban shortwave radiant exchange problems are ray tracing and radiosity. The former is widely used in rendering, and the latter was originally developed for heat transfer exchange computation. The radiosity method is usually more suitable for radiant methods, because physically-realistic data is needed, instead of photo-realistic images like in the rendering field. One of the advantages of using this method is that it can obtain results in the whole scene space, which makes it attractive for urban environment analysis. The radiosity

method is limited to diffuse reflections, and can be inefficient when working with big models.

As an alternative, a previous work for reducing the urban radiosity formulation is the simplified radiosity algorithm (SRA) [Robinson and Stone, 2005]. The basis of the simplification is grouping, for each sky direction, the main obstructions that obscured each surface. Then, for a scene composed of n patches and getting p sky tiles, the system matrix can be reduced to a $n \times p$, that can both be inverted or used to solve the system iteratively. This method is embedded in the CitySim package [Robinson et al., 2009], a multi-purpose system for urban models simulation.

Once the shortwave radiative interactions are computed, aerospace domain techniques can be used to compute coupled heating inside the city. These methods are based in a finite element formulation [Zienkiewicz et al., 1977] of the radiation-conduction problem. An implementation of this technique is presented by Van Eekelen [2012].

A more accurate and integral resolution than radiosity for radiative exchange requires the application of more sophisticated methods. For this, the need to solve the heat conduction equation for solids becomes unavoidable. The discretization of this equation can be performed through many options. The most used one is the *nodal method* [Meyer, 1999], specially designed for thermal balance and heat flux. The defined isothermal nodes are arranged in a network that simulates an electrical circuit to compute radiation. This method outputs an overall distribution of temperature through the simulated model. Other alternatives are the finite element methods [Zienkiewicz et al., 1977], that were originally designed for mechanical and civil engineering. These are more accurate but derive into much heavier computations, especially for bigger models.

Chapter 3

A Hierarchical Factorization Method for Radiosity Calculations on Scenes with High Spatial Coherence [1]

3.1 Introduction

Radiosity is a method for global illumination (GI) calculation on scenes with Lambertian surfaces. This method has been subject of study in several areas such as computer animation, architectural design and heat transfer [Dutre et al., 2006]. To implement this technique, iterative methods have been studied and used [Cohen et al., 1988, Hanrahan et al., 1991], failing to compute solutions at real-time with many bounces. Other GI methods were proposed and developed, such as *instant radiosity* [Keller, 1997], *precomputed radiance transfer* [Sloan et al., 2002], or *GPU-based global illumination* [Wang et al., 2009]. These techniques allow modeling many light effects, taking advantage of the new hardware architectures. Nevertheless, they also fail to provide real-time solutions when simulating many bounces of light. Recently, new techniques have been developed to solve the real-time infinite bounce radiosity problem, for a static geometry and dynamic emission [Fernández, 2009, Fernández and Besuievsky, 2012].

Usually, due to spatial coherence, the form factors matrix [Cohen and Greenberg, 1985] (**F**, see Table 1.1) has a low numerical rank. This fact enables the application of factorization techniques to compute low rank approximations of **F** that can be stored in main memory. Matrix factorization [Golub and Van Loan, 2012] is applied in several subjects related to mathematics and computer sciences. It is useful when matrices with low numerical rank are present in the problem. Nevertheless, it is not always simple to compute the factorization due to memory and execution time reasons. Hence, it is important to continue developing techniques to exploit the different characteristics offered by every specific problem.

[1] This chapter is based on the paper: Aguerre, J. P., & Fernández, E. (2016). *A hierarchical factorization method for efficient radiosity calculations.* Computers & Graphics, 60, 46-54.

In this chapter we present a hierarchical factorization (HF) technique for radiosity calculations. This technique implements a divide-and-conquer strategy based on the calculation of multiple singular value decomposition (SVD). In addition, the sorting of rows/columns of the matrix **RF** by similarity is exploited to speed-up the technique. For this purpose, we use the Z-order curve [Morton, 1966] over the patches of the scene, because each patch is associated to a row/column in the matrix. This method is intended to accelerate the factorization of in-core and out-of-core **RF** matrices, requiring just one pass.

3.2 Method overview

In this section, the HF method is proposed for the factorization of low-rank **RF** matrices. Our algorithm processes groups of columns of the matrix and extracts their principal components. Each group is generated dynamically, making the process suitable for out-of-core matrices. This approach works better with matrices that have a similarity coherence between near columns. Therefore, a method based on the Z-order curve is used.

Using a divide-and-conquer strategy, the input matrix $\mathbf{A}$ is divided into two blocks of columns $\mathbf{A}_1$ and $\mathbf{A}_2$ such that $(\mathbf{A}_1|\mathbf{A}_2)=\mathbf{A}$. These submatrices are factorized, combined and reduced using the SVD method. The technique is based on a recursive scheme, where a binary tree structure is formed (Fig. 3.1). Each node implies a SVD computation. The HF process results in two matrices $\mathbf{Q}$ and $\mathbf{V}$, where $\mathbf{Q}\mathbf{V}^T \approx \mathbf{A}$.

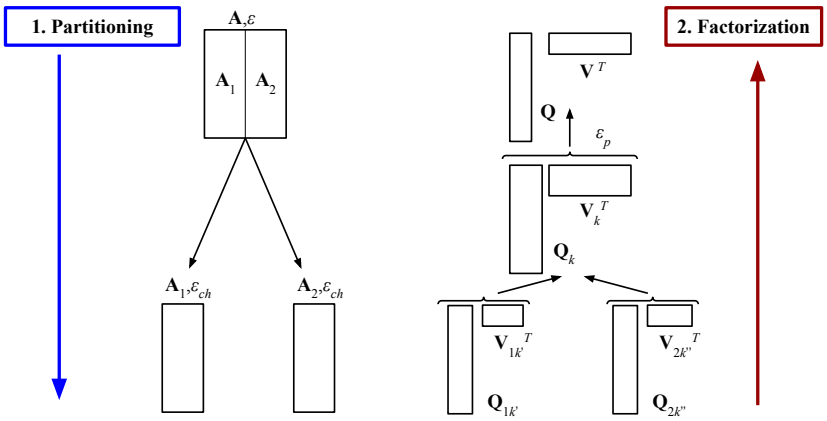


Figure 3.1: Binary tree structure of the recursion. The input matrix $\mathbf{A}$ is divided in two matrices that are factorized separately. Then, the results are combined and another factorization is performed. This results in the outputs $\mathbf{Q}$ and $\mathbf{V}$, such that $\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \varepsilon$.

The base case of the recursion consists in applying the truncated SVD to each matrix on a leaf node to find a factorization at a given error ε_{ch}. On the other hand, the recursive case consists in two recursive calls, one per block of columns. Then, the results of both child nodes are merged and another truncated SVD is executed

at a given error ε_p. The values of ε_{ch} and ε_p must be selected such that the error of the final factorization is bounded by ε.

To explain the method, the recursive case needs to be defined. The approximation of the two child nodes $\mathbf{A}_1$ and $\mathbf{A}_2$, which are $\mathbf{Q}_{1k'}\mathbf{V}_{1k'}^T$ and $\mathbf{Q}_{1k''}\mathbf{V}_{1k''}^T$ respectively, are joined and re-arranged in the following way:

$$\begin{aligned}\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_1 & \mathbf{A}_2 \end{array}\right) &\approx \left(\begin{array}{c|c} \mathbf{Q}_{1k'}\mathbf{V}_{1k'}^T & \mathbf{Q}_{2k''}\mathbf{V}_{2k''}^T \end{array}\right) \\ = \underbrace{\left(\begin{array}{c|c} \mathbf{Q}_{1k'} & \mathbf{Q}_{2k''} \end{array}\right)}_{\mathbf{Q}_k} &\underbrace{\left(\begin{array}{c|c} \mathbf{V}_{1k'}^T & 0 \\ \hline 0 & \mathbf{V}_{2k''}^T \end{array}\right)}_{\mathbf{V}_k^T} = \mathbf{Q}_k\mathbf{V}_k^T\end{aligned} \tag{3.1}$$

$$\text{where } k = k' + k''$$

The columns of the matrix $\mathbf{V}_k^T$ are trivially orthonormal, because it is composed of two matrices with orthonormal columns. Also, the approximations to the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are calculated recursively, such that the Eqs. 3.2 are satisfied for a given ε_{ch}.

$$\|\mathbf{A}_1 - \mathbf{Q}_{1k'}\mathbf{V}_{1k'}^T\|_2 \leq \varepsilon_{ch} \text{ and } \|\mathbf{A}_2 - \mathbf{Q}_{2k''}\mathbf{V}_{2k''}^T\|_2 \leq \varepsilon_{ch} \tag{3.2}$$

Having $\mathbf{Q}_k$, TSVD is applied for a given ε_p, mainly because there still can exist some redundancy between the columns of $\mathbf{Q}_{1k'}$ and $\mathbf{Q}_{2k''}$, which can lead to reduce the number of columns to take into account:

$$\begin{aligned}[\mathbf{Q}_r, \mathbf{V}_r] &= \text{TSVD}\left(\mathbf{Q}_k, \varepsilon_p\right) \\ \text{where} \quad \mathbf{Q}_k &\approx \mathbf{Q}_r\mathbf{V}_r^T\end{aligned} \tag{3.3}$$

$\mathbf{Q}_r$ is a $n \times r$ matrix and $\mathbf{V}_r$ is a $k \times r$ matrix, and by construction of TSVD it holds that

$$\|\mathbf{Q}_k - \mathbf{Q}_r\mathbf{V}_r^T\|_2 \leq \varepsilon_p \tag{3.4}$$

After that, $\mathbf{V}_r^T$ is divided into its first k' columns and its last k'' columns:

$$\mathbf{V}_r^T = \left(\mathbf{V}_{r,1}^T | \mathbf{V}_{r,2}^T\right) \tag{3.5}$$

Using the Eqs. 3.1 and 3.3, the output $\mathbf{Q}$ and $\mathbf{V}$ of the recursive step (Fig. 3.1) is defined:

$$\mathbf{Q} = \mathbf{Q}_r \; ; \; \mathbf{V}^T = \mathbf{V}_r^T\mathbf{V}_k^T = \left(\mathbf{V}_{r,1}^T\mathbf{V}_{1k'}^T \mid \mathbf{V}_{r,2}^T\mathbf{V}_{2k''}^T\right) \tag{3.6}$$

The algorithm applies the recursive case several times, which defines a tree structure. In the following sections the relation between the errors ε, ε_{ch} and ε_p is analyzed, to find $\mathbf{Q}$ and $\mathbf{V}$ that comply with $\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \varepsilon$ for a given ε. Also, the row/column sorting strategy is explained and a pseudocode for the algorithm is presented.

3.2.1 Relation between norms

It is easy to prove that, for any matrix $\mathbf{A}=(\mathbf{A}_1|\mathbf{A}_2)$, $\|\mathbf{A}_1\|_2 + \|\mathbf{A}_2\|_2$ is an upper

bound of $\|\mathbf{A}\|_2$. On the one hand, the following statement is true, using the triangle inequality of the norm:

$$\|(\mathbf{A}_1|\mathbf{A}_2)\|_2 = \|(\mathbf{A}_1|\,0\,) + (\,0\,|\mathbf{A}_2)\|_2 \leq \|(\mathbf{A}_1|\,0\,)\|_2 + \|(\,0\,|\mathbf{A}_2)\|_2 \tag{3.7}$$

Here, $(\mathbf{A}_1|\,0\,)$ represents the matrix $\mathbf{A}$ with a right zero submatrix. Since the column space of $(\mathbf{A}_1|\,0\,)$ is the same as $\mathbf{A}_1$, their left singular vectors are the same, and the only singular values added are zeros. Because the 2-norm of a matrix is its maximum singular value, it can be stated that $\|(\mathbf{A}_1|\,0\,)\|_2 = \|\mathbf{A}_1\|_2$. The same can be expressed for $(\,0\,|\mathbf{A}_2)$ and $\mathbf{A}_2$. Applying this result to Eq. 3.7, $\|\mathbf{A}\|_2$ can be bounded:

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}_1\|_2 + \|\mathbf{A}_2\|_2 \tag{3.8}$$

In spite of this result, next it is proven that $\sqrt{\|\mathbf{A}_1\|_2^2 + \|\mathbf{A}_2\|_2^2}$ is also an upper bound of $\|\mathbf{A}\|_2$, which is a better bound because:

$$\|\mathbf{A}_1\|_2^2 + \|\mathbf{A}_2\|_2^2 \leq \Big(\|\mathbf{A}_1\|_2 + \|\mathbf{A}_2\|_2\Big)^2$$

Hence, as a first step on the error estimation, the following theorem is defined and proved for 2-norm.

Theorem 3.1. *Given any matrix* $\mathbf{A}$ *composed of two column block matrices* $\mathbf{A}_1$ *and* $\mathbf{A}_2$ *such that* $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$*, the following inequality is satisfied:* $\|\mathbf{A}\|_2 \leq \sqrt{\|\mathbf{A}_1\|_2^2 + \|\mathbf{A}_2\|_2^2}$.

Proof. By definition of 2-norm:

$$\|\mathbf{A}\|_2 = \sup\big\{\|\mathbf{A}v\|_2 \text{ such that } \|v\|_2 = 1\big\}$$

Now, v_{max} is an orthonormal vector that reaches the supremum (for 2-norm v_{max} belongs to the subspace of right singular vectors associated to σ_{max}), so

$$\|\mathbf{A}\|_2 = \|\mathbf{A}v_{max}\|_2 = \sigma_{max}$$

By construction, $\mathbf{A}$ also satisfies that:

$$\|\mathbf{A}v_{max}\|_2 = \|\big(\mathbf{A}_1|\mathbf{A}_2\big)\big(v_{max_1}^T|v_{max_2}^T\big)^T\|_2$$

$$= \|\mathbf{A}_1 v_{max_1} + \mathbf{A}_2 v_{max_2}\|_2, \text{ where } v_{max} = \big(v_{max_1}^T|v_{max_2}^T\big)^T$$

Then, using the triangle inequality, it can be stated that:

$$\begin{aligned}\|\mathbf{A}\|_2 &= \|\mathbf{A}_1 v_{max_1} + \mathbf{A}_2 v_{max_2}\|_2 \\ &\leq \|\mathbf{A}_1\|_2\|v_{max_1}\|_2 + \|\mathbf{A}_2\|_2\|v_{max_2}\|_2\end{aligned}$$

Now, we introduce two useful results:

1. If $\|({v_1}^T|{v_2}^T)\|_2^2 = 1$, then $\|v_1\|_2^2 + \|v_2\|_2^2 = 1$.
2. If $a, b, x, y \in \mathbb{R}$ & $x^2 + y^2 = 1$, then $ax + by \leq \sqrt{a^2 + b^2}$.

Therefore, applying them in the triangle inequality:

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}_1\|_2\|v_{max_1}\|_2 + \|\mathbf{A}_2\|_2\|v_{max_2}\|_2 \leq \sqrt{\|\mathbf{A}_1\|_2^2 + \|\mathbf{A}_2\|_2^2}$$

□

3.2.2 The error of HF and its relation with ε_{ch} and ε_p

Given a matrix $\mathbf{A}$ and an error ε, the proposed method allows to find $\mathbf{Q}$ and $\mathbf{V}$ such that $\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \varepsilon$. To obtain this result, it is necessary to estimate the values of ε_{ch} and ε_p. Based on Eqs. 3.1 and 3.3, the following equations are defined:

$$\mathbf{A} = \mathbf{Q}_k\mathbf{V}_k^T + \Delta\mathbf{A} \quad ; \quad \mathbf{Q}_k = \mathbf{Q}_r\mathbf{V}_r^T + \Delta\mathbf{Q}_k \tag{3.9}$$

$\Delta\mathbf{A}$ can be divided into two blocks of columns $(\Delta\mathbf{A}_1|\Delta\mathbf{A}_2) = \Delta\mathbf{A}$, which satisfy that:

$$\Delta\mathbf{A}_1 = \mathbf{A}_1 - \mathbf{Q}_{1k'}\mathbf{V}_{1k'}^T \quad \text{and} \quad \Delta\mathbf{A}_2 = \mathbf{A}_2 - \mathbf{Q}_{2k''}\mathbf{V}_{2k''}^T$$

By operating with Eqs. 3.6 and 3.9, it can be derived that:

$$\mathbf{A} = \mathbf{Q}\mathbf{V}^T + \Delta\mathbf{Q}_k\mathbf{V}_k^T + \Delta\mathbf{A}$$

Now, by substracting $\mathbf{Q}\mathbf{V}^T$ from both sides and applying the triangle inequality, we obtain:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \|\Delta\mathbf{Q}_k\mathbf{V}_k^T\|_2 + \|\Delta\mathbf{A}\|_2$$

Because $\mathbf{V}_k$ is orthonormal, $\|\Delta\mathbf{Q}_k\mathbf{V}_k^T\|_2 = \|\Delta\mathbf{Q}_k\|_2$, and due to Theorem 3.1, $\|\Delta\mathbf{A}\|_2 \leq \sqrt{\|\Delta\mathbf{A}_1\|_2^2 + \|\Delta\mathbf{A}_2\|_2^2}$, therefore the above inequality can be transformed into:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \|\Delta\mathbf{Q}_k\|_2 + \sqrt{\|\Delta\mathbf{A}_1\|_2^2 + \|\Delta\mathbf{A}_2\|_2^2}$$

Finally, substituting the norms by Eqs. 3.2 and 3.4 it can be established that:

$$\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \varepsilon_p + \sqrt{2}\varepsilon_{ch}$$

Then, given an error ε, any pair ε_{ch} and ε_p that satisfies Eq. 3.10 allows to find $\mathbf{Q}$ and $\mathbf{V}$ such that $\|\mathbf{A} - \mathbf{Q}\mathbf{V}^T\|_2 \leq \varepsilon$.

$$\varepsilon_p + \sqrt{2}\varepsilon_{ch} \leq \varepsilon \tag{3.10}$$

There is not a unique set of ε_{ch} and ε_p values that satisfies the inequality. In order to find a useful scheme, it is necessary to study this problem in a deeper way. For this, Eq. 3.11 presents a generalization of the problem, were α is a real number between 0 and 1.

$$\begin{gathered} \varepsilon_p \leq \alpha\varepsilon \\ \sqrt{2}\varepsilon_{ch} \leq (1-\alpha)\varepsilon \;\Rightarrow \varepsilon_{ch} \leq \frac{(1-\alpha)\varepsilon}{\sqrt{2}} \end{gathered} \tag{3.11}$$

Now, the problem to find the values to satisfy the inequality in Eq. 3.10 is reduced to find α. This parameter affects almost every aspect of the algorithm, including execution times, ranks and errors of the resulting factorizations. As the α grows, ε_{ch} gets smaller, which means smaller truncations on the base cases of the recursion (the leaves on the binary tree structure). In the experimental analysis section, α is analyzed, and a value is selected for the rest of the study.

3.2.3 Row/column sorting

In the radiosity problem, each row of the **F** matrix is computed based on the scene view from the corresponding patch. When close patches have similar views, the **F** matrix has many similar rows/columns. The sorting of rows/columns by similarity in the matrix accelerates the factorization method, because the intermediate matrices tend to have smaller ranks on each level of the recursion.

In this work, a 3-Dimensional Z-order curve [Morton, 1966] is used to sort the patches of the scene following the Z scheme. The Z-order curve is chosen because it is less complex than other space-filling curves, but it is still effective for coherence purposes. Refer to Sec. 2.3 to find implementation details.

The patches of the scene are sorted before calculating the form factors. Then, the **F** matrix is computed following this order, which leads to obtain a version of the matrix that is sorted by the similarity of its rows/columns. In Section 3.3 it is shown that this method allows to speed-up the factorization method.

3.2.4 HF pseudocode

In order to clarify some concepts and to provide implementation details, Algorithm 4 presents the pseudocode of the proposed method. Four functions are described: the MAIN routine, the one that performs the factorization (HF), an auxiliary function (COMBINE), and the TSVD function.

In the MAIN routine (lines 1-4), the input data are the geometric model (mdl), the expected error (ε), the α value to use (see Eq. 3.11), and the number of levels used in the recursion (lvl). The patches of the scene are sorted following the Z-order curve, and HF is invoked.

The function HF (lines 5-16) implements Eqs. 3.1, 3.3, 3.5, and 3.6. Its inputs are mdl, 2 column indexes (c_1 and c_2), ε, α, and lvl. This procedure finds a factorization of the matrix formed by the columns $c_1..c_2$ of the form factors matrix. It follows a divide-and-conquer strategy, dividing that matrix in two blocks of columns and invoking recursive calls to factorize them (lines 8 - 9, see Eq. 3.1). After that, the COMBINE function is called (line 9). This function assembles and reduces the factorization of the splitted matrix (lines 17-22, see Eqs. 3.3, 3.5, and 3.6). In the calls to HF and COMBINE functions, the values of ε satisfy Eq. 3.10.

In the base case (lines 13-14), the procedure GENFORMFACTORS takes the geometric model as input and generates the form factors associated to the columns $c_1..c_2$ of the **RF** matrix. After that, TSVD (lines 23-28) is called to factorize it. As can be appreciated, the **RF** matrix is generated by parts, which allows to work with matrices that are bigger than system memory.

Algorithm 4 Hierarchical Factorization.

function MAIN(mdl, ε, α, lvl)
  [mdl, n]=Z-ORDERSORT(mdl)
  [$\mathbf{Q}$, $\mathbf{V}$, k]=HF(mdl, 1, n, ε, α, lvl)
end function

function [$\mathbf{Q}$, $\mathbf{V}$, k]=HF(mdl, c_1, c_2, ε, α, lvl)
  if $lvl \neq 0$ **then** ▷ recursive case
    $\varepsilon_{ch} = \frac{(1-\alpha)\varepsilon}{\sqrt{2}}$ ▷ Eq. 3.11
    [$\mathbf{Q}_{1_{k'}}$,$\mathbf{V}_{1_{k'}}$,k']=HF(mdl, c_1,$\lfloor\frac{c_1+c_2}{2}\rfloor$, ε_{ch}, α, $lvl-1$)
    [$\mathbf{Q}_{2_{k''}}$,$\mathbf{V}_{2_{k''}}$,k'']=HF(mdl,$\lfloor\frac{c_1+c_2}{2}\rfloor+1$,$c_2$,$\varepsilon_{ch}$, α, $lvl-1$)
    $\varepsilon_p = \alpha\,\varepsilon$ ▷ Eq. 3.11
    [$\mathbf{Q}$,$\mathbf{V}$,k]=COMBINE($\mathbf{Q}_{1_{k'}}$,$\mathbf{V}_{1_{k'}}$,$\mathbf{Q}_{2_{k''}}$,$\mathbf{V}_{2_{k''}}$,k',k'',ε_p)
  else ▷ base case
    $\mathbf{RF}_{c_1..c_2}$=GENFORMFACTORS(mdl, c_1, c_2)
    [$\mathbf{Q}$, $\mathbf{V}$, k]= TSVD($\mathbf{RF}_{c_1..c_2}$, ε)
  end if
end function

function [$\mathbf{Q}_r$,$\mathbf{V}$, r]=COMBINE($\mathbf{Q}_{1_{k'}}$,$\mathbf{V}_{1_{k'}}$,$\mathbf{Q}_{2_{k''}}$,$\mathbf{V}_{2_{k''}}$,k',k'',ε)
  [$\mathbf{Q}_r$, $\mathbf{V}_r$, r]=TSVD(($\mathbf{Q}_{1_{k'}}|\mathbf{Q}_{2_{k''}}$), ε) ▷ Eq. 3.3
  $\mathbf{V}_{r,1}$=$\mathbf{V}_r$(1:k', :) ▷ Eq. 3.5
  $\mathbf{V}_{r,2}$=$\mathbf{V}_r$(k'+1:k'+k'', :) ▷ Eq. 3.5
  $\mathbf{V}$=($\mathbf{V}_{r,1}^T\mathbf{V}_{1_{k'}}^T|\mathbf{V}_{r,2}^T\mathbf{V}_{2_{k''}}^T$) ▷ Eq. 3.6
end function

function [$\mathbf{Q}$,$\mathbf{V}$, r]=TSVD($\mathbf{A}$,ε)
  [$\mathbf{U}$, $\mathbf{D}$, $\mathbf{V}$]=SVD($\mathbf{A}$)
  r=$\max_i(\sigma_i \geq \varepsilon)$ ▷ #columns of $\mathbf{Q}$ and $\mathbf{V}$
  $\mathbf{Q}$=$\mathbf{U}$(:,1:r)$\mathbf{D}$(1:r,1:r)
  $\mathbf{V}$=$\mathbf{V}$(:,1:r)
end function

3.2.5 Computational complexity

The theory of computational complexity is a fundamental subject in computer science, and, in particular, in the algorithm design field. This process can provide a clear picture on the performance of methods, and can help on the comparison of techniques. At the same time, the main purpose of the presented algorithm is to offer a different solution to the matrix factorization problem, that can perform better than traditional and not-so-traditional methods. For all this reasons, it becomes necessary to study the complexity of the technique.

As we stated before, the algorithm can be cataloged as deterministic. Nevertheless, its performance depends on the properties of the matrix to factorize. A good scenario would be a matrix whose singular values decay rapidly. Also, another valuable property is the existence of a low numerical rank in its sub-matrices composed of contiguous columns. Following this train of thought, the complexity of the algorithm depends on these factors.

Before beginning the complexity study, we define the following notation for simplicity reasons: ρ is the compression rate resulted from applying a truncated SVD (with rank k) to the matrix $\mathbf{A}_{m\times n}$ ($\rho = k/n$). Also, ρ_i is the average compression rate on level i of the recursion.

The main component on the computational cost of the algorithm is the execution of multiple SVD decompositions, where the cost of computing SVD($\mathbf{A}_{m\times n}$) is $\mathrm{O}(mn\min\{m,n\})$. In this way, the cost of computing SVD in the leaves depends only on the size of the initial division of the matrix, while the rest depends on the results of those decompositions. Therefore, we divide the complexity study in two: the SVD on the leaves and the rest.

Given an amount of levels l, the number of sub-matrices in the leaves (this is level number l) is 2^l, and the size of each sub-matrix is $m \times (n/2^l)$. The number of flops required to compute each SVD is approximately $m\,(n/2^l)^2$. Then, the total number T_l of flops required by this level is:

$$T_l \sim (m\,(\frac{n}{2^l})^2)2^l = \frac{m\,n^2}{2^l} \tag{3.12}$$

Following a bottom-up procedure, the time complexity of computing level $l-1$ is studied next, where ρ_l is the average compression rate on level l. The amount of SVD to compute is 2^{l-1}, and the matrices to process are averagely sized $m\times(n/2^l)(2\rho_l) = m \times (n/2^{l-1})\rho_l$. The total number T_{l-1} of flops required by this level is:

$$T_{l-1} \sim m\,\left(\frac{n\,\rho_l}{2^{l-1}}\right)^2\,2^{l-1} = \frac{m\,n^2\,\rho_l^2}{2^{l-1}} \tag{3.13}$$

With an analogous calculation, the matrices to process on level $l-2$ are averagely sized $m \times (n/2^l)(2\rho_l 2\rho_{l-1}) = m \times (n/2^{l-2})\rho_l\rho_{l-1}$ The number of flops required by level $l-2$ is approximately $m\,n^2\,\rho_l^2\rho_{l-1}^2/2^{l-2}$. This procedure can be generalized for every level:

$$T_{l-i} \sim m\,n^2\frac{1}{2^{l-i}}\prod_{j=l-i+1}^{l}\rho_j^2 \quad \forall i \in 1\,..\,l \tag{3.14}$$

Now, the computational times for computing levels 0 to $(l-1)$ is calculated by the following sum:

$$\sum_{i=1}^{l} T_{(l-i)} \sim m\,n^2\sum_{i=1}^{l}\Big(\frac{1}{2^{l-i}}\prod_{j=l-i+1}^{l}\rho_j^2\Big) \tag{3.15}$$

Finally, the computational complexity of the algorithm is calculated summing up the computational times in Eq. 3.12 and 3.15:

$$T_{HF} = T_l + \sum_{i=1}^{l} T_{(l-i)} \sim \frac{m\,n^2}{2^l} + m\,n^2\sum_{i=1}^{l}\Big(\frac{1}{2^{l-i}}\prod_{j=l-i+1}^{l}\rho_j^2\Big) \tag{3.16}$$

This is the most general equation for the complexity order, which is still $O(mn^2)$. If we suppose $\rho = \rho_1 = .. = \rho_l$, then Eq. 3.16 is simplified:

$$T_{HF} \sim \frac{m\,n^2}{2^l} + m\,n^2 \sum_{i=1}^{l} \left(\frac{\rho^{2i}}{2^{l-i}} \right) \tag{3.17}$$

The summation involved in this equation can be expressed as:

$$\sum_{i=1}^{l} \left(\frac{\rho^{2i}}{2^{l-i}} \right) = \frac{2^{1-l}\rho^2 - 2\rho^{2+2l}}{1 - 2\rho^2} \tag{3.18}$$

Combining Eq. 3.17 and 3.18, an approximation of the total execution time of HF is:

$$\widetilde{T_{HF}} \sim \frac{m\,n^2}{2^l} + m\,n^2\,\frac{2^{1-l}\rho^2 - 2\rho^{2+2l}}{1 - 2\rho^2} \tag{3.19}$$

$\widetilde{T_{HF}}$ from Eq. 3.19 is much lower than mn^2 when $0 < \rho < 1$ and l is a natural number. It approximates T_{HF} if the supposition $\rho \approx \rho_1 \approx ... \approx \rho_l$ is fairly satisfied. However, this is not the case for many matrices, depending on the level of spatial coherence of the corresponding scene. We can approximate ρ using the geometric mean of $\rho_1..\rho_l$: $\bar{\rho} = \sqrt[l]{\rho_1 \times \rho_2 \times ... \times \rho_l}$. Sec. 3.3.2 shows an experimental study of these concepts, where ρ_i and $\bar{\rho}$ are analyzed for an example case.

3.3 Experimental analysis

In order to analyze our proposal, the algorithm was implemented using MATLAB and the SVD function presented by Vijayan [2014]. Note that, for simplicity reasons, the only parallelism used is the MATLAB native multi-threading, leaving other possible optimizations for future works. All tests were performed on an Intel i7 processor along with 16GB of RAM memory. The form factors are computed using the hemi-cube technique [Cohen and Greenberg, 1985], implemented with OpenGL and CUDA, and executed on a Geforce GTX 780 with 4GB of RAM.

The hemi-cube algorithm uses five plane projections per patch. These projections are implemented using some basic OpenGL operations. Since the patches are fixed through the computation of every patch's form factors, the vertex buffer storing the corresponding vertexes is transferred only once to the GPU. This prevents the memory allocation and transference for every hemi-cube, which allows a more efficient use of the GPU.

After the five projections of patch i are obtained, a CUDA kernel performs the rest of the algorithm in parallel, exploiting the capabilities of modern GPUs. Each pixel of the projections is decoded by color to find the corresponding projected patch. The pixels are first weighted by the cosine law depending on its position inside the hemi-cube. Then, for $j = 1..n$, the number of pixels associated with patch j is divided by the total number of pixels (which is also called hemi-cube resolution), and this results in $\mathbf{F}_{ij}$. This procedure is repeated for all the patches, obtaining the matrix $\mathbf{F}$.

As each column can be computed independently, the matrices are generated dynamically on each leaf of the recursive tree, allowing to process matrices that cannot

be allocated in system memory. As can be seen, this hemi-cube implementation fully exploits the specialized graphics hardware to obtain an efficient approach.

3.3.1 Algorithm calibration

In this section, we study the impact on the algorithm of parameters such as the number of levels in the recursion and ε. For this purpose, the radiosity matrix corresponding to the Cornell Box scene is used (Fig. 3.2), with different number of patches. The number of patches used are 2560, 10240, 40960, and 163840. The execution times needed to generate all the column blocks of $\mathbf{F}$ are 3.76s, 13.8s, 61.0s, and 369s respectively.

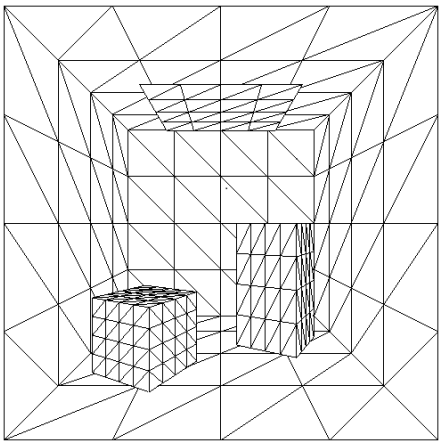


Figure 3.2: Cornell box subdivided into 640 triangles.

Choosing a configuration for the values of ε_p and ε_{ch}

To study the effect of the parameter α (defined at Eq. 3.11) on the method, Table 3.1 presents the execution results using different α values on the factorization of an example matrix ($\mathbf{F}$ matrix of the Cornell box scene subdivided into 2560 patches, $\varepsilon = 0.1$). As can be seen, the obtained errors get closer to ε as the α grows, and the execution times get larger. This shows that the algorithm is able to result in approximations that are very close to the expected error, but this signifies a slight growth in the execution times.

In Table 3.1, it can be appreciated that for most values of α, the real error converges roughly to $\alpha\varepsilon$. The experiment leads to think that a different scheme can be used, where Eq. 3.10 is not satisfied, and yet get a bounded error. This happens because the spaces defined by the columns of $\mathbf{Q}_{1k'}$ and $\mathbf{Q}_{2k''}$ are often similar when there is a high spatial coherence in the scene. More work is needed to take advantage of this aspect. Beyond this fact, in the rest of the experimental section, we use the configuration described in Eq. 3.20, where ε is equally distributed in both terms of Eq. 3.10 ($\alpha = 0.5$):

$$\varepsilon_{ch} = \frac{\varepsilon}{2\sqrt{2}} \;\; ; \;\; \varepsilon_p = \frac{\varepsilon}{2} \tag{3.20}$$

α	time (s)	rank	real ε
0.1	7.68×10^{-1}	641	0.0214
0.2	5.68×10^{-1}	585	0.0204
0.3	6.00×10^{-1}	535	0.0301
0.4	6.19×10^{-1}	462	0.0401
0.5	6.45×10^{-1}	416	0.0500
0.6	7.04×10^{-1}	392	0.0595
0.7	8.15×10^{-1}	373	0.0696
0.8	1.01×10^{0}	357	0.0800
0.9	1.27×10^{0}	344	0.0892

Table 3.1: Results for $\varepsilon = 0.1$ and different α values.

Therefore, looking at the results at Table 3.1, it is expected that the real error will be approximately $\varepsilon/2$.

Levels of the recursion

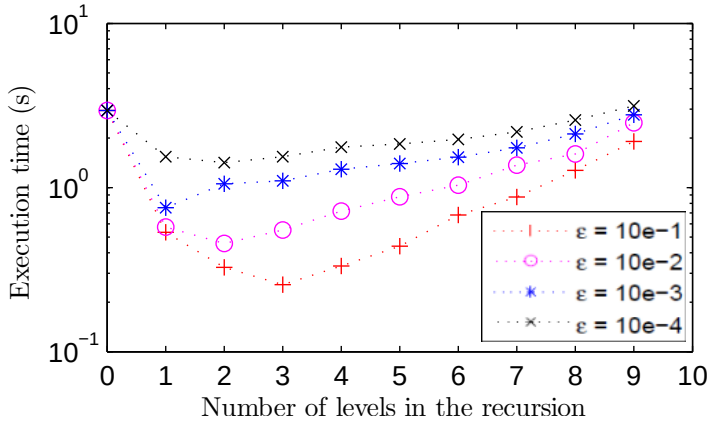
There is a relation between the recursion levels and the execution times, for a given ε. Moreover, this relation also depends on the scene and its number of patches. Fig. 3.3 presents the execution times of the algorithm for the Cornell Box scene with different number of patches and different ε. In the case of 0 levels (where just TSVD is applied), the results do not depend on ε. This happens because the implementation of the TSVD technique does not take advantage of ε to speed up the process. Also, the optimal number of levels depends on the expected error.

Z-order curve technique

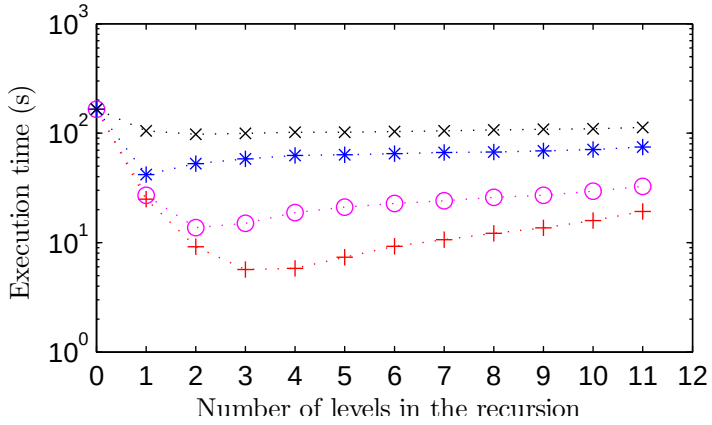
Next, a test to show the usefulness of the Z-order curve technique (described in Sec. 3.2.3) is presented. It is important to say that this technique is a linear process, and that its execution times are very small when compared to the factorization time, for every size. Table 3.2 shows the execution times, obtained errors and ranks for the algorithm with and without the use of this technique. All the results were executed using the optimal number of levels in the recursion. As can be seen, the use of Z-order curve makes the factorization code more efficient, reaching a speedup up to $2\times$ for large matrices.

On the other hand, the real ε displayed corresponds to the 2-norm of the difference between the original and factorized matrix; for size $n = 163840$, ε could not be calculated due to memory reasons. As stated in Sec. 3.3.1, the error converges roughly to $\varepsilon/2$, due to the configuration selected for ε_{ch} and ε_p, described in Eq. 3.20.

Fig. 3.4 shows the execution times for the Cornell box scene with 10240 patches and different recursion levels. The results are presented with and without the use of the Z-order curve technique. Besides the fact that both strategies show its optimum at 3 levels, better execution times are obtained when the row/column sort is applied.



(a) Results for 2560 patches



(b) Results for 10240 patches

Figure 3.3: Execution times for the Cornell Box scene with different number of patches.

Size	No Z-order			Z-order			Gain
	time (s)	real ε	rank	time (s)	real ε	rank	
2560	1.31×10^{0}	0.0501	416	2.57×10^{-1}	0.0500	416	5.10 ×
10240	8.24×10^{0}	0.0505	874	5.67×10^{0}	0.0498	877	1.45 ×
40960	2.60×10^{2}	0.0500	1786	1.41×10^{2}	0.0496	1791	1.84 ×
163840	4.65×10^{4}	-	3981	2.32×10^{4}	-	3990	2.00 ×

Table 3.2: Execution times and obtained error for desired $\varepsilon = 0.1$, with and without the use of Z-order curve.

Matrix rank and memory consumption

Since one of the purposes of this work is to reduce the memory size of the input matrix, it is important to study the relation between the initial dimensions and the obtained ranks. This helps to show that the proposed algorithm can take advantage

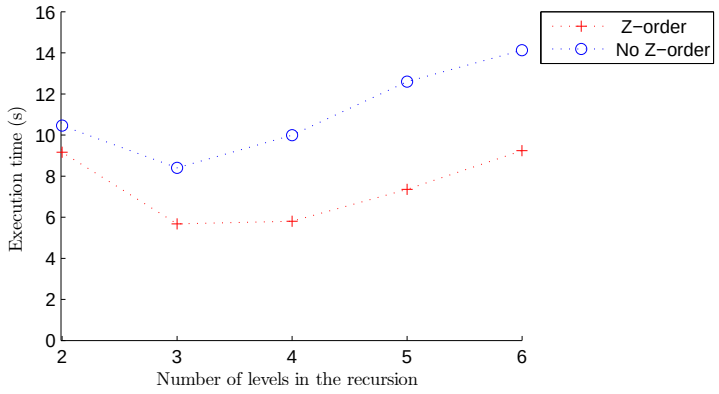


Figure 3.4: Execution times for the Cornell Box with 10240 patches.

of the low numerical rank of the matrices and, therefore, be utilized in radiosity calculations. Fig. 3.5 presents the relation between the number of patches and the obtained rank for a Cornell Box scene and different errors.

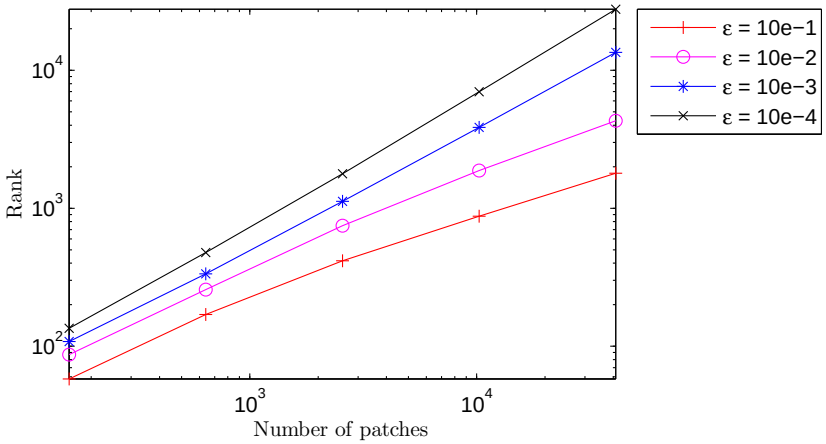


Figure 3.5: Relation between #patches and rank for the Cornell Box.

On the other hand, it is important to study the memory consumption throughout the execution of the algorithm. This determines the memory requirements to perform the factorization, which are usually bigger than its final memory size. Table 3.3 shows the memory size of the resultant factorizations for the Cornell Box scene with different number of patches and $\varepsilon = 0.1$, as well as the memory peak reached during the process. The memory peak depends on the number of levels used in the recursion. For each matrix, the results are reported using the number of levels that produce shorter execution times.

The experiment shows that the memory reduction is significant when comparing the size of the full matrix and the size of the factorization. For all the test cases, the factorization is small enough to fit in the system memory of a regular PC. The memory peaks are reached during the execution of the SVD routine, and are always higher than the final size of the factorization. In spite of this, every test case was

Size	k	Full matrix Memory size	Factorization Memory size	Process Memory peak
2560	416	25.0 MB	8.12 MB	184 MB
10240	874	400 MB	68.3 MB	209 MB
40960	1786	6.25 GB	558 MB	820 MB
163840	3981	100 GB	4.86 GB	7.55 GB

Table 3.3: Memory consumption for the Cornell Box and $\varepsilon = 0.1$.

successfully executed without exceeding the memory limits (16 GB).

3.3.2 Analysis of the compression rates (ρ)

The value of ρ (see Eq. 3.19) depends on many factors, such as the numerical rank of the matrix to factorize, the values of ε, and the number of levels. As an example, we present the obtained values of $\rho_1..\rho_l$ and $\bar{\rho}$, using the Cornell Box scene ($n = 2560$), different number of levels (l), and different ε values. The results can be seen in Table 3.4.

	ε	ρ_1	ρ_2	ρ_3	ρ_4	$\bar{\rho}$	T_{HF}(Eq.3.16)	$\widetilde{T_{HF}}$(Eq.3.19)
l=1	1.0×10^{-1}	0.21	-	-	-	0.21	9.13×10^{9}	9.13×10^{9}
	1.0×10^{-2}	0.35	-	-	-	0.35	1.04×10^{10}	1.04×10^{10}
	1.0×10^{-3}	0.59	-	-	-	0.59	1.42×10^{10}	1.42×10^{10}
	1.0×10^{-4}	0.93	-	-	-	0.93	2.29×10^{10}	2.29×10^{10}
l=2	1.0×10^{-1}	0.87	0.29	-	-	0.50	5.97×10^{9}	7.34×10^{9}
	1.0×10^{-2}	0.82	0.50	-	-	0.64	9.11×10^{9}	1.04×10^{10}
	1.0×10^{-3}	0.80	0.84	-	-	0.82	1.77×10^{10}	1.74×10^{10}
	1.0×10^{-4}	0.85	0.98	-	-	0.91	2.39×10^{10}	2.26×10^{10}
l=3	1.0×10^{-1}	0.77	0.82	0.39	-	0.63	4.61×10^{9}	6.13×10^{9}
	1.0×10^{-2}	0.70	0.82	0.71	-	0.74	9.84×10^{9}	9.66×10^{9}
	1.0×10^{-3}	0.81	0.91	0.92	-	0.88	1.92×10^{10}	1.82×10^{10}
	1.0×10^{-4}	0.87	0.96	1.00	-	0.94	2.57×10^{10}	2.39×10^{10}
l=4	1.0×10^{-1}	0.77	0.68	0.81	0.57	0.70	4.43×10^{9}	5.04×10^{9}
	1.0×10^{-2}	0.70	0.74	0.90	0.87	0.80	1.08×10^{10}	9.12×10^{9}
	1.0×10^{-3}	0.80	0.90	0.97	0.96	0.91	2.00×10^{10}	1.83×10^{10}
	1.0×10^{-4}	0.89	0.96	0.97	1.00	0.96	2.59×10^{10}	2.52×10^{10}

Table 3.4: Compression rates for the Cornell Box scene, with different l and ε values

.

Note that $\bar{\rho}$ decreases when ε decreases, which implies smaller factorization matrices. In the case of ε=1.0×10^{-1}, the decomposition in the leaves always produces the bigger compression. This means that, for this scene, the spatial coherence is bigger for closer patches. Nevertheless, the compression rates at the rest of the levels is still significant.

For ε=1.0×10^{-3} , 1.0×10^{-4}, the number of levels that leads to the smallest amount of operations seems to be $l = 2$, while for other values of ε it is $l = 4$. On the other hand, the strategy of using the geometric mean ($\bar{\rho}$) instead of each ρ_i

seems to produce only a negligible error $|T_{HF} - \widetilde{T_{HF}}|$ for this scene. This fact allows to work with $\bar{\rho}$, which simplifies the study.

3.3.3 Comparison with other algorithms

In order to compare the proposed algorithm with other existing methods, we studied several factorization techniques that compute a low-rank approximation of a given matrix. In this stage, two algorithms stood out both in execution time and precision of the resultant factorization. The first method is the Randomized Subspace Iteration, presented by Halko [2011]. It offers an iterative solution to find a basis that approximates most of the action of the input matrix to factorize. Each iteration derives into more precise solutions, but each one passes 2 times over the matrix, slowing the total execution time. The second technique is the Sub-Gaussian-based Randomized SVD Decomposition (SGRSVD), presented by Aizenbud and Averbuch [2016], which uses sub-Gaussian random matrices to compute an approximated truncated SVD. This method requires only two passes over the input matrix.

It is important to highlight that both algorithms solve the fixed-rank problem (see Sec. 2.4.3). On the other hand, HF solves the so called fixed-precision problem, where the expected error is taken as input, instead of the expected rank. For the radiosity problem, the latter scheme is more useful because it allows to bound the error of the radiosity results (as shown in Eq. 2.9). Despite the fact that it is still possible to obtain a desired radiosity error using the fixed-rank approach, this would lead to greater execution times. For this, a larger k value should be used, and then the resultant matrices should be truncated measuring the generated error. The value of k should be estimated by the user, which can lead to several trials due to underestimations, or greater execution times due to overestimations.

Therefore, to perform the comparison, we first execute the HF algorithm to factorize the form factors matrix of the Cornell Box scene for different sizes. Then, the output rank k is used as input to factorize the same matrices using the other algorithms. Table 3.5 presents these results along with the regular SVD routine. The speedups of HF over the other methods and the obtained errors are reported.

Size	k	HF		RSI		SGRSVD		SVD	
		time (s)	real ε	speedup	real ε	speedup	real ε	speedup	real ε
2560	416	2.57×10^{-1}	0.0500	1.11 ×	0.0570	2.83 ×	0.0513	11.3 ×	0.0500
10240	877	5.67×10^{0}	0.0505	1.25 ×	0.0586	1.60 ×	0.0872	28.8 ×	0.0504
40960	1791	1.41×10^{2}	0.0500	1.38 ×	0.0600	7.66 ×	0.0986	39.6 ×	0.0500
163840	3990	2.32×10^{4}	-	1.45 ×	-	15.23 ×	-	-	-

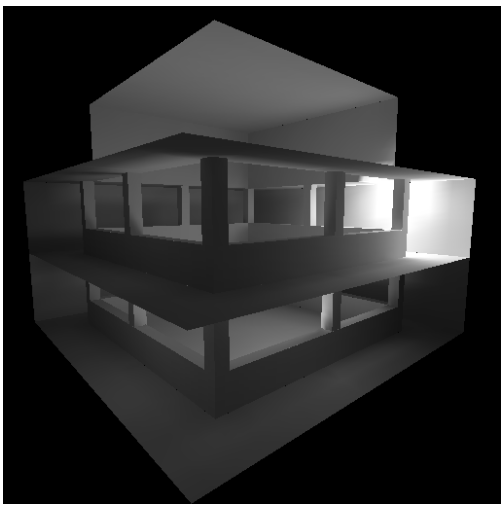
Table 3.5: Results for factorizing the Cornell Box form factors matrix, using the three algorithms to be compared. Speedup is equal to $T_{algorithm}/T_{HF}$.

Our technique works faster than the rest and its accuracy is very near the optimal (SVD). Also, the SGRSVD method is the slowest for all test cases. The out-of-core matrix used corresponds to the model with n=163840. For this model, HF works faster than RSI and SGRSVD algorithms, which need more passes over the input matrix than ours.

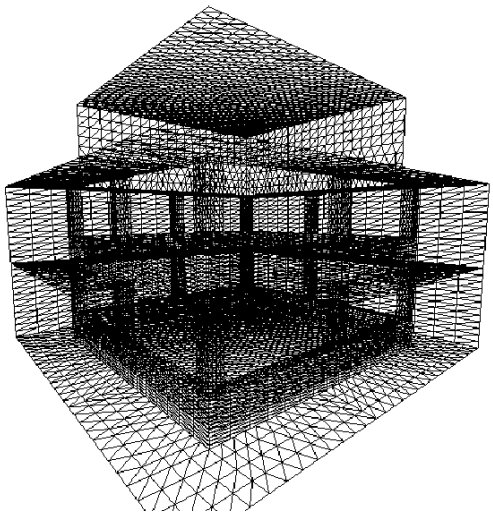
3.3.4 Real-Time Radiosity results

It is important to analyze the results of the proposed algorithm when applied to solve the radiosity (B) of a scene. To solve radiosity, we apply the factorization results $\mathbf{Q}$ and $\mathbf{V}$ on Eq. 2.8, where $\mathbf{U}_k$=$\mathbf{Q}$ and $\mathbf{V}_k^T$=$\mathbf{V}^T$.

Two more scenes were used. Fig. 3.6a presents the Patio scene composed of 21886 patches, while Fig. 3.6c shows the Sponza Atrium, composed of 79232 patches. Computing the elements of the $\mathbf{F}$ matrices takes about 30 and 130 seconds respectively.

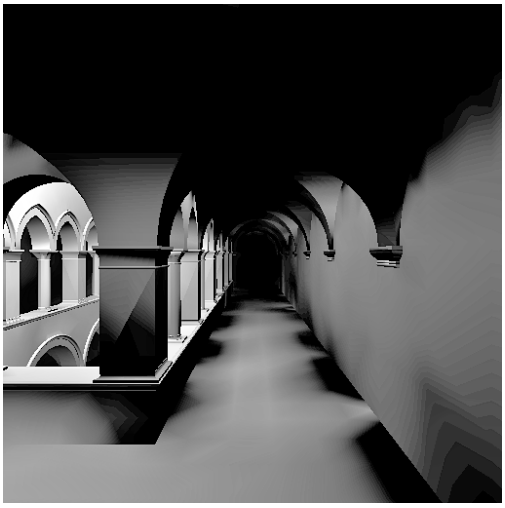


(a) Patio.



(b) Wireframe view.

(c) Sponza Atrium.



(d) Upper corridor.

Figure 3.6: Two example scenes.

The obtained execution times and speed (in Frames per Second) for solving radiosity are presented in Table 3.6. The execution times for computing $\mathbf{F}$ are not

included. The pre-computation times for the calculation of $\mathbf{Q}$ and $\mathbf{V}$ by factorization, and $\mathbf{Y_k}$, are much higher than the B calculation times, which for some cases are shorter enough for a Real-Time execution (>20 fps). This result makes the method suitable for problems where the radiosity values of a static geometry with dynamic emission must be calculated many times, like in video games or inverse lighting problems [Fernández and Besuievsky, 2012].

Scene	k	factorization (s)	Y_k (s)	B (s)	fps
Patio	3920	97	7.42	0.040	25.3
$n = 21888$	3114	71	4.70	0.031	32.6
	2227	58	2.45	0.024	41.5
	1617	49	1.29	0.016	62.1
Sponza	17857	19225	329.42	0.921	1.1
$n = 79232$	3483	2023	19.66	0.128	7.8
	1751	1987	5.25	0.067	14.8
	376	1524	0.33	0.015	66.7

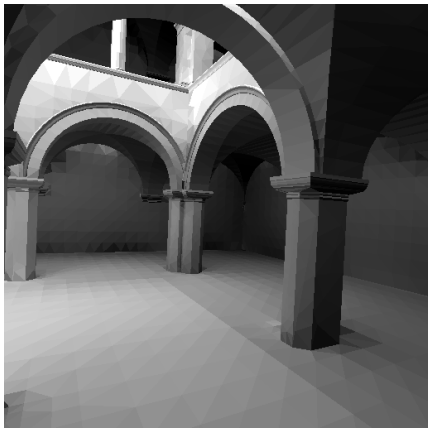
Table 3.6: Execution times for computing radiosity of the Patio and Sponza atrium, with different dimensions.

Table 3.7 shows the radiosity relative error results for different dimensions on the Patio and Cornell Box scenes. The different factorizations were tested for radiosity (using Eq. 2.8 to calculate $\widetilde{B}$) with 500 random emission vectors E, where only one patch is selected as emitter in each test. The exact radiosity B is calculated using Eq. 2.3 and the inverse operation. The mean, standard deviation and maximum values are reported. The minimum value is near 0 for every case. This shows that for these scenes, the mean relative errors in the radiosity results are about 20 times smaller than the value of ε defined for the approximation of $\mathbf{RF}$.

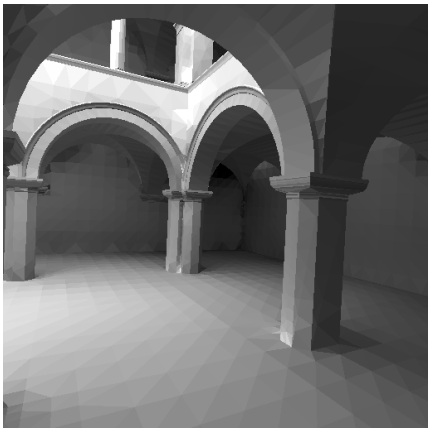
	Patio (n=21888) $\|\widetilde{B} - B\|/\|B\|$				Cornell Box (n=2560) $\|\widetilde{B} - B\|/\|B\|$			
ε	k	μ	σ	Max	k	μ	σ	Max
0.05	3920	0.0025	0.0013	0.0073	560	0.0021	0.0017	0.0094
0.1	3114	0.0048	0.0030	0.0170	416	0.0045	0.0042	0.0213
0.2	2227	0.0096	0.0073	0.0395	328	0.0088	0.0075	0.0448
0.3	1617	0.0142	0.0124	0.0635	274	0.0133	0.0122	0.0839
0.5	932	0.0232	0.0224	0.1205	139	0.0255	0.0299	0.1713

Table 3.7: Relative error for radiosity calculations using different dimensions.

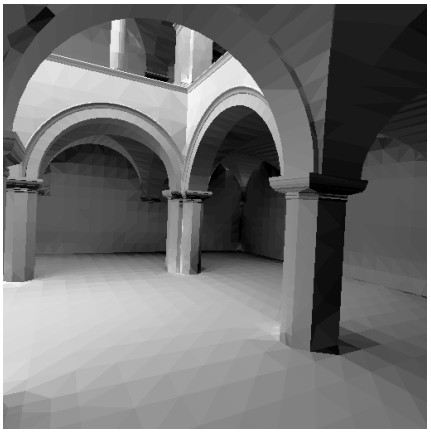
Fig. 3.7 shows radiosity results for the Sponza atrium scene. It is important to highlight that the relative error is not reported for this scene because the size of the matrix is too large (the exact radiosity B was not calculated). Three examples of the same scene are shown, calculated with different ε values. The illumination looks more real and the memory cost increases for smaller values of ε.



(a) ε=0.9 ; size = 79232×376 ≡ 227MB.



(b) ε=0.5 ; size = 79232×1751 ≡ 1.06GB.



(c) ε=0.2 ; size = 79232×3482 ≡ 2.10GB.

Figure 3.7: Radiosity results for the Sponza Atrium.

3.4 Overview and conclusions of the chapter

This chapter proposed a hierarchical technique to factorize low numerical rank matrices, applying multiple decompositions to sections of the matrix and, then, joining and reducing the results. Moreover, the sorting of rows/columns of the matrix by similarity can be used to accelerate the process. This one-pass algorithm works faster than the other studied methods for medium-to-large sized problems, and allows to work with matrices that do not fit into system memory.

The proposed method is used to solve the radiosity problem efficiently for static scenes by factorizing the matrix **RF**. Furthermore, the Z-order curve is used to sort the patches of the scene, which allows to exploit its spatial coherence. As a result, it is possible to calculate in real-time the global radiosity (with infinite bounces) for multiple scenes (2560, 21888, and 79232 patches) with negligible relative error.

Regarding the hierarchical factorization method, several aspects were analyzed. In the first place, the error of the technique was evaluated. For this, we bounded the error of the result of one step of the recursion, using the 2-norm difference on the results of the previous step (ε_{ch}), and the error produced by the extra decomposition applied to reduce the concatenation (ε_p). Applying this bound in every step of the recursion allows to bound the final error of the resulting factorization. Finally, given a goal error bound (ε), multiple configurations of ε_{ch} and ε_p can be used, leading to different final errors and execution times.

On the other hand, the computational complexity of the hierarchical factorization was studied. This depends highly on the compression rates of each step of the recursion ($\rho_1, \rho_2, .., \rho_l$), which is directly related to the spatial coherence of the scene. Eq. 3.16 expresses the computational complexity of the algorithm in terms of the compression rates. Also, if we suppose $\rho_1 = \rho_2 = .. = \rho_l$, the complexity becomes simpler (see Eq. 3.19).

In the experimental analysis, the Cornell box scene was used to calibrate the algorithm, evaluating the effect of changing the number of levels in the recursion, and the use of the Z-order curve. We conclude that the sorting of rows/columns of **RF** is a powerful strategy to reduce the execution times by exploiting the spatial coherence of the scene.

The execution times and memory consumption of performing the factorization were studied using different number of patches (n), and different expected errors. As n grows, the final compression rate becomes smaller, but the execution times and memory consumption grows. Also, a comparison between the algorithm and other existing methods was performed. For the tested matrices, our technique works faster and more accurately than the others.

Finally, to test radiosity results, two more scenes were used: the Patio and the Sponza Atrium. HF was applied to factorize their **RF** matrices. As a result, it was possible to calculate the global radiosity (with infinite bounces) for multiple scenes (2560, 21888, and 79232 patches) with negligible relative error. After a pre-computation stage, this was performed in real-time for the static geometries, and without the use of any specialized hardware resource (like GPU) to compute the matrix-vector and matrix-matrix multiplications.

Chapter 4

A Radiosity Method for Highly Occluded Environments[1]

4.1 Introduction

In the radiosity problem, if two patches are completely occluded from each other, they do not exchange energy directly. If this property is satisfied for most pair of patches on a scene, then the form factors matrix is a sparse matrix. Inverting or factorizing sparse matrices is not always possible, since the results are frequently full matrices that may exceed the memory limits. In the case of radiosity calculations, this sparsity can be exploited directly using iterative methods [Chelle and Andrieu, 1998], or the inverse can be approximated by another sparse matrix [Kontkanen et al., 2006].

In this chapter, we analyze the case of City models, which are an example of a highly occluded environment [Wimmer and Bittner, 2015]. Computational simulation for radiative transfer on an urban scale, where thousands of buildings have to be considered, is a challenge. The main problem is how to deal with the huge amount of data required to represent such models. Despite the fact that the factorization of the **F** matrix is usually not an option, the sparsity of the radiosity matrix can be exploited. An approach for radiative exchange computation that can approximate the inverse of the radiosity matrix is proposed. The problem is formulated as a Neumann series [Golub and Van Loan, 1996] and the inverse is approximated by eliminating unimportant terms. This technique is applied to compute urban radiation exchange.

Urban physics simulation has become a major topic of interest, due to the increasing need of energy assessment tools at large scale. The evaluation of annual solar irradiance and the analysis of the spatial variation over building facades have a relevant interest for urban planning and building design. Numerical simulation of cities generates highly complex computational challenges. Many existing computer models should be adapted to consider the physical and social phenomena that are developed in urban environments.

[1]This chapter is based on the paper: Aguerre J. P., Fernández E., Besuievsky G. & Beckers B. *Computing urban radiation exchange: a sparse matrix solution*, October 2016. First International Conference on Urban Physics.

Our study on different kinds of urban model configuration shows that, for models composed of thousands of patches, we can provide an accurate approximation of the inverse radiosity matrix that can also be stored in main memory. In this way, it can be used to solve efficiently annual based radiative simulations. This is a promissory result, concerning its potential use for radiative exchange and analysis.

4.2 Our Proposal

This section presents the main ideas of the chapter, and proposes an algorithm to compute radiosity solutions exploiting the properties of the studied matrices. In the first place, the sparsity of **F** in city environments is analyzed. Secondly, a method to approximate the inverse of the radiosity matrix is proposed. After this, an experimental analysis is performed to test the proposed ideas.

4.2.1 Why not factorizing?

As stated in the previous chapter, a good approach to accelerate the radiosity calculations is to factorize the form factors matrix generated by the scene. When there is a high spatial coherence, the factorization allows to drastically reduce the amount of data, without losing much information about the model. This factorization is used to approximate the inverse of the radiosity matrix, which is used to compute radiosity solutions efficiently for a static geometry. Nevertheless, this is not always the case for many scenes.

City environments composed of thousands of buildings disposed over a terrain, do not have enough spatial coherence to exploit the previous properties. In these scenes, the singular values of the corresponding **F** matrices usually decay slowly, which prevents the use of factorization techniques to accelerate radiosity calculations. Therefore, other strategies need to be studied in order to work with big city models. To understand these concepts, Fig. 4.1 shows the singular values for two different scenes: the Cornell box, which has a high spatial coherence, and an example city model. As can be seen, the singular values decay rapidly in the high coherence case, while for the city environment they decay more slowly. For example, to get an error of 0.1, the city needs three times more singular values than the Cornell box.

4.2.2 Studying the Sparsity of a City's Form Factors Matrix

The density factor (sparsity) of a matrix is the fraction of non-zero elements over the total number of elements. In the form factor matrices, this factor depends on how many patches are seen from each patch: if patch j sees few patches, then row j of **F** has few elements different than zeros, and vice versa. Figure 4.2 shows two urban scenes where each patch is colored by checking how many elements are seen from it. For example, the upper elements on the tallest buildings are red while the ones on houses are blue. These results allow to predict that the **F** matrix corresponding to a city, where each patch sees few others, is very sparse.

The previous fact derives into the main conjecture of the present work: different kinds of cities have sparse **F** matrices with different density factors. This sparsity

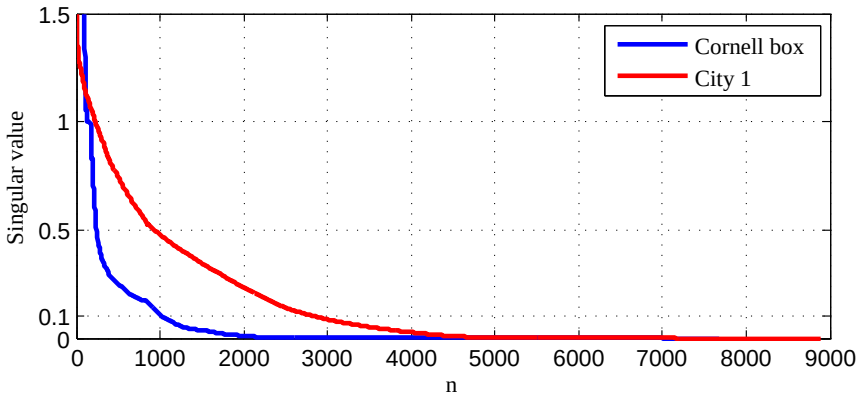
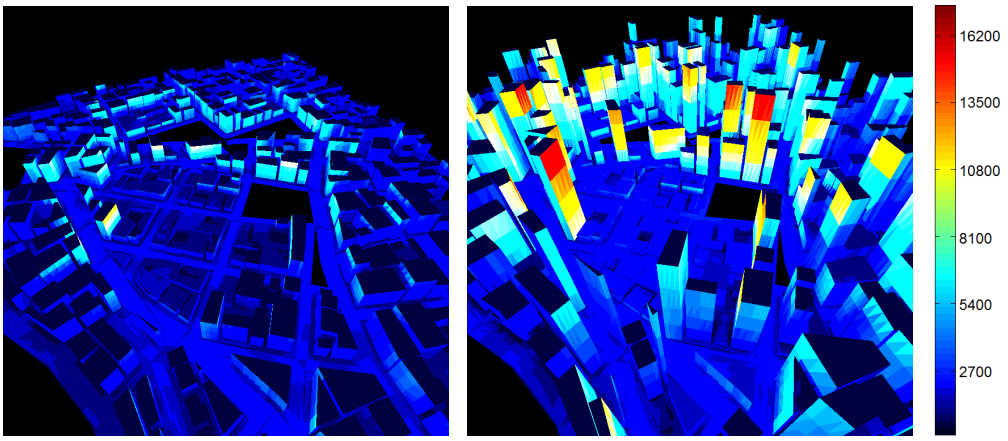


Figure 4.1: Singular values of $\mathbf{F}$ for the Cornell box scene and for City 1.



(a) City 1. (b) City 3.

Figure 4.2: Two example urban scenes. The color of a patch indicates the number of patches that are seen from it. Both models are composed of 142k patches.

depends on many factors. For example, orography, construction type, buildings disposition and heights are expected to have a great influence on the structure of the matrices. As a first step, we focus on the variation of building heights. A city with big variance on its buildings height (as a typical contemporary downtown with skyscrapers) should generate less sparse matrices than a city with uniformly elevated buildings (as, for example, Haussmann's Paris [Loyer, 1988]). In following sections the orography is also taken into account.

4.2.3 An Approximation of $\mathbf{M} = (\mathbf{I} - \mathbf{RF})^{-1}$

The inverse of a sparse matrix is usually a full matrix [Duff, 1977], where its calculation is computationally expensive, and has memory limitations for medium to large size matrices. However, in the case of the radiosity matrix, its inverse $\mathbf{M}$ has many

elements below a small threshold. This is because the indirect energy exchange between two occluded patches is predominantly small, in such way that it can be ignored in many cases. Therefore, in relation to $\mathbf{M}$, our proposal consists in finding a sparse approximation ($\tilde{\mathbf{M}} \approx \mathbf{M}$) that allows to compute low-error radiosity values.

In order to compute $\tilde{\mathbf{M}}$ efficiently, we use a method based on the work by Kontkanen [2006]. As described in Sec. 2.2.5, the algorithm is based on the use of Neumann series and a compression strategy based on removing all elements below a threshold ε. To obtain even sparser matrices, we apply this compression to $\mathbf{RF}$ before starting the process. Algorithm 5 describes the proposed method, where the function "remove" eliminates $|\mathbf{T}(i,j)| < \varepsilon, \forall ij$.

Algorithm 5 Calculate $\tilde{\mathbf{M}}$.

1: $\tilde{\mathbf{M}} = \mathbf{0}$
2: $\mathbf{T} = \mathbf{I}$
3: **while** $\mathbf{T} \neq \mathbf{0}$ **do**
4: $\quad \tilde{\mathbf{M}} = \tilde{\mathbf{M}} + \mathbf{T}$
5: $\quad \mathbf{T} = \mathbf{T}\,\mathbf{RF}$
6: $\quad \mathbf{T} = \text{remove}(\mathbf{T}, \varepsilon)$
7: **end while**

Once the sparse approximation is computed, it is relatively inexpensive to calculate the radiosity results for k different emissions (Eq. 4.1):

$$\begin{aligned} \tilde{\mathbf{M}} &\approx (\mathbf{I} - \mathbf{RF})^{-1} \\ \tilde{\mathbf{B}} &= \tilde{\mathbf{M}}\mathbf{E} \end{aligned} \tag{4.1}$$

where the i^{th} column of $\mathbf{E}$ is an emission and the i^{th} column of $\tilde{\mathbf{B}}$ is the approximation of its corresponding radiosity result, $\forall i \in 1\,..\,k$.

4.2.4 Daylight Simulation

Computing urban radiation exchange can have increasing interest if it is efficiently calculated. In this work, we apply the described techniques on urban daylight simulation, such kind of simulation has been applied in different fields such as design [Baker and Steemers, 2014], building energy consumption [Hviid et al., 2008] or ecology [Longcore and Rich, 2004].

In order to simulate the sky and its interaction with the city, a hemisphere containing the city is added to the model. This hemisphere is divided into m elements and each element is given its corresponding emittance, simulating the skylight. For more details on these concepts, refer to Sec. 2.5.2.

Once the sky is added to the model, we use the strategy described by Beckers [2013] to calculate the first bounce of light from the sky in the city. We use this as the urban emission, which allows us to work only with the form factors between patches of the city. For this purpose, the discrete radiosity equation (Eq. 2.3) is re-written in the following way:

$$B = E + \mathbf{RF}B \tag{4.2}$$

Let us separate the sky (index s) and city (index u) contributions in Eq. 4.2:

$$\begin{bmatrix} B_s \\ B_u \end{bmatrix} = \begin{bmatrix} E_s \\ E_u \end{bmatrix} + \begin{bmatrix} (\mathbf{RF})_{ss} & (\mathbf{RF})_{su} \\ (\mathbf{RF})_{us} & (\mathbf{RF})_{uu} \end{bmatrix} \begin{bmatrix} B_s \\ B_u \end{bmatrix}$$

The sky is considered as a black surface, with zero reflectance, while the city has no emission. Therefore, as $(\mathbf{RF}_{ss}) = 0$, $(\mathbf{RF})_{su} = 0$, and $E_u = 0$:

$$\begin{bmatrix} B_s \\ B_u \end{bmatrix} = \begin{bmatrix} E_s \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ (\mathbf{RF})_{us} & (\mathbf{RF})_{uu} \end{bmatrix} \begin{bmatrix} B_s \\ B_u \end{bmatrix}$$

In the previous equation, $B_s = E_s$. This leads to the following statement:

$$B_u = (\mathbf{RF})_{us}B_s + (\mathbf{RF})_{uu}B_u = (\mathbf{RF})_{us}E_s + (\mathbf{RF})_{uu}B_u$$

Now, grouping the radiosities from both sides:

$$(\mathbf{I} - (\mathbf{RF})_{uu})B_u = (\mathbf{RF})_{us}E_s$$

The left side of this equation is the radiosity matrix $(\mathbf{I} - (\mathbf{RF})_{uu})$ times the radiosity result for the city. Therefore, following Eq. 2.3, the new emission is $E = (\mathbf{RF})_{us}E_s$, which is the first bounce of light coming from the sky in the city.

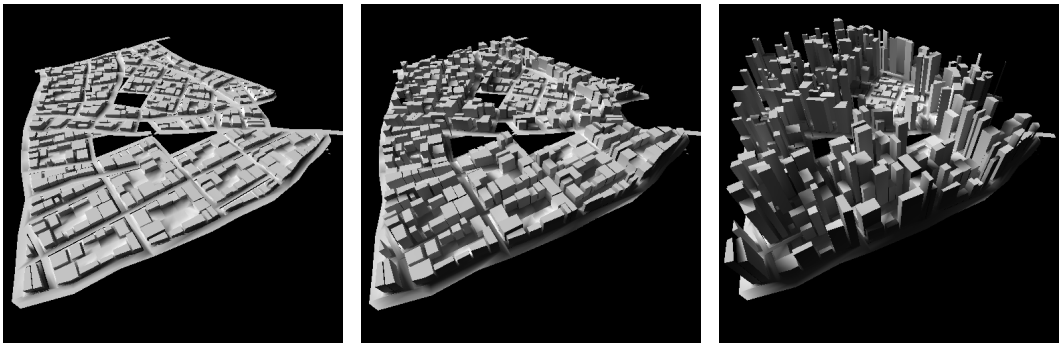
4.3 Experimental Analysis

The results of the presented set of experiments were conducted on a desktop computer, with Intel quad-core i7 processor and 16 Gbytes RAM. The calculation of each $\mathbf{F}$ matrix was performed using the hemi-cube technique with a resolution of 512 $\times$ 512 pixels, where the graphic component was executed on a NVIDIA GeForce-780 GPU processor. The code was implemented on C++, OpenGL, CUDA [Kirk and Hwu, 2010], and MATLAB [MATLAB, 2010].

4.3.1 Example City Models

The following analysis is performed using three different urban scenes, which are generated from the same cadastral plan. The first model contains only flat houses, the second low and middle-sized buildings, and the third is composed of different sized buildings, including tall skyscrapers. The urban scenes can be observed in Fig. 4.3.

As can be seen, the first model has a small variance on building heights, while the third one has a big variance. The three models are composed of 8897 patches.



(a) City 1. (b) City 2. (c) City 3.

Figure 4.3: Three different urban scenes to experiment with.

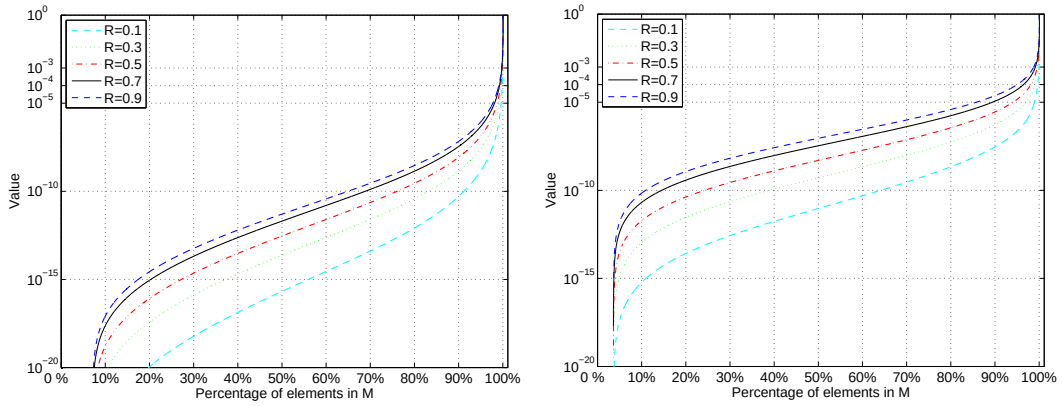
4.3.2 Sparsity Results for F and $\tilde{M}$

We show the sparsity results for the described city models. For each model, three variants are studied: the original (n=8897), dividing each patch into 4 (n=35588) and into 16 (n=142352), where n is the number of patches. This allows to analyze the proposed algorithm for bigger models, as well as the effect of dividing patches in the sparsity factor. The calculation of $\mathbf{F}$ takes about 20s, 90s, and 600s for n=8897, 35588, and 142352, respectively.

First of all, we explore the density of $\mathbf{M}$ matrices (n=8897, inverted with MATLAB) and the distribution of their elements, for different reflectivity indexes $\mathbf{R}$ (Fig. 4.4). It can be appreciated that most of the matrices elements are non-zero, and also that most of them have very small values. An increment in the value of $\mathbf{R}$ is related to an increment in the values of the matrix elements. The matrices of City 1 have smaller elements than those related to City 3. For example, only 3% of the elements corresponding to City 1 are greater than 10^{-5} for $\mathbf{R} <= 0.7$, while approximately 10% of the elements in City 3 satisfy this property. In the rest of the paper, a reflectivity index of 0.7 is used. This value is higher than the expected for cities, but it is useful for challenging the sparsity of the matrices $\tilde{\mathbf{M}}$.

The sparsity results and memory storage for $\mathbf{F}$ and $\tilde{\mathbf{M}}$ matrices can be seen in Table 4.1. As expected, the density factor of $\mathbf{F}$ grows as the city model becomes less homogeneous, which implies the use of a larger memory space. Nevertheless, the density reported in the worst case (City 3 with $n = 8897$) signifies a storage of 1.23% of the total elements of the matrix. On the other hand, the density factors are shorter for finer meshes of the same city model.

The density factor of $\tilde{\mathbf{M}}$ has a similar behavior to that described for $\mathbf{F}$. Also, for all cases, the density increases as the threshold ε becomes smaller. The memory required to store the sparse matrices $\mathbf{F}$ and $\tilde{\mathbf{M}}$ is always much less than its full version, for every of the test cases executed.



(a) City 1. (b) City 3.

Figure 4.4: Distribution function of the **M** elements, for different reflectivity indexes and cities.

		Density	Density of $\tilde{\mathbf{M}}$			Memory size (GB)			Gain	
	n	of **F**	$\varepsilon = 10^{-3}$	10^{-4}	10^{-5}	Full	**F**	$\tilde{\mathbf{M}}$ (10^{-5})	**F**	$\tilde{\mathbf{M}}$ (10^{-5})
City 1	8897	0.68 %	0.22 %	0.61 %	1.48 %	0.60	0.01	0.02	60×	30×
	35588	0.48 %	0.10 %	0.31 %	0.83 %	9.66	0.09	0.17	107×	57×
	142352	0.35 %	0.04 %	0.15 %	0.45 %	154.60	1.05	1.46	147×	106×
City 2	8897	0.84 %	0.30 %	1.01 %	2.58 %	0.60	0.01	0.03	60×	20×
	35588	0.60 %	0.15 %	0.51 %	1.41 %	9.66	0.12	0.28	80×	34×
	142352	0.44 %	0.05 %	0.24 %	0.75 %	154.60	1.34	2.42	115×	64×
City 3	8897	1.23 %	0.55 %	2.07 %	6.24 %	0.60	0.01	0.08	50×	7.5×
	35588	0.87 %	0.20 %	0.88 %	2.98 %	9.66	0.17	0.59	57×	16×
	142352	0.64 %	0.06 %	0.36 %	1.40 %	154.60	1.94	4.52	80×	34×

Table 4.1: Density and memory size of the form factors matrices and the approximated inverse. The gain equals to the full memory size over the sparse memory size.

4.3.3 Execution Times

In order to study the computational performance of the proposed algorithm, we calculate the daylight illumination for a whole year. For this, we use the 3 city models with the 3 mesh variants, along with 3650 sky configurations. There are several ways to mesh the sky [Mardaljevic, 1999, Tregenza, 1987], though we use a simple division with parallels and meridians (m=132) as a proof of concept. The obtained data is compared to the execution times of solving the same radiosity problem iteratively, using the Jacobi iteration (Eq. 2.5).

Table 4.2 shows the obtained execution times for the described test cases. Table 4.3 shows the speedup over the Jacobi based method. It is important to highlight that this speedup is calculated taking into account both time to compute $\tilde{\mathbf{M}}$ and time to compute $\tilde{\mathbf{M}}\mathbf{E}$. That is: Speedup $= T_J/(T_{\tilde{\mathbf{M}}} + T_{\tilde{\mathbf{M}}\mathbf{E}})$, where T_J is the execution time needed to calculate the radiosity using Jacobi. For each case, the number of iterations (light bounces) used to compute T_J is the same than the used for $\tilde{\mathbf{M}}$.

As can be appreciated in the tables, the execution times depend highly on the correspondent density factor of $\tilde{\mathbf{M}}$. The sparser this matrix is, the lower the execution times are. For the considered example problem, the proposed algorithm works faster than the Jacobi iteration method for all the test cases.

		Time for $\tilde{\mathbf{M}}$ ($T_{\tilde{\mathbf{M}}}$)			Time for $\tilde{\mathbf{M}}\mathbf{E}$ ($T_{\tilde{\mathbf{M}}\mathbf{E}}$)			Time for Jacobi (T_J)		
	n	$\varepsilon = 10^{-3}$	10^{-4}	10^{-5}	$\varepsilon = 10^{-3}$	10^{-4}	10^{-5}	$\varepsilon = 10^{-3}$	10^{-4}	10^{-5}
City 1	8897	0.17s	0.68s	2.51s	0.52s	1.22s	2.88s	32.78s	51.68s	79.77s
	35588	1.10s	7.08s	32.80s	3.72s	10.20s	26.70s	334.51s	594.43s	993.65s
	142352	7.33s	60.90s	440.00s	26.40s	89.80s	264.00s	3710.30s	6615.73s	14150.40s
City 2	8897	0.26s	1.49s	5.80s	0.71s	1.99s	4.99s	43.84s	76.56s	117.61s
	35588	2.06s	15.20s	79.60s	5.17s	16.20s	45.70s	530.68s	963.98s	1691.55s
	142352	11.80s	136.00s	1180.00s	34.20s	140.00s	442.00s	5704.00s	11509.20s	27411.80s
City 3	8897	0.52s	4.10s	18.40s	1.09s	4.04s	11.90s	72.93s	129.43s	203.01s
	35588	3.35s	38.00s	264.00s	6.91s	29.10s	100.00s	826.96s	1758.02s	3385.20s
	142352	15.80s	306.00s	4070.00s	37.70s	223.00s	837.00s	10058.00s	22323.80s	72132.90s

Table 4.2: Execution times (in seconds) of radiosity calculations for the test cases. Both $T_{\tilde{\mathbf{M}}}$ and T_J are computed using 40 iterations (light bounces).

		Speedup $= T_J/(T_{\tilde{\mathbf{M}}} + T_{\tilde{\mathbf{M}}\mathbf{E}})$		
	n	$\varepsilon = 10^{-3}$	10^{-4}	10^{-5}
City 1	8897	47.5 ×	27.2 ×	14.8 ×
	35588	69.4 ×	34.4 ×	16.7 ×
	142352	110.0 ×	43.9 ×	20.1 ×
City 2	8897	45.2 ×	22.0 ×	10.9 ×
	35588	73.4 ×	30.7 ×	13.5 ×
	142352	124.0 ×	41.7 ×	16.9 ×
City 3	8897	45.3 ×	15.9 ×	6.7 ×
	35588	80.6 ×	26.2 ×	9.3 ×
	142352	188.0 ×	42.2 ×	14.7 ×

Table 4.3: Time speedup for radiosity calculations.

4.3.4 Radiosity Results

In this section we study the impact of the proposed algorithm on the radiosity results. We use 132 different sky configurations, each one with a unique sky tile illuminating the scene, to compute 132 radiosity solutions of the city. Given a patch of the city, the radiosity value calculated for each of the sky configurations is related to the concept of Daylight Coefficient [Tregenza and Waters, 1983]. The linear combinations of the radiosity solutions for the 132 skies allow to find the radiosity of the city for any other sky configuration. Fig. 4.3 shows the radiosity values of the three cities for the same sky configuration, when ε=10^{-5}.

Comparison with Jacobi

Table 4.4 shows the relative errors of the 132 radiosities obtained, comparing $\widetilde{B}$=$\widetilde{\mathbf{M}}E$ to the solution (B_J) of the Jacobi iteration methodology (Eq. 2.5). The initial emission is the first bounce of the light emitted from the sky (Sec. 4.2.4). The mean, standard deviation and maximum values are reported. As expected, the error

gets smaller as the city homogeneity increases and as the truncation factor decreases. For every case, the standard deviation is small, as well as the maximum error is close to the mean value.

		Relative error of $\widetilde{B}$: $\frac{\|\widetilde{B}-B_J\|}{\|B_J\|}$ ($\times 1000$)								
		$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-4}$			$\varepsilon = 10^{-5}$		
	n	μ	σ	Max	μ	σ	Max	μ	σ	Max
City 1	8897	27.80	2.29	29.40	7.83	0.73	8.29	1.88	0.19	2.00
	35588	53.00	3.17	55.70	16.90	1.25	17.90	4.53	0.36	4.78
	142352	48.60	5.09	52.80	15.30	1.88	16.70	4.15	0.57	4.57
City 2	8897	87.70	6.44	95.20	31.70	2.99	34.20	9.50	1.04	10.30
	35588	86.50	9.15	97.80	35.00	5.42	39.80	11.00	2.20	12.80
	142352	134.00	11.70	154.00	61.80	6.24	69.30	23.00	3.41	25.90
City 3	8897	92.60	3.92	98.20	33.30	2.11	35.30	9.86	0.72	10.50
	35588	141.00	8.67	155.00	59.30	4.80	64.20	19.70	1.93	21.40
	142352	189.67	17.74	221.03	99.20	8.45	113.05	42.14	4.77	47.12

Table 4.4: Radiosity errors for the test cases (all numbers are $\times 1000$).

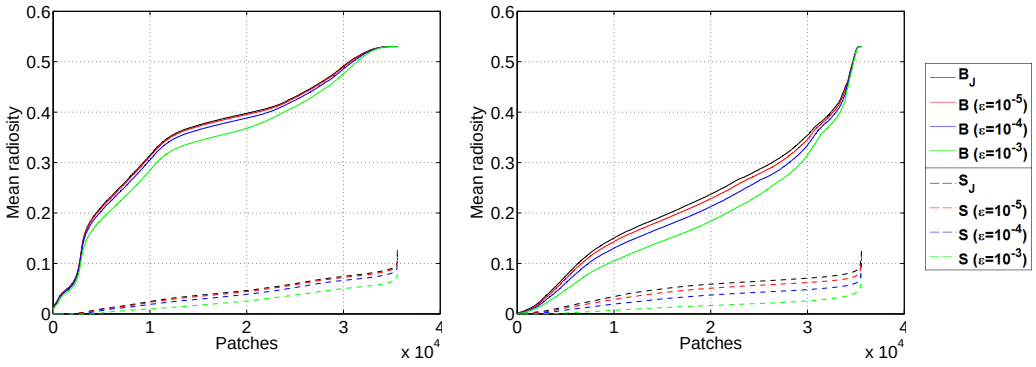
3^{rd} and successive light bounces

The indirect lighting is an important component of the illumination of the city. Next, a test is performed to measure the precision of the radiosity results without taking the direct and first bounces into account. In Fig. 4.5, the average radiosity values for the 132 different sky configurations are shown (for two city models). All the radiosity curves are sorted from lowest to highest values. In both plots, we present the results using the Jacobi iteration method for computing the full radiosity (B_J) and the radiosity considering only the third and successive bounces (S_J). Also, we show the same results using the proposed algorithm for different truncation thresholds.

As can be appreciated, the illumination is much higher in B_J than in S_J, because the first two bounces are the main component of the total radiosity. Nevertheless, the rest of the bounces together are not negligible, which means that they cannot be discarded in the calculations. Taking a closer look into the results of the proposed algorithm, it is evident that a higher truncation threshold implies a higher error. When compared to the Jacobi solution, the results seem close enough for most practical applications, when ε=10^{-4} and 10^{-5}. Finally, the absolute errors $\|B-B_J\|$ seem to have similar values to the errors $\|S-S_J\|$, which leads to think that most of the radiosity error is produced after the 2^{nd} bounce.

4.3.5 Orography

Orography is the study of the topographic relief of a terrain [Glickman, 2000]. Identifying features and recognizing typical landform patterns are part of the field. It has a major impact on several subjects such as heat exchange, air movement and daylighting [Collier, 2006]. Topographic studies may have different goals: geological exploration, planning and construction for civil engineering projects, or even neuroimaging [Chakraborty, 2005].

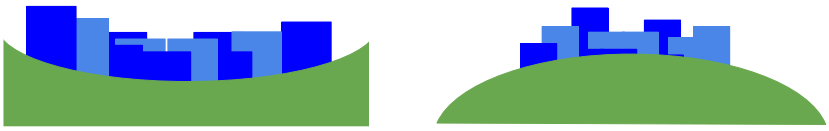


(a) City 1. (b) City 3.

Figure 4.5: Comparison of radiosity results between Jacobi ($\mathbf{B}_j$), third and successive bounces ($\mathbf{S}_j$), and their approximations using different thresholds.

This field becomes very important for city planning purposes, because building strategies depend highly on the properties of the surface to construct on. Moreover, in an urban environment, there is a relation between orography and daylighting [Johnson, 1981]. Different terrains lead to different occlusions between buildings, which affects the light and heat interaction within the city. In this section, we study the effect of changing the terrain where the city lies in. In particular, the density factor of the correspondent form factors matrices is measured.

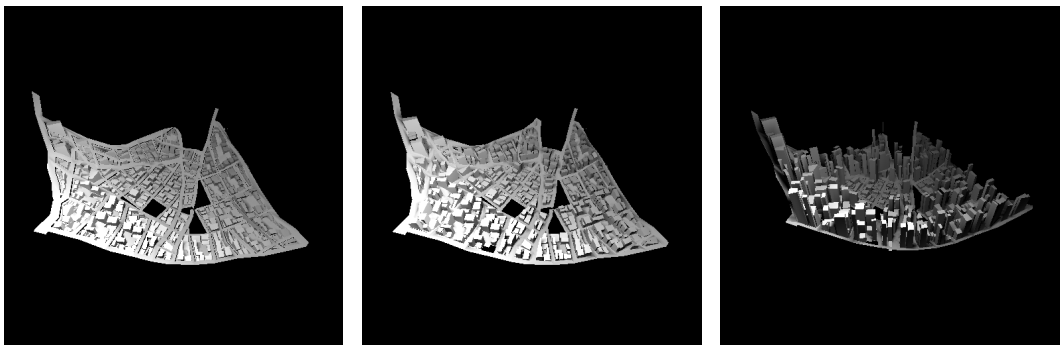
To perform this study, three different terrains are used: a flat terrain, a hill terrain, and a valley terrain. Then, the three city models presented at Fig. 4.3 are skewed according to the terrains shapes, creating six new scenes. Fig. 4.6 illustrates the different terrains. Example pictures of the new models are shown in Fig. 4.7.



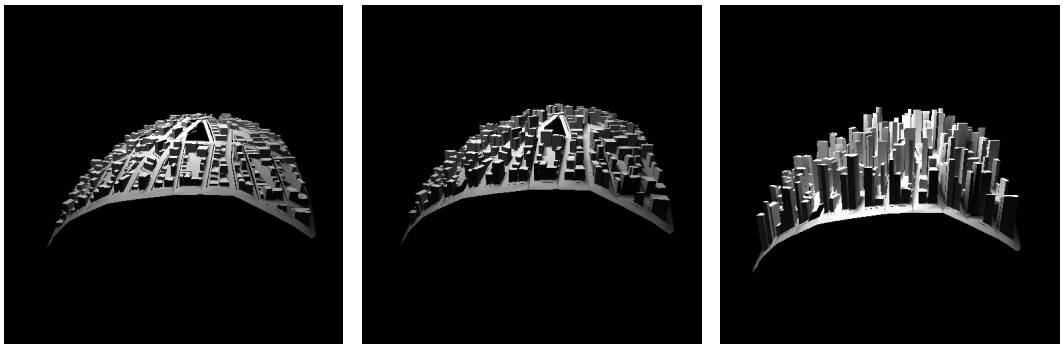
(a) City over valley. (b) City over hill.

Figure 4.6: Illustration of the valley and hill terrains.

As shown in Sec. 4.3.3, the execution times of computing radiosity solutions depends highly on the density factor of the matrix $\mathbf{F}$. Regardless of the strategy to use, this property helps to reduce the memory usage and the number of operations to perform. Fig. 4.2 presented the number of patches that are seen from each patch in a flat city, and this affected directly on the sparsity results. The same study can be performed over the new scenes, in order to observe their patch classification graphically. Fig 4.8 presents this result for the three different variations of City 3, using 35588 patches. The valley terrain leads the city patches to see much more elements than over flat or hill terrains. This is because the "U" shape of the landform creates more visibilities between patches.

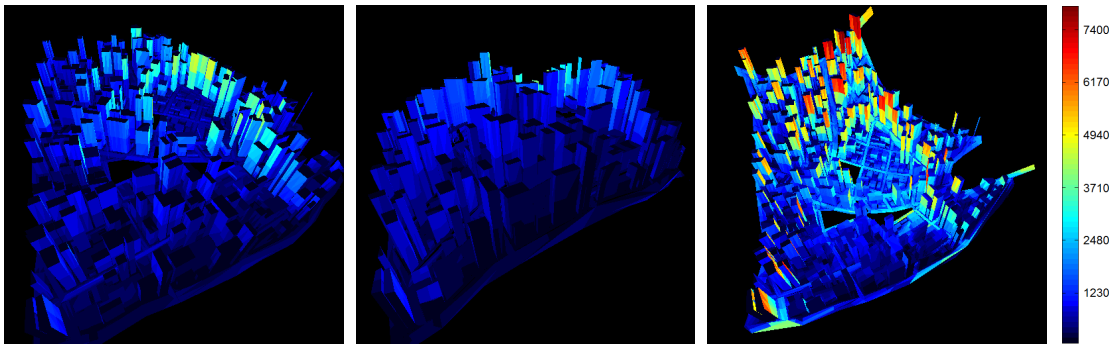


(a) City 1 over valley (b) City 2 over valley (c) City 3 over valley



(d) City 1 over hill (e) City 2 over hill (f) City 3 over hill

Figure 4.7: Six urban scenes generated by posing the three cities into different terrains.



(a) City 3 flat. (b) City 3 over hill. (c) City 3 over valley.

Figure 4.8: Three variations of City 3. The color of a patch indicates the number of patches that are seen from it. All three models are composed of 35k patches.

Next, the sparsity results are reported at Table 4.5, combining terrains, city types and number of patches. Once again, the results show the same behavior as the patch classification shown in Fig. 4.8. The valley terrain makes **F** much more

dense (reaching a maximum value of 14.3%), while the hill landform produces similar results than the flat one. Moreover, for City 3, results using the hill terrain are less dense than using the flat. These results can be explained by taking a look into the different types of patches in the models. For example, in the case of patches belonging to a building's ceiling, the visibility increases drastically in the lower part of a valley terrain, because almost all the city is now seen from it. On the other hand, their visibility is reduced when they are located at the top of a hill. These facts, along with other similar phenomena, have a direct incidence in the density results for **F**.

	n	Flat	Valley	Hill
City 1	8897	0.68%	14.3%	0.87%
	35588	0.48%	9.10%	0.60%
	142352	0.35%	4.87%	0.42%
City 2	8897	0.84%	6.48%	0.96%
	35588	0.60%	4.19%	0.66%
	142352	0.44%	2.48%	0.47%
City 3	8897	1.23%	2.99%	1.14%
	35588	0.87%	1.97%	0.79%
	142352	0.64%	1.31%	0.58%

Table 4.5: **F** density factors for different cities, orographies and n values.

4.4 Alternative optimizations

The described algorithms have shown good results for the tested city models. The sparse radiosity matrix allows to speedup the computations, while the method for approximating its inverse is suitable when the geometry is static and several thousand radiosity calculations must be performed. However, there are still many optimizations to be added to the algorithms. In this section, we propose a variation to accelerate the form factors calculation on highly occluded environments, as well as a corrective technique to improve the precision of radiosity results when using the approximated inverse of the radiosity matrix.

4.4.1 Form factors generation

In the previous sections, the form factors were generated using the hemi-cube technique. This technique is thoroughly described in Chapter 2. An important variable in this algorithm is the hemi-cube resolution, which determines the precision used in the calculation. A bigger resolution leads to good form factors approximations, but implies heavier computations. This is because the graphic hardware that executes the rendering (using the Z-buffer technique) works slower for larger image resolutions, which makes sense since more calculations at pixel level need to be performed. In this section, the effect of changing the hemi-cube resolution for computing the form factors of a city is studied.

As stated in Sec. 4.2.2, the sparsity of $\mathbf{F}$ is related to the number of patches that are seen from each patch on the scene. For every view link between two patches, $\mathbf{F}$ has a new element different than zero. Given the obtained experimental results, it can be said that every patch in a regular city scene is view-linked with few others. This fact leads to think that, when using the hemi-cube algorithm to compute the form factors, there should be a big coherence at pixel level in the resultant images. If this is true, lower resolutions could be used with small errors in the form factors results. Fig 4.9 presents an example hemi-cube view using different resolutions.

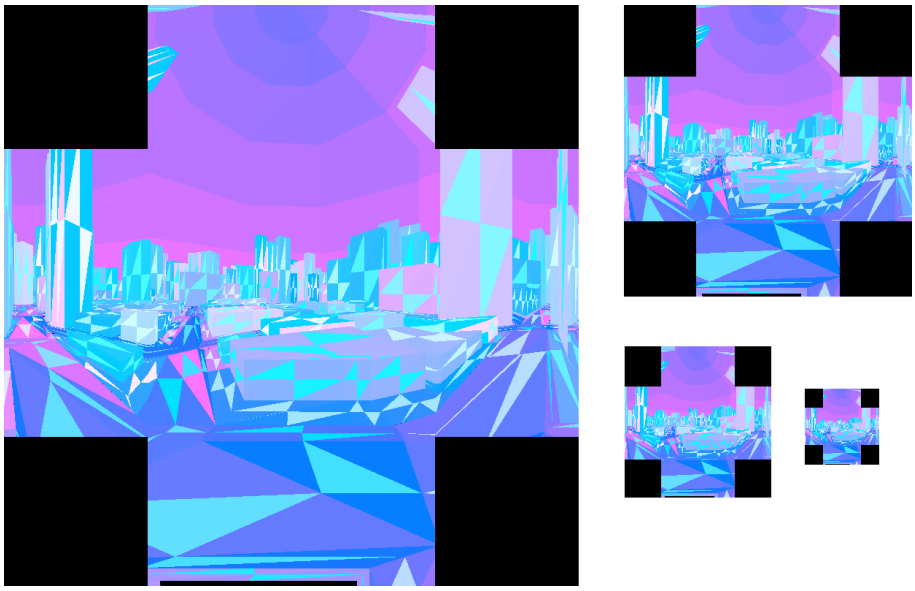


Figure 4.9: Same hemi-cube view with different resolutions.

To test this hypothesis, the following experiment is performed: for the three cities that were presented, the matrix $\mathbf{F}$ is generated using different hemi-cube resolutions. The sparsity results, execution times and obtained relative errors are reported in Table 4.6. $\mathbf{F}_{res}$ is the form factors matrix generated using a hemi-cube with resolution equal to $res \times res$. The precision is calculated comparing each matrix $\mathbf{F}_r$ to $\mathbf{F}_{512}$, which is the highest resolution used.

res	Density of $\mathbf{F}_{res}$			Speedup to $\mathbf{F}_{512} \approx 90s$			$\|\mathbf{F}_{512} - \mathbf{F}_{res}\|/\|\mathbf{F}_{512}\|$		
	City 1	City 2	City 3	City 1	City 2	City 3	City 1	City 2	City 3
512	0.48%	0.60%	0.87%	1.0 ×	1.0 ×	1.0 ×	0	0	0
256	0.34%	0.46%	0.69%	1.8 ×	1.8 ×	1.8 ×	0.0072	0.0072	0.0076
128	0.22%	0.32%	0.49%	2.8 ×	2.8 ×	2.7 ×	0.0610	0.0125	0.0121
64	0.12%	0.19%	0.30%	3.1 ×	3.1 ×	3.1 ×	0.0642	0.0360	0.0346
32	0.06%	0.09%	0.15%	3.2 ×	3.2 ×	3.2 ×	0.1444	0.1281	0.1279

Table 4.6: Generation of $\mathbf{F}$ using different hemi-cube resolutions, $n = 35588$.

For the test scene used, the relative errors are not very high for resolutions 256 and 128, and up to 2.8× speedups are obtained. These results are obtained for

the three different city models, which seems promising if this technique needs to be applied to other city models. On the other hand, for resolutions 64 and 32, the errors are larger, and the speedups do not grow significantly.

With regard to the density factors, lower resolution hemi-cubes generate more sparse matrices, because the city patches that are far from the hemi-cube tend to disappear from the view. As shown in Sec. 4.3.3, the sparsity of $\mathbf{F}$ has a direct impact on the execution times and memory consumption when solving radiosity. Therefore, using lower resolutions can be a good alternative to speedup not only the form factor calculation but the radiosity computation.

4.4.2 Considering discarded elements

The main error of the algorithm that computes $\tilde{\mathbf{M}}$ is related to the removal of terms below the input threshold. In the previous experimental analysis, these elements were eliminated and not taken into account. In this section, we propose an alternative method for including them in the radiosity calculations in order to improve the precision of the solutions.

Let's consider $\mathbf{\Delta}$ to be the error produced by the method to approximate the inverse of the radiosity matrix ($\mathbf{M}$):

$$\mathbf{M} = \widetilde{\mathbf{M}} + \mathbf{\Delta} \tag{4.3}$$

By construction, $\mathbf{\Delta}$ can be approximated by accumulating the elements discarded in the Neumann iteration (elements below the input threshold). Then, similarly to Eq. 4.1, the radiosity vector B corresponding to an emission E can be approximated in the following manner:

$$B \approx (\widetilde{\mathbf{M}} + \mathbf{\Delta})E = \widetilde{\mathbf{M}}E + \mathbf{\Delta}E \tag{4.4}$$

Because of the results shown at Fig. 4.4, we can predict that the matrix $\mathbf{\Delta}$ is a full matrix with small elements. This fact prohibits its calculation. Therefore, the term $C = \mathbf{\Delta}E$ needs to be approximated using other techniques.

The i-th element of C is computed in the following way, where $\delta_{i,j}$ are the elements of $\mathbf{\Delta}$, and e_i is the i-th element of E:

$$C_i = \delta_{i,1}e_i + \delta_{i,2}e_i + ... + \delta_{i,n}e_i \ , \ \forall i \in 1..n \tag{4.5}$$

When the values of e_i are similar (for instance, in the city there are many patches receiving direct sky and sun light), they can be approximated by their mean $\mu(E)$=$\sum_{i=1}^{n} e_i$. Therefore, Eq. 4.5 can be substituted by Eq. 4.6.

$$C_i \approx (\delta_{i,1} + \delta_{i,2} + ... + \delta_{i,n})(\frac{e_1 + e_2 + .. + e_n}{n}) = \mu(E)\sum_{j=1}^{n}\delta_{i,j} \tag{4.6}$$

Now, $\mathbf{\Delta}$ is approximated accumulating the discarded elements by row, at each iteration. This is equivalent to accumulate the discarded elements by scene patch, which gives an idea of how much error is produced by the technique at every polygon. Besides, the emission is approximated by its mean value, which can be an inaccurate approach if only few patches are emitters.

Algorithm 6 shows the new scheme, which is very similar to the previous algorithm (Algorithm 5). $\mathbf{\Delta}$ is the matrix that contains all the elements that were removed from $\mathbf{T}$ at each iteration. The operation "sum" performs the summation by row, and the result is accumulated into vector δ.

Algorithm 6 Calculate $\tilde{\mathbf{M}}$ and accumulate discarded elements.

1: $\tilde{\mathbf{M}} = \mathbf{0}$
2: $\delta = \mathbf{0}$
3: $\mathbf{T} = \mathbf{I}$
4: **while** $\mathbf{T} \neq \mathbf{0}$ **do**
5: $\quad \tilde{\mathbf{M}} = \tilde{\mathbf{M}} + \mathbf{T}$
6: $\quad \mathbf{T} = \mathbf{T}\,\mathbf{R}\mathbf{F}$
7: $\quad [\mathbf{T}, \mathbf{\Delta}] = \text{remove}(\mathbf{T}, \varepsilon)$
8: $\quad \delta = \delta + \text{sum}(\mathbf{\Delta})$
9: **end while**

The vector δ is calculated at almost no computational cost, and is then applied to compute the radiosity vector. Once $\tilde{\mathbf{M}}$ and δ are computed, and given an emission vector E, the radiosity solution is calculated following Eq. 4.7. Here, $\mu(E)$ means the average emission value.

$$\widetilde{B} = \widetilde{\mathbf{M}}E + \delta\mu(E) \tag{4.7}$$

As can be seen, this strategy adds an average energy value to the radiosity of each patch, weighted by the sum of terms discarded when computing $\tilde{\mathbf{M}}$. Next, some experimental analyses are performed to test this technique in daylighting calculations.

The new algorithm is re-executed for the three city examples presented before (over flat terrain). In the same way than Sec. 4.3.4, we use 132 different sky configurations, each one with a unique sky element illuminating the scene, to compute 132 radiosity solutions of the city. Table 4.7 shows the relative errors of the 132 radiosities obtained, comparing $\widetilde{B} = \widetilde{\mathbf{M}}E$ to the solution (B_J) of the Jacobi iteration methodology (Eq. 2.5). The mean, standard deviation and maximum values are reported.

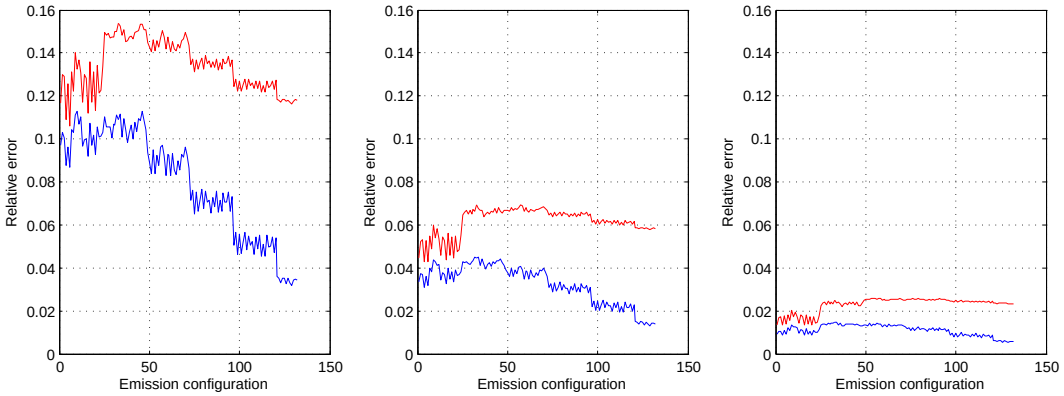
The new results are much better than the previous (see Table 4.4). The mean error is lower for all executed cases, which confirms that the new strategy allows to reduce the relative error in radiosity results. Additionally, despite the fact that the new standard deviations are greater, the maximum error values are still lower than before.

In order to get a graphical view of the new error, Fig. 4.10 presents the relative error for the 132 emissions, using both previous and new algorithms. These plots

		Relative error of $\widetilde{B}$: $\frac{\|\widetilde{B}-B_J\|}{\|B_J\|}$ ($\times 1000$)								
		$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-4}$			$\varepsilon = 10^{-5}$		
	n	μ	σ	Max	μ	σ	Max	μ	σ	Max
City 1	8897	11.30	2.10	15.30	2.73	0.53	3.74	0.58	0.11	0.80
	35588	23.00	4.82	32.00	6.31	1.38	8.96	1.51	0.35	2.19
	142352	22.40	4.62	30.70	6.26	1.32	8.69	1.57	0.34	2.20
City 2	8897	43.00	9.69	60.50	13.70	3.01	19.40	3.80	0.85	5.38
	35588	48.30	13.30	67.70	17.20	4.20	23.20	4.98	1.10	6.57
	142352	79.30	24.50	113.00	32.30	9.08	45.20	11.10	2.55	14.80
City 3	8897	42.90	10.60	61.80	13.30	3.11	19.20	3.42	0.85	5.05
	35588	74.10	18.70	105.00	27.30	6.22	38.70	8.15	1.84	11.60
	142352	117.16	38.09	177.58	55.12	16.81	79.35	20.92	5.64	29.41

Table 4.7: Radiosity errors using the new algorithm (all numbers are $\times 1000$).

correspond to City 3 with $n = 35588$, but it is necessary to highlight that the rest of the environments show similar results. It can be observed that the new relative errors are lower than the previous, for every emission configuration and ε value.



(a) $\varepsilon = 10^{-3}$ (b) $\varepsilon = 10^{-4}$ (c) $\varepsilon = 10^{-5}$

Figure 4.10: Relative error: $\frac{\|\widetilde{B}-B_J\|}{\|B_J\|}$ for the 132 emissions. These results correspond to City 3 with $n = 35588$. The red plot is the error for the original algorithm, while the blue plot is the error for the new algorithm.

4.4.3 A better approach for a small number of emitters

The computation of $\mathbf{M}$ is useful when a big portion of the total amount of patches are potential emitters. Nevertheless, when only a small number s of patches are emitters (for example, a coarse sky for daylighting, or a few luminaries for artificial lighting), another pre-computation step can be executed to speed-up the process. A $n \times s$ matrix $\mathbf{B}_s$ containing s radiosity results is computed using the Jacobi iteration (Eq. 4.8) for s emission vectors ($\mathbf{E}_s$ is an $n \times s$ matrix).

$$\mathbf{B}_s^{(i+1)} = \mathbf{RFB}_s^{(i)} + \mathbf{E}_s \quad , \quad \text{where } \mathbf{B}_s^{(0)} = \mathbf{E}_s \tag{4.8}$$

The i^{th} column of $\mathbf{B}_s$ corresponds to the radiosity result of the i^{th} potential emitter patch emitting only by itself (the i^{th} column of $\mathbf{E}_s$ has all zeros except for a 1 in the index of the i^{th} patch). The contribution of each emitter does not interfere with the contribution of other emitters, therefore the radiosity result B corresponding to a set of emitters can be computed by a linear combination of the previous vectors ($B = \mathbf{B}_s S$, where S is a vector of length s containing the emission of each emitter). After the computation of $\mathbf{B}_s$ using Eq. 4.8, many thousand emission configurations can be solved in very short execution times.

This approach was exploited for computing daylighting in our work [Fernandez et al., 2016], where City 3 with a Tregenza sky dome (145 tiles [Tregenza, 1987]) was used along with an hourly test reference weather data [EERE, 2016] from London. The daylighting results were an input for optimizing opening shapes for a standard office [Nabil and Mardaljevic, 2005], located at several positions in the same building. The results show that the city illumination affects the window configuration drastically, as well as it has a direct impact on the number of hours in the year that have a satisfactory lighting condition.

In this work, we used City 3 with 142k patches. The calculation of $\mathbf{F}$ takes about 600s. The matrix $\mathbf{F}$ is sparse, with only 0.64% of non-zero elements. Its sparseness allows to store it in main memory, with a size of almost 2 GBytes. The calculation of $\mathbf{B}_s$ considering each of the 145 sky tiles as emitters takes about 140s after 5 iterations of Eq. 4.8. This iterative process was also executed 42 times, stopping when $\mathbf{B}_s^{(i+1)}$=$\mathbf{B}_s^{(i)}$ was satisfied. This last process takes about 1000s to finish. Once $\mathbf{B}_s$ is calculated, the illumination of the city for 3650 emissions (10 working hours per day) is computed in less than 5 seconds in a regular desktop computer.

4.5 Overview and conclusions of the chapter

With the goal of studying environments with a high occlusion rate between its elements, several city models were used. The correlation between the characteristics of a city and the sparsity of its form factors matrix ($\mathbf{F}$) was analyzed. Additionally, an iterative method to approximate the inverse of the radiosity matrix was proposed, based on the use of Neumann Series. At each iteration, all elements below a threshold are removed, which leads to very sparse matrices.

The proposed method was intended for daylight simulation. There are several strategies to simulate the skylight. We used a dome model composed of not-many tiles, where each one represents a section of the celestial sphere. Then, meteorological data can be used to determine the energy contribution of each tile. Once a sparse approximation of the radiosity matrix is computed, the radiosity corresponding to a sky configuration can be calculated using a single matrix-vector operation. This allows to determine the city illumination for a whole year of daylight at reasonable execution times.

In the experimental analysis, the sparsity of $\mathbf{F}$ was analyzed for three different types of cities with different variation on their building heights. Also, various number of patches were tested for each scene. We found that more homogeneous cities produce more sparse $\mathbf{F}$ matrices. The matrices are sparse enough to be stored in the main memory of a desktop computer, considering city scenes that contain up to

140k patches.

Another result is the calculation of a sparse approximation to the inverse of the radiosity matrix ($\tilde{\mathbf{M}}$). This approximation is based on the use of Neumann series and the elimination of all terms with lower values than a given threshold. $\tilde{\mathbf{M}}$ is also sufficiently sparse as to be stored in main memory.

These matrices were tested doing radiosity calculations for several sky configurations. We compared the results with a Jacobi iteration method, and found that the radiosities have low relative errors. This is accomplished in much better execution times than the Jacobi method. For example, it reaches a 42.2× speedup to find a radiosity solution with less than 0.1 relative error.

Regarding orography, it was found that the terrain were the city lies in has a big influence on $\mathbf{F}$. For example, if the landform is a valley, $\mathbf{F}$ becomes more full than in a flat terrain. On the other hand, when using hill-type topographies, the results do not vary significantly with respect to flat-type.

Chapter 5

Conclusions and future work

This chapter presents the conclusions of the research work related to the hierarchical factorization technique, and to the proposed methods for radiosity calculations in urban environments. The principal lines of future work emerged during the development of this thesis are described next.

5.1 Conclusions

This thesis was dedicated to the study of different strategies applied to the resolution of the radiosity equation. The goal of achieving fast global illumination solutions is still a subject of study in computer graphics, which has emphasized the development of new ray tracing based methods in recent years. As stated in Chapter 2, several works related to radiosity have been developed, exploiting general and particular opportunities and advantages. These are usually based on iterative methods, which slow down the process, and require large volumes of memory to achieve precise solutions. In this way, we studied techniques to reduce the representations of the radiosity matrix for two different types of scenes: high spatial coherence models, and highly occluded environments.

Since high spatial coherence scenes are related to low numerical rank form factors matrices **F**, we proposed a new hierarchical technique to factorize them (HF). This one-pass algorithm works faster than other studied methods for medium-to-large sized problems, and allows to work with matrices that do not fit into system memory. Furthermore, the Z-order curve is used to sort the patches of the scene, grouping them by nearness, which accelerates the process by exploiting the spatial coherence. We conclude that the sorting of rows/columns of the **RF** matrix is a powerful strategy to reduce the execution times of the algorithm.

In order to test radiosity results, two models were used. HF was successfully applied to factorize their **RF** matrices. As a result, it was possible to calculate the global radiosity (with infinite bounces) for multiple scenes (2560, 21888, and 79232 patches) with negligible relative error. After a pre-computation stage, this was performed in real-time for the static geometries, and without the use of any specialized hardware resource (like GPU) to compute the matrix-vector and matrix-matrix multiplications.

On the other hand, with the goal of studying environments with a high occlusion rate between its elements, several city models were used. The correlation between the characteristics of a city and the sparsity of its form factors matrix was studied. In the experimental analysis, the sparsity of $\mathbf{F}$ was analyzed for three different types of cities with different variation on their building heights. Also, different mesh resolutions were tested for each scene. We found that cities with more homogeneous buildings produce more sparse $\mathbf{F}$ matrices. The matrices are sparse enough to be stored in the main memory of a desktop computer, considering city scenes that contain up to 140k patches.

Another result is the calculation of a sparse approximation of the inverse of the radiosity matrix ($\tilde{\mathbf{M}}$), which is also sufficiently sparse as to be stored in main memory. These matrices were tested doing radiosity calculations for several sky configurations. This is accomplished in much better execution times than the Jacobi method, with low relative errors. For example, it reaches a 42.2× speedup to find a radiosity solution with less than 0.1 relative error.

Regarding orography, it was found that the terrain where the city lies in has a big influence on $\mathbf{F}$. For example, if the landform is a valley, $\mathbf{F}$ becomes denser than in a flat terrain. On the other hand, when using hill-type topographies, the results do not vary significantly with respect to flat-type. This study allows to conclude that it is possible to work with different types of terrains for city models, but this will have an impact on the execution times and memory consumption of radiosity calculations.

5.1.1 Summary of contributions

In summary, the principal contributions of this thesis are the following:

1. The proposal and implementation of a novel one-pass hierarchical technique to factorize low numerical rank matrices, applying multiple decompositions recursively. Several theoretical studies were performed to analyze the proposal. In the experimental stage, this new method was applied to factorize $\mathbf{RF}$, allowing to compute real-time radiosity results with low relative error, for static scenes.

2. The factorization of matrices that do not fit into system memory using the proposed method, which allows to compute radiosity for problems that are not solvable with the classic formulation.

3. The use of the Z-order curve as a row/column sorting strategy to accelerate the factorization method.

4. The advance on the study of the factorization of form factors matrices and its relation to the spatial coherence.

5. The study of the sparsity properties of $\mathbf{F}$ in urban models, and on how to take advantage of them in radiosity calculations.

6. The analysis of $\mathbf{F}$ for different types of cities, taking building height homogeneity, mesh granularity, and orography into account.

7. The implementation and experimental analysis of a method to approximate the inverse of the radiosity matrix by a very sparse matrix, useful for solving global illumination problems efficiently.

5.2 Future Work

This thesis leads to several lines of future work. The proposed methods are intended to solve the radiosity problem efficiently, but more specific applications should be studied. Both the HF method and the sparse **F** solution could be used in diverse areas of computer graphics, such as inverse lighting problems and Computer Aided Design.

5.2.1 Scenes classification

It is important to highlight that both techniques solve what seem to be problems associated with orthogonal concepts, because big spatial coherence is not usually reported when there is a high occlusion factor in the scene. For example, the Cornell Box scene presented at Chapter 3 has a big spatial coherence, i.e. most of the action of its corresponding **F** matrix is captured using few singular values. Nevertheless, its density factor is around 20%, which makes it unsuitable for the sparse matrix approach. On the contrary, City 1 from Chapter 4 has a density factor of less than 1%, but its singular values decay slowly.

A future work should address the study of the properties of the models that lead to low numerical rank and/or sparse matrices. A classification of problems would be very important in order to assess the election of the correct technique in every case. In this regard, there are scenes where neither sparse nor low-rank matrices are generated. Also, both sparse and low-rank **F** matrices could be associated with some kinds of scenes. Fig. 5.1 presents a diagram associated with these ideas, using four example models, each one with different properties. A picture of the scene, the sparsity structure of its associated **F**, and a plot of its singular values are shown.

The upper left model corresponds to City 1, whose **F** is sparse and its associated singular values decay slow. The lower right scene is the Cornell Box, whose **F** is full and its numerical rank is low. The other two models were generated to test the existence of other scenes with different properties.

The upper right scene is composed of several rooms, and corridors connecting them. The rooms are simple boxes composed of a fine mesh, which makes them numerically low-rank by their own. Each room “sees” almost nothing of the others. This makes its form factors matrix to be sparse, as can be seen in the structure, but its singular values decay fast enough to be considered numerically low-rank. On the other hand, the lower left scene represents an anechoic chamber, which is a room designed to absorb wave reflections. For this, its walls are filled with pyramids pointing in. This particular property makes **F** to be highly dense and not numerically low-rank.

More work is needed to study these aspects, and to find new techniques to represent the radiosity matrix efficiently in the above mentioned cases. The upper right case opens the door to combine techniques, in order to exploit both properties

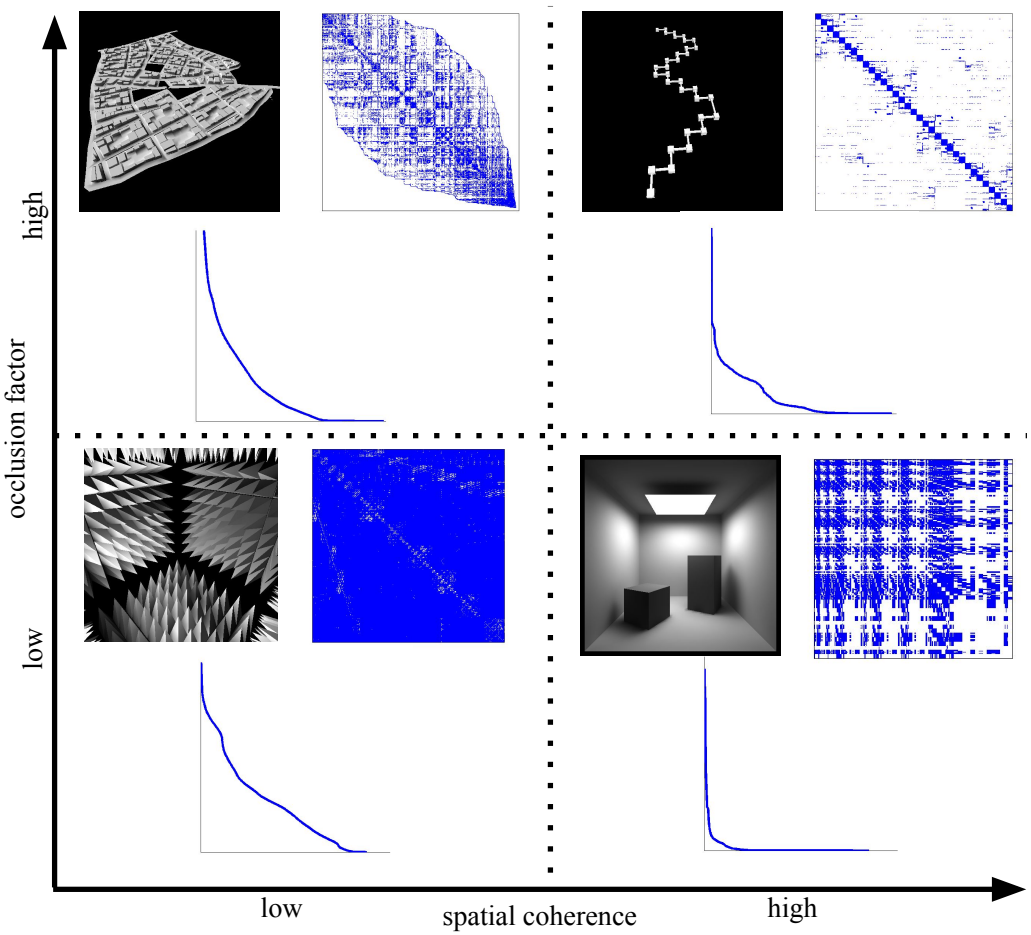


Figure 5.1: Diagram of sparsity and low-rank properties of four example scenes.

in a productive manner. On the contrary, new techniques should be developed to compute radiosity in models similar to the lower left one, especially if they are composed of a large number of elements.

5.2.2 Further works

It is necessary to continue the study and development of the HF algorithm. There are plenty of areas of scientific work in which this new factorization can be applied and exploited, such as signal processing, inverse problems, and principal component analysis.

A better estimation of the error bound should be found, where the spatial coherence of the scene could be taken into consideration. In this sense, other types of norms such as the Frobenius norm could be used. On the other hand, another pending work is to develop a strategy to predict the optimal number of levels in the recursion that minimizes the number of operations to perform. This is closely related to the computational complexity study presented before.

Despite the fact that SVD is a well-known factorization strategy, the use of other

factorization techniques in the base case of the recursion could be analyzed. The only restriction is that the new factorization needs to produce two $n \times k$ matrices, such that its produced error is less than a given (expected) error. For example, the RRQR decomposition is a factorization strategy that can be adapted to satisfy this restriction.

Regarding the implementation, several optimizations are pending, for example the use of multiple threads to execute each leaf decomposition in parallel. Besides, specialized hardware such as GPUs could be used to perform a faster factorization in the base case, or matrix-vector and matrix-matrix products.

The sparsity of $\mathbf{F}$ is a promising property for solving radiosity in other scenes with high occlusion, such as mazes or plant canopies. It would be very important to develop a classification of these type of models, in order to choose the correct method for global illumination calculations. Also, a more efficient method to compute the form factors matrix could be studied, taking the high occlusion factor as an advantage to speedup the process.

Further works should address the study of the form factors associated with real city models, and also take other characteristics into account, such as different orographies or the building density. These approaches can lead to simulate the effect of light over cities, which can be the empirical basis to the proposal and/or modification of city regulations. Other possible line of work is related to the design of new city elements, like buildings and public places, taking into consideration the main characteristics of the surroundings.

Finally, it would be interesting to use the proposed methods as additional elements for computing global radiation exchange in the city, including heat transfer calculations. This requires the inclusion of the heat equation.

Bibliography

Aizenbud, Y. and Averbuch, A. (2016). Matrix decompositions using sub-gaussian random matrices. Manuscript submitted for publication., http://www.cs.tau.ac.il/~amir1/publications.html:Accesed: May 2016.

Alexander, C., Ishikawa, S., and Silverstein, M. (1977). A pattern language: towns, buildings, construction, volume 2. Oxford University Press.

Aliaga, D. G., Beckers ed., B., and Wiley, J. (2012). Geometrical models of the city. Solar Energy at Urban Scale, pages 191–203.

Appel, A. (1968). Some techniques for shading machine renderings of solids. In Proceedings of the April 30–May 2, 1968, spring joint computer conference, pages 37–45. ACM.

Ashdown, I. (2001). Eigenvector radiosity. Master's thesis, Department of Computer Science, University of British Columbia, Vancouver, British Columbia.

Badt, S. (1988). Two algorithms for taking advantage of temporal coherence in ray tracing. The Visual Computer, 4(3):123–132.

Baker, N. and Steemers, K. (2014). Daylight design of buildings: A handbook for architects and engineers. Routledge.

Baranoski, G. V., Bramley, R., and Rokne, J. G. (1997). Eigen-analysis for radiosity systems. In Proceedings of the Sixth International Conference on Computational Graphics and Visualization (Compugraphics' 97), pages 193–201. Citeseer.

Beckers, B. (2013). Taking advantage of low radiative coupling in 3d urban models. In Proceedings of the Eurographics Workshop on Urban Data Modelling and Visualisation, pages 17–20. Eurographics Association.

Beckers, B. and Masset, L. (2008). Heliodon2 software, references & manuals. http://www.heliodon.net/ (in French & Spanish).

Beckers, B., Masset, L., and Beckers, P. (2011). The universal projection for computing data carried on the hemisphere. Computer-Aided Design, 43(2):219–226.

Beckers, B. and Rodríguez, D. (2009). Helping architects to design their personal daylight. WSEAS transactions on environment and development, 5(7):467–477.

Beckers, P., Beckers ed., B., and Wiley, J. (2012). Radiative simulation methods. Solar Energy at Urban Scale, pages 205–236.

Beran, M. and Parrent, G. (1974). Theory of partial coherence. Society of Photo-optical Instrumentation Engineers.

Bischof, C. H. and Quintana-Ortí, G. (1998). Computing rank-revealing qr factorizations of dense matrices. ACM Transactions on Mathematical Software (TOMS), 24(2):226–253.

Bolz, J., Farmer, I., Grinspun, E., and Schröoder, P. (2003). Sparse matrix solvers on the gpu: conjugate gradients and multigrid. In ACM Transactions on Graphics (TOG), volume 22, pages 917–924. ACM.

Borel, C. C., Gerstl, S. A., and Powers, B. J. (1991). The radiosity method in optical remote sensing of structured 3-d surfaces. Remote Sensing of Environment, 36(1):13–44.

Bouatouch, K. and Bouville, C. (2013). Photorealism in computer graphics. Springer Science & Business Media.

Catmull, E. E. (1974). A Subdivision Algorithm for Computer Display of Curved Surfaces. PhD thesis. AAI7504786.

Chace, M. A. (1984). Methods and experience in computer aided design of large-displacement mechanical systems. In Computer aided analysis and optimization of Mechanical System Dynamics, pages 233–259. Springer.

Chakraborty, A. (2005). Impact of orography on the simulation of monsoon climate in a general circulation model.

Chandrasekar, S. (1950). Radiative Transfer. Oxford Univ. Press.

Chelle, M. and Andrieu, B. (1998). The nested radiosity model for the distribution of light within plant canopies. Ecological Modelling, 111(1):75–91.

Choi, J. W., Singh, A., and Vuduc, R. W. (2010). Model-driven autotuning of sparse matrix-vector multiply on gpus. In ACM sigplan notices, volume 45, pages 115–126. ACM.

Christensen, P. (2008). Point-based approximate color bleeding. Pixar Technical Notes, 2(5):6.

Cichocki, A. and Amari, S.-i. (2002). Adaptive blind signal and image processing: learning algorithms and applications, volume 1. John Wiley & Sons.

Cohen, M., Greenberg, D., Immel, D., and Brock, P. (1986). An efficient radiosity approach for realistic image synthesis. IEEE Comput. Graph. Appl., 6:26–35.

Cohen, M., Wallace, J., and Hanrahan, P. (1993). Radiosity and realistic image synthesis. Academic Press Professional, Inc., San Diego, CA, USA.

Cohen, M. F., Chen, S. E., Wallace, J. R., and Greenberg, D. P. (1988). A progressive refinement approach to fast radiosity image generation. In ACM SIGGRAPH Computer Graphics, volume 22, pages 75–84. ACM.

Cohen, M. F. and Greenberg, D. P. (1985). The hemi-cube: A radiosity solution for complex environments. In ACM SIGGRAPH Computer Graphics, volume 19, pages 31–40. ACM.

Cohen, M. F. and Wallace, J. R. (2012). Radiosity and realistic image synthesis. Elsevier.

Collier, C. (2006). The impact of urban areas on weather. Quarterly Journal of the Royal Meteorological Society, 132(614):1–25.

Crocker, G. A. (1984). Invisibility coherence for faster scan-line hidden surface algorithms. SIGGRAPH Comput. Graph., 18(3):95–102.

Dachsbacher, C., Stamminger, M., Drettakis, G., and Durand, F. (2007). Implicit visibility and antiradiance for interactive global illumination. ACM Trans. Graph., 26(3).

Danby, J. (1992). Fundamentals of celestial mechanics. Richmond: Willman-Bell,—c1992, 2nd ed., 1.

Dong, Z., Kautz, J., Theobalt, C., and Seidel, H.-P. (2007). Interactive global illumination using implicit visibility. In Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, PG '07, pages 77–86, Washington, DC, USA. IEEE Computer Society.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. (2004). Clustering large graphs via the singular value decomposition. Machine learning, 56(1-3):9–33.

Drineas, P. and Kannan, R. (2003). Pass efficient algorithms for approximating large matrices. In SODA, volume 3, pages 223–232.

Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. SIAM Journal on Computing, 36(1):158–183.

Duff, I. S. (1977). A survey of sparse matrix research. Proceedings of the IEEE, 65(4):500–535.

Dutre, P., Bekaert, P., and Bala, K. (2006). Advanced global illumination. CRC Press.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. Psychometrika, 1(3):211–218.

EERE (2016). Weather data downloaded from the website: http://energy.gov/eere/office-energy-efficiency-renewable-energy, visited in.

Felsen, L. B. and Marcuvitz, N. (1994). Radiation and scattering of waves, volume 31. John Wiley & Sons.

Fernández, E. (2009). Low-rank radiosity. In Proceedings of the IV Iberoamerican symposium in computer graphics. Sociedad Venezolana de Computación Gráfica, pages 55–62.

Fernandez, E., Aguerre, J. P., Beckers, B., and Besuievsky, G. (2016). Optimizing window shape for daylighting: An urban context approach. In Accepted in Eurographics Workshop on Urban Data Modelling and Visualisation. Eurographics Association.

Fernández, E. and Besuievsky, G. (2012). Inverse lighting design for interior buildings integrating natural and artificial sources. Computers & Graphics, 36(8):1096–1108.

Foley, J. D., Van Dam, A., Feiner, S. K., Hughes, J. F., and Phillips, R. L. (1994). Introduction to computer graphics, volume 55. Addison-Wesley Reading.

Freed, L. and Ishida, S. (1995). History of computers. Ziff-Davis Publishing Co.

Frieze, A., Kannan, R., and Vempala, S. (2004). Fast monte-carlo algorithms for finding low-rank approximations. Journal of the ACM (JACM), 51(6):1025–1041.

Galin, E., Peytavie, A., Maréchal, N., and Guérin, E. (2010). Procedural generation of roads. In Computer Graphics Forum, volume 29, pages 429–438. Wiley Online Library.

Garland, M., Willmott, A., and Heckbert, P. S. (2001). Hierarchical face clustering on polygonal surfaces. In Proceedings of the 2001 symposium on Interactive 3D graphics, I3D '01, pages 49–58, New York, NY, USA. ACM.

Glassner, A. S. (1994). Principles of Digital Image Synthesis. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Glickman, T. (2000). Ams glossary of meteorology. American Meteorology Society, Boston, USA.

Goel, N. S., Rozehnal, I., and Thompson, R. L. (1991). A computer graphics based model for scattering from objects of arbitrary shapes in the optical region. Remote Sensing of Environment, 36(2):73–104.

Golub, G. H. and Van Loan, C. F. (1996). Matrix computations (3rd ed.). Johns Hopkins University Press, Baltimore, MD, USA.

Golub, G. H. and Van Loan, C. F. (2012). Matrix computations, volume 3. JHU Press.

Gonzalez, P. and Gisbert, F. (1998). Object and ray coherence in the optimization of the ray tracing algorithm. In Proceedings. Computer Graphics International (Cat. No.98EX149). Institute of Electrical & Electronics Engineers (IEEE).

Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. (1984). Modeling the interaction of light between diffuse surfaces. In ACM SIGGRAPH Computer Graphics, volume 18, pages 213–222. ACM.

Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. (1993). Wavelet radiosity. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 221–230. ACM.

Groller, M. E. and Purgathofer, W. (1993). Coherence in computer graphics. In Conner, Hernandez, M. P., editor, Visualization and Intelligent Design in Engineering and Architecture. Elsevier Science Publishers.

Gu, M. and Eisenstat, S. C. (1996). Efficient algorithms for computing a strong rank-revealing qr factorization. SIAM Journal on Scientific Computing, 17(4):848–869.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review, 53(2):217–288.

Hanrahan, P., Salzman, D., and Aupperle, L. (1991). A rapid hierarchical radiosity algorithm. In ACM SIGGRAPH Computer Graphics, volume 25, pages 197–206. ACM.

Harbrecht, H., Peters, M., and Schneider, R. (2012). On the low-rank approximation by the pivoted cholesky decomposition. Applied numerical mathematics, 62(4):428–440.

Hašan, M., Pellacini, F., and Bala, K. (2007). Matrix row-column sampling for the many-light problem. ACM Trans. Graph., 26(3).

Hilbert, D. (1891). Ueber die stetige abbildung einer line auf ein flächenstück. Mathematische Annalen, 38(3):459–460.

Holmes, M. P., Isbell, J., Lee, C., and Gray, A. G. (2009). Quic-svd: Fast svd using cosine trees. In Advances in Neural Information Processing Systems, pages 673–680.

Hughes, J. F., Van Dam, A., Foley, J. D., and Feiner, S. K. (2014). Computer graphics: principles and practice. Pearson Education.

Hviid, C. A., Nielsen, T. R., and Svendsen, S. (2008). Simple tool to evaluate the impact of daylight on building energy consumption. Solar Energy, 82(9):787–798.

Jensen, H. W. (1996). Global illumination using photon maps. In Proceedings of the eurographics workshop on Rendering techniques '96, pages 21–30, London, UK, UK. Springer-Verlag.

Jensen, H. W. (2001). Realistic Image Synthesis Using Photon Mapping. A. K. Peters, Ltd., Natick, MA, USA.

Jensen, H. W., Arvo, J., Dutre, P., Keller, A., Owen, A., Pharr, M., and Shirley, P. (2003). Monte carlo ray tracing. In ACM SIGGRAPH, pages 27–31.

Johnson, T. (1981). Solar architecture: the direct gain approach. McGraw Hill Book Company.

Kajiya, J. T. (1986). The rendering equation. SIGGRAPH Comput. Graph., 20(4):143–150.

Keller, A. (1997). Instant radiosity. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97, pages 49–56, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Kirk, D. B. and Hwu, W.-m. W. (2010). Programming Massively Parallel Processors: A Hands-on Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Kontkanen, J., Turquin, E., Holzschuch, N., and Sillion, F. X. (2006). Wavelet radiance transport for interactive indirect lighting. In Rendering Techniques 2006 (Eurographics Symposium on Rendering), pages 161–171.

Krüger, J. and Westermann, R. (2003). Linear algebra operators for gpu implementation of numerical algorithms. In ACM Transactions on Graphics (TOG), volume 22, pages 908–916. ACM.

Kuchkuda, R. (1988). An introduction to ray tracing. In F40, Theoretical Foundations of Computer Graphics and CAD, edited by R.A. Earnshaw, Springer-Verlag, pages 1039–1060. Springer.

Labayrade, R., Jensen, H. W., and Jensen, C. (2009). Validation of velux daylight visualizer 2 against cie 171: 2006 test cases. In Proceedings 11th International IBPSA Conference, International Building Performance Simulation Association, pages 1506–1513.

Liberty, E. (2013). Simple and deterministic matrix sketching. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 581–588. ACM.

Liberty, E., Woolfe, F., Martinsson, P.-G., Rokhlin, V., and Tygert, M. (2007). Randomized algorithms for the low-rank approximation of matrices. Proceedings of the National Academy of Sciences, 104(51):20167–20172.

Light, R. A. and Gossard, D. C. (1983). Variational geometry: a new method for modifying part geometry for finite element analysis. Computers & Structures, 17(5-6):903–909.

Lipp, M., Scherzer, D., Wonka, P., and Wimmer, M. (2011). Interactive modeling of city layouts using layers of procedural content. In Computer Graphics Forum, volume 30, pages 345–354. Wiley Online Library.

Longcore, T. and Rich, C. (2004). Ecological light pollution. Frontiers in Ecology and the Environment, 2(4):191–198.

Loyer, F. L. (1988). Paris nineteenth century: Architecture and urbanism. Number 711: 72 (44). Abbeville Press Publishers,.

Malley, T. J. (1988). A shading method for computer generated images. master's thesis. Computer Science Dept., Univ. of Utah, Salt Lake City, June.

Mangkuto, R. A. (2016). Validation of dialux 4.12 and dialux evo 4.1 against the analytical test cases of cie 171: 2006. LEUKOS, 12(3):139–150.

Mardaljevic, J. (1999). Daylight simulation: validation, sky models and daylight coefficients. De Montfort University.

Marsh, A. (2003). Ecotect and energyplus. Building Energy Simulation User News, 24(6):2–3.

MATLAB (2010). version 7.10. The MathWorks Inc., Natick, Massachusetts.

Meyer, Q., Eisenacher, C., Stamminger, M., and Dachsbacher, C. (2009). Data-parallel hierarchical link creation for radiosity. In Proceedings of the 9th Eurographics conference on Parallel Graphics and Visualization, EG PGV'09, pages 65–70, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Meyer, R. X. (1999). Elements of space technology. Academic Press.

Millhauser, G. L., Carter, A. A., Schneider, D. J., Freed, J. H., and Oswald, R. E. (1989). Rapid singular value decomposition for time-domain analysis of magnetic resonance signals by use of the lanczos algorithm. Journal of Magnetic Resonance (1969), 82(1):150–155.

Morton, G. M. (1966). A computer oriented geodetic data base and a new technique in file sequencing. International Business Machines Company New York.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Gool, L., and Purgathofer, W. (2013). A survey of urban reconstruction. In Computer graphics forum, volume 32, pages 146–177. Wiley Online Library.

Nabil, A. and Mardaljevic, J. (2005). Useful daylight illuminance: a new paradigm for assessing daylight in buildings. Lighting Research and Technology, 37(1):41–57.

Parish, Y. I. and Müller, P. (2001). Procedural modeling of cities. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 301–308. ACM.

Peano, G. (1890). Sur une courbe, qui remplit toute une aire plane. Mathematische Annalen, 36(1):157–160.

Pharr, M. and Fernando, R. (2005). Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation. Addison-Wesley Professional.

Prusinkiewicz, P. and Lindenmayer, A. (2012). The algorithmic beauty of plants. Springer Science & Business Media.

Reinhart, C. (2011). Daylight performance predictions. Building performance simulation for design and operation, page 235.

Ritschel, T., Dachsbacher, C., Grosch, T., and Kautz, J. (2012). The state of the art in interactive global illumination. Comput. Graph. Forum, 31(1):160–188.

Ritschel, T., Engelhardt, T., Grosch, T., Seidel, H.-P., Kautz, J., and Dachsbacher, C. (2009). Micro-rendering for scalable, parallel final gathering. In ACM SIGGRAPH Asia 2009 papers, SIGGRAPH Asia '09, pages 132:1–132:8, New York, NY, USA. ACM.

Robinson, D., Haldi, F., Kämpf, J., Leroux, P., Perez, D., Rasheed, A., and Wilke, U. (2009). CitySim: Comprehensive micro-simulation of resource flows for sustainable urban planning. In Proc. Building Simulation.

Robinson, D. and Stone, A. (2005). A simplified radiosity algorithm for general urban radiation exchange. Building Services Engineering Research and Technology, 26(4):271–284.

Rudelson, M. (2014). Recent developments in non-asymptotic theory of random matrices. Modern Aspects of Random Matrix Theory, 72:83.

Sillion, F. and Puech, C. (1989). A general two-pass method integrating specular and diffuse reflection. In ACM SIGGRAPH Computer Graphics, volume 23, pages 335–344. ACM.

Simon, H. D. and Zha, H. (2000). Low-rank matrix approximation using the lanczos bidiagonalization process with applications. SIAM Journal on Scientific Computing, 21(6):2257–2274.

Sloan, P.-P., Kautz, J., and Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In ACM Transactions on Graphics (TOG), volume 21, pages 527–536. ACM.

Smits, B., Arvo, J., and Greenberg, D. (1994). A clustering algorithm for radiosity in complex environments. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, pages 435–442, New York, NY, USA. ACM.

Stewart, G. (1998). Matrix Algorithms: Volume 1: Basic Decompositions. Society for Industrial and Applied Mathematics.

Sutherland, I. E., Sproull, R. F., and Schumacker, R. A. (1974). A characterization of ten hidden-surface algorithms. ACM Comput. Surv., 6(1):1–55.

Teller, S. and Hanrahan, P. (1993). Global visibility algorithms for illumination computations. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 239–246. ACM.

Tregenza, P. (1987). Subdivision of the sky hemisphere for luminance measurements. Lighting Research and Technology, 19(1):13–14.

Tregenza, P. and Waters, I. (1983). Daylight coefficients. Lighting Research and Technology, 15(2):65–71.

van Eekelen, T., Beckers ed., B., and Wiley, J. (2012). Radiation modeling using the finite element method. Solar Energy at Urban Scale, pages 237–257.

Vershynin, R. (2010). Introduction to the non-asymptotic analysis of random matrices. arXiv preprint arXiv:1011.3027.

Vijayan, V. (2014). Fast svd and pca. `http://www.mathworks.com/matlabcentral/fileexchange/47132-fast-svd-and-pca/content/svdecon.m`. Accessed: May 2016.

Vollmer, M. and Möllmann, K.-P. (2010). Infrared thermal imaging: fundamentals, research and applications. John Wiley & Sons.

Voorhies, D. (1991). Space-filling curves and a measure of coherence. Graphics Gems II, pages 26–30.

Wallace, J. R., Elmquist, K. A., and Haines, E. A. (1989). A ray tracing algorithm for progressive radiosity. In ACM SIGGRAPH Computer Graphics, volume 23, pages 315–324. ACM.

Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., and Greenberg, D. P. (2005). Lightcuts: a scalable approach to illumination. In ACM SIGGRAPH 2005 Papers, SIGGRAPH '05, pages 1098–1107, New York, NY, USA. ACM.

Wang, C. (1992). Physically correct direct lighting for distribution ray tracing. In Graphics Gems III, pages 307–313. Academic Press Professional, Inc.

Wang, R., Wang, R., Zhou, K., Pan, M., and Bao, H. (2009). An efficient gpu-based approach for interactive global illumination. In ACM Transactions on Graphics (TOG), volume 28, page 91. ACM.

Ward, G. J. (1994). The radiance lighting simulation and rendering system. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques, pages 459–472. ACM.

Whitted, T. (1979). An improved illumination model for shaded display. In ACM SIGGRAPH Computer Graphics, volume 13, page 14. ACM.

Wilkinson, J. (1971). The algebraic eigenvalue problem. In Handbook for Automatic Computation, Volume II, Linear Algebra. Springer-Verlag New York.

Wimmer, M. and Bittner, J. (2015). Gpu gems 2: chapter 6. hardware occlusion queries made useful. 2005.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant architecture, volume 22. ACM.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T. S., and Yan, S. (2010). Sparse representation for computer vision and pattern recognition. Proceedings of the IEEE, 98(6):1031–1044.

Yin, X., Wonka, P., and Razdan, A. (2009). Generating 3d building models from architectural drawings: A survey. IEEE Computer Graphics and Applications, (1):20–30.

Zienkiewicz, O. C., Taylor, R. L., Zienkiewicz, O. C., and Taylor, R. L. (1977). The finite element method, volume 3. McGraw-hill London.