Vers la modélisation grand échelle d'environnements urbains à partir d'images

Oussama Moslah

UNIVERSITY OF CERGY - PONTOISE

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION
ET DE LA COMMUNICATION

PHD THESIS

to obtain the title of

PhD of Science

of the University of Cergy - Pontoise

Speciality : Computer Science

Defended by

Oussama Moslah

Towards Large-Scale Urban Environments Modeling from Images

Thesis Advisor: Sylvie Philipp-Foliguet

prepared at:

THALES D3S SBL Simulation, Cergy-Pontoise, France.
ETIS - UMR CNRS 8051, ENSEA, Cergy-Pontoise, France.

Jury :

Reviewers :	Nicolas Paparoditis	-	IGN, Paris, France.
	Peter Sturm	-	INRIA Alpes, Grenoble, France.
Advisor :	Sylvie Philipp-Foliguet	-	ETIS - UMR CNRS 8051, Cergy, France.
President :	Serge Couvet	-	THALES Simulation, Cergy, France.
Examinators :	Peter Wonka	-	Arizona State University, Tempe, USA.
	Thorsten Thormählen	-	MPII, Sarbrucken, Germany.

Acknowledgments

First, I wish to aknowledge particlurarly my PhD supervisors Mme Sylvie Philipp-Foliguet and Mr Serge Couvet for their supervision, assistance, and helpfull suggestions and guidelines during the thesis.

I wish also to aknowledge the members of the jury, Mr Nicolas Paparoditis, Mr Peter Sturm, Mr Peter Wonka, and Mr Thorsten Thormählen for their acceptation to read the PhD manuscript and assist to my PhD thesis defense.

I wish also to aknowledge my colleagues and particularly Mr Vincent Guitteny for its help during the PhD thesis and its assistance during the writing of this manuscript.

I wish to aknowledge all the students that did internships with me in Thales and strongly contribute to the work presented in this manuscript and the different research papers.

Finally, I wish to acknowledge the Cap Digital Business Cluster Terra Numerica project for sponsoring the research reported in this manuscript.

Contents

Chapter 1

Introduction

The context of this thesis is the growing interest in large-scale modeling of cities. This thesis is a part of the Terra Numerica project whose aim is to develop new technologies for the large-scale reconstruction of urban environments and new virtual/augmented reality applications based on the city 3D model. This work was carried out in Thales Training and Simulation a division of the Thales group and the ETIS lab in Cergy.

Terra Numerica project :

The access to accurate and geo-localized informations of the territories is a crucial issue for a broad spectrum of applications involving individuals, governments and businesses. The introduction of the third dimension in the representation of these areas provides a unique potential to visualize information and simulations used for the study and management of these territories.

The TerraNumerica project aims to develop technologies needed to produce the most automated and most accurate possible representation of large 3D urban areas with high resolutions, and the use of these visual representations through online applications (Internet), mobile applications (mobile phone or PDA) devices and virtual reality and augmented reality.

The developped technologies include: the acquisition of geo-referenced buildings from platforms and mobile ground stations and airborne platforms, the fusion and the alignment of geo-referenced data from different sources and different acquisition devices, the automated 3D reconstruction of buildings and vegetation using image-based and model-based approaches, the segmentation, compression, and transmission of reconstructed urban 3D data, and the use of 3D urban databases through online applications, mobile devices (phones, PDAs), and virtual/augmented reality.

Thales Training and Simulation :

Thales Training & Simulation is a subsidiary of the Thales group, in the Security Solutions division & Services. Thales Training & Simulation design and integrates simulators and training systems for nearly 50 years and offers a wide range of products and services. At first, it only produced aircraft simulators, and then demand has diversified. Today's products and services delivered by the subsidiary cover areas such as civil aviation, military and energy. Europe accounts for about 70 % revenue of the company are divided equally between the fields of defense and civilian. Thales Training & Simulation is present in over 60 countries worldwide with over 800 systems in operation simulations.

The simulators offered allow students to learn how to behave in case of emergencies or unexpected situations.

Figure 1.1: An aircraft simulator developped by Thales.

Military forces also rely on simulation systems to perform repeated tasks. The business simulation offers a wide range of possibilities, from training systems based on computers and full flight simulators to the synthetic environments in the most comprehensive network, enabling the simulation of large-scale military operations.

The simulators manufactured by Thales include flight simulators for civil aviation used by Airbus and Boeing simulators and military training systems for the Mirage 2000, Rafale , Eurofighter Typhoon, the Leclerc tanks and other vehicles, and also the truck Trust simulator that I had the chance to try during my internship. This is a real cab mounted on jacks and really giving the impression of being in a truck with a field of vision 180 degrees.

Thesis objectives and contributions :

The main goal of this thesis is to develop innovative and practical tools for the reconstruction of buildings from images. The typical input to our work is a set of facade images, building footprints, and coarse 3D models reconstructed from aerial images. The main steps include the calibration of the photographs, the registration with the coarse 3D model, the recovery of depth and semantic information, and the refinement of the coarse 3D model.

To achieve this goal, we use computer vision, pattern recognition, and computer graphics techniques. Contributions in this approach are presented in two parts.

In the first part, we focused on multiple view reconstruction techniques with the aim to automatically recover the depth information of facades from a set of uncalibrated photographs. First, we use structure-from motion techniques to automatically calibrate the set of photographs. Then, we propose techniques for the registration of the sparse reconstruction to a coarse 3D model. Finally, we propose an accelerated multi-view stereo and voxel coloring framework using graphics hardware to produce a textured 3d mesh of a scene from a set of calibrated images. Multi-view stereo is a direct approach (2D images -> 3D scene) where stereo techniques are used to produce depth maps and then produce a 3D surfacic model. Voxel coloring is an indirect method (3D scene -> 2D images) in which the volume encompassing the 3D scene is discretized as voxels that are re-projected onto images and colored if they are consistent.

The second part is dedicated to single view reconstruction and its aim is to recover the semantic structure of a facade from an ortho-rectified image. The novelty of this approach is the use of a stochastic grammar describing an architectural style as a model for facade reconstruction. We combine bottom-up detection with top-down proposals to optimize the facade structure using the Metropolis-Hastings algorithm.

Part I

Multiple View Reconstruction

Chapter 2

Structure from Motion

This chapter addresses the problem of fitting a coarse 3D model to a structure from motion sparse reconstruction. Fitting the model to the sparse reconstruction is defined and solved as an absolute orientation problem. The novelty of our approach is the use of facade correspondences to compute the absolute orientation. We also propose a solution to automatically establish facade correspondences using a pseudo-RANSAC procedure. Our system consists of four parts: a robust structure from motion recovery framework able to register accurately a set of digital photographs, a method to recover the principal walls of a scene from a 3d sparse reconstruction using a robust RANSAC scheme, a method to fit automatically a 3d model to a set of 3d wall planes, and finally, a user interface allowing scene visualisation and photorealistic rendering of urban 3d models from a set of calibrated photographs.

2.1 Introduction

Our work is mainly related to Structure From Motion (SFM) techniques. Structure from motion techniques are able to automatically recover the sparse structure of a scene together with the motion of the camera using multiple view geometry techniques [Hartley 2003]. There are three main Structure From Motion methods: (1) Factorization based methods [Sturm 1996, Triggs 1996] consist of SVD (Singular Value Decomposition) decomposition of a matrix containing the images of points in all views in order to recover a projective structure and motion of the scene. The metric reconstruction is then obtained using self-calibration methods [Hartley 2003, Ponce 2005, Pollefeys 2004], (2) Trifocal tensor based methods [Hartley 2003] use image triplets to iteratively recover the structure and motion from images sequences, (3) Sequential methods [Pollefeys 2004, P.A. 1997] use the motion computed from the fundamental matrix between a pair of images as initialization and then iteratively update the structure and motion by resection. Our implementation is similar to the PhotoTourism project which is a recent sequential structure from motion research work [N. 2006] able to calibrate a large set of digital photographs taken by different cameras.

2.2 The classical pinhole camera model

The classic pinhole model representing the central projection of points in space on a plane is shown schematically in Fig. 2.1. The projection center, or optical center C is the origin of the Euclidean coordinate system ; the plane $Z = f$ where f is the focal length of the camera is called the image plane or focal plane. The point p is called the central point and is obtained by intersecting the main axis Z with the image plane.

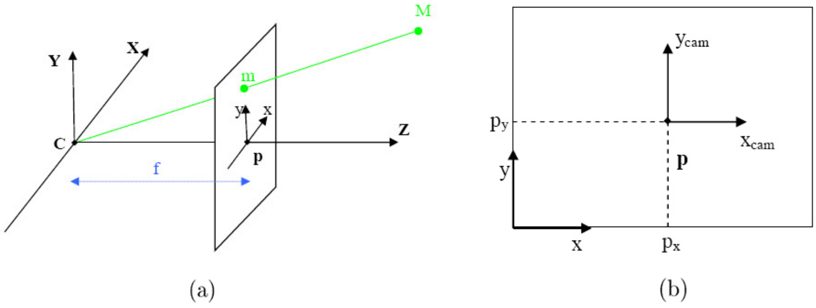


Figure 2.1: *Projection of a 3D in the image plane using the pinhole camera model : (a) pinhole camera model (b) image coordinates* (x, y) *and camera coordinates* $(xcam, ycam)$

A 3D point M in the space is projected onto a 2D point m on the image plane as the beam emitted from the point M to the center of camera C through the image plane at that point. The central projection between the 3D Euclidean space and the 2D Euclidean space is such that:

$$\Re^3 \to \Re^2 : M(X,Y,Z)^T \mapsto (\frac{fX}{Z}, \frac{fY}{Z})^T \tag{2.1}$$

2.2.1 Central projection in homogeneous coordinates

The central projection can be described by the matrix relation $m\ PM$ with $M = (X,Y,Z,1)^T$ the 3D point of space, $m = (fX/Z, fY/Z)^T$ its projected point into the image plane and P the projection matrix[3 × 4]. The projection of a point can then be written:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.2}$$

2.2.2 Principal point offset

The above relations imply that the origin of the coordinate system in the image plane is the principal point p. In practice, this is not always the case as described in [Hartley 2003]. Thus, the central projection is then defined by:

$$\Re^3 \to \Re^2 : M(X,Y,Z)^T \mapsto (\frac{fX}{Z} + p_x, \frac{fY}{Z} + p_y)^T \tag{2.3}$$

where (p_x, p_y) are the coordinates of principal point in the coordinate system of the image plane. This projection is described in homogeneous coordinates by:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.4}$$

If one writes:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The projection than is written as follows:

$$m = K[I|0]M_{cam} \tag{2.6}$$

where K is defined as the intrinsic camera matrixa and M_{cam} is the 3D representation of the point M in the camera coordinate system. The notation $[.|.]$ is a block representation of a matrix. In this example, $[I|0]$ is a $3x4$ matrix composed by a 3×3 identity matrix and in the last a column a 3×1 zeros vector.

2.2.3 Rotation and translation of the camera

The points of space are described in another Euclidean coordinate system: the world coordinate frame. The latter is related to the camera coordinate frame by a rotation and a translation, as shown in Fig 2.2. Let M be the vector representing the inhomogeneous coordinates of a point in space in the world coordinate frame and M_{cam} its representation in the camera coordinate frame, then $M_{cam} = R(M - C)$, where C represents the optical center in the world coordinate frame and R is the rotation matrix $[3 \times 3]$ representing the orientation of the camera. This relationship can be written in homogeneous coordinates as follows:

$$M_{cam} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} M \tag{2.7}$$

This leads to the relation $m = KR[I| - C]M$ where M is now expressed in the world coordinate frame. The projection matrix $P = KR[I| - C]$ has then 9 degrees of freedom: 3 for matrix K (the focal length f and the coordinates of the central point $p = (p_x, p_y)$, 3 to rotation matrix R and 3 for the optical center C. Parameters contained in K are called internal or intrinsic parameters of the camera while parameters of R and C are called external or extrinsic parameters of camera and represent the external orientation of the camera. In general, it is more appropriate to represent the transformation (world coordinate frame / camera coordinate frame) as follows: $M_{cam} = RM + t$ with $t = -RC$. In this case, the matrix of the camera is just $P = K[R|t]$.

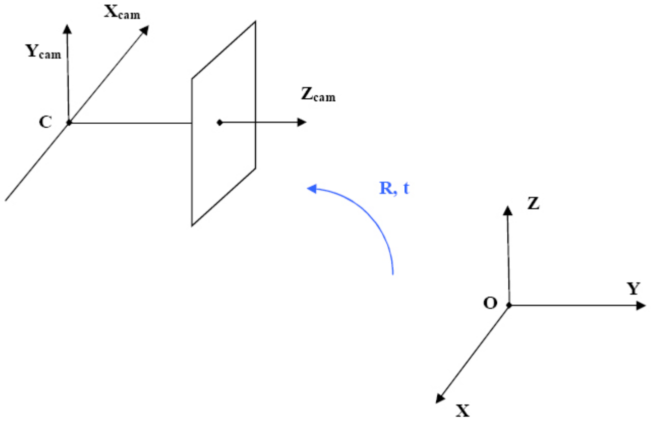


Figure 2.2: *Euclidean transformation between world (right) and camera coordinate frame (left).*

2.3 Keypoints detection and matching

The first step of any structure from motion recovery technique consists in matching feature points between the set of photos. We used SIFT [Lowe 2003] keypoints and descriptors since they are invariant to changes of scale, rotation, illumination and partially to the point of view. As SIFT descriptors are of dimension 128, we used a K-d tree to match keypoints between pairs of photos. Other methods have been explored to address the problem of matching image features such as the optical flow method and particularly the pyramidal implementation of Lucas-Kanade algorithm [Bouguet 1999], these techniques are particularly suitable for processing video images (small changes between two points of view). FIG 2.3 presents the results obtained using SIFT on the first two images of the Wadham College image sequence.

2.4 Epipolar geometry and the fundamental matrix

The epipolar geometry models the geometric relationship between two image points x and $x^{'}$ projected from the same 3D point in space on two different views. Indeed, the two image points, the 3D point in space and the two optical centers are coplanar and belong to the same plane called epipolar plane. This plan intersects the two image planes in two epipolar lines.

The fundamental matrix is the algebraic representation of epipolar geometry. It models the algebraic mapping between an image point in one view and its corre-

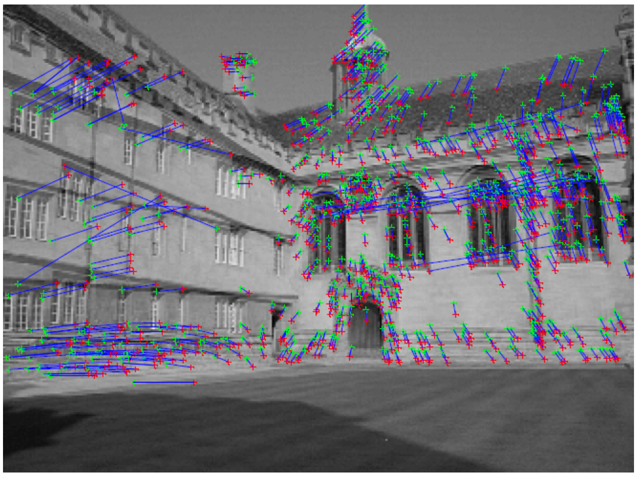


Figure 2.3: *Keypoints matching using SIFT: the number of matched points is 921. The points of the first image are displayed in green and those of the second in red. The lines connecting pairs of matched points between the two images are displayed in blue. Both images are displayed in superposition. Notice the presence of false matches. Those outliers can be detected and rejected using the epipolar geometry.*

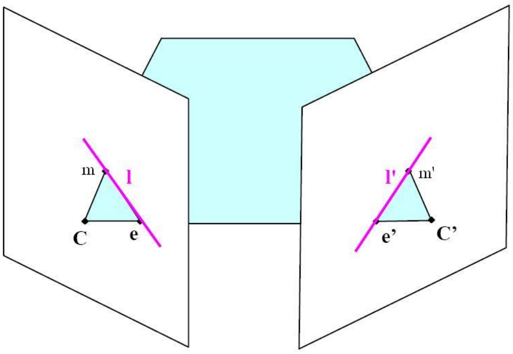


Figure 2.4: *Epipolar geometry.*

sponding epipolar line in another view. The fundamental matrix is a matrix [3 × 3] of rank 2. Given two image points $(m, m^{'})$ projected from the same 3D point M on images I and $I^{'}$ and $(l^{'}, l)$ the two corresponding epipolar lines respectively in images $I^{'}$ and I, then $l^{'} = Fm$ and $l = F^T m^{'}$.

According to epipolar geometry image point $m^{'}$ belongs to the epipolar line $l^{'}$, then: $m^{'T} l^{'} = 0$, hence:

$$m^{'T} Fm = 0 \tag{2.8}$$

Thus the fundamental matrix is defined as the unique homogeneous

[3 × 3] matrix of rank 2 satisfying this relationship for all previous correspondences $(m_i(x_i, y_i)m_i^{'}(x_i^{'}, y_i^{'})$, $i = 1, ..., n$, between two distinct views. The relation $m_i^{'T} F m_i = 0$ can then be written in algebraic form as follows [Zhang 1995, G. Xu 1996, Zhang 1996]:

$$u_i T f = 0 \tag{2.9}$$

with :

$$f = [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T \tag{2.10}$$

$$u_i = [x_i x_i^{'}, x_i y_i^{'}, x_i, y_i x_i^{'}, y_i y_i^{'}, y_i, x_i^{'}, y_i^{'}, 1]^T \tag{2.11}$$

where F_{ij} represents the element of the fundamental matrix F at row i and column j.

The concatenation of vectors u_i, $i = 1, ..., n$ of the relationship leads to the following linear system:

$$U_n f = 0 \tag{2.12}$$

With:

U$_n = [u_1, u_2, ..., u_n]^T$ (2.13)

All these linear equations and the rank constraint on the fundamental matrix allows then to estimate the epipolar geometry between two views. We can classify the methods of estimation as follows: linear, iterative, and robust methods.

2.4.1 Linear methods

2.4.1.1 Exact solution with 7 points

The fundamental matrix is of rank 2 and is defined up to a scale factor then it has 7 degrees of freedom [Hartley 2003, Zhang 1996]. Thus, 7 points (minimum) are needed to solve the linear system of the previous relationship. For $n = 7$ points, an eigen value decomposition using SVD (Singular Value Decomposition) of matrix U_n yields to two vectors f_1 and f_2 leading to matrices F_1 and F_2. The fundamental matrix is then a linear combination $\alpha F_1 + (1 - \alpha) F_2$ with $\alpha \in [0, 1]$. The rank 2 constraint on the fundamental matrix requires:

$$det[\alpha F_1 + (1 - \alpha) F_2] = 0 \tag{2.14}$$

This gives a polynomial of degree 3 whose maximum number of real solutions is 3. For each value of α, the fundamental matrix is then given by:

$$F = \alpha F_1 + (1 - \alpha) F_2 \tag{2.15}$$

2.4.1.2 Normalized 8 points algorithm

This method is known as the normalized 8-point algorithm [Hartley 2003]. The principle is described in algorithm 1. It consists in solving the following problem in the sense of least squares:

$$\min_{F} \sum_{i=1}^{n} (m_i^{'T} F m_i)^2, n \geq 8 \tag{2.16}$$

The normalization of points (before the resolution of the linear relationship) improves the conditioning of the problem and therefore the stability of the final result. The normalization used here is that proposed in [Hartley 2003] which consists of a translation and a rescaling of the homogeneous points in each image so that the mean point is at the origin and the average distance of points to the origin at $\sqrt{2}$.

Algorithm 1 Normalized 8 point algorithm

Input : Set of 8 image matches $m_i(x_i, y_i) \Leftrightarrow m_i^{'}(x_i^{'}, y_i^{'})$ or more.
Output : Fundamental matrix F.

(i) Normalization : Transform the coordinates of points such that $\widehat{m}_i = T m_i$ and $\widehat{m}_i^{'} = T^{'} m_i^{'}$; T and $T^{'}$ are two normalization matrices consisting on a translation and a scaling.

(ii) Find the fundamental matrix $\widehat{F}$ for the correspondances $\widehat{m}_i \Leftrightarrow \widehat{m}_i^{'}$ corresponding to the smallest eigenvalue of U_n. Strengthen the rank 2 constraint by an SVD decomposition.

(iii) Denormalization : The matrix F is then obtained by $F = T^{'T} \widehat{F} T$ and corresponds to the initial image matches $m_i \Leftrightarrow m_i^{'}$.

2.4.2 Iterative methods

Iterative methods can be classified into two groups: those based on minimizing distances between points and epipolar lines and those which rely on the gradient. Minimizing distances points / lines is defined by the following equation:

$$\min_{F} \sum_{i=1}^{n} d^2(m_i, F^T m_i^{'}) + d^2(m_i^{'}, F m_i) \tag{2.17}$$

Where $d(.,.)$ is the euclidean distance. This equation can be solved by applying a minimization of Newton-Raphson or Levenberg-Marquardt. Iterative methods based on the gradient solves the following equation:

$$\min_{F} \sum_{i=1}^{n} \frac{(m_i^{'T} F m_i)^2}{g_i^2} \tag{2.18}$$

with :

$$g_i^2 = \sqrt{l_1^2 + l_2^2 + l_1'^2 + l_2'^2} \tag{2.19}$$

$$Fm_i^{'} = (l_1, l_2, l_3)^T \tag{2.20}$$

$$F^T m_i = (l_1^{'}, l_2^{'}, l_3^{'})^T \tag{2.21}$$

2.4.3 Robust methods

Point correspondences are obtained using corner detection and correlation techniques or similarity between two images [Zhang 1996]. This process necessarily includes bias. Indeed, on the set of points obtained in matches, there is a proportion of false points due especially to:

- **bad locations**: when estimating the fundamental matrix, the error on the location of points of interest is assumed to follow a Gaussian distribution. A localization error of more than 3 pixels considerably degrades the estimation of fundamental matrix.

- **false match**: during the process of mapping several points there are false matches. This would completely undermine the process of estimating the fundamental matrix.

The methods described above are extremely sensitive to these errors since mapping is based on techniques like least squares. Several robust regression methods exist in the literature, the most popular algorithms are M-estimator, RANSAC and LMedS. These methods are detailed in next paragraphs.

2.4.3.1 M-Estimator

Algorithms for M-Estimator attempt to reduce the effect of false points in the minimization of the residual error using a weight function [Zhang 1995, G. Xu 1996, Zhang 1996]. Thus the M-estimators are based on solving the following equation:

$$\min_F \sum_{i=1}^{n} w_i r_i^2 \tag{2.22}$$

with

$$r_i^2 = (m_i^{'T} F m_i)^2 \tag{2.23}$$

Several weighting functions exist in the literature, including the Tukey [F. Mosteller 1977] and Huber [Huber 1981] weighting functions. The Tukey's weight function is defined by:

$$w_i = \begin{cases} \left(1 - \left(\frac{r_i}{4.6851}\right)^2\right)^2 & \text{si } |r_i| \leq 4.6851\sigma \\ 0 & \text{sinon} \end{cases} \tag{2.24}$$

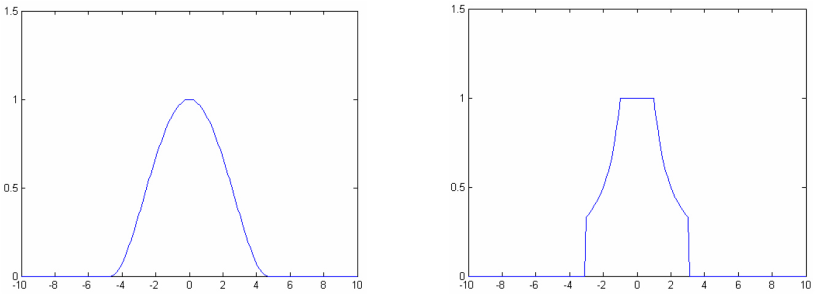
A robust estimate of standard deviation σ related to the median value of residual errors and the number of points n is given by:

$$\sigma = 1.4826 \left(1 + \left(\frac{5}{n-7}\right)\right) median_i \, |r_i| \tag{2.25}$$

Among other functions of weight used in the M-estimators we can cite functions of type Lp, Cauchy, Welsch and Geman-McClure. For each weight function used a different M-Estimator is obtained. However, the minimization procedure of M-estimators is the same and is described by algorithm 2.

Algorithm 2 M-Estimator algorithm

```
Input : A set of 8 correspondances $m_i(x_i, y_i) \Leftrightarrow m_i^{'}(x_i^{'}, y_i^{'})$ or more and an initial estimate of the fundamental matrix $F_0$
Output : Fundamental matrix $F$.
(i) Normalization : Transform the coordinates of points such that $\widehat{m}_i = T m_i$ and $\widehat{m}_i^{'} = T^{'} m_i^{'}$ ; $T$ et $T^{'}$ are two transformation matrices consisting of a translation and scaling.
(ii) $k = 1$, Calculation of the residual $r_{k-1}$ and the robust standard deviation $\sigma$ using equations (2.23) and (2.25)
(iii) Calculation of the Tukey's weights $w_{k-1}$ using equation (2.24).
(iv) Solve the linear system $w_{k-1} U_n f_k = 0$. Strengthen the rank 2 constraint on $F_k$ using an SVD decomposition.
(v) If $|F_k - F_{k-1}| < \varepsilon$ or $k > N$ go to (vii)
(vi) $F_{k-1} = F_k$, $k = k + 1$, go to (iii)
(vii)Denormalization The fundamental matrix is obtained by $F = T^{'T} F_k T$ and corresponds to the initial correspondences $m_i(x_i, y_i) \Leftrightarrow m_i^{'}(x_i^{'}, y_i^{'})$.
```



(a) (b)

Figure 2.5: *Tukey and Huber weighting functions.*

The robustness of this method relies on the weight assignement to selective points to give less importance to the false points in the resolution of linear system. Assessments have shown that this algorithm is robust to false points due to poor locations but not to false matches. This is because it does not remove false matches but only assignes a low weight to them.

2.4.3.2 RANSAC (RANdom SAmple Consensus)

The RANSAC algorithm (see algorithm3) has been proposed by Fischler and Bolles [A.M. Fischler 1981]. Unlike conventional regression techniques that use as many points as possible, the RANSAC algorithm (see algorithm 3) is based on the estimation of a model from a minimum of s points (eg: $s = 2$ points for a line, $s = 4$ points for a homography, $s = 7$ points for a fundamental matrix) randomly drawn. This estimation is repeated several times, the selected model is the one that has received the greatest support. The support is defined as the number of points whose distance from the model is below a threshold. The implicit idea is that if a false match has been used to estimate the model, so it will not have a great support.

Algorithm 3 RANSAC (RANdom SAmple Consensus) algorithm

```
Input : A set of de 8 matches $m_i(x_i, y_i) \Leftrightarrow m_i^{'}(x_i^{'}, y_i^{'})$ or more.
Output : Fundamental matrix $F$.
(i)Select randomly $m$ subsets of 8 matches.
(ii) For each subset $k$ compute the fundamental matrix $F_k$ using the normalized 8 point algorithm.
(iii) For each fundamental matrix $F_k$ compute the residual error $d_i$ associated to each match $m_i \Leftrightarrow m_i^{'}$ and a robust estimation of the standard deviation $\sigma$ and the threshold $t = 1.96\sigma$ :
$\widehat{\sigma} = 1.4826\left[1 + \frac{5}{(n-7)}\right]\sqrt{median_i\,|d_i|}$.
(iv)Compute the number of inliers such that $d_i < t$.
(v) Select the best solution corresponding to the largest number of inliers.
(vi) Compute the fundamental matrix $F$ using the inliers.
```

Suppose we have a proportion ε of data contaminated by false points and the probability P that at least one of m randomly drawn subsets does not contain false points, then $(1 - (1 - \varepsilon)^s)^m = 1 - P$. Thus, we deduce the minimum number of iterations [Hartley 2003, Zhang 1996]:

$$m = \frac{log(1 - P)}{log(1 - (1 - \varepsilon)^s)}$$

The RANSAC algorithm [Hartley 2003, Zhang 1996] includes the automatic calculation of the threshold (see algorithm 3), thus allowing the calculation of the support of each estimated model. This algorithm is robust to false matches and allows an automated estimation of the fundamental matrix. Another advantage is the ability to make an adaptative estimate of the fundamental matrix F by the calculation of the ratio ε of data contamination at each iteration, by estimating the support of the model.

2.4.3.3 LMedS (Least Median of Squares)

The principle of the LMedS algorithm originally proposed by Rousseeuw [P. J. Rousseeuw 1987] is to solve the nonlinear minimization problem: $min(median_i r_i^2)$, where r_i^2 is the squared residual error associated with the correspondence between the points m_i and $m_i^{'}$ (see equation (2.23)). The principle of solving this problem is described by the algorithm 4 [Zhang 1996] and is quite similar to the RANSAC algorithm. The difference is based on the criterion of model selection. Contrary to RANSAC, which selects the model with the greatest support, the LMedS algorithm retains the model giving the smallest median squared residual error.

Algorithm 4 Algorithm LMedS (Least Median of Squares)

Input : A set of de 8 matches $m_i(x_i, y_i) \Leftrightarrow m_i^{'}(x_i^{'}, y_i^{'})$ or more.
Output : Fundamental matrix F.
(i) Select randomly m subsets of 8 matches.
(ii) For each subset k compute the fundamental matrix F_k using the normalized 8 point algorithm.
(iii) For each fundamental matrix F_k compute the median M_k of the squared residual errors : $M_k = median_{i=1,\dots,n}\left[d^2(m_i, F_k^T m_i^{'}) + d^2(m_i^{'}, F_k m_i)\right]$
(iv) Select the fundamental matrix F_k associated to the smallest residual error in M_k.
(v) Compute a robust estimation of the standard deviation σ given by :
$\widehat{\sigma} = 1.4826\left[1 + \frac{5}{(n-7)}\right]\sqrt{M_k}$.
(vi) For each match assign a weight w_i, such that :

$$w_i = \begin{cases} 1 & \text{si } \left|r_i^2\right| \leq (2.5\widehat{\sigma})^2 \\ 0 & \text{else} \end{cases}$$

with $r_i^2 = d^2(m_i, F_K^T m_i^{'}) + d^2(m_i^{'}, F_K m_i)$ **(vii)** Compute the fundamental matrix F on the set of correspondences whose weight is not zero.

Correspondences whose weight is zero are considered as false points and are no longer taken into account in estimating the fundamental matrix. Thus the fundamental matrix is correctly estimated because the wrong points are discarded before the final estimate of F. Performance improvements of this algorithm is to have sub-sets of points scattered in the image. Indeed, if we chose the subset of correspondences randomly, without taking into account their spatial distribution in the image, we can select points close to each others and the estimation of the epipolar geometry becomes ill conditioned. A solution to this problem is to use the technique known as "bucketing". The principle of this method involves cutting the image into grids and randomly choose $s = 8$ matches from different grids. The technique proposed by Zhang [Zhang 1996] is implemented here, and its principle is described in the algorithm 5. This technique ensures that each point has the same probability of being selected. Thus, the calculation of the number of iterations needed for the LMedS algorithm remains valid.

Algorithm 5 Bucketing algorithm.

Input: Set of n points $m_i(x_i, y_i)$, b^2: number of buckets.
Output: s points drawn randomly.
(i) Calculate the min and max of x_i and y_i (bounding rectangle).
(ii) Cut the bounding rectangle into $b \times b$ buckets.
(iii) The buckets that have no points attached are excluded.
(iv) Cut the interval $[0, 1]$ into L subintervals such that the length of the i^{th} interval is equal to $\frac{n_i}{n}$ with n_i the number of points included in the bucket number i and L the total number of non empty buckets.
(v) During the selection procedure of s buckets, a number produced by a uniform random generator between 0 and 1 belonging to the i^{th} interval implies that bucket number i is selected.
(vi) For each bucket chosen randomly select one of its points.

Figure 2.6 illustrates the different phases of selection of points by the bucketing technique on an image of the Wadham College sequence [Hartley 2003].

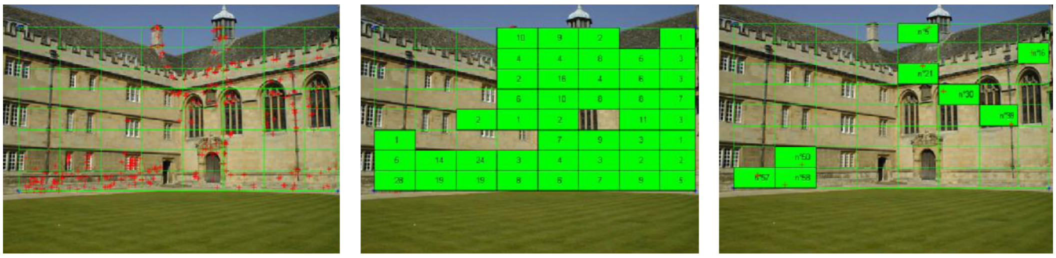


Figure 2.6: *Principle of the bucketting technique for selecting candidate points : (left) cut the region into* 8×8 *buckets, (middle) buckets with no points are removed , (right) select* $s = 8$ *buckets.*

2.4.3.4 Evaluation

The evaluation procedure of the quality of the estimated fundamental matrix and epipolar geometry is defined by the average of the residual error between epipolar lines and points correspondences :

$$Q_F(pixels) = \sum_{i=1}^{n} \frac{d(m_i, F^T m_i^{'}) + d(m_i^{'}, F m_i)}{n}$$

where $(m_i, m_i^{'})$ represents a pair of points in correspondence and d the Euclidean distance.

Figure 2.7 represents the quality assessment of the fundamental matrix estimation algorithms obtained by RANSAC and LMedS according to the proportion of data contamination by false points on two images of the Wadham College sequence.

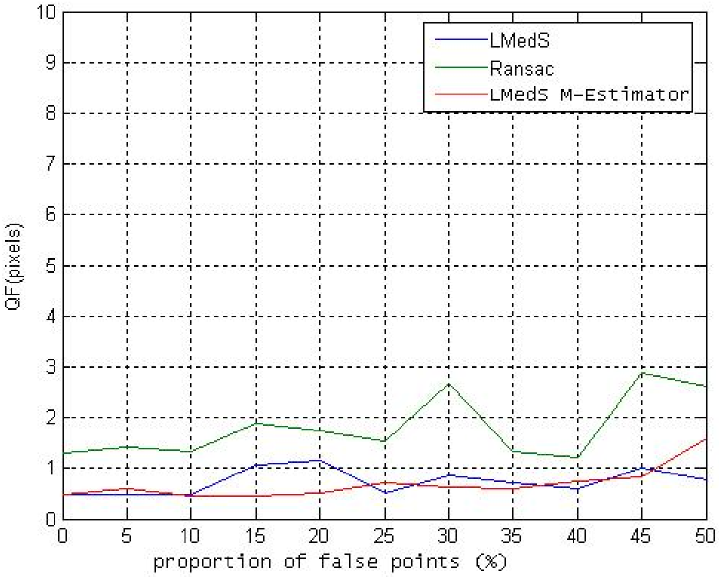


Figure 2.7: Error evaluation of algorithms for estimating the fundamental matrix on two images of the Wadham College sequence.

The LMedS algorithm achieves more reliable results than RANSAC and this for a proportion of false points below 50%. Once the LMedS algorithm rejected the "false points" we use the normalized 8 point algorithm to estimate the fundamental matrix. However, the points selected by the LMedS algorithm may still contain false points that introduce bias to the estimation process. The solution to this problem is to use the M-estimator algorithm to reduce the effect of these false points.

Estimating the fundamental matrix, and thus the epipolar geometry is an important stage within a structure from motion framework. Several algorithms and techniques exist in the literature for estimating this matrix. Among the algorithms robust to false matches we selected the LMedS algorithm originally proposed by [P. J. Rousseeuw 1987] combined to the bucketing technique proposed by Zhang [Zhang 1996]. The contribution consists to use an M-estimator using Huber's weight function [Huber 1981] as a final step of re-estimating the final fundamental matrix on all the inlier points instead of using the normalized 8 point algorithm. Tests on several image sequences were used to validate the combined algorithm LMedS / M-estimator and showed its relevance with regard to conventional methods. Figure 2.8 illustrates the filtering of false matches by the estimation of the epipolar geometry.

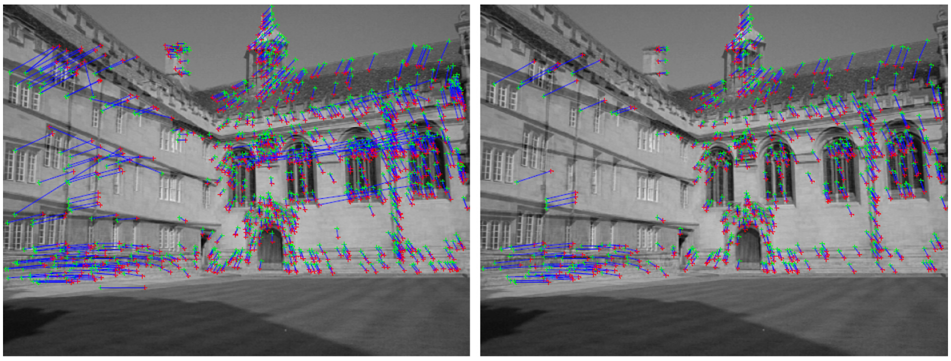


Figure 2.8: *Filtering image correspondences using the epipolar geometry : (left) initial point matching using SIFT : 921 points ; (right) result after estimating the fundamental matrix: 365 outlier points were rejected (556 inlier point correspondances).*

2.5 Structure from motion

2.5.1 Overview

The relationships between the different views (epipolar geometry) will be used to jointly estimate the camera motion and scene structure. This problem is known under the term "Structure from Motion" (SFM) [P.A. Beardsley 1997]. The SFM approach consists in initializing a 3D reconstruction using 2 views, then in estimating the projection matrices for each view using the $2D-3D$ point correspondences. The results of this method are strongly related to the quality of the initial reconstruction. The SFM process is summarized in Algorithm 6. The different steps of our SFM framework including the initialization of the structure, the iterative calculation of the different views camera pose and the bundle adjustment are explained in the following sections.

2.5.2 Initial reconstruction

The SFM pipeline starts by initializing the structure and the motion with a convenient pair of photos. The essential matrix is derived from the fundamental matrix and it represents the calibrated epipolar geometry between two views:

$$E = K_2^T F K_1 = \left[R_2(T_1 - T_2)\right]_x R_2 R_1^T \tag{2.26}$$

Where R_1, R_2, T_1 and T_2 are repectiveley the camera orientation and translation of the two cameras and $K1$, $K2$ their intrinsic matrices. The first camera is chosen so that it is aligned with the world coordinate frame [Pollefeys 2004] and the second

Algorithm 6 Structure from Motion

```
Input : $N$ images, taken by different cameras at different conditions.
1. Linking
    (a) Keypoint detection for all the views using SIFT [Lowe 2003]
    (b) Keypoint matching using a K-D tree data structure [Lowe 2003]
    (c) Fundamental matrix estimation and outlier removal [et R.C. Bolles , Hartley 2003]
    (d) Remove the connections between images whose number of matches is below $n_{min}$ (For example, 20 matches)
    (e) Link keypoint matches over all the views as tracks.
2. Initial reconstruction using 2 views.
    (a) Select the 2 appropriate camera views with known intrinsic parameters such that they have the greatest number of matches and wich can not be well modeled using a homography [Lourakis 2006] to avoid degenerate cases.
    (b) Using the fundamental matrix and the intrinsic camera parameters compute the essential matrix and then the relative camera pose between the initial 2 views (rotation and translation).
    (c) Align the first camera with the world coordinate frame. The second camera pose corresponds to the relative camera motion computed from the essential matrix.
    (d) Compute the projection matrices for the 2 initial views.
    (e) Compute 3D points using the optimal triangulation method on 2D points.
3. Adding views : while there are images to be processed
    (a) Select the camera which has the highest number of reconstructed points
    (b) Robust Estimation of the projection matrix using the DLT (Direct Linear Transform) method [Hartley 2003] inside a RANSAC procedure
    (c) Compute intrinsic and extrinsic camera parameters.
    (d) Add, Remove, and Update 3D points using the optimal triangulation method.
    (e) Local bundle ajustement to optimize the overall reprojection error [Lourakis 2004]
    (f) Exclude cameras with a high average reprojection error.
Output : $N_{cal}$ cameras poses ($N_{cal} \leq N$), reconstructed 3D points.
```

camera is chosen to correspond to the relative camera motion (R,T) computed by an SVD decomposition of essential matrix:

$$E = K_2^T F K_1 = [T]_x R \tag{2.27}$$

Thus, the projection matrices of the initial pair of cameras are given by:

$$P_1 = [I|0]$$
$$P_2 = [R|t]$$

Let S, U and V be respectively the diagonal matrix and two unitary matrices produced by the SVD decomposition of the essential matrix E:

$$E = USV^T$$

Then the rotation and translation are defined by: $R = UWV^T$ or UW^TV^T, and $t = u_3$ or $-u_3$

with:

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This gives four different combinations of rotations and translations which represent combinations for the second projection matrix. We can determine the correct geometric configuration interpretation. The right pair of projection matrices, as shown in Fig 2.9, is the one that provides the 3D points in front of two cameras.

To automatically determine the right pair of projection matrices we can proceed as follows:

- Choose a pair of point matches between the first two images,
- Obtain the corresponding 3D point using the optimal triangulation method,
- Compute the depth of the 3D point for the two cameras for each configuration,
- Select the configuration that gives a positive depth for the two cameras.

The depth of a 3D point X is defined as follows:

$$Depth(X,P) = \frac{sign(Det(M))w}{T \left\| m_3 \right\|}$$

with :

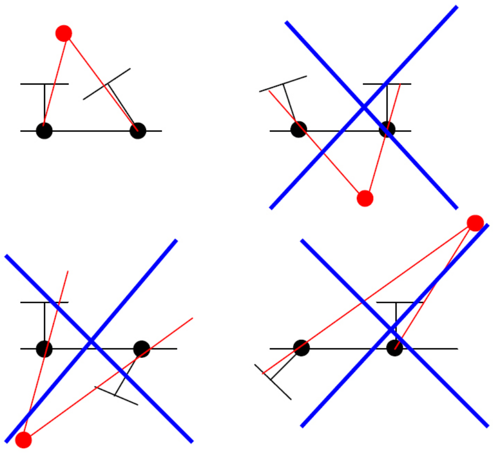


Figure 2.9: *Combinations of orientations and rotations from the SVD decomposition: the configuration chosen is the one which provides the 3D points (gray points) in front of the two cameras (black points).*

- $P = [M|p_4]$ the camera projection matrix (with M a $3x3$ matrix and p_4 a $3x1$ vector)
- $\widehat{X} = [XYZT]^T$
- $P\widehat{X} = w\,[xy1]^T$
- $M = [m_1 m_2 m_3]^T$, m_i rows of M.

The two initial views should be close enough to get enough point correspondences, and far enough such that the problem of triangulation is well conditioned. Indeed, uncertainty on the 3D position of a point obtained by triangulation depends on the angle between the projection rays. For too close views, the angle is low and uncertainty is high. Further details regarding the selection of reference images for initialization of the reconstruction are defined in [Pollefeys 2004]. Once the projection matrices of the first two views are defined, an initial reconstruction of 3D points can be obtained using the optimal triangulation method [Hartley 1997].

2.5.3 Adding views

We iteratively select a new camera and compute extrinsic parameters using the direct linear transform method [Hartley 2003] within a RANSAC scheme followed by an optimization of the re-projection error through gradient descent. The structure is updated by removing, adding or refining 3d points [Pollefeys 2004]. We refine the results through a local bundle adjustment which consists in finding the parameters of cameras and 3d points that minimize the re-projection error. So for m views and n tracks, we try to minimize the following criterion:

$$\min_{P_i, M_j} \sum_{i=1}^{m} \sum_{j=1}^{n} d(P_i M_j, m_j^i)^2 \tag{2.28}$$

We use the sparse bundle adjustment library of Lourakis and Agyros [Lourakis 2004] based on the non linear minimization method of Levenberg-Marquardt to minimize this criterion. Figure 2.10 and Figure 2.11 respectively show an example of a photo sequence and a sparse reconstruction generated with our SFM pipeline.

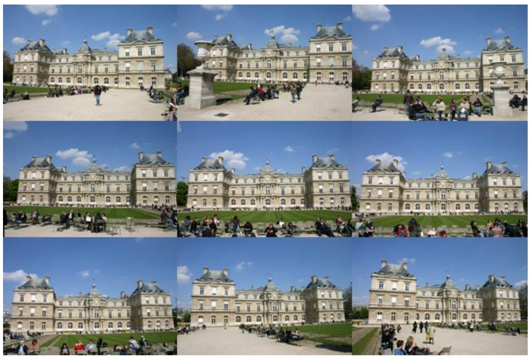


Figure 2.10: *The Luxembourg palace sequence (9 photos, 3072x2048, Canon EOS 300D Digital Camera).*

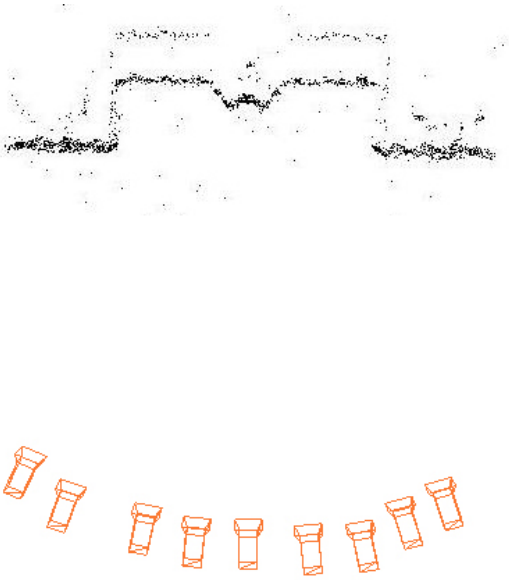


Figure 2.11: *Structure and camera poses of the Luxembourg palace sequence (9 cameras, 2345 3d points).*

2.5.4 Results

Table 2.1 shows the computation times and results for some image sequences. The calculations were performed on a 1.4 GHz PC with 2GB RAM.

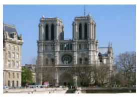
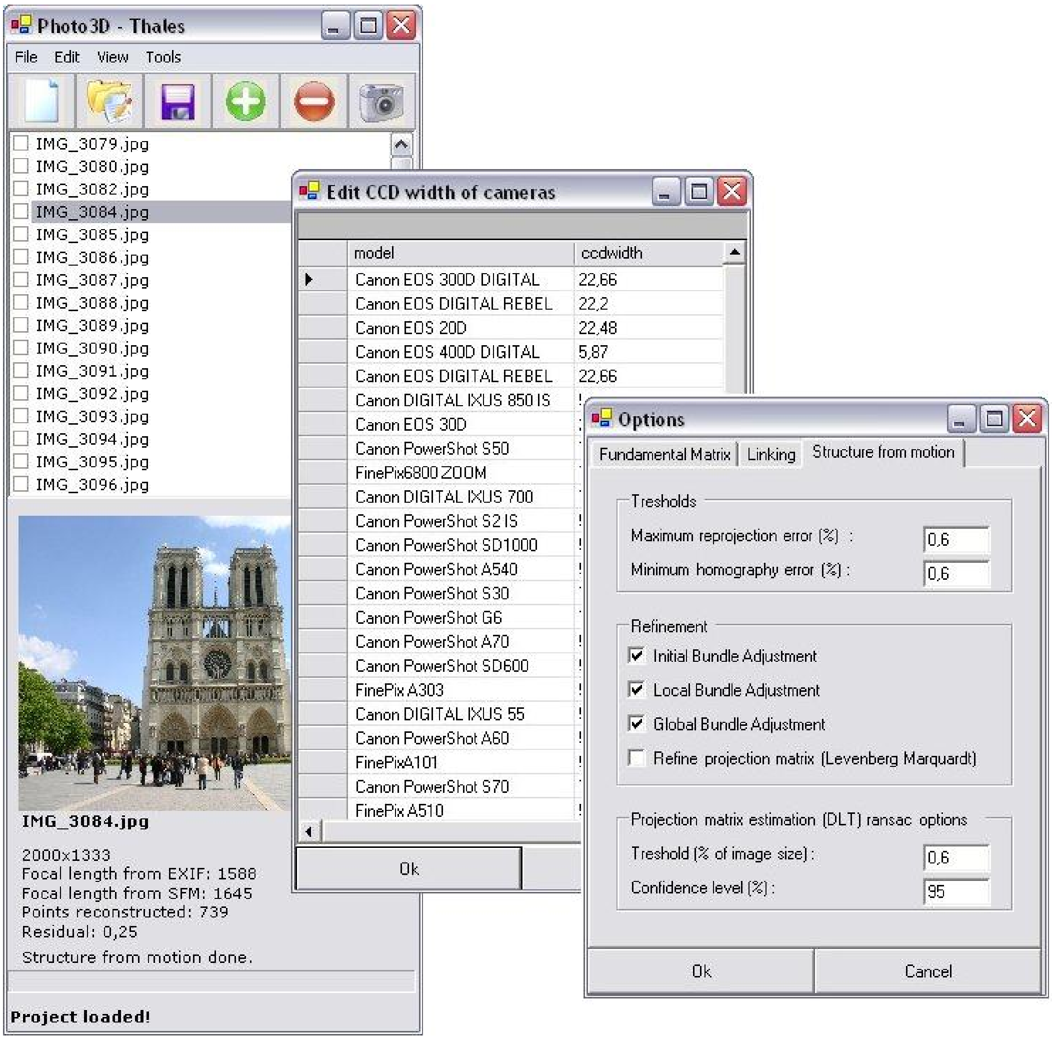
Name	Luxembourg	Notre-Dame 	Arenberg	Temple	Triomphe
Resolution	$1536 * 1024$	$1896 * 1350$	$768 * 576$	$2050*1543$	$1912*1440$
N	9	277	22	39	48
N_{cal}	9	170	22	29	25
$t_{detection\ SIFT}$	2 min	3 h	4 min	15 min	25 min
$t_{matching}$	3 min 15 s	4 days	3 min	45 min	2 h
$t_{FM\ estimation}$	15 s	2 h	10 s	1 min	3 min
$t_{linking}$	30 s	8 h	30 s	3 min	12 min
t_{SFM}	1 min	3 days	50 s	12 min	15 min
t_{total} (approx.)	7 min	8 days	9 min	2 h 11 min	2 h 55 min
n_{SIFT} (mean)	9457	11002	6100	8152	10050
$n_{points\ 3D}$	5773	97418	10086	10355	13606
Error	0.38 pixels	1.23 pixels	0.19 pixels	0.84 pixels	1.05 pixels

Table 2.1: Structure from motion computation times for different photo collections

The user interface of the Photo3D software that allows the user to create SFM projects using images sequences is illustrated in figure 2.12.

The 3D user interface integrated to the Photo3D software is illustrated in figures 2.13 and 2.14 and shows the Notre Dame de Paris image sequence calibrated using our SFM framework. This sequence was downloaded from The FlickR website and consists in 300 photographs of Notre Dame de Paris taken by different cameras at different conditions and times. Among the 300 photographs, 170 cameras were succefully calibrated. The intrinsic camera parameters are extracted from the Exif header in JPEG images. This header contains for example the focal length of the camera.



IMG_3084.jpg

2000x1333
Focal length from EXIF: 1588
Focal length from SFM: 1645
Points reconstructed: 739
Residual: 0,25
Structure from motion done.

Project loaded!

Edit CCD width of cameras

model	ccdwidth
Canon EOS 300D DIGITAL	22,66
Canon EOS DIGITAL REBEL	22,2
Canon EOS 20D	22,48
Canon EOS 400D DIGITAL	5,87
Canon EOS DIGITAL REBEL	22,66
Canon DIGITAL IXUS 850 IS	[illegible]
Canon EOS 30D	[illegible]
Canon PowerShot S50	[illegible]
FinePix6800 ZOOM	[illegible]
Canon DIGITAL IXUS 700	[illegible]
Canon PowerShot S2 IS	[illegible]
Canon PowerShot SD1000	[illegible]
Canon PowerShot A540	[illegible]
Canon PowerShot S30	[illegible]
Canon PowerShot G6	[illegible]
Canon PowerShot A70	[illegible]
Canon PowerShot SD600	[illegible]
FinePix A303	[illegible]
Canon DIGITAL IXUS 55	[illegible]
Canon PowerShot A60	[illegible]
FinePixA101	[illegible]
Canon PowerShot S70	[illegible]
FinePix A510	[illegible]

Options

Tresholds

Maximum reprojection error (%) : 0,6

Minimum homography error (%) : 0,6

Refinement

☑ Initial Bundle Adjustment

☑ Local Bundle Adjustment

☑ Global Bundle Adjustment

□ Refine projection matrix (Levenberg Marquardt)

Projection matrix estimation (DLT) ransac options

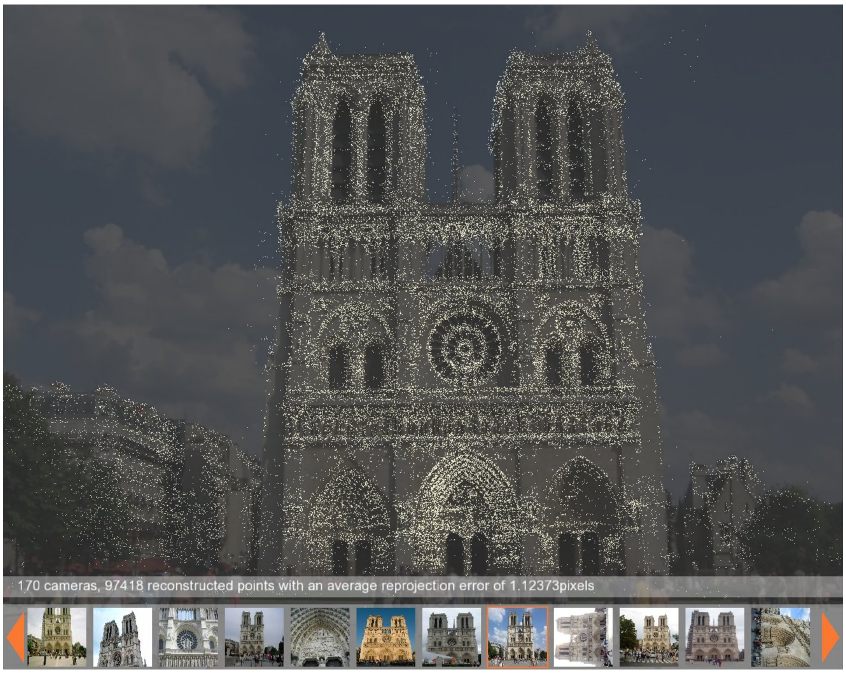
Treshold (% of image size) : 0,6

Confidence level (%) : 95

Figure 2.12: Photo3D User Interface : the user interface allows to create, load and update SFM projects. The user can also tune the SFM parameters and manually update the camera database.

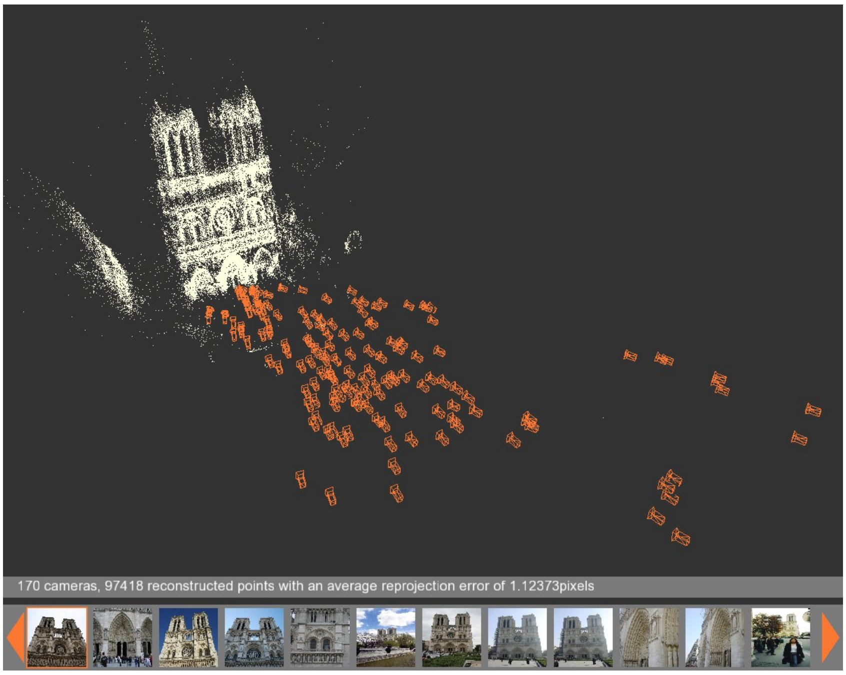
2.6 Registration

The proposed solution described by Figure 2.15 consists of four parts. Here we give a short description of the individual parts, with details provided in the later sections. (1) Structure from motion: The input is a set of un-calibrated photographs. We use SFM techniques to iteratively recover a sparse structure of a scene and the motion of the camera. (2) Recovering walls: The input of this part is a sparse set of 3d points and camera projection matrices. We use an iterative RANSAC (RANdom SAmple Consensus) scheme together with a recursive facade splitting algorithm to recover the principal walls of a scene. (3) Model fitting: The input is a CAD model and a set of principal walls of a scene. Fitting the CAD model to a sparse SFM reconstruction



170 cameras, 97418 reconstructed points with an average reprojection error of 1.12373pixels

Figure 2.13: The sparse 3D reconstruction of Notre Dame de Paris seen from a selected camera viewpoint.



170 cameras, 97418 reconstructed points with an average reprojection error of 1.12373pixels

Figure 2.14: The sparse 3D reconstruction of Notre Dame de Paris and the camera poses seen from an arbitrary camera viewpoint.

is then defined and solved as an absolute orientation problem. (4) Visualisation and rendering: Once the model is fitted we use projective texture mapping technique to obtain a photorealistic rendering of the CAD model.

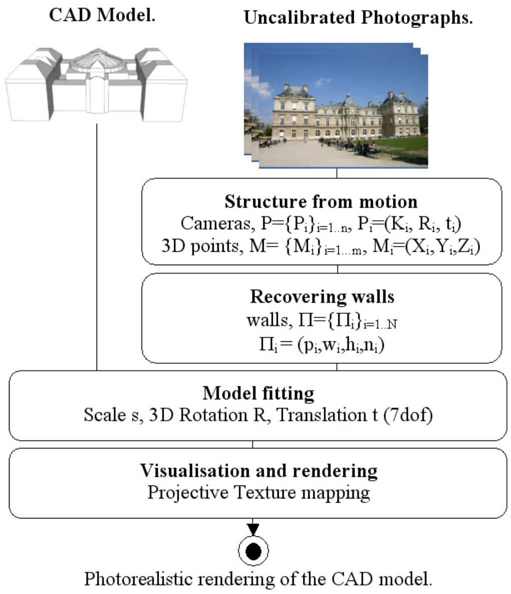

CAD Model.
Uncalibrated Photographs.
Structure from motion
Cameras, $P=\{P_i\}_{i=1..n}$, $P_i=(K_i, R_i, t_i)$
3D points, $M=\{M_i\}_{i=1...m}$, $M_i=(X_i,Y_i,Z_i)$
Recovering walls
walls, $\Pi=\{\Pi_i\}_{i=1..N}$
$\Pi_i=(p_i,w_i,h_i,n_i)$
Model fitting
Scale s, 3D Rotation R, Translation t (7dof)
Visualisation and rendering
Projective Texture mapping
Photorealistic rendering of the CAD model.

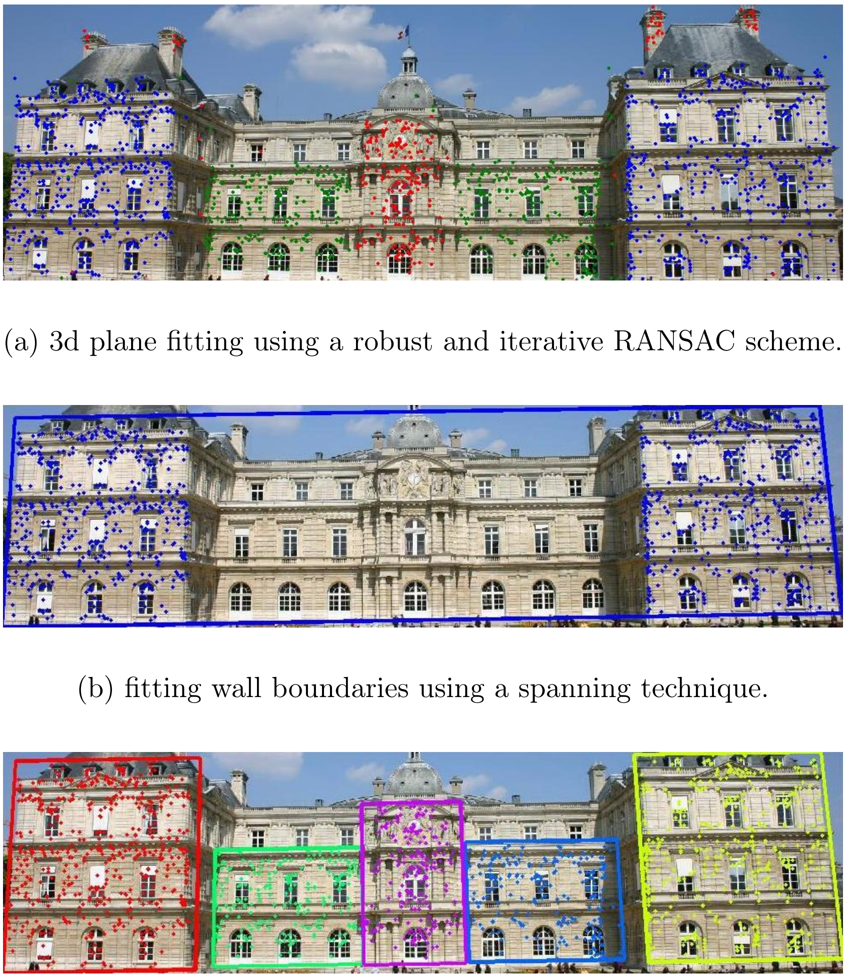
Figure 2.15: System overview.

2.6.1 Recovering walls

We use an iterative and robust RANSAC scheme to fit 3d dominant planes of a building from a sparse reconstruction. Each wall is represented by a rectangle and has 4 parameters: height h, width w, position p and normal n. We begin by estimating the normal to the ground floor from the camera centers. Then we iteratively recover 3d dominant planes of the scene from the set of sparse 3d points using a robust RANSAC scheme. The 3d planes are constrained to be orthogonal to the ground floor plane. Figure 2.16 (a) shows the re-projection of different sets of 3d points belonging to fitted 3d planes. Note that there are some points belonging to the same 3d plane (red points in chimney and roof windows) but do not belong to the same wall plane. Another problem is that two different walls belonging to the same 3d plane gives rise to the same entity.

Second, we recover the boundaries of each wall plane by spanning each 3d point set belonging to the same 3d plane in the horizontal and vertical directions [A. 2003]. Figure 2.16 (b) illustrates a resulting boundary of a wall plane together with the set of inlier 3d points.

Finally, we use a recursive splitting method to subdivide and filter wall planes



(a) 3d plane fitting using a robust and iterative RANSAC scheme.

(b) fitting wall boundaries using a spanning technique.

(c) wall subdivision and filtering using a recursive and robust splitting technique.

Figure 2.16: Robust wall fitting using a combined spanning and splitting technique. Each 3d point set belonging to a different wall plane is plotted with a different color.

into different facades. This technique consists of splitting in the vertical and horizontal directions the in-lying 3d points of a wall plane and recursively subdividing around lines that separate parts containing significant 3d points. Figure 2.17 shows an illustration of the principle of this method: the input is a wall plane and in-lying 3d points and the output is a subdivision into main facades. Note that the badly localized points are removed by defining a threshold for the size of facades and the number of in-lying 3d points which must be significant to obtain a real facade. Figure 2.16 (c) shows the final result obtained with this method. Note that we solve at the same time the problem of badly localized points (red points in chimney and roof in Figure 2.16 (a)) and the subdivision of wall planes containing more than one facade.

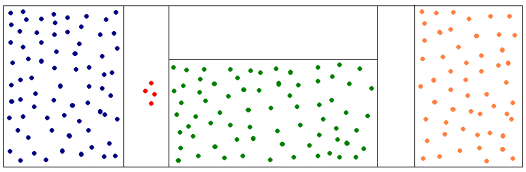


Figure 2.17: *Recursive vertical and horizontal splitting technique.*

2.6.2 Model fitting

Fitting the 3d model of a building to a sparse set of facade planes can be formulated as an absolute orientation problem. The unknown parameters to fit are: the scale s (1 parameter), the 3d orientation R (3 parameters) and the 3d translation T (3 parameters). The idea is to use the fitted walls from the sparse reconstruction and the CAD model of the building to dynamically establish a list of facade correspondences and then compute the absolute orientation. Figure 2.18 illustrates the absolute orientation problem between two facades.

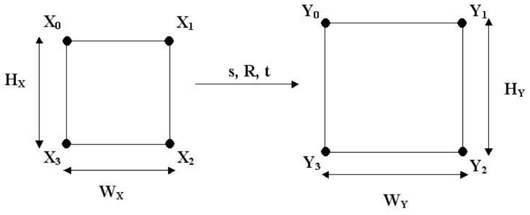


Figure 2.18: *Fitting a model facade to a fitted facade wall plane.*

We compute for each facade four points: top left, top right, bottom right and bottom left. Each facade vertex correspondence are then related by the formula:

$$Y_i = s(R.X_i + T) \tag{2.29}$$

The scale parameter s is computed as the ratio of the widths or heights of both facades. Each point correspondence gives rise to 2 equations, the minimum number of correspondences needed to estimate the rotation and the translation (6 degrees of freedom) is 3 points.

$$X_{mean} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad Y_{mean} = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{2.30}$$

The rotation matrix R is recovered by an SVD decomposition of the matrix A:

$$A = (Y - Y_{mean}).(X - X_{mean})^T \tag{2.31}$$

The translation vector T is then recovered from the rotation matrix using the simple formula:

$$T = Y_{mean} - R.X_{mean} \tag{2.32}$$

Finally, we optimize the results through gradient descent by minimizing the following criterion:

$$\min_{s,R,t} \sum_{i=1}^{N} d(Y_i, s(R.X_i + T))^2 \tag{2.33}$$

To dynamically establish the list of facade correspondences the solution is implemented as a pseudo-RANSAC procedure. For each fitted facade we try to find a facade in the model that has almost the same height/width ratio. This allows to directly eliminating false facade correspondences. Then, we estimate the absolute orientation and compute the consensus. We chose the resulting facade correspondences that give rise to the largest consensus and robustly estimate the absolute orientation using only the in-lying facade correspondences. Figure 2.19 shows an illustration of a final result obtained with our method for fitting a CAD model to the SFM sparse reconstruction.

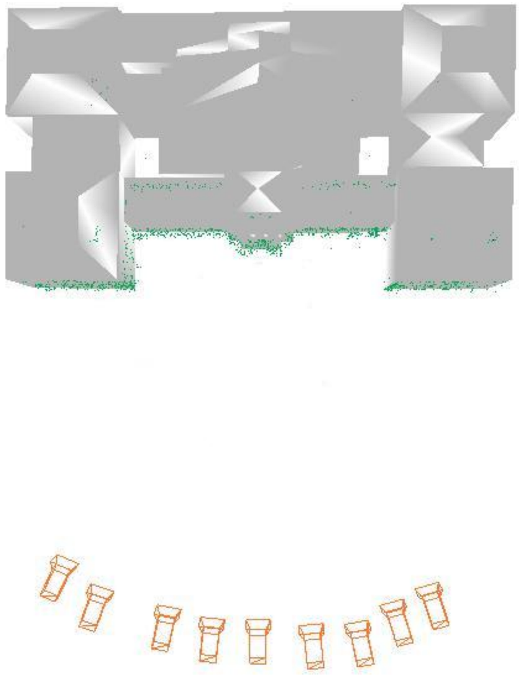


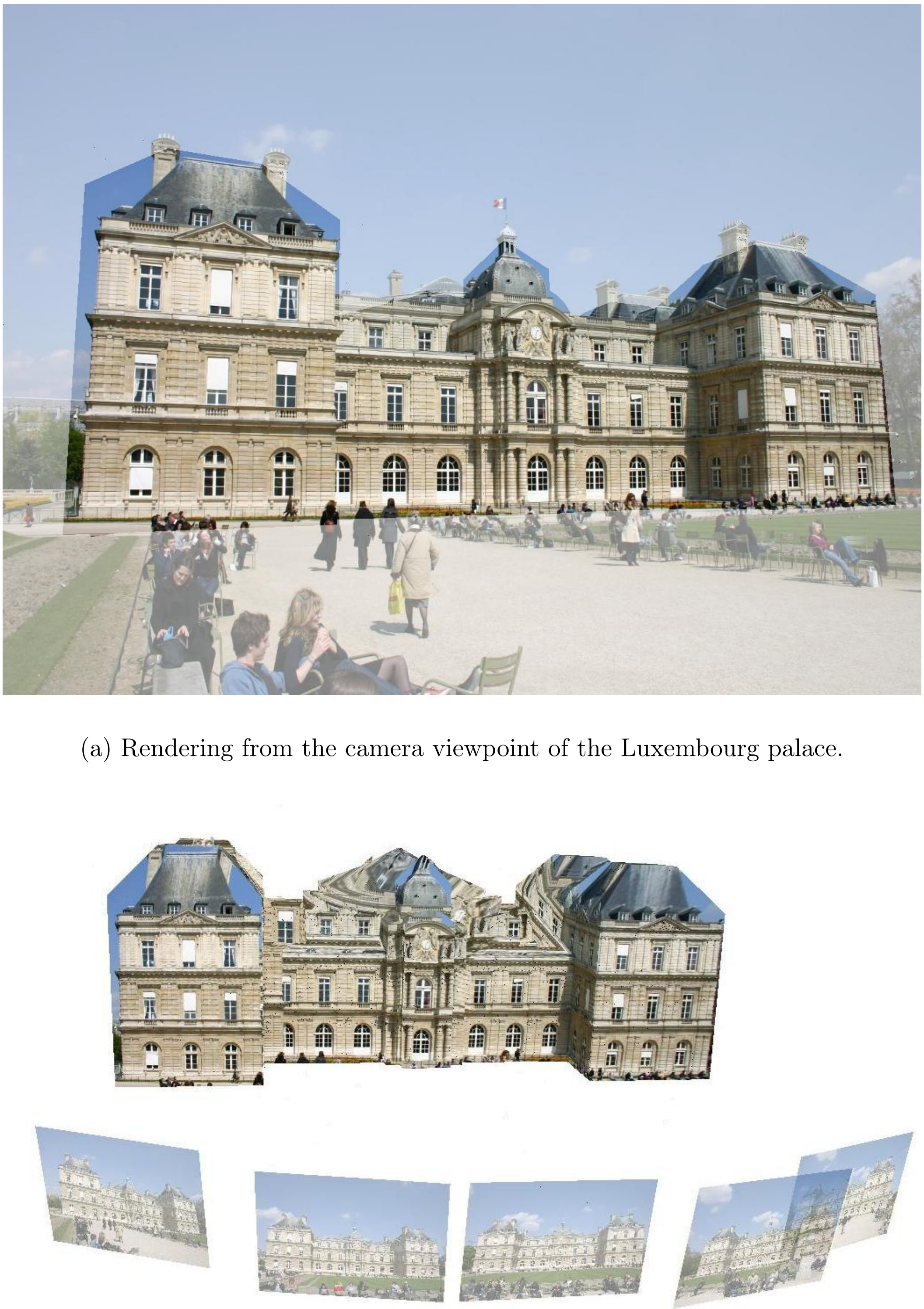
Figure 2.19: CAD model of the Luxembourg palace automatically fitted to the computed structure and motion.

2.6.3 Visualisation and rendering

We developed an OpenGL-based user interface for the visualization and rendering of the CAD model using projective texture mapping technique. The camera selected by the user is used to project texture onto the model. Figure 2.20 shows two different views for the rendering of a CAD model using our interface.

2.7 Conclusion

This chapter introduces methods and techniques for recovering wall planes from a sparse reconstruction and for fitting CAD models to a set of un-calibrated photographs. These two elements form the major contribution of this chapter. Our structure from motion pipeline is mainly inspired from state of the art techniques but involved some contribution such the combination of LMedS and the M-Estimator algorithms for estimating the fundamental matrix and the use of the bucketing technique to improve the spatial distribution of points.



(a) Rendering from the camera viewpoint of the Luxembourg palace.

(b) Rendering from an arbitrary viewpoint.

Figure 2.20: A photorealistic rendering of the CAD model using projective texture mapping.

Chapter 3

Multi-View Stereo

This chapter presents an accelerated implementation of a multi-view stereo pipeline using parallel processing capabilities of the GPUs. Our system takes as input a set of calibrated photographs and produces a textured 3D mesh of the scene. The pipeline is divided into three parts: dense stereo matching, multi-view correspondence linking and 3D model generation. First, we use a combined vertical aggregation and dynamic programming (DP) scheme to produce disparity maps between pairs of photographs. Then, the depth maps are computed using a multi-view correspondence linking algorithm. Finally, we use a Delaunay triangulation algorithm and texture mapping to produce the 3D model of the scene.

3.1 Introduction

We present in this chapter an accelerated implementation of a multi-view stereo pipeline that takes advantage of the parallel processing capabilities of actual GPUs. Stereovision is a very challenging research topic in computer vision. The research community has devoted a lot of effort to developp algorithms and techniques for an automatic 3D reconstruction of a scene from a set of uncalibrated photographs [Hartley 2003, Pollefeys 2004]. Scharstein and Szeliski [Scharstein 2002] have established a classification of the different dense stereo matching algorithms into local and global methods. Local methods use the intensity of a pixel and of its neighbours to compute the disparity. Global methods make use of optimization techniques (dynamic programming, graph cuts, ...) for the computation of the disparities. Acceleration using graphics hardware to estimate depth was first explored by Yang et al. [Yang 2002] using a plane sweep approach. Cornells and Van Gool [Cornelis 2005] combined this with the iterative refinement from Zach et al. [Zach 2003] to generate quality depth maps for fine 3D structures.

3.1.1 GPU pipeline and GPGPU

Acceleration using graphics hardware has been for a long time restricted to purely graphical processing. With the constant evolution of graphics hardware and the emerging GPGPU techniques and technologies such Cg [CG: 2008] and CUDA [CUD 2008] researchers start to re-design their algorithms to benefit from the parallel capabilities of modern GPUs. GPGPU stands for General Purpose Graphics Processing Unit [GPG 2008]. Stating it briefly, GPGPU is a combination between hardware components and software that allows the use of a traditional GPU to perform computing tasks that are extremely demanding in terms of processing power.

The graphics pipeline is designed to allow hardware implementations to maintain high computation rates through parallel execution. The pipeline is divided into several stages; all geometric primitives pass through every stage. In hardware, each stage is implemented as a separate piece of hardware on the GPU in what is termed a task-parallel machine organization. Figure 3.1 shows the pipeline stages in current GPUs. This pipeline is described in more detail in the OpenGL Programming Guide [Shreiner 2003].

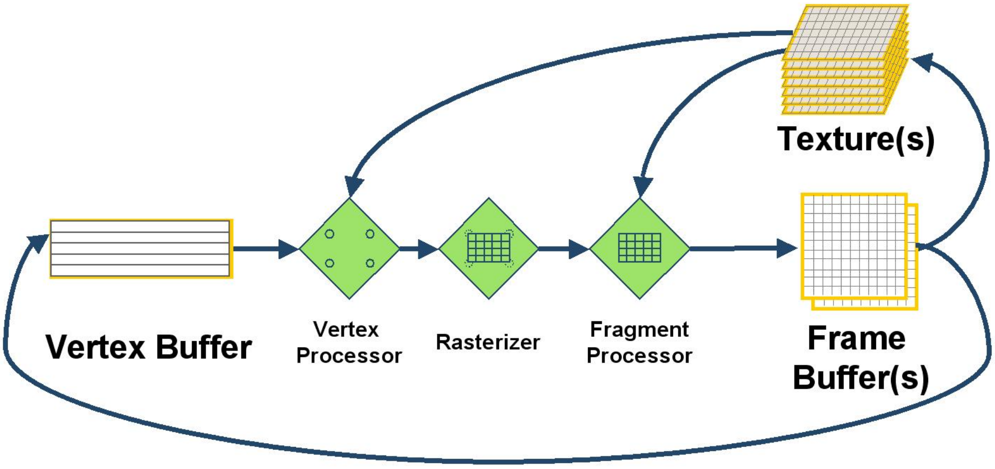


Figure 3.1: *The modern graphics hardware pipeline: the vertex and fragment processor stages are both programmable by the user.*

3.1.2 System overview

The 3D reconstruction pipeline presented in this paper consists of three parts. (1) Dense Stereo Matching: match all the pixels between consecutive image pairs and produces the disparity maps. We use two different dense stereo matching methods: the winner takes all approach, that runs fully in GPU and a dynamic programming approach that runs in both GPU and CPU. (2) Multi-View Correspondence Linking: produces the depth maps using multiple view triangulation of all correspondences. This process fully runs in GPU and forms the major contribution of this chapter. (3) 3D Mesh Generation and Texture Mapping: produces the final 3D model using a Delaunay triangulation and texture mapping.

3.2 Dense stereo matching

Dense stereo matching consists in matching all the pixels between two consecutive images in order to produce disparity maps. Our algorithm has three major steps:

matching cost computation, cost aggregation and disparity selection. To compute the cost volume, we draw a rectangle aligned with the two input rectified images stored as textures, and one of them shifted by d pixels. We adopt the following matching cost criterion:

$$|p(x, y) - q(x + d, y)| \tag{3.1}$$

where p and q are the corresponding matched pixels and d is the hypothesized disparity value. We use a fragment program to calculate the color absolute difference and output it to a texture. After doing this process over all the disparity hypothesis d we have the entire cost volume separated in different textures. One 4-channel texture for each 4 disparity hypothesis. The matching cost computation is done entirely in the graphics processing unit. We compute the weight masks in a similar way using the following equation:

$$w(p, q) = \exp\left(-(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_g})\right) \tag{3.2}$$

where p and q are pixels, Δc_{pq} their color difference, Δg_{pq} their Euclidean distance and γ_c and γ_g are weighting constants determined empirically. The cost aggregation is implemented as a pixel shader that can index between both cost and weight textures to produce a new set of textures that will conform the aggregated cost volume. The aggregated cost is computed as a weighted sum of the per-pixel cost [Wang 2006] and is implemented as a pixel shader. The number of rendering passes needed to do this process is $\lceil \frac{N*H}{16} \rceil$ where N is the number of disparity hypothesis and H is the size of the vertical window. To select the best disparity for each pixel in the image we use both Winner Takes All (WTA) approach and a Dynamic Programming (DP) scheme [Forstmann 2004]. WTA approach simply selects the disparity that has the lower cost while DP selects the disparity trying to minimize a global energy function [Wang 2006]. The WTA algorithm can be fully implemented in the GPU. The DP algorithm can be implemented also in GPU, but as reported in [Wang 2006, Gong 2005], a GPU-based DP program is actually slower than it's CPU counterpart. Figure 3.2 shows the differences between the disparity maps calculated with the two different methods: WTA and DP. Table 3.1 illustrates the real-time performances of the dense stereo matching algorithm.

3.3 Multi-view correspondence linking

The pairwise disparity estimation allows to compute correspondences between adjacent rectified image pairs and independent depth estimates for each camera viewpoint. In order to fuse those separate depth estimates into a common 3D model we use the multi-view correspondence linking algorithm described in [Pollefeys 2004]. For each image point m_k we create two chains of correspondence links one up m_{k+1} and one down m_{k-1} as follows :

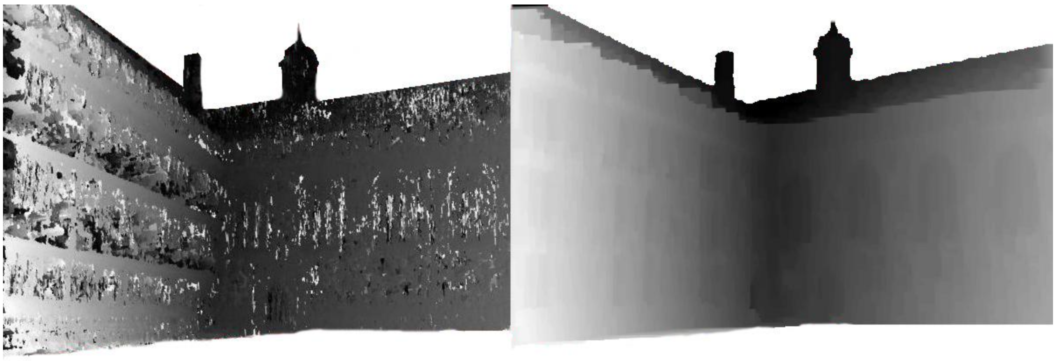


Figure 3.2: *Resulting disparity maps using WTA and DP methods.*

Image Size	DR	VA + DP			WTA
		GPU	CPU	Total	GPU
640x480	16	0.024	0.382	0.407	0.026
	32	0.051	0.731	0.783	0.054
	48	0.076	1.082	1.158	0.081
320x240	16	0.007	0.098	0.105	0.007
	32	0.013	0.182	0.196	0.014
	48	0.019	0.271	0.290	0.020

Table 3.1: *Real-time Performance in seconds. The test system is a 3.0Ghz with a NVIDIA GForce 8800 GTS 512 Mb graphics card. VA+DP denotes Vertical Aggregation + Dynamic Programming, WTA denotes Winner Takes All and DR the Disparity Range.*

$$m_{k+1} = (H^k_{k+1})^{-1} D_{(k,k+1)} [H^{k+1}_k m_k] \tag{3.3}$$

$$m_{k-1} = (H^k_{k-1})^{-1} D_{(k,k-1)} [H^{k-1}_k m_k] \tag{3.4}$$

This linking process is repeated along the image sequence for a reference view i to create a chain of correspondences upwards $(i, i+1, .., n)$ and downwards (i,i-1,...,1) . Every correspondence link requires 2 mappings and 1 disparity lookup. The Disparity map $D_{(k,k-1)}$ holds the downward correspondences from image I_k to I_{k-1} while the map $D_{(k,k+1)}$ contains the upward correspondences from I_k to I_{k+1} . Rectification of image points for a stereoscopic pair of images (I_k, I_{k+1}) is done using transformation matrices H^{k+1}_k and H^k_{k+1}. Once we have located the same pixel in a sequence of images, we can proceed to triangulate them to get a set of depths. And last, we use a Kalman filter to estimate the final depth. We also have to detect when a depth estimation falls out of our previous mean estimation. Outliers can be detected comparing the depth estimation of the new point with the previous filtered mean and taking into account the error in the current step. If we find an outlier, the correspendance chain ends and we have to assure that this pixel will not be triangulated in further steps. Figure 3.3 illustrates the principle of depth estimation and outlier detection using a chain of correspondence link.

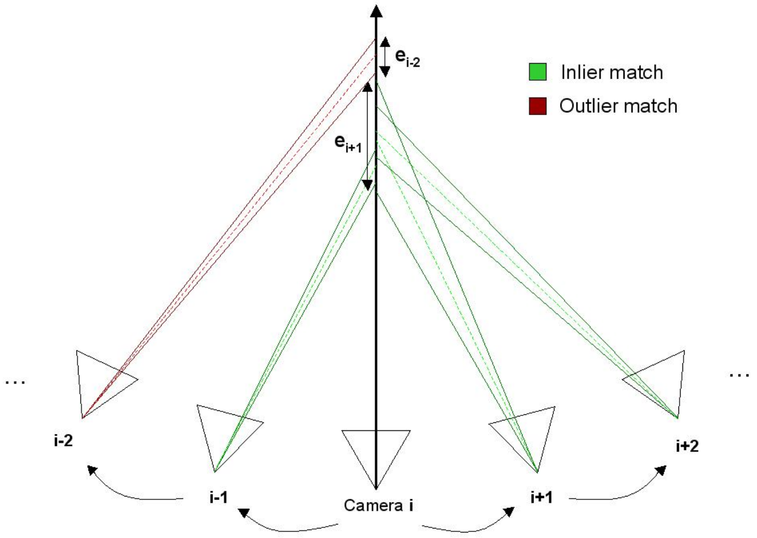


Figure 3.3: *Depth fusion, uncertainty estimation and outlier detection from correspondence linking.*

We have implemented this algorithm in the GPU by performing the linking in an incremental way. We select a camera for which we want to compute the depth map then we step forward and backward to triangulate all the correspondences. At each step we triangulate correspondent image points and estimate the pixels depth values. For each step and for each image point we save the following informations in textures: the coordinates of correspondant point in the next image, the filtered depth estimation and the number of estimations. By this way the algorithm is suitable to run in GPU and generates the depth maps in a very fast way. The CPU implementation takes about 112 s to calculate the depth maps of the Wadham College example, our GPU implementation takes less than 0.5 s. We can see an example of a computed depth map using this algorithm in figure 3.4.

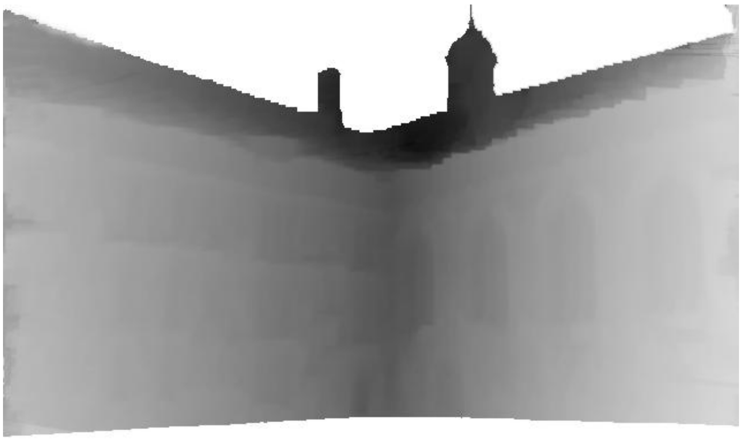


Figure 3.4: *Resulting depth map using the multi-view correspondence linking algorithm.*

3.4 3D mesh generation and texture mapping

The 3D mesh generation is performed using a 2D Delaunay triangulation of a selected depth map followed by a inverse projective transformation. We compute normal vectors of each triangle and extract textures from the original images depending on the camera viewpoint. The visualisation and rendering of the final 3D model is then obtained using the OpenSceneGraph library [OSG 2008]. Figure 3.5 shows some final 3D models generated with our system and Table 3.2 illustrates the computation times.

3.5 Conclusion

This chapter presented algorithms and techniques to accelerate a multi-view stereo pipeline using parallel processing capabilities of the GPUs. Most previous work focus on the acceleralation of two-frame dense stereo matching algorithms. Our GPU implementation of a multi-view correspondence linking algorithm allows 3D reconstruction from multiple images.

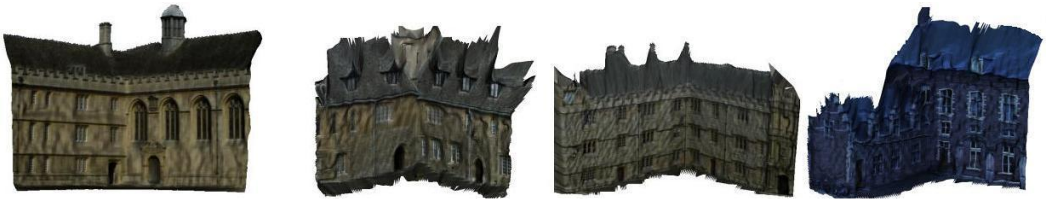


Figure 3.5: *Four views of the reconstructed models. From left to right: Wadham College, Merton College I, Merton College II [VGG 2008a] and Arenberg Castle [MP: 2008].*

Image Sequence	MDR	DSM	MVCL	3DMG&TM	Total
Wadham College (5 images 640x480)	[-34,94]	40.987	0.414	1.122	42.623
Arenberg Castle (22 images 768x576)	[-8,52]	89.301	3.053	1.774	94.128
Merton College I (3 images 640x480)	[-248,90]	418.214	2.942	0.969	422.125
Merton College II (2 images 1024x768)	[-65,106]	244.812	2.370	2.430	249.612

Table 3.2: *Computation times in seconds. The test system is a 3.0Ghz with a NVIDIA GForce 8800 GTS 512 Mb graphics card. MDR, DSM, MVCL and 3DMG&TM respectively denotes for Maximum Disparity Range, Dense Stereo Matching, Multi-View Correpondence Linking and 3D Mesh Generation and Texture Mapping.*

Chapter 4

Voxel Coloring

This chapter presents algorithms and techniques towards a real-time and accurate Voxel Coloring framework. We combine Visual Hull, Voxel Coloring and Marching Cubes techniques to derive an accurate 3D model from a set of calibrated photographs. First, we adapted the Visual Hull algorithm for the computation of the bounding box from image silhouettes. Then, we improved the accuracy of the Voxel Coloring algorithm using both colorimetric and geometric citerions. The calculation time is reduced using an Octree data structure. Then, the Marching Cubes is used to obtain a polygonal mesh from the voxel reconstruction. Finally, we propose a practical way to speed up the whole process using graphics hardware capababilities.

4.1 Introduction

In this chapter we address the problem of real-time 3D reconstruction from photographs. Our framework consists of three parts: (1) computation of the bounding box of the object we want to reconstruct using a Visual Hull approach, (2) a voxel reconstruction based on both colorimetric and geometric criterions (3) and a generation of a polygonal mesh using Marching Cubes techniques. The context of this work is the growing interest in automatic reconstruction techniques from photographs. With the increasing capababilities of modern graphics hardware 3D reconstruction techniques can be accelerated to obtain accurate models in real-time.

4.2 Related Work

The original Voxel Coloring paper described in [Seitz 1997] uses only colorimetric criterions to reconstruct an object consistent with the input images. This algorithm starts by discretizing the 3D space into voxels and projects them on each image. The voxels that are consistent from a colorimetric viewpoint with the images are retained. The complexity of this algorithm is $O(N^3 * n)$ with N^3 is the number of voxels and n the number of images. In order to improve the accuracy of this method we use both colorimetric and geometric criterions to derive 3D models from image silhouettes. The calculation time is improved using an Octree data structure. The Visual Hull algorihm [Franco 2003] operates in a different manner by projecting the image silhouettes into the 3D space. The intersection of the silhouettes cones produces the 3D polygonal model. We adapted this algorithm to compute the bounding box of the 3D object which is needed for the voxel reconstruction. Instead of projecting the image silhouettes into the 3D space we project their 2D bounding boxes.

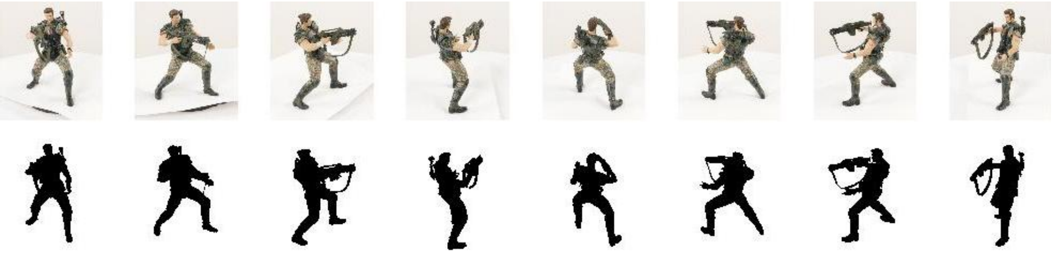


Figure 4.1: A sample data set: Our system takes as input a set of calibrated images and silhouettes and produces a textured polygonal model. [VGG 2008b]

The Marching Cubes technique [Lorensen 1987, F. Goetz 2005] takes as input a 3D point cloud and produces a textured polygonal mesh. Acceleration using graphics hardware has been for a long time restricted to purely graphical processing. With the constant evolution of graphics hardware and the emerging GPGPU (General Purpose GPU) techniques and technologies such Cg [W. R. Mark 2003] and CUDA [Cuda 2008] researchers start to re-design their algorithms to benefit from the parallel capabilities of modern GPUs [Trendall 2000, Krueger 2003, F. Goetz 2005].

4.3 Our Approach

Given a set of calibrated images and silhouettes our system produces a textured polygonal model. Figure 1 illustrates a sample input data set used for evaluating our reconstruction pipeline. The proposed solution described consists of three parts. First we compute the bounding box of the object we want to reconstruct using a Visual Hull approach. Then we reconstruct the 3D object with a colorimetric and geometric consistency based Voxel Coloring scheme. Finally, we produce a textured polygonal model using the Marching Cubes technique. The Voxel Coloring algorithm is accelerated using graphics hardware.

4.3.1 Visual Hull

The Visual Hull algorihm [Franco 2003] computes a 3D coarse representation of an object from it's 2D projections in a set of images. Figure 2 illustrates this algorithm in the simple case of 2 calibrated images. Given (1) two projection matrices P_A and P_B (2) two 2D regions D_A and D_B representing the projection of the same 3D object we compute the 3D cones V_A and V_B and intersect them to compute the coarse 3D model C.

We adapted this algorithm to compute the bounding box of the 3D object which is needed for the voxel reconstruction. Instead of projecting the image silhouettes into the 3D space we project their 2D bounding boxes.

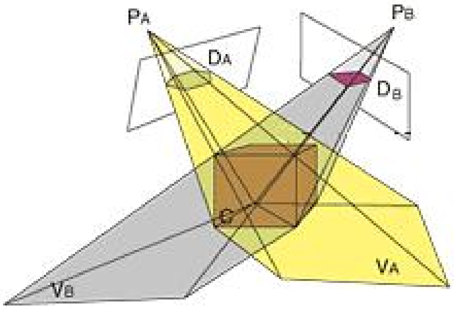


Figure 4.2: Illustration of the Visual Hull principle.

4.3.2 Voxel Coloring

The Voxel Coloring algorithm described in [Seitz 1997] uses a colorimetric criterion to decide if a voxel is consistent or not. Thus voxels can be colorized even if their projection is totally outside of the object silhouette. In order to improve the accuracy of the Voxel Coloring algorithm we add a geometric criterion [Kuzu 2001]. Figure 3 illustrates the principle of the use of silhouettes: voxels are identified respectively as gray, black or white depending if their projection into images falls in the boundary, outside or inside of the silhouettes.

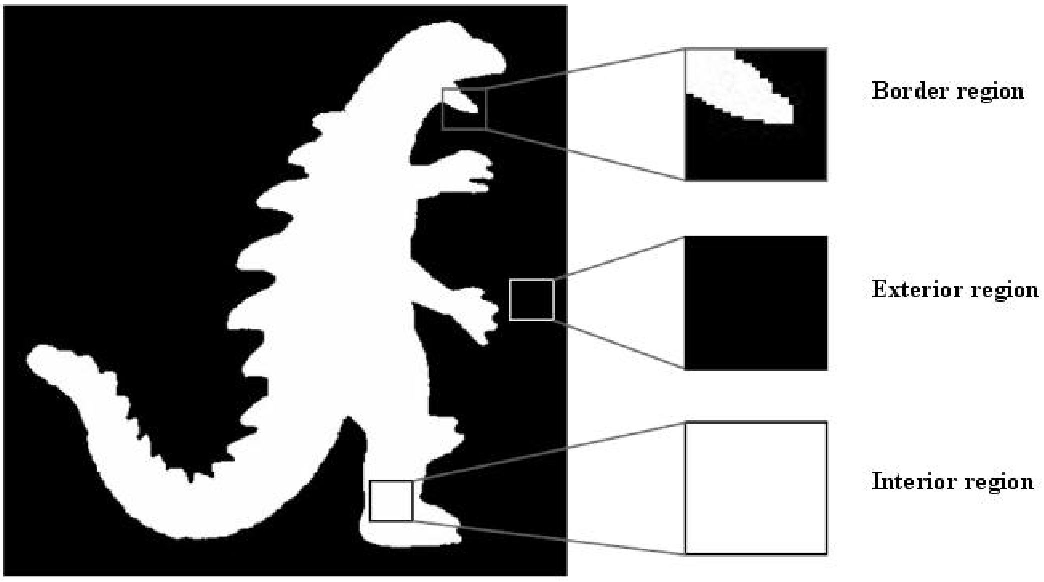


Figure 4.3: Illustration of the geometric consistency check.

To reduce the complexity of the algorithm we use an Octree data structure [A. W. Fitgibbon 1998]. We recursively subdivide the volume into 8 subvolumes. The subdivision of a volume is made only if it's projection into images is on the boundary of the silhouette. Thus the object is reconstructed in an economic way. Figure 4 shows an illustration of a volume subdivision and it's associated data structure.

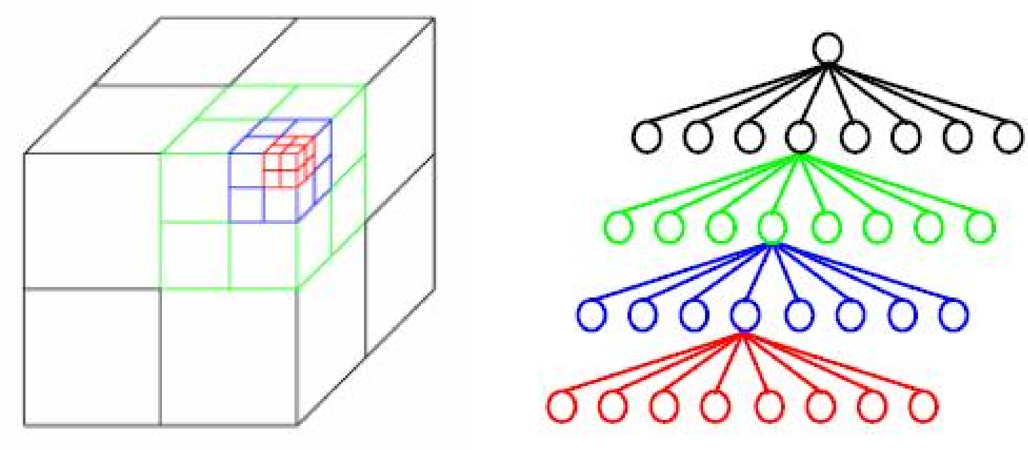


Figure 4.4: Recursive subdivision of a volume and the associated octree.

We also use a 3D connexity check to improve the surface of the voxel reconstruction. We proceed as follows to update the voxel classification: (1) we check the 6 neighbours of each gray voxel and if no black voxel is found we identify it as white, (2) we also check the neighbourhood of each white voxel and if at least one black voxel is found we identify it as gray. Figure 5 illustrates the principle of this algorithm.

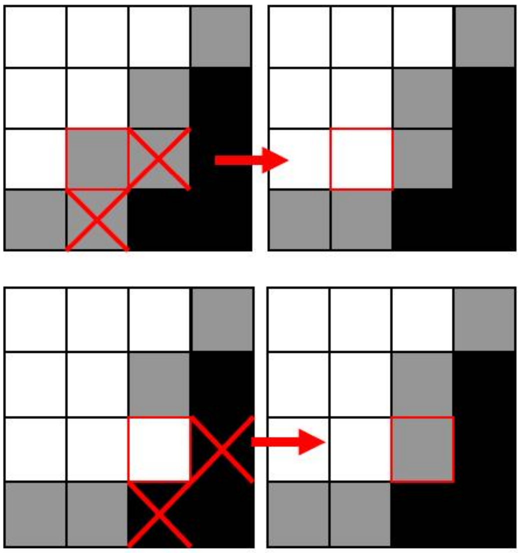


Figure 4.5: Illustration of the voxels connexity check.

4.3.3 Marching Cubes

The Marching cubes algorithm [Lorensen 1987] is used to obtain a polygonal model from a scattered set of voxels. The algorithm starts by taking eight neighbor locations to construct a cube, then determine the polygons that passes through this cube. The individual polygons are then fused into the model surface. We use an index of precalculated array of 256 possible polygon configurations ($2^8 = 256$) within the cube. This array of 256 cube configurations is obtained by reflections and symmetrical rotations of the basic cases illustrated by Figure 6.

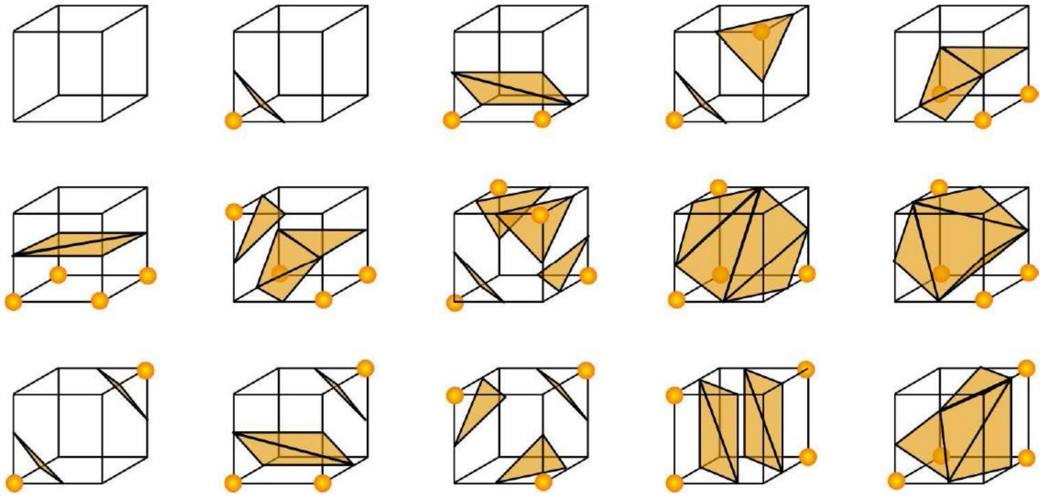


Figure 4.6: 15 basic configurations of polygons.

4.3.4 Acceleration Using Graphics Hardware

The GPU architecture is specialized for parallel computing tasks. The graphics hardware consists of a set of processors grouped together in a common multiprocessors block. Figure 7 illustrates the hierarchy of the parallel architecture of actual GPUs.

For example the device used in this work is the NVidia GeForce 8800GTS card. This card mainly consists of 12 multiprocessors and a 512 MB device memory. Each multiprocessor is composed itself by 16 processors, a shared memory and an instruction unit. This card can thus make 192 calculations in parallel. Unlike the device memory the shared memory consists of 16 Kb and is accessible only by processors belonging to the same multiprocessor unit. However this memory is very usefull and can be accessible much more faster than the device memory. To execute hundreds of processes working in various programs, multi-processors use new architecture called SIMT (Single Instruction Multiple Thread).

The Voxel Coloring process is accelerated using the parallel capabilities of modern

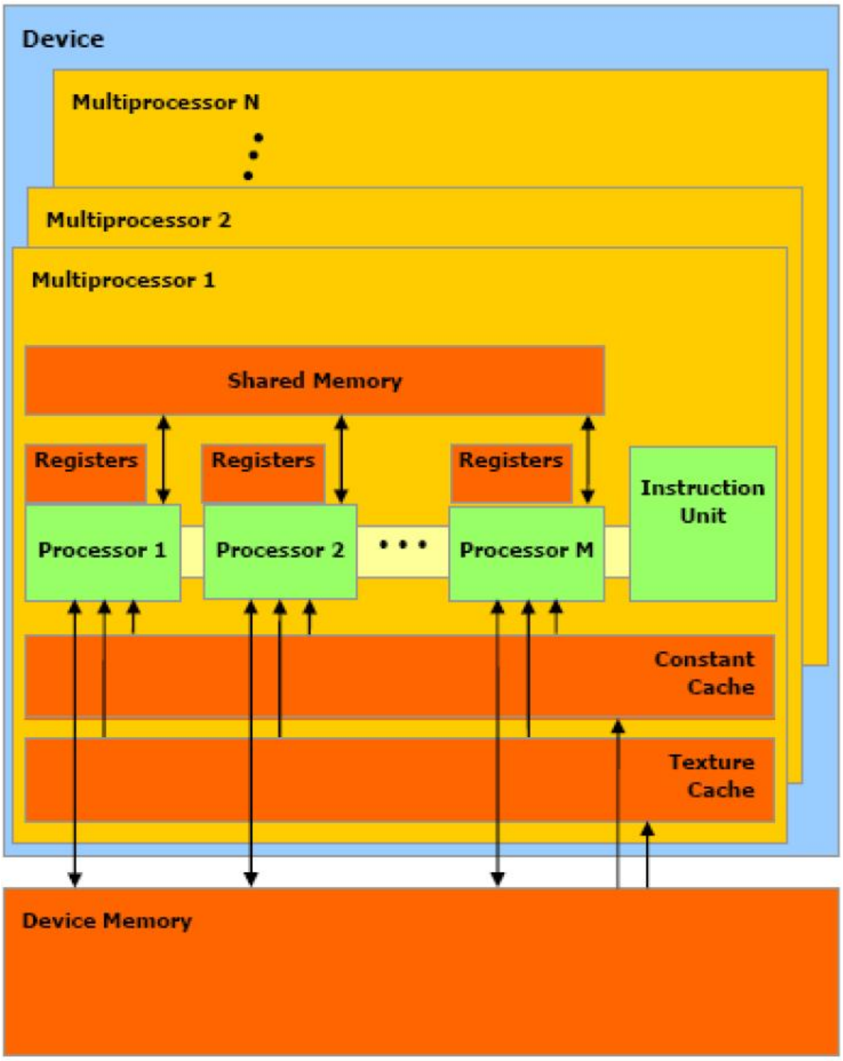


Figure 4.7: Graphics hardware parallel architecture

graphics hardware. Unlike the original Voxel Coloring algorithm [Seitz 1997] we use an Octree data structure [Szelinski 1993]. First, each volume is subdivided into 8 subvolumes. This gives rise to 27 points (we remove redundant 3D points). The projection is made using the well known pinhole camera model [Zissermann 2003] that describes how a 3D point M with coordinates (X, Y, Z) in the world coordinate space projects into an image point m with coordinates (u,v) in pixels using the classic perspective transformation:

$$m \cong K\left[R^T| - R^T t\right] M = PM \tag{4.1}$$

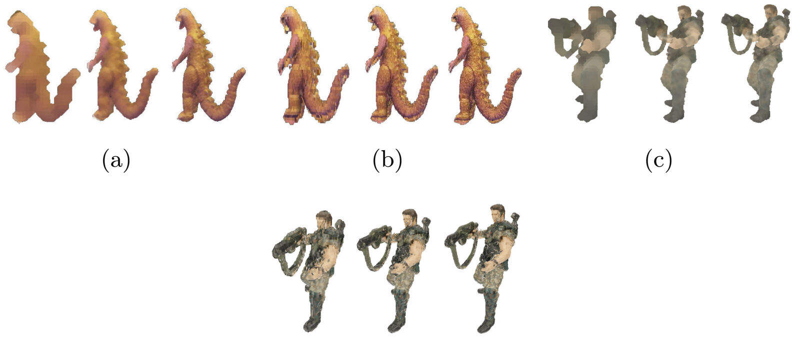


Figure 4.8: Reconstruction results for three different level of details ($64 = 2^6$, $128 = 2^7$, $256 = 2^8$) (a,c) Calculation using the CPU (b,d) Calculation using the GPU

Where R and t respectively represent the camera orientation and position, K the camera matrix or matrix of intrinsic parameters and P the projection matrix. In order to optimize the use of parallel architecture of the graphics hardware we assign a different process for image point coordinate computation using the simple formula:

$$m[p + 27n] = \sum_{k=0}^{3} P[4n + 12i + k] * M[4p + k] \tag{4.2}$$

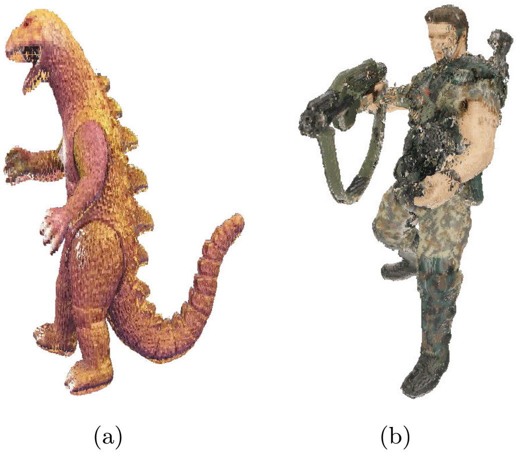
Where m, P and M respectively represent the concatenated matrices of 2D image points, projection matrices and 3D points. Then the result is normalized and we obtain 27 points in homogenous coordinates (u,v,1) for each image using the formula :

$$m[p + 27n] = \frac{m[p + 27n]}{m[p + 54]} \tag{4.3}$$

Finally, a subvolume can be recursively subdivided if the projection into images falls in the boundary of the silhouette. Thus we get better level of details without the need of subdividing all the voxels.

4.4 Results

Figure 8 illustrates the final results obtained with our Voxel Coloring framework for different level of details. The level of detail corresponds to the depth of the Octree data structure. It is the maximum number of recursive subdivisions of a voxel. Figure 9 shows the reconstruction results of the Dinosaur and the Soldier for the maximum level of detail. Evaluation of the results and the computing times of our framework are presented in Table 1.



(a) (b)

Figure 4.9: Results using the highest level of detail ($256 = 2^8$). (a) the Dinosaur reconstructed (b) the Soldier reconstructed

		Dinosaur			Soldier		
		64	128	256	64	128	256
Visual Hull (seconds)		0.442			3.703		
Voxel Coloring (seconds)	CPU	92.4	518	5449	56.47	200	1614
	GPU	1.50	3.25	16.44	0.79	1.98	8.33
Marching Cubes (seconds)		2.37	12.49	97.16	3.13	13.56	90.34
Mesh (number of triangles)		18736	73672	308956	24020	93652	392468

Table 4.1: Evaluation results of our Voxel Coloring framework using the Nvidia card 8800GTS 512MB on a Dualcore Intel Pentium 4 3.2Ghz with 2GB RAM.

4.5 Discussion

Comparison to previous work: Most previous works on reconstruction techniques from image silhouettes using Voxel Coloring algorithm either use geometric or colorimetric approach. In our work we combine those two criteria to get an accurate reconstruction. We also use an Octree data structure to improve the computing times and propose a way to accelerate the algorithm using parallel processing capabilities of modern GPUs.

Robustness and Limitations: The robustness of our approach strongly depends on the quality of silhouettes. Actually only the Voxel Coloring algorithm is accelerated using the GPU. Thus our framework is a mixed CPU/GPU implementation.

4.6 Conclusion

This chapter introduced methods and techniques for real-time recovering of accurate textured 3D models from image silhouettes. The main contributions of this work is the way we compute the bounding box using a Visual Hull approach, the combined colorimetric and geometric criteria used inside the Voxel Coloring algorithm and the way the computations are accelerated using the parallel capabilities of modern GPUs.

Part II

Single View Procedural Modeling

Chapter 5

Procedural Modeling

5.1 Introduction

Procedural modeling focuses on automatically creating 3D models and textures from a set of rules by using algorithms, rather than manually editing by using generic 3D modellers. Procedural modeling is generally applied to create complex or large scale content. The procedural content does not need to be stored, since the algorithm used can produce the same result using the same random seed. Moreover, procedural techniques are generally context-sensitive and thus allow to automatically fit user input. The set of rules may either be embedded into the algorithm, configured by parameters, or the set of rules is separate from the production engine. L-Systems, fractals, generative modeling, and shape grammars are procedural modeling techniques since they apply algorithms to produce content. In this chapter we will present a general overview of some procedural techniques and specifically shape grammars for architecture modeling.

5.2 Fractals

A fractal is "a rough or fragmented geometric shape that can be split into parts, each of which is (at least approximately) a reduced-size copy of the whole", [1] a property called self-similarity.

The term fractals was first introduced by Benoît Mandelbrot in 1975 and was derived from the Latin word "fractus" meaning "broken" or "fractured". They are generally used to produce complex real-world scenes such as clouds, mountain surfaces, and snow flakes. They can be easily generated using computer algorithms. They can be defined using a reccurrence relation, a fixed geometric replacement rule, or differential equations. They can be classified according to their self-similarity. The strongest type of self-similarity is the exact self-similarity where the fractal appears the same at different scales. In the quasi self-similarity case, fractals appears approximately identical at different scales. The statistical self-similarity is the weakest case where the fractals preserve a statistical measure at different scales.

[1] Barnsley, Michael F., and Hawley Rising. Fractals Everywhere. Boston: Academic Press Professional, 1993. ISBN 0-12-079061-0

The Barnsley Fern is a fractal introduced by the mathematician Michael Barnsley who first described it in his book Fractals Everywhere [1]. This fractal was made to resemble the Black Spleenwort, Asplenium adiantum-nigrum plant (figure 5.1).

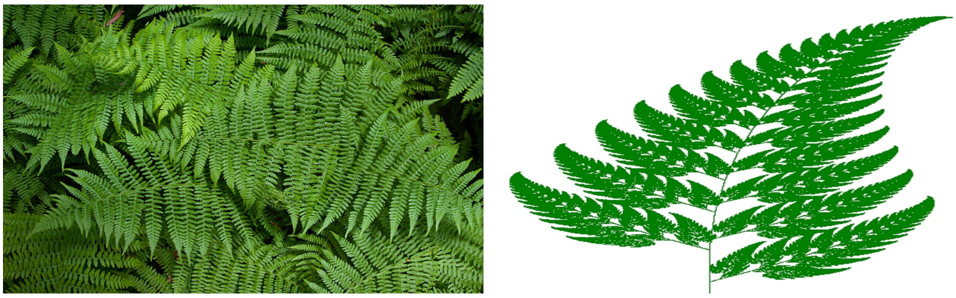


Figure 5.1: The Barnsley Fern. Left : a real Asplenium fern. Right : an Aspelenium fern fractal.

5.3 Generative Modeling Language (GML)

Generative Modeling Language (GML) is a simple programming language where complex scenes can be modeled using a list of operations rather than a hierarchy of objects. The same list of operations can be applied to different user input data to produce different results. This makes this approach context-sensitive and very powerfull for the modeling of complex scenes. GML provides a list of operators for the creation and transformation of 3D models. Similarly to other procedural approaches the model complexity is no longer directly (ie: linearly) related with the file size. The Procedural Cathedral, a basic model of the Cologne Cathedral, contains about 7 million triangles generated from only 126 KB of GML code (18 KB zipped)[1].

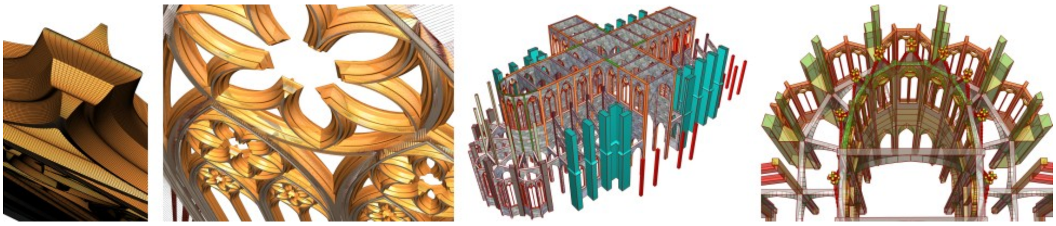


Figure 5.2: The Procedural Cathedral, a basic model of the Cologne Cathedral, contains about 7 million triangles generated from only 126 KB of GML code (18 KB zipped).

[1] http://en.wikipedia.org/wiki/Generative_Modelling_Language

5.4 L-systems

An L-system or Lindenmayer system is a parallel rewriting system and a variant of a formal grammar generally used to model the growth processes of plant development. L-systems were introduced and developed in 1968 by Aristid Lindenmayer.

L-system grammars are very similar to formal grammar (Chomsky grammars). L-systems are defined as a tuple $G = (V, S, \omega, P)$, where:

- V (the alphabet) is a set of symbols containing elements that can be replaced (variables).
- S is a set of symbols containing elements that remain fixed (constants).
- ω (axiom) is a string of symbols from V defining the initial state of the system.
- P is a set of production rules defining the way variables can be replaced with combinations of constants and other variables. A production rule consists of two strings, the predecessor and the successor.

The rules of the L-system grammar are applied iteratively starting from the initial state. As many rules as possible are applied simultaneously, per iteration; this is the distinguishing feature between an L-system and the formal language generated by a formal grammar.

An L-system is context-free if each production rule refers only to an individual symbol and not to its neighbours.

If a rule depends not only on a single symbol but also on its neighbours, it is called a context-sensitive L-system.

If there is exactly one production for each symbol, then the L-system is said to be deterministic. If there are several, and each is chosen with a certain probability during each iteration, then it is a stochastic L-system.

Figure 5.3 shows different trees generated using a real-time L-systems implementation [1].

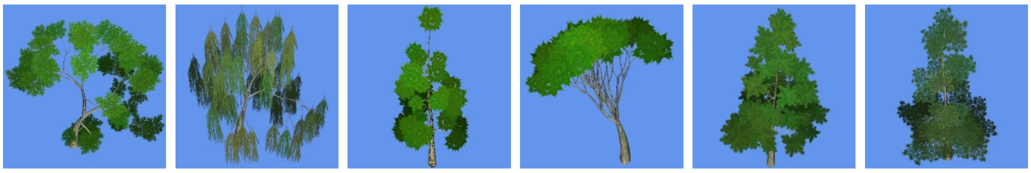


Figure 5.3: Realistic trees generated using the L-Trees application. From left to right : Rug, Willow, Birch, Garden tree, Gray wood and Pine.

[1] http://ltrees.codeplex.com/

5.5 Shape grammars

In our work we build on the CGA shape grammar [Mueller 2007] which we will briefly review in the following. Shape grammars were first introduced by Stiny and Gips [Stiny 1972] and Wonka et al. [Wonka 2003] extended the concept to computer graphics . CGA shape grammar also builds on concepts introduced for the modeling of plants using L-systems [Prusinkiewicz 1991].

5.5.1 Production system

Procedural modeling of architecture using shape grammars is based on a production system. The process takes as input a basic shape called the axiom and a set of production rules and sequentially generates what is called the shape tree. A shape is mainly defined by a symbol, a geometry (mesh) and a scope. A scope is an oriented bounding box composed of an origin P, three orthogonal axis X, Y, and Z and three numeric attributes S_x, S_y, and S_z defining its size in meters (see figure 5.4). At each iteration of the process the system chooses an active shape A in the shape tree and a production rule with A as shape predecessor. Then it creates a set of new shapes according to the commands on the right hand side of the rule and adds them as children of the shape A in the shape tree. The shape A is then marked as inactive and another iteration starts till the shape tree contains no more active and non-terminal shapes or there is no more applicable production rules.

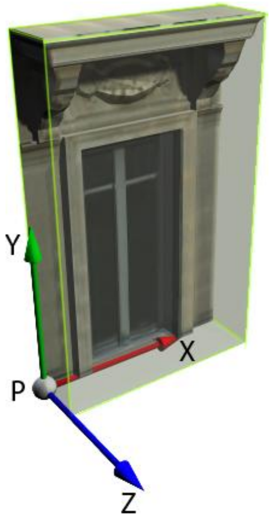


Figure 5.4: A rendering of a 3d asset of a Haussmannian tile : The scope is an oriented bounding box composed of an origin P, three orthogonal axis X, Y, and Z and three numeric attributes S_x, S_y, and S_z defining its size in meters.

A typical Haussmannian building model generated using our procedural modeling tool (ie: BuildingMaker) from a set of 60 production rules and a basic Haussmanian shape library (3d assets and textures of some architectural elements such windows, balconies, and chimney) is shown in figure 5.5.

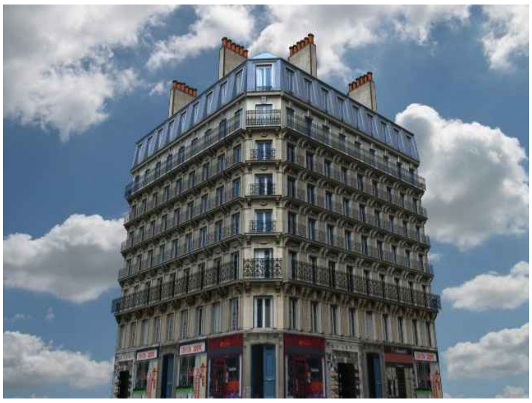


Figure 5.5: A typical Haussmannian building model generated using our procedural modeling tool with a set of 60 rules and rendered using mental ray.

Notation: The production rules are defined as follows:

$$predecessor : condition \rightarrow successor : probability$$

where *predecessor* is a symbol representing a shape that will be replaced by *successor* (a set of CGA commands). The *condition* is a logical expression that has to evaluate to true in order to select the rule and *probability* is used to add stochastic variability during the production process.

5.5.2 CGA commands

Scope commands: we use scope commands to modify shapes: T(tx, ty, tz) translates a shape and its scope position P, R(rx,ry,rz) rotates a shape and its local coordinate system, and S(sx, sy, sz) scales a shape and sets the size of the scope. We also use [and] to push and pop the current shape on a stack.

Extrude: This command allows to extrude a polygon into a volume. For example, in order to extrude a building lot into a volume wich height is 30 meters we use the following rule:

$$Lot \rightarrow Extrude(30)\{Building\}$$

Component split: This command allows to explode a shape into a lower dimension shapes. For example the following command explodes a building volume into facades (here only the side faces are selected):

$$Building \rightarrow Comp("sidefaces")\{Facade\}$$

Subdiv: This command allows to split a shape along a scope axis into smaller shapes of different sizes using a plane clipping algorithm. For example, we use

Subdiv to split a facade into floors:

$$\begin{aligned}&Facade \rightarrow Subdiv(\text{"}Y\text{"})\{\\ &\quad 4 : GroundFloor|\\ &\quad 3 : Floor1|\\ &\quad 3 : Floor2|\\ &\quad ...|\\ &\quad 3 : Floor6\\ &\quad \}\end{aligned}$$

Repeat: This command is similar to Subdiv except the fact that it splits a shape into a collection of shapes of the same size. The size is adjusted depending on the scope size. For example, we use Repeat to split a floor into tiles of the same approximate size (2.5 meters):

$$Floor1 \rightarrow Repeat(\text{"}X\text{"}, 2.5)\{Tile1\}$$

Insert: This command allows to insert a 3d asset and fit it to the scope of the current shape. For example, the following command inserts a 3d model of a window in Wavefront OBJ format into a tile.

$$Tile1 \rightarrow I(\text{"}window1.obj\text{"})$$

SetupUV: This command allows to setup the UV coordinates for the texture of a shape:

$$SetupUV(channel, u, v, z - factor)$$

where the *channel* parameter is used for multitexturing purposes, (u, v) parameters in meters are used to define the UV coordinates for the texture. The $z - factor$ parameter is related to 3D textures but it still not used in our actual implementation. The following rule shows an example of the use of UV coordinates for texture mapping (see figure 5.6).

$$\begin{aligned}&1 : tile \rightarrow SetupUV(0, scope.sx, scope.sy, 1)Texture(\text{"}uvtest.tif\text{"})\\ &2 : tile \rightarrow SetupUV(0, 4, 5, 1)Texture(\text{"}uvtest.tif\text{"})\end{aligned}$$

BakeUV: The BakeUV command when applied to a shape allows to 'bake' (ie: all the derived shapes in the shape tree will be assigned the same texture than the

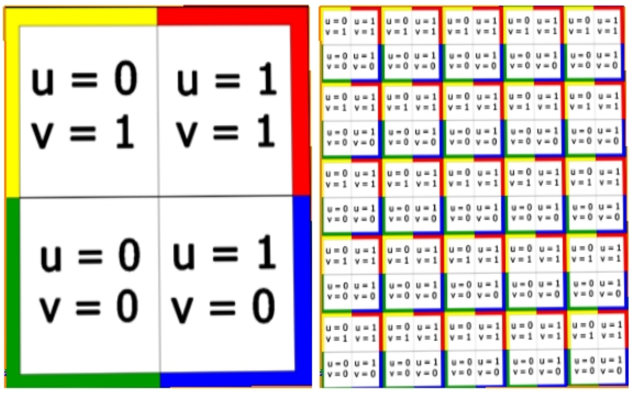


Figure 5.6: SetupUV operator. Left : the (u, v) parameters are set to the scope size of the shape (here $scope.sx = 20meters$ and $scope.sy = 25meters$) thus the texture fits the shape. Right : the (u, v) parameters are set to 4 and 5 meters thus the texture is repeated 5 times along the X and Y axis.

original shape) a texture. The UV coordinates will be automatically computed for each created shape depending on its scope and the scope of the original shape.

OffsetUV: The OffsetUV command when applied allows to make an offset on the UV coordinates in meters along the X and Y scope axis.

ScaleUV: The ScaleUV command when applied allows to make a scale on the UV coordinates along the X and Y scope axis.

Color: The Color command allows to define an (R, G, B) color with a transparency parameter A to a shape in the range $[0, 1]$. The notation of this command is as follows:

$$Color(R, G, B, A)$$

Texture: This command allows to define the texture file of a shape. For example, the following production rule combines the SetupUV, Texture and BakeUV commands to 'bake' a facade texture and thus allows the automatic recursive texturing of all the derived shapes (floors, tiles, windows, ...) using the original facade texture: (1) The SetupUV command is used here so that the texture fits the facade shape since the UV parameters are set to the scope size along the X and Y axis (ie: $scope.sx$ and $scope.sy$) (2) The Texture command is used to define the texture file (ie: 'facade19.jpg') (3) The BakeUV command is used to 'bake' the texture file to all created child shapes (ie: for example, here the $GroundFloor$, $Floor1$, ..., and $Floor6$ created shapes using the Subdiv command applied to the $Facade$ shape will be assigned the same texture file 'facade19.jpg' but different UV coordinates depending on their relative scope with the *Facade* shape).

```
$Facade \rightarrow$
    $SetupUV(0, scope.sx, scope.sy)$
    $Texture("facade19.jpg")$
    $BakeUV(0)$
    $Subdiv("Y")\{$
    $4 : GroundFloor|$
    $...|$
    $3 : Floor6$
    $\}$
```

Center: This command is usefull when we want to center a shape towards its predecessor shape. For example the following production rules show the result of the center operator on the generation of balconies (see figure 5.7).

$$1 : tile \rightarrow windowS('1.2,' 0.3,' 1)balcony$$
$$2 : tile \rightarrow windowS('1.2,' 0.3,' 1)Center("X")balcony$$

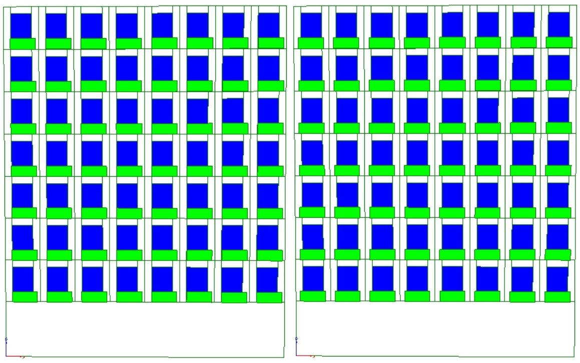


Figure 5.7: Center operator. Left : a balcony shape generated using the first rule (without the center operator). Right : a balcony shape generated using the second rule (with the center operator).

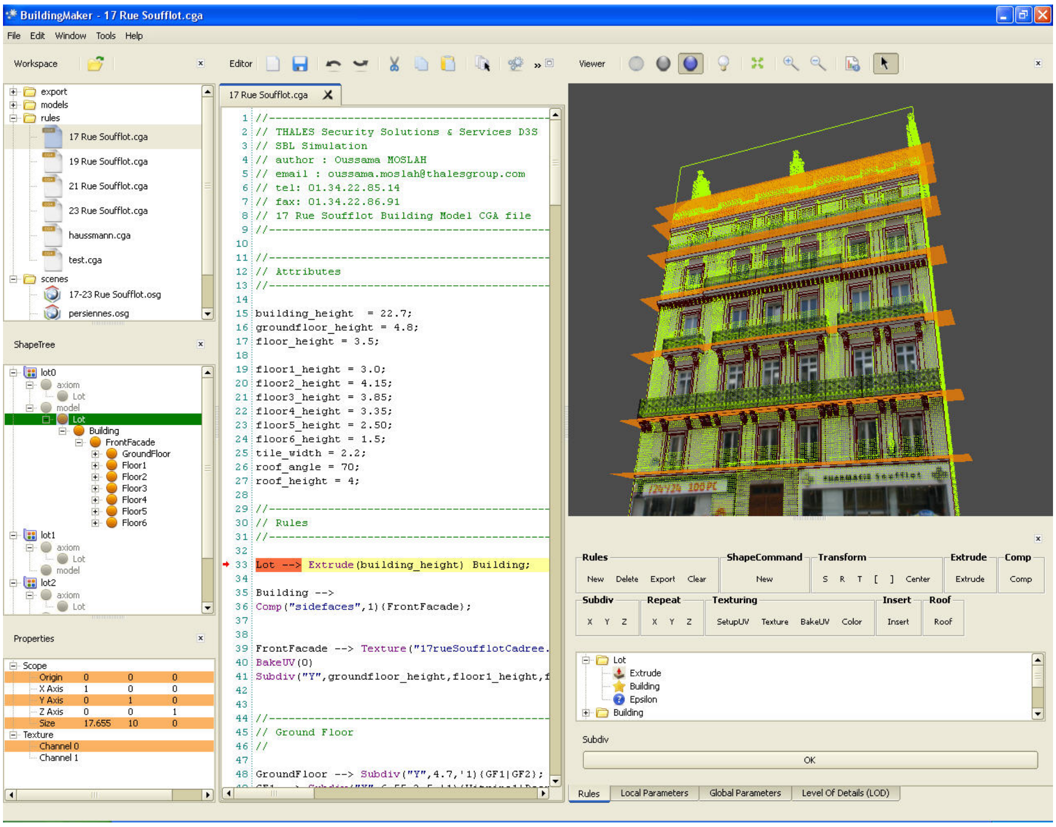
Roof: The Roof command creates a roof structure (pyramid, gable, hipped, mansard) on a base polygon. For example, the following production rule allows to crate a mansard roof with a 75° angle and a height equal to 3 *meters*.

$$top \rightarrow Roof("mansard", 75, 3)\{roof\}$$

NIL: The NIL command allows to delete a shape from the shape tree.

5.6 Interactive editing

Most procedural modeling tools using grammars are text-based editing systems. The user has to manually write some rules and then the production system automatically produces the 3d model. In order to refine or modify the obtained 3d model the user has to modify the text several times. We found that this process can become very time consuming for the user mainly when modeling real facades using photographs. The parameters tuning of the corresponding shape grammars rules takes the most of the time since the user has no direct control over the created 3d model.



```
//----------------------------------------------------------
// THALES Security Solutions & Services D3S
// SBL Simulation
// author : Oussama MOSLAH
// email : oussama.moslah@thalesgroup.com
// tel: 01.34.22.85.14
// fax: 01.34.22.86.91
// 17 Rue Soufflot Building Model CGA file
//----------------------------------------------------------

//----------------------------------------------------------
// Attributes
//----------------------------------------------------------

building_height  = 22.7;
groundfloor_height = 4.8;
floor_height = 3.5;

floor1_height = 3.0;
floor2_height = 4.15;
floor3_height = 3.85;
floor4_height = 3.35;
floor5_height = 2.50;
floor6_height = 1.5;
tile_width = 2.2;
roof_angle = 70;
roof_height = 4;

//----------------------------------------------------------
// Rules
//----------------------------------------------------------

Lot --> Extrude(building_height) Building;

Building -->
Comp("sidefaces",1){FrontFacade};


FrontFacade --> Texture("17rueSoufflotCadree.
BakeUV(0)
Subdiv("Y",groundfloor_height,floor1_height,f


//----------------------------------------------------------
// Ground Floor
//

GroundFloor --> Subdiv("Y",4.7,'1){GF1|GF2};
```

Figure 5.8: BuildingMaker : an interactive procedural modeling tool of architecture using CGA shape grammars.

Thus, we developped a similar interactive editing framework following the approach described in [Lipp 2008] and integrate it into our procedural modeling tool illustrated in figure 5.8. Instead of using a text-based system to write rules the user has a direct control over the derivation process using a visual user interface. The user can create or modify rules using user interface components and also directly tune the rule parameters in the 3d viewer. For example, the visual implementation of the Subdiv and Repeat CGA commands (split operators) that allows to subdivide a shape into smaller shapes through an axis direction (X, Y, or Z) allows the user to directly tune the parameters by splitting the clipping planes inside the 3d viewer and adjust them to fit the input photograph 5.8. Following a similar reasoning we implemented a visual and interactive editing capabilities and integrated them into our procedural modeling tool to speed up and offer a full control over the procedural modeling process.

We also created additionnal user interface components to improve the usability of our procedural modeling tool and to offer advanced capabilities to designers. For example, the user can select a component such a shape in the 3d viewer or its corresponding node in the shape tree user interface component and then its corresponding rule will be automatically selected inside the text editor and vice versa. Moreover, we added import/export capabilities using standard formats for the interface with other modeling tools. The user can also import calibrated photographs and use them to texture the 3d model using projective texture mapping. We also added the import of the LIDAR data (3d point cloud) into the 3d viewer to allow the user to tune the depth parameters of facades.

The modeling of a real Haussmannian facade in Paris such those presented in figure 5.9 could need between 30 to 60 min of user time using a text-based procedural scheme and only 5 to 10 min thanks to the interactive modeling capabalities. The perspective rectification can be done as a pre-processing stage through homographic rectification or using projective texture mapping (see figures 5.10 and 5.11). Realistic and semantic 3d models obtainted using our procedural modeling tool for the urban dataset sequence in Paris 5.9 are shown in figure 5.12.

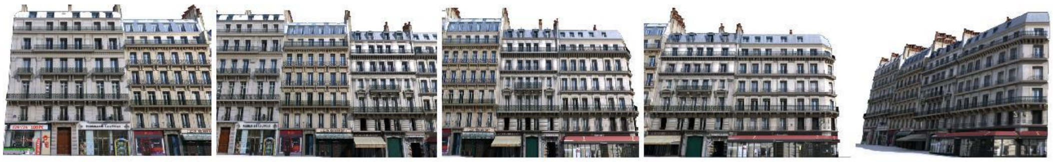


Figure 5.9: Urban dataset: the input dataset is composed of a set of calibrated photographs (pixel image resolution 1920x1080)and a coarse 3D model of the observed Parisian buildings.

5.7 Conclusion

This chapter presented some procedural modeling techniques and more specifically shape grammars for the modeling of architecture. We found that CGA shape grammars are suitable for the modeling of Parisian facades. Using simple and concise grammars, realistic and semantic facade models were modeled using rectified facades images and some 3d assets and textures representing some architectural elements (modilions, ornaments, balconies, ...). The interactive editing capabilities improves the usability of the tool but we still need to manually edit one shape grammar per

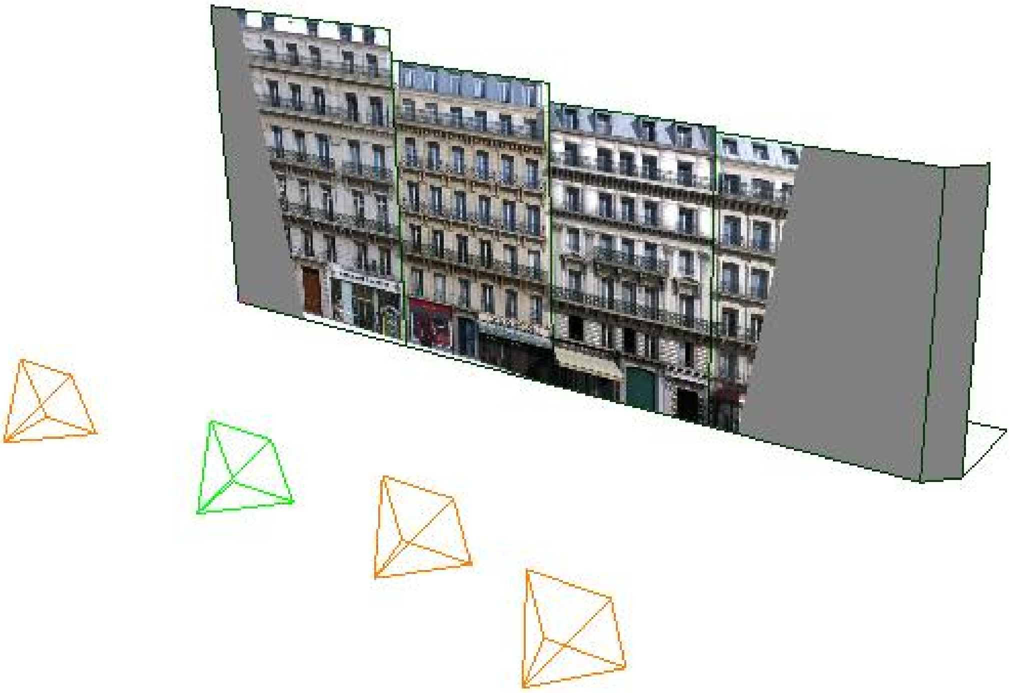


Figure 5.10: The camera selected by the user is projected onto the 3D model.

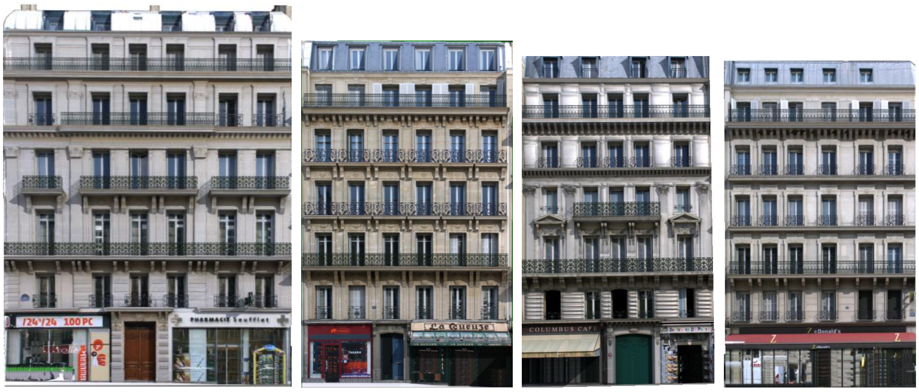


Figure 5.11: Perspective rectification is obtained using an orthographic rendering of a 3D facade with projective texture mapping. From left to right the real-world dimensions are: 17.6x24.4m, 13.5x22.2m, 12.6x20.9m and 15.1x20.6m and and the pixel image resolutions are: 333x465, 487x799, 392x642 and 372x519.

facade. Thus we realize that we need to develop automatic facade elements detectors in order to obtain a large scale reconstruction of Paris city. The idea is basically to write a stochastic shape grammar that can describe the variety inside one particular architectural style and to use it as a model combined with bottom-up detectors for the modeling of several facades of the style. This approach is described in the next chapter.

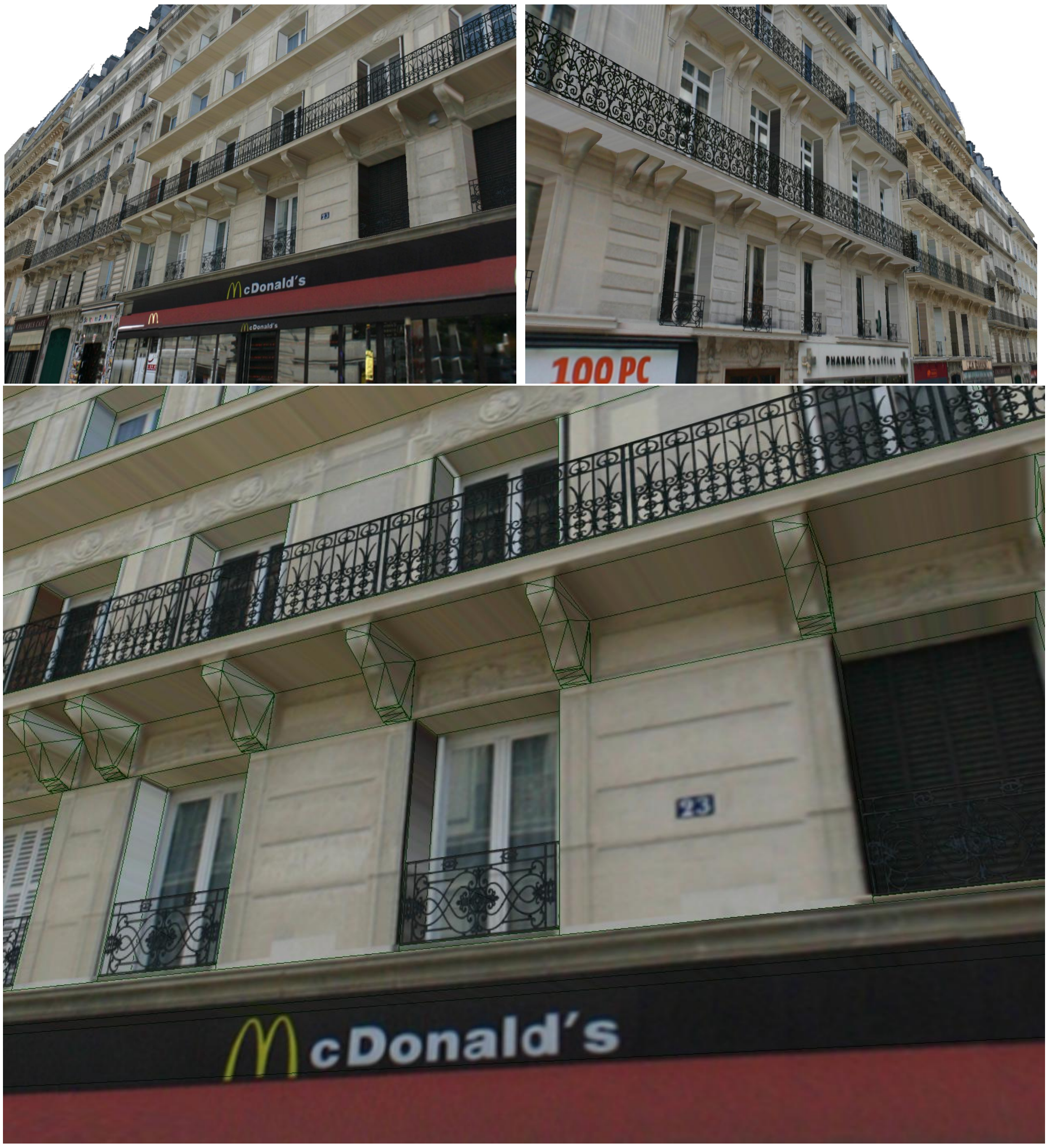


Figure 5.12: Semantic and realistic 3D models of Facades generated using our procedural modeling tool.

Chapter 6

Grammar-driven Reconstruction

This chapter presents an approach to reconstruct 3d facade models from a single image. The novelty of our approach is the use of stochastic shape grammars as architectural prior instead of deterministic shape grammars. We combine bottom-up detection using image-based classifiers with top-down proposals generated by the stochastic shape grammar using Metropolis-Hastings as optimization algorithm. We demonstrate the robustness of our approach on a variety of complex facade images with challenging architectural elements and occlusions due to balconies.

6.1 Introduction

This paper addresses the problem of 3d facade reconstruction from a single image. The output of our method is a 3d model including textures and a semantic segmentation. We believe that there is widespread interest in urban reconstruction and our models could be used in applications like internet based mapping, urban planning, architecture, computer games, movies, tourism, and training simulations.

The problem of 3d facade reconstruction from a single image is very difficult. There are various forms of image noise, occlusions due to objects such as vegetation and balconies, and no depth information is available. Therefore, previous work always use a combination of image analysis, architectural knowledge, and user interaction. Approaches vary from image-based reconstruction approaches [Mueller 2007, Xiao 2008] where architectural knowledge is encoded in parameter settings and implicit restrictions on the alignment of facade elements, to parametric architectural models [Dick 2003], and more recently deterministic (shape) grammars [Alegre 2004, Ripperda 2008, Hohmann 2009, Koutsourakis 2009]. Deterministic shape grammars lead to very strong results, but a lot of time is spend in modeling because one shape grammar has to be modeled for each facade image.

The main motivation of our work is to extend the idea of grammar-based reconstruction to stochastic shape grammars. A stochastic shape grammar can encode the variation within an architectural style (a larger class of facades) and it becomes possible to use one shape grammar to reconstruct many facades. Our proposed solution is a framework that uses CGA (Computer Generated Architecture) shape grammars [Mueller 2006] (including the use of parameters, conditions, and probabilities) to encode knowledge about an architectural style. We combine a set of bottom-up detectors for locating architectural elements on a facade with a top-down optimization stage that uses the Metropolis-Hastings algorithm to guide the sampling of facade structures from the grammar. While the use of stochastic shape grammars is the main novelty of this paper, we also propose new algorithms for window detection, balcony detection, and inpainting. These algorithms are important, because we found that previous work was not sufficiently robust for difficult architectural styles, such as Louis XIII-XVI (1595 – 1790), Empire (1800 – 1815), Restauration (1815 – 1830) and Haussmanian (1850 – 1870) style (figure 5.5) in Paris. These styles contain balconies resulting in self occlusions and several complex ornaments such as consoles and modillions. The main contributions of this paper are:

- We are the first to use a full parametric, conditional, and stochastic shape grammar as prior for facade reconstruction instead of parametric models or deterministic grammars.
- We combine bottom-up detection with top-down proposals to optimize the facade structure using the Metropolis-Hastings algorithm making full use of the prior knowledge encoded inside the stochastic grammar.
- We introduce a new algorithm for window detection and a new algorithm for balcony detection and inpainting. These algorithms are more efficient when processing facade images with complex ornaments and occlusions due to balconies than previous work.

6.2 Related work

Different authors adressed the problem of semantic 3d reconstruction of buildings from images. The proposed approaches fall into two categories: image-based [Lee 2004, Mueller 2007, Xiao 2008] or model-based [Dick 2003, Alegre 2004, Ripperda 2008, Hohmann 2009, Koutsourakis 2009].

The image-based techniques focus on using image processing and recognition methods to produce 3d models from images. The architectural style prior is generally not explicitly introduced within the reconstrucion process. Therefore, it is likely that these methods miss some architectural elements in the presence of occlusions or difficult noise.

The model-based techniques try to explicitly incorporate prior knowledge to optimize the reconstruction. The prior can contribute to add constraints on the reconstruction and thus makes the results consistent. This helps to fill in missing model parts making the methods robust to partial occlusions and to drive the semantic building understanding which can considerably reduce the search space inside images and improve the accuracy of pure image-based fitting techniques. The way the prior knowledge is encoded within model-based approaches ranges from parametric models [Dick 2003] to deterministic shape grammars [Alegre 2004, Ripperda 2008, Hohmann 2009, Koutsourakis 2009]. However in the case of existing grammar-based models the authors optimize the parameters of a given derivation tree (a set of deterministic rules) and not the tree itself. Therefore, these methods are actually very similar to parametric models. The use of a deterministic grammar as a model cannot capture the variation within an architectural style. In our work we propose to extend the previous shape grammar based approaches by allowing stochastic shape grammars and including more bottom-up image classifiers.

6.3 System overview

Our system takes as input a rectified facade image and produces a semantic and photo-realistic 3d model (figure 6.1). The a priori knowledge on the architectural style is encoded inside a stochastic grammar. The proposed pipeline consists of two stages. Here we give a short description of the individual stages, with details provided in the later sections:

Bottom-up detection: The input is a rectified facade image. We detect windows using a combined contour-based and texture-based profile projection method (see section 4.1). We also automatically detect balconies using both a color-based pixel clustering and Gabor filtering (see section 4.2). The balconies are then removed using a specific inpainting technique based on pixel neighboorhood. The cornices are detected using elongated segments in a fine granularity segmentation (see section 4.3). Finally, we use classifiers as generic detectors to locate additional architectural elements such as modillions, and doors (see section 4.4).

Top-down optimization: The input to this stage is a collection of bottom-up detected elements and a stochastic grammar describing the corresponding architectural style (see section 5). The problem is formulated using a Bayesian approach (see section 6.1) where the facade prior is computed from the stochastic grammar (see section 6.2) and the likelihood is defined as a top-down/bottom-up evidence score (see section 6.3). The optimization is done using the Metropolis-Hastings algorithm where the random facade structures are sampled from the stochastic grammar (see section 6.4).

6.4 Bottom-up detection

In this section we will introduce classifiers for windows, balconies, cornices, and general elements. The goal of this section is to detect and label as many facade elements as possible. Our framework does not require perfect element detection. The optimization algorithm will combine the output of the classifiers with prior architectural knowledge to improve the results. The first technical challenge of this work was to analyze several existing algorithms and see how well they work in combination. Our main results were that 1) existing window detection algorithms cannot handle the difficult inputs we consider (see Fig. 3). 2) occlusion due to balconies has not been sufficiently explored in previous work, but explicit balcony detection and removal

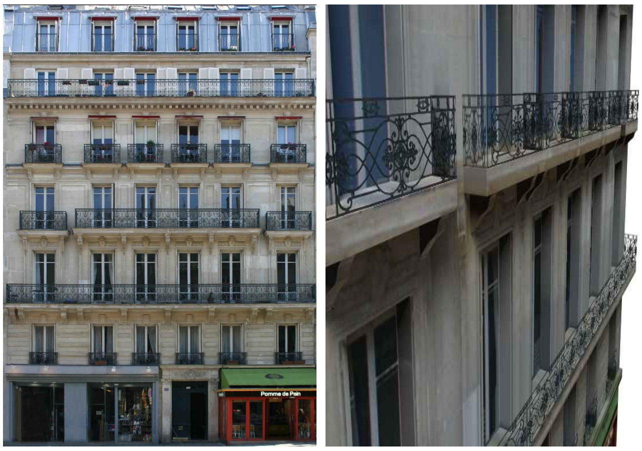


Figure 6.1: System overview. Left : original rectified facade image. Right : reconstructed 3d model. Our system allows the creation of semantic and photo-realistic 3d models of complex facades including windows, balconies, cornices, and modillions.

is very important. 3) detection of other elements such as cornices, modillions, and doors is very difficult, but a well engineered heuristic is an important complement to window and balcony detection. We developped specific detectors for windows, balconies, and cornices and we use generic Viola-Jones [Viola 2002] trained classifiers to detect other elements such modillions and doors. Windows are detected using a combined contour-based and texture-based profile projection method (see section 4.1). Balconies are detected using both a color-based pixel clustering and Gabor filtering (see section 4.2). The balconies are then removed using an inpainting technique based on pixel neighboorhoods. The cornices are detected using elongated segments in a fine granularity segmentation (see section 4.3). We also use Viola-Jones [Viola 2002] classifiers as generic detectors to locate additional architectural elements such as modillions and doors (see section 4.4).

6.4.1 Window detection

Window detection is probably the most important bottom-up detector for a facade reconstruction pipeline. The detected window grid has a strong impact on the perception of the facade and it gives additional information for balcony detection and for obtaining the ground floor limit. Due to its importance it is vital to have a robust and precise algorithm. This is difficult because the intra-class variety is significant: windows can be closed or opened, can be protected by shutters, can reflect the sky or another building and usually have various curtains behind. Some attempts have been made to create a window detector. Dick et al [Dick 2003] use a correlation grid (with prior examples), Werner and Zisserman [Werner 2002] use learned correlations over contours to directly detect and classify windows. Mueller et al [Mueller 2007] use the mutual information to detect similarities between regions and split the facades into tiles and then further split tiles into walls and windows. Lee and Nevatia [Lee 2004] use a contour profile projection method based on the vertical/horizontal alignment of windows. The last method performs well with facades composed of a simple window grid. However, when dealing with complex facades this method fails (Figure 6.2) due to the noise introduced by architectural elements such as balconies, cornices, modillions and ornaments. Thus we introduce a texture profile projection method similar to Lee and Nevatia [Lee 2004] technique which uses texture descriptors rather than contours. The combination of both methods provides a robust window detection technique in presence of occlusions and noise.

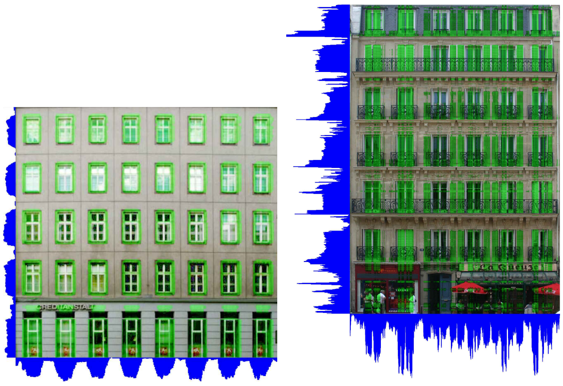


Figure 6.2: The horizontal/vertical profile projection method evaluation. Left: a facade with a simple window grid. Right: a Haussmannian facade with perturbations due to balconies, cornices, ornaments and modillions.

To avoid the problem of false contour lines and noise edges we introduce a new descriptor for detecting major borderlines between architectural elements based on texture (figure 6.3). The materials that appear on a facade (stone, iron, wood, glass, ...) have strong stochastic texture components (randomness). According to previous work [Liu 1996] the best model for this textural components is an AR (Auto-Regressive) model. AR [Mao 1992] is a model where a pixel c of intensity $I(c)$ is a linear combination of its neighbors $r \in N$. The descriptors $D(r)$ are the parameters of this combination:

$$I\left(c\right) = \mu + \sum_{r \in N} D(r) I(r + c) + \varepsilon(c) \tag{6.1}$$

Where ε is the error of this model at c and μ the intensities mean value of its neighbors $r \in N$. The parameter estimation is done over a window so that the model is overdetermined. For our purpose we want to detect the regions where the texture changes. Intuitively and empirically the region where two textures meet has a higher model error ε than the model error of the two textures. We create probability blobs that correspond to window locations by projecting vertically and horizontally the texture model error. Let ε be the texture model error as defined

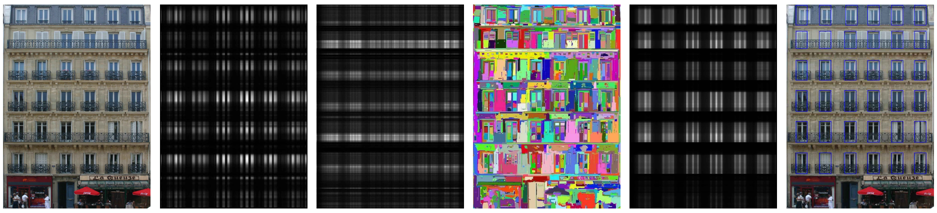


Figure 6.3: Window detection. From left to right: original image, texture blobs, contour blobs, fine granularity segmentation, final averaged blobs, detected windows. Texture derived blobs are more reliable than contour based blobs.

in equation 6.1, and KxL the image resolution. The texture-based blobs are then obtained using the formula:

$$B(i,j) = \left(\frac{1}{K}\sum_{k=0}^{K}\varepsilon(i,k)\right)\left(\frac{1}{L}\sum_{l=0}^{L}\varepsilon(l,j)\right) \tag{6.2}$$

We compute a pixel-wise multiplication between the texture-based blobs (figure 6.3 (b)) and the contour-based blobs (figure 6.3 (c)) and average the results over a fine granularity segmentation (using the algorithm described in [Felzenszwalb 2004]) to create sharp edges between window blobs and the facade wall. The averaged combined blobs (noted $\tilde{B}$) are then reprojected to obtain the final precise window blobs B_f (figure 6.3 (e)) using a similar formula:

$$B_f(i,j) = \left(\frac{1}{K}\sum_{k=0}^{K}\tilde{B}(i,k)\right)\left(\frac{1}{L}\sum_{l=0}^{L}\tilde{B}(l,j)\right) \tag{6.3}$$

The blobs are then thresholded at different levels and a window energy function $E(t)$ is computed over the set of detected windows $W(t)$ at each threshold level t:

$$E(t) = \sum_{w_1,w_2\in W(t)} Al(w_1,w_2) + \sum_{w\in W(t)} Pr(w) \tag{6.4}$$

Here $Al()$ is a function that decreases exponentially with the misalignment of two windows in the same row or column and is zero otherwise. $Pr()$ is a function that encodes the architectural prior for windows in the current architectural style. It can be obtained by sampling windows from the grammar. It decreases exponentially when the size of the window is different from the average grammar sampled window size. The optimization is done through a gradient descent to minimize the variations in position and size between the set of windows according to the vertical/horizontal alignement assumption. The final set of detected windows is the one that maximizes the energy function $E(t)$.

Method	Recall	Overlap	Notes
Dick et al[Dick 2003]	$< 20\%$	$> 80\%$	correlation; works well if test data similar to prior data
Viola-Jones trained detection[Ali 2007]	53%	>50%	trained detector, no window-grid assumptions
Lee and Nevatia[Lee 2004]	60%	>80%	edge blobs, works well in the absence of noise, evaluated on a dataset of haussmanian facades, irregular window grid-assumptions
Texture and Edge based blobs(our method)	85%	$> 80\%$	Assumes an irregular-grid of windows, handles architectural noise (e.g. Haussmannian architecture)

Table 6.1: Comparative evaluation of window detection algorithms

6.4.1.1 Comparative evaluation of the algorithm

While testing Dick's window detection method[Dick 2003] we realized that using correlation techniques with prior examples worked well only if the prior examples are very similar to the test data (recall bellow 20% for windows overlapping at more than 80%). Searching for correlations at different scales also generated performance problems. We have also experimented with mutual information techniques [Mueller 2007] but the hypothesis of the mutual information approach was invalidated by the fact that windows in our test data sets are very different on the same facade (e.g. shutters, reflections, etc.) (recall was not computed). Experiments done by [Ali 2007] and our own efforts to train a Viola-Jones detector failed because of the intra-class variety (reported recall was 53% for windows that overlapped by more than 50% with ground truth, thus finding the window grid is not possible for such a low recall). Lee and Nevatia's method of blob-construction [Lee 2004] showed the most promising results but has two issues: the user must manually select a threshold for blob extraction and the method has a low robustness due to noise introduced by complex facade elements (see projected contour histograms for the right (noisy) facade in figure 6.2). We realized that false contour lines are the major problem on complex facades. We have then replaced the classic contour lines (e.g. Canny) with texture-based descriptors in order to detect changes in materials. We also introduced the window grid energy function to thresold the blobs without the need to manually select a threshold. In the following we compare our method to various other algorithms for window detection. The general algorithm behavior is the following: it either succeeds (recall $> 85\%$ for 80% overlap) or it fails completely (recall $< 20\%$ for 50% overlap). Such a failure case is easily detected by inspecting the maximum value of the window energy function in equation 6.4 for abnormally low values. Known failure cases are: window missing in the grid and false-positives introduced, wrong size of windows due to slight misalignment of one window in a column, and strong occlusions by trees. The comparative evaluation of window detection techniques are listed in table 6.1.

6.4.2 Balcony detection and removal

Balconies have a strong impact on the visual realism of a 3d facade model. The reason why we developed a specific balcony detector is because balcony detection does not work with general classifiers. We started out using a trained Viola-Jones [Viola 2002] detector and we found the following obstacles: trained Viola-Jones like detectors are inappropriate to detect balconies because of their variable aspect ratio, the strong texture variability, and the fact that the balconies are occluding elements with transparent parts on different facade materials such as walls and windows. That is why we propose a specialized detector for iron-made railing balconies. The color of balconies is a system parameter (eg: dark-colored, red-colored) used for a pixel-based clustering of regions that are more likely to be a balcony on the facade. Moreover, we developed an inpainting technique that generates a facade texture without balconies. This removes texture artifacts that happen if we directly texture map the facade 3d model using the original image and allows the creation of photo-realistic 3d models in the presence of occlusions. Synthetic balconies are then automatically added by the grammar using generic textures.

Based on the initial description of the balconies we search for regions of the image that are more likely to be a balcony (eg: dark-colored pixels) and have a high energy texture [Petkov 1997]. We cluster the pixels into two groups one that is close to the user given color and one that is far. The centroid estimation can be easily done using an EMGM (Expectation Maximization Gaussian Mixture) with 2 gaussians (one for the wall, the other for the balconies). We can then assign to each pixel a probability that it belongs to the balconies cluster (equation 6.5). Assuming the wall cluster is called W and the balconies cluster is B (parameters for the two Gaussian clusters have been estimated by the EMGM procedure) we can compute the posterior probability (equation 6.5) that a pixel of color intensity i_{rgb} belongs to the balconies cluster B (figure 6.4 (b)). Low probability values (e.g. less than 2%) should be thresholded to 0.

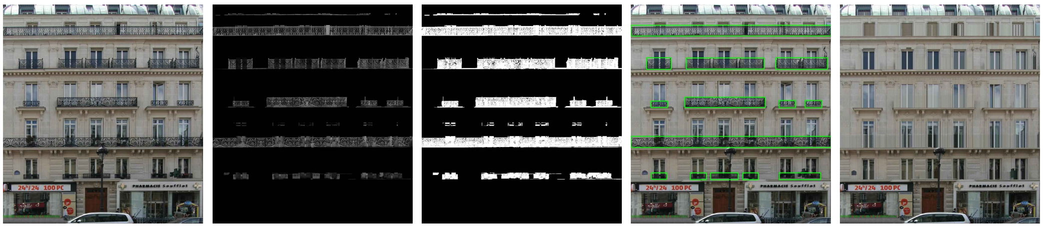


Figure 6.4: Balcony detection and removal. From left to right: original image, color-based pixel clustering, Gabor filtering, detected balconies, inpainted image.

$$P(B|i) = \frac{P(i|B)\,P(B)}{P(i|B)\,P(B) + P(i|W)\,P(W)} \tag{6.5}$$

For high energy texture areas we use a combination of 4 standard Gabor filters (figure 6.4 (c)) ($G_0, G_{45}, G_{90}, G_{135}$) with the following orientations [0°, 45°, 90°, 135°] [Grigorescu 2002]. We have chosen four directions instead of two in order to handle complex (artistic) balconies. Since we search for texture energy in all directions we multiply the resulting responses and then scale the result to the interval [0..1]. Extremely low values are thresholded (e.g. less than 0.02).

$$E_{Tex}(x,y) = G_0 G_{45} G_{90} G_{135}(x,y) \tag{6.6}$$

We then combine (multiply) the balconies cluster probability (equation 6.5) of each pixel with its texture energy (equation 6.6) and scale back the result to [0..1] to obtain the soft mask $S(x,y)$.

$$S(x,y) = P(B|i_{x,y})\,E_{Tex}(x,y) \tag{6.7}$$

To obtain each balcony size and position (figure 6.4 (d))we morphologically close the refined soft mask, threshold at a very low value (e.g. 0.02) and use a flood fill algorithm to extract connected components and perform size/position sanity checks on these components. The kernel for all morphological operations is a square whose size is chosen based on the image resolution (e.g. ksize = 21 pixels for resolutions close to 100 pixels/meter). We compared our balcony detection algorithm with the method described in [Hernandez 2009] and an example is shown in figure 6.5.

Both methods have the same limitation when dealing with close balconies (spacing<20cm). However, our method can manage balconies with different heights since we process separatly each balcony while the method described in [Hernandez 2009] can not since they use a horizontal profile projection method (see the small balconies of the 3^{rd} floor in figure 6.5).

The facade inpainting is done for all non zero values in the refined soft mask. Each pixel color $I(x,y)$ is replaced by a weighted combination of its original color $I(x,y)$, the color in the morphologically opened image $M(x,y)$ and the color of a pixel N lines above $I(x,y-N)$ (e.g. N = 5 pixels for resolutions close to 100

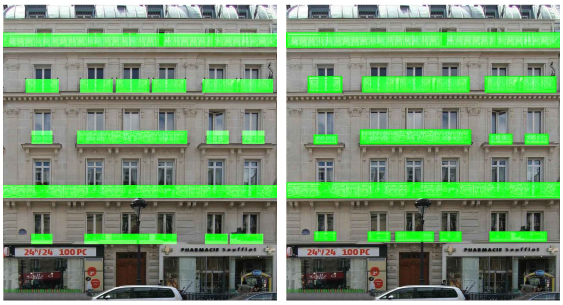


Figure 6.5: Balcony detection comparison. Left : a combined morphological and horizontal profile projection method. Right : our balcony detection method.

pixels/meter). The last component is introduced so that vertical lines are continued in the inpainting image, and this also propagates horizontal texture with a period of less than N.

$$T(x, y) = \alpha \cdot M(x, y) + (1 - \alpha)I(x, y - N) \tag{6.8}$$

$$C(x, y) = S(x, y) \cdot T(x, y) + (1 - S(x, y)) \cdot I(x, y) \tag{6.9}$$

Despite its empiric construction the balconies inpainting algorithm has proven to be robust in practice and it gives good results on several facades. We have comapared it with a recently published inpainting technique [Liu 2009] and it gives better results on different facades (Figure 6.6). Alternative methods using Dynamic Programming and Belief Propagation such [Sun 2005] can be investigated to improve or extend the concept to other occluding architectural elements or vegetation to automatically produce a high-quality facade texture without artifcats.

6.4.3 Cornice detection

To detect cornices we propose a simple heuristic that detects and merges elongated segments in a fine granularity segmentation. Horizontal elongated segments (aspect ratio inferior to 20) are grouped, merged and filtered using the detected window positions. The presence of a cornice is signaled by a group of elongated segments that spans more than 50% of the facade's width. The results of Gao and Bischof [Gao 2009] could be a basis for further developements to improve the cornice detector. Based on some small scale experiments we have performed, for this task trained Viola-Jones like detectors are inappropriate because of the aspect ratio of cornices. Similarly pixel-wise trained detectors do not work well because the cornice has almost the same color/texture as the facade wall. Cornice detection results are shown in figure 6.12. Problems are mostly due to low precision(more then 50% false positives). The unreliable results from the cornice detector are compensated by a

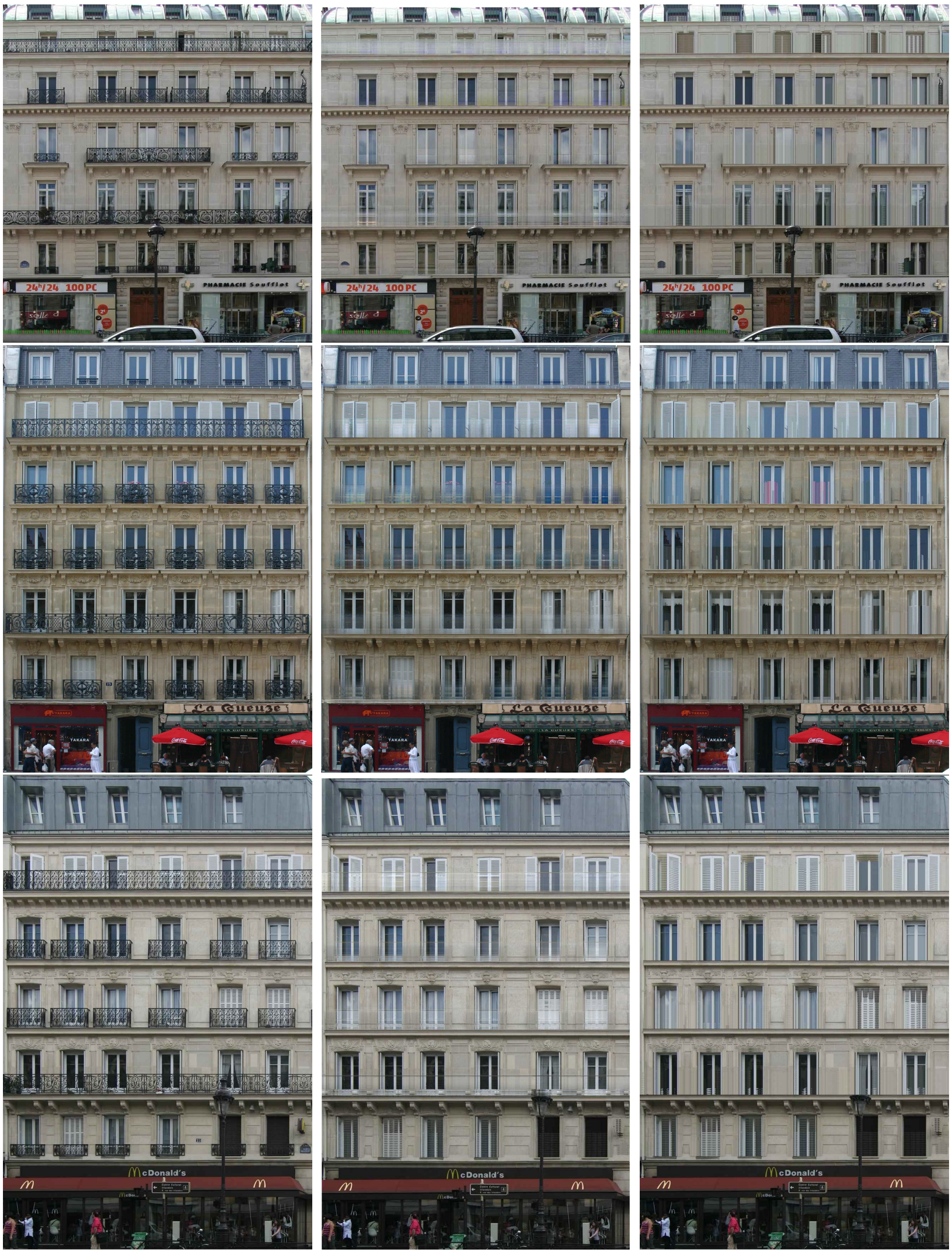


Figure 6.6: Facade inpainting results. Left : original image. Middle : using a more generic image inpaintig technique [Liu 2009]. Notice the horizontal artifacts after inpainting the balconies. Right : using our specialized inpainting method. The inpainted image is used to texture the procedural facade 3d model derived from the stochastic grammar. We found that if we map the original image directly into the 3d model artifcats appear because the balcony texture is mapped into the windows and facade wall. Since our window detection algorithm is robust to noise and occlusions we get similar results when using the original image or the inpainted image as input for window detection.

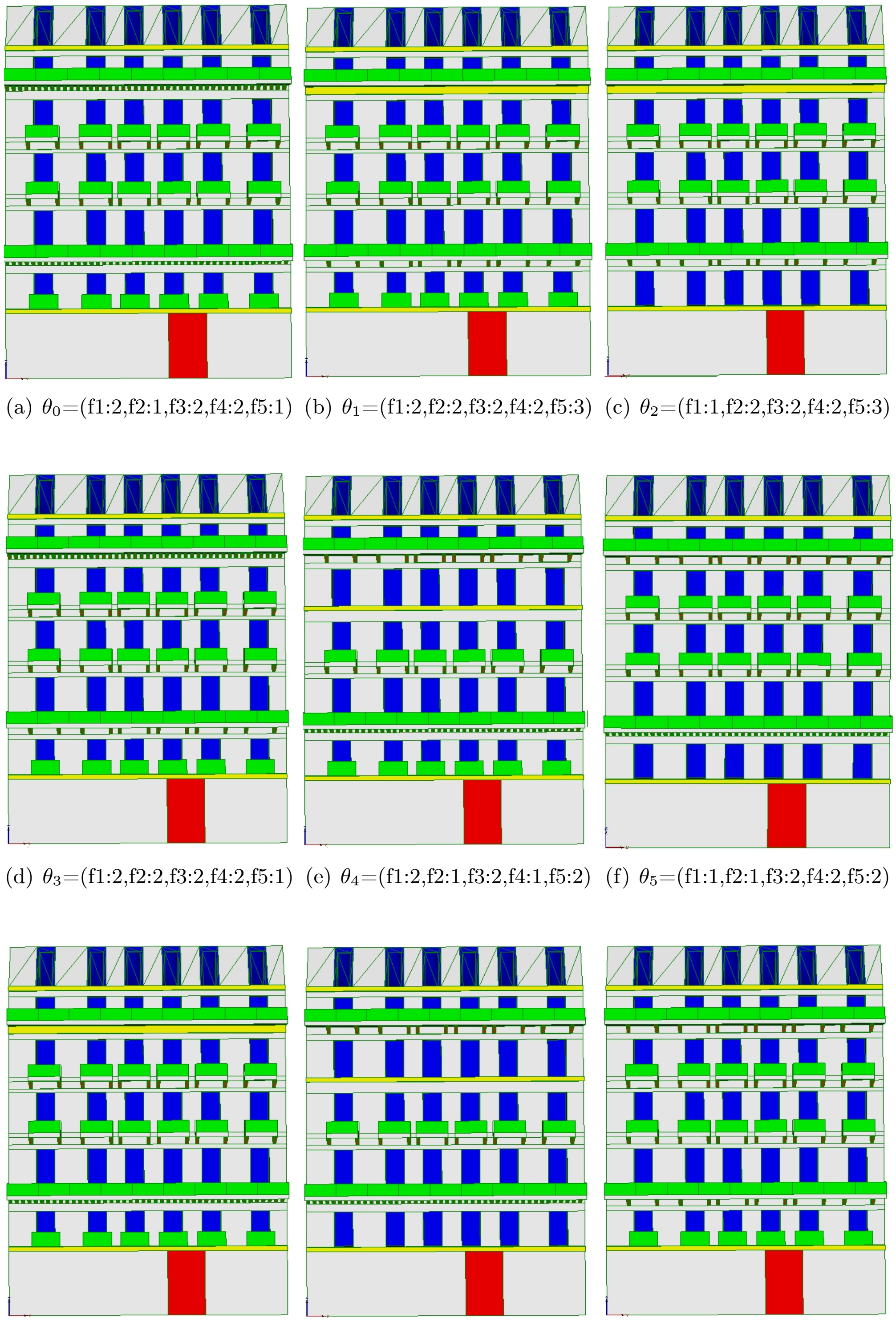
strict definition of the relative position and size in the grammar file. This technique can be generalized, meaning that if you have a detector that yields unreliable results you should restrict as much as possible the parameter space proposed by the grammar and contextually link the existance of the element with another reliable element. Cornices are linked(probabilistically derived from the same rule) as window rows and modilions/consoles are linked with windows.

6.4.4 A Generic Element Detector

In order to detect other architectural elements such as doors, modillions and ornaments (Figure 6.12) we use a trained Viola-Jones [Viola 2002] detector which uses Haar descriptors as image features. The modillions detector is trained on a database of 226 positive examples and 344 negatives. For door detection we use 56 positive images and 711 negatives. The size of the positive and negative database for each detector are then increased by an automatic procedure that adds random noise. The generic element detector has shown very good results for detecting architectural elements that show little intra-class variations. For modillion detection the recall was superior to 80% and the precision was superior to 80% for all datasets. However results were significantly worse for elements that have a high degree of variability: windows - 53% recall [Ali 2007], and doors <30% recall. In our pipeline it used for detecting modilions and doors. While we think that the detection of modilions is sufficient, the classifier for doors should be refined in future work. However, the door detection algorithm is not the most critical part of our pipeline and most of the time there is only one door in a facade.

6.5 The stochastic grammar

In this section we describe how we modeled stochastic grammars for our test data set of facade images from Paris. We developed three different grammars for three different styles: Haussmannian $(1850 - 1870)$, Louis XIII $(1595 - 1660)$, and Louis XIV $(1660 - 1700)$ and we will describe the Haussmannian style in more detail. The Haussmannian style is probably one of the most interresting architectural styles in Paris since it is the most frequent and has challenging architectural elements such as balconies, cornices, modillions, and ornaments. George Eugene Haussmann, was the prefect of Seine in Paris from June 23^{rd}, 1853 till January 5^{th}, 1870. He supervised the architectural transformations of Paris under the Second Empire by working on a vast plan of renovation. During this period strict architectural laws were in effect to regulate the construction of new buildings. Haussmanian facades follow these rules and a stochastic shape grammar was written with the help of an architect to setup the variations on the style and the corresponding probabilities. A typical Haussmannian facade sampled from a stochastic grammar is shown in figure 6.8 together with a corresponding real facade in Paris that follows the same semantic structure. Following the same approach we also developed a grammar



(a) θ_0=(f1:2,f2:1,f3:2,f4:2,f5:1) (b) θ_1=(f1:2,f2:2,f3:2,f4:2,f5:3) (c) θ_2=(f1:1,f2:2,f3:2,f4:2,f5:3)

(d) θ_3=(f1:2,f2:2,f3:2,f4:2,f5:1) (e) θ_4=(f1:2,f2:1,f3:2,f4:1,f5:2) (f) θ_5=(f1:1,f2:1,f3:2,f4:2,f5:2)

(g) θ_6=(f1:2,f2:1,f3:2,f4:2,f5:3) (h) θ_7=(f1:1,f2:1,f3:2,f4:1,f5:2) (i) θ_8=(f1:2,f2:2,f3:2,f4:2,f5:2)

Figure 6.7: Visualization of the procedural space: some facade variations sampled from the stochastic grammar of the Haussmannian architectural style (windows in blue, balconies in green, cornices in yellow, modillions and doors in red). Each facade structure θ_i is the result of the derivation of different variations of the production rules. The facade prior probability is computed using the product of the corresponding rule variations probabilities. In the examples the position of windows and the door position are fixed.

for the following styles: Louis XIII-XVI (1595 – 1790), Empire (1800 – 1815), and Restauration (1815 – 1830).

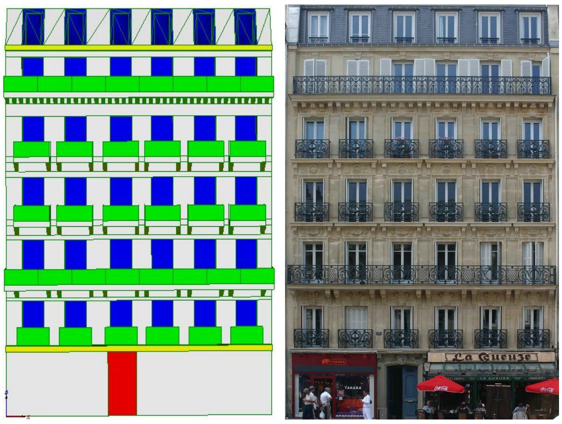


Figure 6.8: A typical Haussmannian facade structure sampled from the stochastic shape grammar. The 2^{nd} and the last floor has generaly a long balcony with a support base and the other floors has small balconies with a support base while the first floor can be composed of flat iron balconies without a support base. Each floor can also be composed of modillions and ornaments. This style is similar to other architectural styles in France and several European cities. This particular Haussmannian facade structure is one of the most frequent thus it has a high prior probability according to equation 6.13.

Some other semantic facade structure variations sampled from the Haussmannian stochastic grammar (see table 6.2) are illustrated in figure 6.7. We envision that users of our system can also use a similar approach to generate new grammars for a wide variety of architectural styles. While in our system implementation scripting experience is necessary to define new styles, recent work showed how to define grammars with a visual user interface [Lipp 2008]. Incorporating a similar approach could allow a larger number of users to model stochastic grammars.

6.6 Top-Down optimization

The motivation behind our approach is to use a Bayesian inference framework where the architectural knowledge is encoded inside the stochastic shape grammar and used as prior to improve the facade reconstruction process. The detected architectural elements are the evidence in the image and our aim is to maximize the posterior probability. Since the procedural space derived from a stochastic grammar is in general very large, the inference problem cannot be solved analytically in an efficient way. Thus, we use Markov chain Monte Carlo (MCMC) methods where the facade structures are randomly sampled from the stochastic grammar. The problem is formulated using a Bayesian approach (see section 6.1) where the facade prior is computed from the stochastic grammar using the rule variations probabilities (see

rule	variations
ground floor	fixed : contain a door
first floor	f1:1 windows
	f1:2 windows, flat balconies
second floor	f2:1 windows, long balcony and a modillions row
	f2:2 windows, long balcony and a 2 modillions/window
third floor	f3:1 windows, small balconies and 2 modillions/window
	f3:2 windows with a long cornice
fourth floor	f4:1 windows, small balconies and 2 modillions/window
	f4:2 windows and a long cornice
fifth floor	f5:1 windows, long balcony and a modillions row
	f5:2 windows, long balcony and a 2 modillions/window
	f5:3 windows, long balcony and a long cornice
last floor	fixed : contain roof windows

Table 6.2: The Haussmanian stochastic grammar: variations on the different floor rules for a facade with 6 floors.

section 6.2) and the likelihood is defined as a top-down/bottom-up scoring function that measures the evidence of the proposed top-down facade structure towards the bottom-up detected elements in the image (see section 6.3). The optimization is done using the Metropolis-Hastings algorithm where the random facade structures are sampled from the stochastic grammar (see section 6.4).

6.6.1 Problem formulation

In our shape grammar a facade structure θ is represented by a tree containing non-terminal (eg: floor, tile, etc.) and terminal shapes (eg: window, balcony, cornice, etc.). Each shape has a symbol, a scope, a mesh, and a texture associated with it. Finding the optimal structure θ of a facade given its architectural style M (encoded by a stochastic grammar) and a set of bottom-up detected elements D can be formulated as a maximum a posteriori problem [Dick 2003]. The facade energy function is then defined as follows:

$$\begin{aligned} P(\theta|MD) &= \frac{P(D|\theta M)\,P(\theta|M)}{P(D|M)} && (6.10)\\ &= \frac{P(D|\theta M)\,P(\theta|M)\,P(M)}{P(M|D)\,P(D)} && (6.11)\\ &\propto P(D|\theta M)\,P(\theta|M) && (6.12)\end{aligned}$$

Each factor in equations 6.10, 6.11 and 6.12 has an intuitive interpretation:

$\mathbf{P}(\theta|\mathbf{MD})$ is the probability of the facade described by θ knowing the architectural style M and the set of bottom-up detected elements D. The best structure will maximize this probability.

$\mathbf{P}\left(\mathbf{D}|\theta\mathbf{M}\right)$ is the likelihood of the detected data D given the structure described by θ and the architectural style M. It is the evidence in the image of the given facade θ.

$\mathbf{P}\left(\theta|\mathbf{M}\right)$ is the prior probability of the facade structure θ given the architectural style M.

$\mathbf{P}\left(\mathbf{M}\right)$ is the probability of the given architectural style. As the architectural style is provided as user input this probability is always 1.

$\mathbf{P}\left(\mathbf{M}|\mathbf{D}\right)$ is the probability of the architectural style knowing the detected data. This probability is always 1 for the same reason.

$\mathbf{P}\left(\mathbf{D}\right)$ is the probability of the detected data. Our specialized window and balcony detectors return a probability, Viola-Jones derived detectors do not return such a probability. Under the assumption of independent detection the cumulative probability $P\left(D\right)$ is equal to: $P\left(D\right) = \prod_{i=0}^{|D|} P\big(D^i\big)$. However, since this probability is constant per facade it is not used to compute the facade energy.

The main conclusion of equation 6.12 is that the best reconstructed facade should be the one that maximizes the product of its prior probability given its style and the evidence in the image (the likelihood of the image D given a facade structure θ and its style M).

6.6.2 The facade prior

Since the facade structures are directly sampled from the stochastic grammar we can compute $P\left(\theta|M\right)$ using the rule probabilities (see the rule notation in section 2.1). Assuming that the root of the shape tree (the axiom) has the probability 1 we assign to each shape θ^i created by a production rule its corresponding probability $P(\theta^i)$. The probability of the facade structure θ given its architectural style M is then defined as follows:

$$P(\theta|M) = \prod_{\theta^i \in \theta} P(\theta^i) \tag{6.13}$$

6.6.3 The facade likelihood

In this subsection we propose a practical way to compute $P\left(D|\theta M\right)$, the likelihood of the image given a facade structure θ and the architectural style M. We define a top-down/bottom-up score (see figure 6.9) to measure the evidence of detected elements D towards a proposed structure θ.

Let $\theta^i \in \theta$ be a top-down shape proposed by the grammar and $D^j \in D$ a bottom-up detected architectural element in the input image. We define the evidence of θ^i in the input image as follows:

$$E(\theta^i) = \max_{D^j \in D} \delta_{ij} E_P(\theta^i, D^j) E_S(\theta^i, D^j) E_T(\theta^i, D^j) \tag{6.14}$$

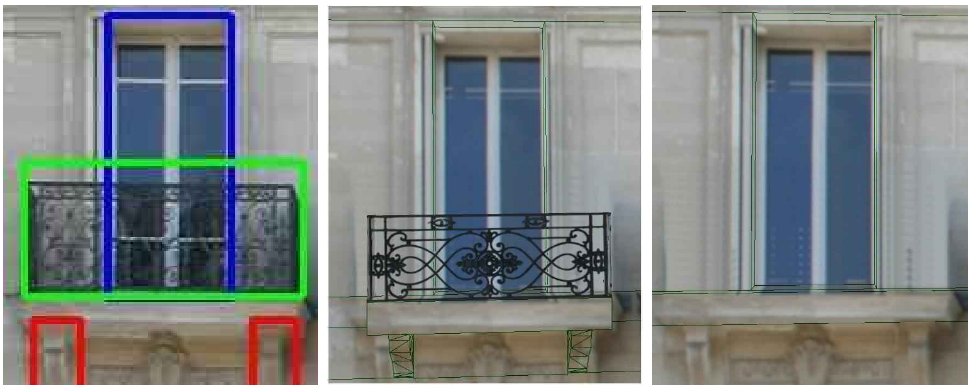


Figure 6.9: Top-down/bottom-up evidence score. From left to right : bottom-up detected elements, top-down candidate structure θ_1, top-down candidate θ_2. Here $likelihood(\theta_1) > likelihood(\theta_2)$ since the first structure contains a window, a balcony, and two modillions where the second structure contains only a window. The textures are extracted from the same tile in a facade image: the left figure is extracted from the original image where the middle and the right figures are extracted from the inpainted facade image.

where $\delta_{ij} = 1$ if the elements θ^i and D^j are of the same type and 0 otherwise. E_P, E_S and E_T are respectively the position, size and texture evidence scores. All these functions must take into account the presence of noise in the image and we define them as follows:

$$E_P\left(\theta^i, D^j\right) = \exp - \frac{\|Pos\left(\theta^i\right) - Pos\left(D^i\right)\|^2}{2\sigma^2_{Pos}} \tag{6.15}$$

$$E_S\left(\theta^i, D^j\right) = \exp - \frac{\|Size\left(\theta^i\right) - Size\left(D^i\right)\|^2}{2\sigma^2_{Size}} \tag{6.16}$$

Here σ_{Pos}, σ_{Size} are acceptable noise parameters in position and size (determined empirically at 0.3 meter and 0.1 meter). $Pos()$, $Size()$ are functions that give metric position and size of an element.

The texture evidence is based on the Mahalanobis distance d_M between the joint MRSAR [Mao 1992, Liu 1996] texture features (15) and 4 Gabor filter [Randen 1999] responses that were precomputed in the bottom-up detections stage (see section 6.4). The 19x19 covariance matrix is computed over joint features of the entire image. Let V_i, V_j be the average joint feature vectors for the elements θ^i, D^j. The Mahalanobis distance is:

$$d_M\left(\theta^i, D^j\right) = \sqrt{\left(V_i - V_j\right)^T C^{-1}\left(V_i - V_j\right)} \tag{6.17}$$

Similarly we define:

$$E_T\left(\theta^i, D^j\right) = \exp -\frac{d_M^2\left(\theta^i, D^i\right)}{2\sigma_{tex}^2} \tag{6.18}$$

We combine the evidence scores for individual elements to provide an approximation for $P\left(D|\theta M\right)$:

$$P\left(D|\theta M\right) \propto \sum_{\theta^i \in \theta} E\left(\theta^i\right) \tag{6.19}$$

Using equations 6.13 and 6.19 and according to equation 6.12 the facade energy function is computed as follows:

$$P(\theta|MD) \propto \prod_{\theta^i \in \theta} P(\theta^i) \cdot \sum_{\theta^i \in \theta} E(\theta^i) \tag{6.20}$$

6.6.4 The optimization algorithm

As optimization algorithm we experimented with several alternatives: simulated annealing, greedy optimization, and Metropolis Hastings. Since we cannot generate and evaluate a huge number of samples the theoretical properties in the limit are not that interesting in the practical application. We therefore selected an algorithm that conducts a global search within the search space and that is easy to implement and replicate. The optimization of the posterior distribution given in equation 6.20 is done using a Markov Chain Monte Carlo (MCMC) method. MCMC approximates a probability distribution using a set of discrete samples within a Markov Chain mechanism (figure 6.10).

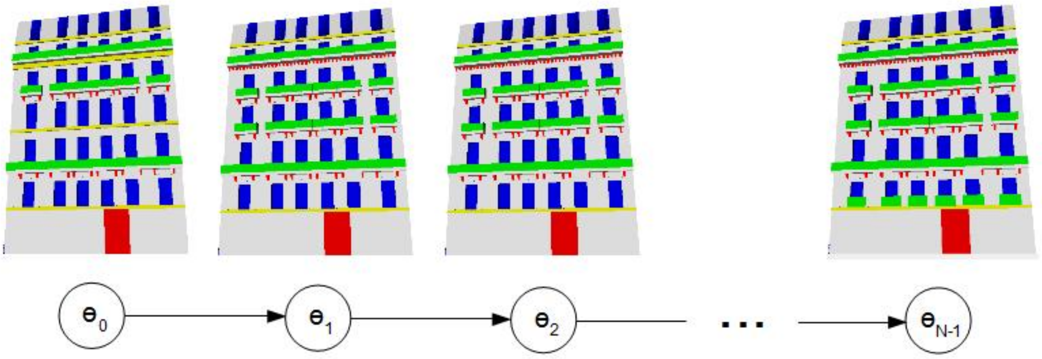


Figure 6.10: MCMC Simulation: a representation of the Markov chain, each state represents a random sampled facade structure. New elements in the chain are computed based on the direct predecessor.

The Metropolis-Hastings (MH) algorithm is the most popular variant of MCMC methods. The MH algorithm explores the state space (the space of possible facade structures within a given architectural style sampled from the stochastic grammar) in a random walk fashion. The MH algorithm starts from an initial facade structure

θ_0. Then a new structure θ is generated by sampling from the proposal distribution, $g(\theta|\theta_{n-1})$. A sample from the proposal distribution $g(\theta|\theta_{n-1})$ is generated as follows: the shape tree of the previous facade structure θ_{n-1} is cut at a random non-terminal node and the non-terminal node is derived again by the grammar. The sample is then either accepted and added to the Markov Chain as the n^{th} sample, or rejected and the previous sample is added to the chain as the n^{th} sample. The decision to accept or reject the proposed facade structure depends on the acceptance ratio, α. The acceptance ratio measure the improvement in the quality of the sample θ over the previous sample θ_{n-1}. We automatically accept samples showing improvement, and accept the rest with the probability α (Algorithm 7).

Algorithm 7 Top-down optimization algorithm

```
Sample a random initial facade structure $\theta_0$.
$\theta_* \leftarrow \theta_0$. $max \leftarrow energy(\theta_*)$.
for $n = 1$ to $N$ do
  Sample a facade structure $\theta$ from the grammar
  using the proposal distribution $g(\theta|\theta_{n-1})$
  Evaluate the transition probability:
  $\alpha \leftarrow \min\left[\frac{energy(\theta)}{energy(\theta_{n-1})}, 1\right]$
  Sample a probability $t$ from $U(0,1)$.
  if $t < \alpha$ then $\theta_n \leftarrow \theta$ else $\theta_n \leftarrow \theta_{n-1}$ end if.
  if $energy(\theta_n) > max$ then
    $max \leftarrow energy(\theta_n)$. $\theta_* \leftarrow \theta_n$.
  end if
end for
Return the most visited facade structure $\theta$
(optionally return the highest energy structure $\theta_*$).
```

The Metropolis-Hastings algorithm explores areas of high energy structures by accepting samples which increase the overall energy. The samples resulting in lower energy can still be added to the Markov chain with probability α in order to keep the MH algorithm from becoming stuck in local maxima. The Markov chain is constructed from drawing $N = N_{burn} + N_{mix}$ samples where N_{burn} is the number of burn-in samples and N_{mix} is the number of samples required to reach the mixing time after burn-in until the equilibrium distribution is reached. The facade structures sampled during the burn-in period are removed from the chain in order to make the Metropolis-Hastings algorithm less sensitive to the impact of a badly located intial structure θ_0. An alternative simpler computation can just return the structure with the highest energy θ_*.

6.7 Discussion

Comparison to previous works: The main problem of facade optimization is that the facades in a given style have a complex distribution from which it is impossible to draw samples. The technique used by most authors [Dick 2003, Alegre 2004, Ripperda 2008] in this case is to simulate the drawing of samples using a Markov Chain. This technique is known by the community as Monte Carlo Markov Chain (MCMC) method [Green 1995]. The Metropolis-Hastings algorithm [Hastings 1970, Metropolis 1949] is the most popular brand of MCMC methods. Dick [Dick 2003] was the first to use the MCMC algorithm using parametric models and a bayesian formulation for modeling architecture from images. However, in his approach we have to manually define the jump categories and their respective probabilities within the Markov chain. Other authors [Koutsourakis 2009] use a Markov Random Field (MRF) formulation and parametric grammars for building reconstruction. Our proposed approach using a stochastic shape grammar as architectural style prior instead of parametric models or specific grammars (a set of deterministic rules) together with an efficient Metropolis-Hastings optimization algorithm within a bayesian formulation completes and improves the state of the art towards a full model-based and photo-realistic 3d facade reconstruction. We believe that one important conclusion from our work is that top-down optimization has to be complimented with bottom-up detectors to be successful.

Results: The advantage of a model-based approach (as opposed to an image-based approach) is the robustness to missing data, noise and occlusions. More over, the reconstructed facade structures are consistent with the input grammar (the model). Our system takes advantage of this features and offers a flexible framework for facade reconstruction. The bottom-up detectors can be enriched if we want to manage additional architectural styles. We use a dataset consisting of more than 30 rectified facade images belonging to five different architectural styles. The shape grammars for these architectural styles have between 40 and 50 rules. We show selected final reconstruction results for facades belonging to different architectural styles in figure 6.12 and 6.13. The reconstruction process takes around 3-5 minutes depending on the image resolution. The bottom-up detection results take about 3 minutes and the optimization step takes about 2 minutes on average. We used a computer with a core duo processor 1.6GHZ and 2GB of RAM to time the results.

Limitations: The major limitation of our approach lies in the window detection algorithm which works only under the horizontal and vertical windows alignment assumption (see Figure 6.11). We are currently working on a generic windows classification approach which will remove this limitation.

Future work: While the proposed optimization algorithm works well for our test cases, we envision two additional changes that could improve our method. First, we would like to extend our shape grammar definitions to introduce randomness on global parameters. Second, we would like to investigate an alternative formulation using rjMCMC. rjMCMC has a better theoretical framework for evaluating structural changes. We are also interested in extending our conceptual framework to

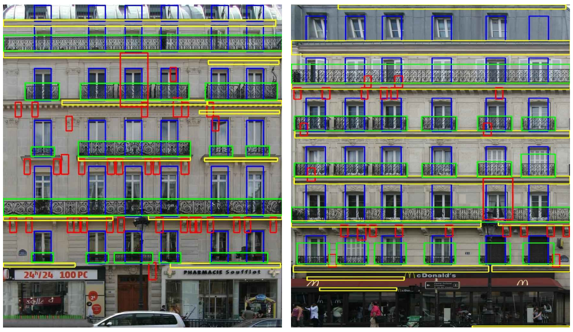


Figure 6.11: Failure cases. Left : the roof windows are not aligned with the facade windows grid. Right : on the top right of the facade a window is over detected. These examples show the typical limitations of our contour and texture based profile projection method.

other domains, such as plant modeling and texure modeling.

6.8 Conclusion

We have presented a framework to reconstruct 3d facade models from single images. The novelty of our approach is the use of stochastic shape grammars as architectural prior together with a combined top-down/bottom-up facade optimization scheme using the Metropolis-Hastings algorithm. In contrast to previous works we can model one stochastic shape grammar for a larger class of facade images instead of modeling one deterministic shape grammar for each input image. Additionally, we propose a new image-based window detector and a new balcony detection and inpainting method that is necessary to handle difficult inputs, such as images with balconies and many ornaments.

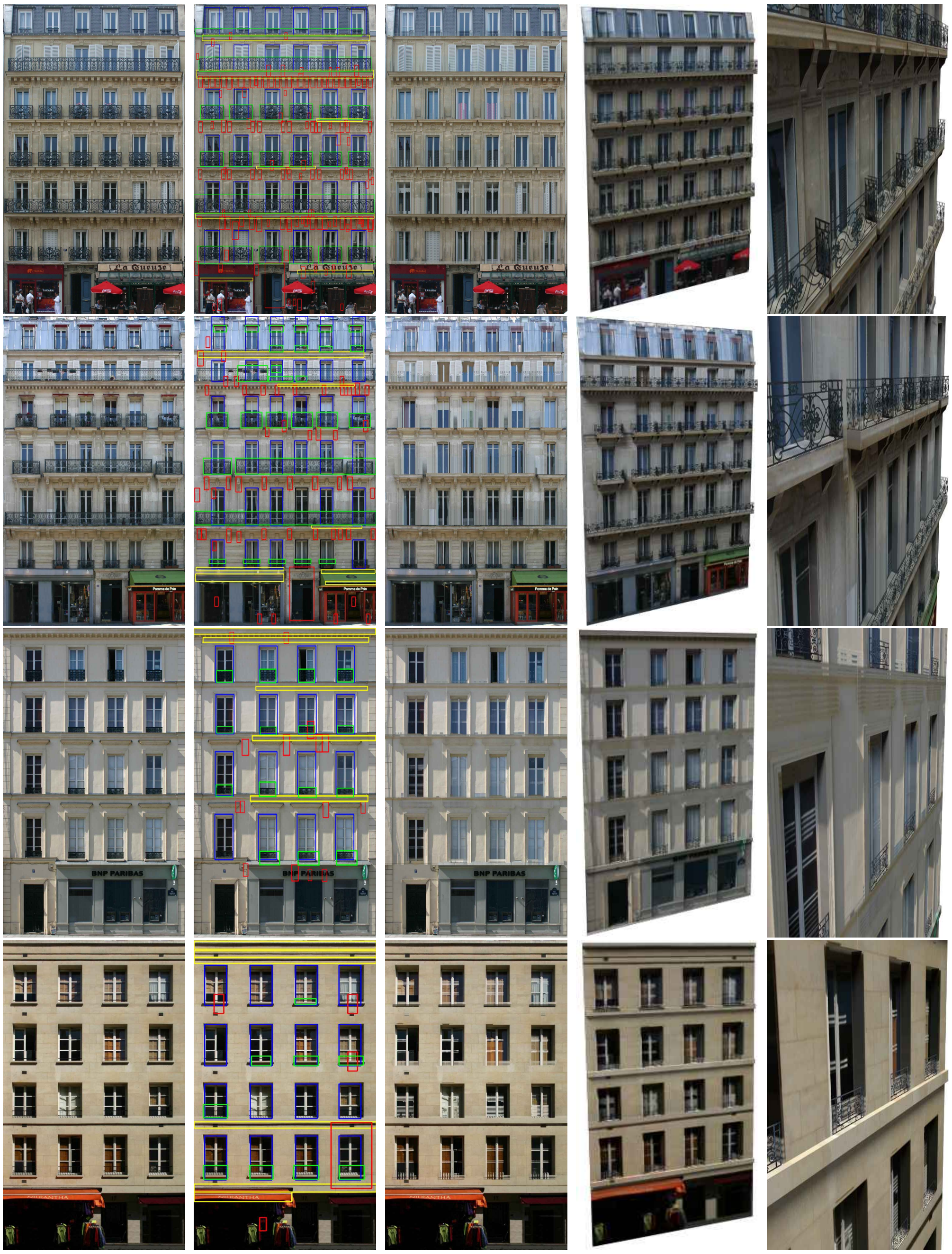


Figure 6.12: Reconstruction results. From left to right: original image, bottom-up detected elements(windows in blue - balconies in green - cornices in yellow - consoles, modillions and doors in red), inpainted facade image, reconstructed facade and facade details. The two first facades belongs to the Haussmannien style (1850 – 1870), the third facade to the Louis XIV style (1660 – 1700) and the fourth facade to the Louis XIII style (1595 – 1660).

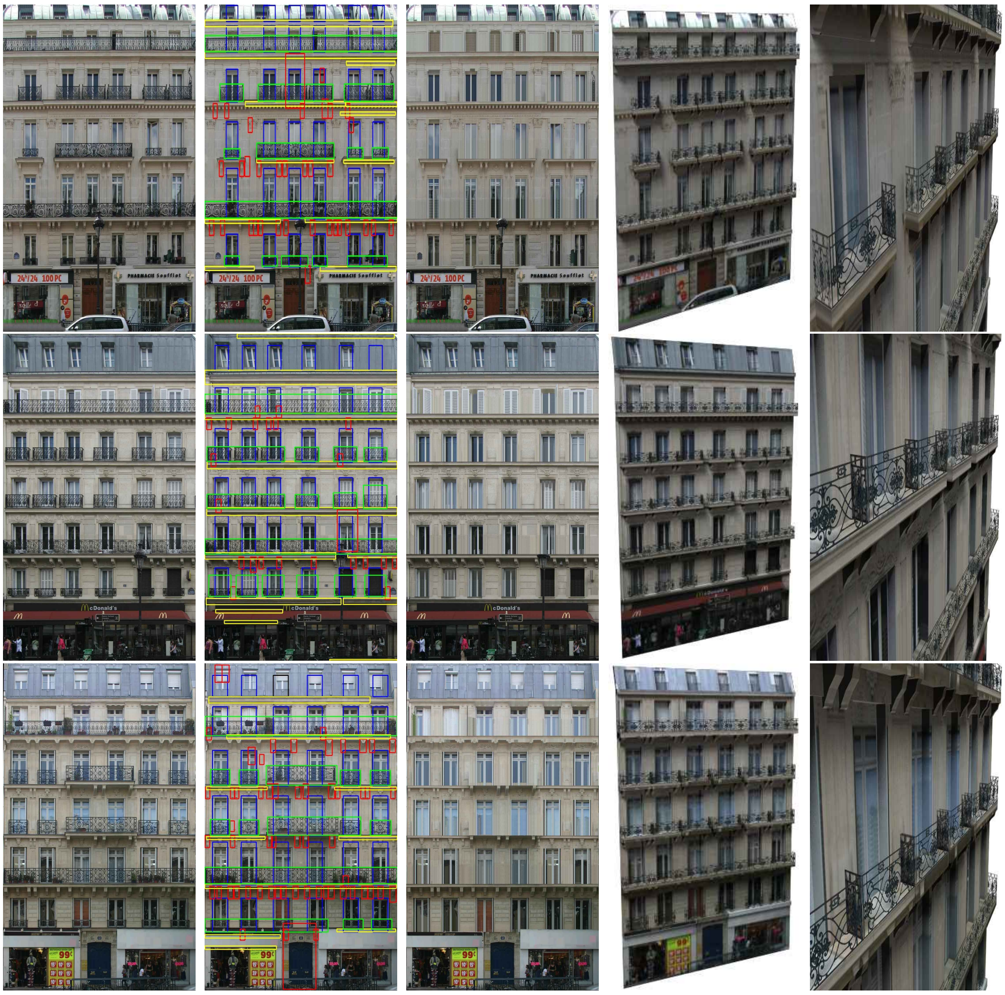


Figure 6.13: Additionnal reconstruction results. From left to right: original image, bottom-up detected elements(windows in blue - balconies in green - cornices in yellow - consoles, modillions and doors in red), inpainted facade image, reconstructed facade and facade details. The three facades belongs to the Haussmannien style (1850 – 1870).

Chapter 7

Conclusion

During this thesis we developped several tools for the reconstruction of buildings from images. The techniques used come from computer vision, pattern recognition and computer graphics fields. The main developped tools include techniques such as structure from motion, multi-view stereo, voxel coloring, and procedural modeling. Contributions in this approach was presented in two parts.

In the first part, we focused on multiple view reconstruction techniques with the aim to automatically recover the depth information of facades from a set of uncalibrated photographs. First, we implemented a structure from motion framework able to automatically calibrate a set of unordered photographs [4]. Our structure from motion implementation is sequential and uses the Exif header of JPEG images to initialize the intrinsic camera parameters. We combined the LMedS and M-Estimator algorithms to address the problem of estimating the fundamental matrix. We also used the bucketing technique to improve the spatial distribution of the randomly selected keypoints during the fundamental matrix estimation. We demonstrate the robustness of our structure from motion framework on several state of the art image sequences and on a large set of unordered photographs of Notre Dame de Paris (300 images taken by different cameras at different times and conditions). Then, we proposed techniques for the registration of the sparse reconstruction to a coarse 3d model based on the fitting of 3d planes [2,4,5]. The novelty of the approach is the use of facade correspondences to compute the absolute orientation. This offer an automatic and low-cost procedure for the georeferencing of uncalibrated photographs using a coarse 3d model. We also proposed a method for fitting the 3d facade planes from the sparse 3d reconstruction (3d point cloud and cameras poses). Finally, we proposed an accelerated multi-view stereo [1] and voxel coloring framework [3] using graphics hardware to produce a textured 3d mesh of a scene from a set of calibrated images. We demonstrate the robustness of our GPU implementaions on state of the art image sequences.

The second part was dedicated to single view reconstruction and its aim is to recover the semantic structure of a facade from an ortho-rectified image. The novelty of this approach is the use of a stochastic grammar describing an architectural style as a model for facade reconstruction. The stochastic grammar describing Paris architectural styles was modeled with the help of an architect/designer based on a large set of facade photographs. In our approach we combine bottom-up detection with top-down proposals to optimize the facade structure using the

Metropolis-Hastings algorithm. During each iteration a facade structure is randomly sampled from the stochastic shape grammar and an energy function is used for its evaluation. The energy function is based on the consistency of the sampled facade structure with the detected facade elements. To address the problem of window detection we proposed a combined contour and texture profile projection method. We also developped a method for balconies detection and removal using inpainting techniques. In order to detect additionnal facade elements such as doors and modillions we used trained Viola-Jones classifiers based on a large database of image samples. We demonstrate the robustness of our approach on several facade images belonging to different Paris architectural styles.

Further investigations should be carried in order to combine multiple view reconstruction techniques with single view procedural modeling. As first step, we can use depth maps as an additionnal input for single view facade reconstruction. This will make the semantic recognition process easier and will allow the automatic tuning of model parameters such as windows, balconies, and cornices depth.

The procedural modeling tool developped in this thesis was succefully used within the Terra Numerica project to produce 3d models of large-scale urban environments for the Paris city and is being under integration within the DataBase Generator System (DBGS) in Thales company for training and simulation markets. The developped procedural modeling tool and associated urban shape grammars are also used in other projects such as "Lyon Confluence" whose aim is to produce a 3d model of a future urban part of the Lyon city and the "Bus Training Game" (BTG) project whose aim is to develop a bus training simulator in urban environments to prevent accidents.

During the PhD thesis in Thales a patent was pended for a process to observe scenes partially covered by a set of cameras using a reduced number of screens [8]. The idea consists in a 3d video surveillance system where the user can see the whole set of cameras within a textured 3d model of the scene. The moving objects such as persons are then tracked and represented in real-time in 3D. This process offers a global view of the surveillance system and a flexible way for the user to move from a camera viewpoint to another one in order for example to track peoples.

Publications:

[1] Oussama Moslah, Alex Valles-Such, Vincent Guitteny, Serge Couvet and Sylvie Philipp-Foliguet. Accelerated Multi-View Stereo using Parallel Processing Capababilities of the GPUs. 3DTV-conf 2009.

[2] Oussama Moslah, Vincent Guitteny, Serge Couvet. Geo-Referencing Uncalibrated Photographs using Aerial Images and 3D Urban Models. (CORESA2009).

[3] Oussama Moslah, Arnaud Debeugny, Vincent Guitteny, Serge Couvet and Sylvie Philipp-Foliguet. Towards Real-Time and Accurate Voxel Coloring Framework. International Conference on Computer Vision Theory and Applications (VISAPP 2009)

[4] Oussama Moslah, Mathieu Klee, Antoine Grolleau, Vincent Guitteny, Serge Couvet and Sylvie Philipp-Foliguet. Urban Models Texturation from Uncalibrated Photographs. 23rd International Conference Image and Vision Computing New Zealand (IVCNZ 2008).

[5] Oussama Moslah, Vincent Guitteny, Serge Couvet. Fitting Uncalibrated Photographs to Geo-Referenced Urban 3D Models. Association Française de Réalité Virtuelle (AFRV 2008)

[6] Vincent Guitteny, Oussama Moslah and Serge Couvet. Infrared Based Camera Registration for In-Door/Out-Door Augmented Reality. Association Française de Réalité Virtuelle (AFRV 2008).

[7] Vincent Guitteny, François-Pierre Robert and Oussama Moslah. Infrared Based Camera Calibration for Urban Augmented Reality. 2nd International WorkShop on Mobile Geospatial Augmented Reality (REGARD 2008).

[8] Olivier Desmaison, Vincent Guitteny and Oussama Moslah. Procedé dobservation de scenes couvertes au moins partiellement par un ensemble de cameras et visualisables sur un nombre reduit decrans. French patent, 6 June 2008, N° 08/03 168.

Bibliography

[A. W. Fitgibbon 1998] A. Zissermann A. W. Fitgibbon G. Cross. *Automatic 3D Model Construction for Turn-Tables Sequences.* Lectures notes in Computer Sciences, 1998. 45

[A. 2003] Dick A., Torr R., Ruffle S. and Cipolla R. *Modeling and interpretation of architecture from several images.* IJCV, 2003. 28

[Alegre 2004] Fernando Alegre and Frank Dellaert. *A Probabilistic Approach to the Semantic Interpretation of Building Facades.* In Proc. CIPA, 2004. 67, 68, 86

[Ali 2007] Haider Ali, Christin Seifert, Nitin Jindal, Lucas Paletta and Gerhard Paar. *Window Detection in Facades.* ICIAP, 2007. 73, 78

[A.M. Fischler 1981] R.C. Bolles A.M. Fischler. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.* ACM Communication, vol. 24, pages 381–395, 1981. 16

[Bouguet 1999] J-Y. Bouguet. *Pyramidal Implementation of the Lucas-Kanade Feature Tracker.* In OpenCV Documentation, Microprocessor Research Labs, Intel Corporation., 1999. 10

[CG: 2008] *NVidia Cg programming language homepage http://developer.nvidia.com/page/cg_ main.html.* 2008. 35

[Cornelis 2005] N. Cornelis and L. V. Gool. *Real-time connectivity constrained depth map computation using programmable graphics hardware.* In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), pages 1099–1104, 2005. 35

[CUD 2008] *NVidia Cuda API technology homepage http://www.nvidia.com/cuda.* 2008. 35

[Cuda 2008] Cuda. *CUDA: Compute Unified Device Architecture, www.nvidia.com/cuda.* NVidia, July 2008. 44

[Dick 2003] A. R. Dick, P. H. S. Torr and R. Cipolla. *Modeling and interpretation of architecture from several images.* IJCV, 2003. 67, 68, 70, 73, 81, 86

[et R.C. Bolles] M.A. Fischler et R.C. Bolles. *Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography.* In Communication Association and Computing Machine, volume 24(6), pages = 381-395, year=1981. 21

[F. Goetz 2005] G. Domik F. Goetz T. Junklewitz. *Real-Time Marching Cubes on the Vertex Shader.* In Eurographics, 2005. 44

[F. Mosteller 1977] J. W. Tukey F. Mosteller. Data analysis and regression: A second course in statistics. 1977. 14

[Felzenszwalb 2004] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. *Efficient Graph-Based Image Segmentation.* International Journal of Computer Vision, vol. 59, no. 2, September 2004. 72

[Forstmann 2004] S. Forstmann, J. Ohya, Y. Kanou, A. Schmitt and S. Thuering. *Realtime stereo by using dynamic programming.* In Proc. of CVPR Workshop on Real-time 3D Sensors and Their Use, 2004. 37

[Franco 2003] J.S. Franco and E. Boyer. *Exact Polyhedral Visual Hull.* In British Machine Vision Conference (BMVC'03), volume I, pages 329–338, September 2003. 43, 44

[G. Xu 1996] Z. Zhang G. Xu. Epipolar geometry in stereo, motion and object recognition : A unified approach. Springer, 1996. 12, 14

[Gao 2009] Rui Gao and Walter F. Bischof. *Detection of Linear Structures in Remote-Sensed Images.* ICIAR, 2009. 76

[Gong 2005] M. Gong and Y.-H. Yang. *Near real-time reliable stereo matching using programmable graphics hardware.* In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), pages 924–931, 2005. 37

[GPG 2008] *GPGPU, General-Purpose Computation Using Graphics Hardware homepage http://www.gpgpu.org/.* 2008. 35

[Green 1995] Peter J. Green. *Reversible jump markov chain monte carlo computation and bayesian model determination.* Biometrika, vol. 87, pages 711–732, 1995. 86

[Grigorescu 2002] S.E. Grigorescu, N. Petkov and P. Kruizinga. *Comparison of texture features based on Gabor filters.* IEEE Trans. on Image Processing, vol. 11, no. 10, pages 1160–1167, 2002. 75

[Hartley 1997] Richard I. Hartley and Peter Sturm. *Triangulation.* Computer Vision and Image Understanding, vol. 62(2), pages 146–157, 1997. 23

[Hartley 2003] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision.* Cambridge University Press, Second Edition, 2003. 7, 8, 12, 13, 16, 18, 21, 23, 35

[Hastings 1970] W. K. Hastings. *Monte carlo sampling methods using markov chains and their applications.* Biometrika, vol. 57, pages 97–109, 1970. 86

[Hernandez 2009] Jorge Hernandez. *Morphological Image Analysis for Urban Modeling.* PhD thesis, CMM - Mines ParisTech, 2009. 75

[Hohmann 2009] Bernhard Hohmann, Ulrich Krispel, Sven Havemann and Dieter Fellner. *CityFit: High-Qualiy Urban Reconstructions by Fitting Shape Grammarsto Images and Derived Textured Point Clouds.* In Proc. 3DARCH, 2009. 67, 68

[Huber 1981] P.J. Huber. Robust statistics. 1981. 14, 19

[Koutsourakis 2009] P. Koutsourakis, O. Teboul, L. Simon, G. Tziritas and N. Paragios. *Single View Reconstruction Using Shape Grammars for Urban Environments.* In Proc. ICCV, 2009. 67, 68, 86

[Krueger 2003] J. Krueger and R. Westermann. *Acceleration Techniques for GPU-Based Voluume Rendering.* In IEEE Visualization'03, 2003. 44

[Kuzu 2001] Y. Kuzu and V. Rodehorst. *Volumetric Modeling Using Shape from Silhouette, Photogrammetry and Cartography.* 2001. 45

[Lee 2004] Sung Chun Lee and Ram Nevatia. *Extracting and integration of window in a 3D building model from ground view image.* In Proc. CVPR, 2004. 68, 71, 73

[Lipp 2008] Markus Lipp, Peter Wonka and Michael Wimmer. *Interactive Visual Editing of Grammars for Procedural Architecture.* In Proc. SIGGRAPH, 2008. 63, 80

[Liu 1996] Fang Liu and Rosalind W. Picard. *Periodicity, Directionality, and Randomness: Wold Features for Image Modeling and Retrieval.* IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI, vol. 18, no. 7, pages 722–733, 1996. 71, 83

[Liu 2009] J. Liu, P. Musialski, P. Wonka and J. Ye. *Tensor Completion for Estimating Missing Values in Visual Data.* In ICCV, 2009. 76, 77

[Lorensen 1987] W. E. Lorensen and H. E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* In SIGGRAPH'97, August 1987. 44, 47

[Lourakis 2004] M.I.A. Lourakis and A.A. Argyros. *The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm.* In ICS/FORTH Technical Report No. 340, 2004. 21, 24

[Lourakis 2006] M.I.A. Lourakis. *Non-linear, robust homography estimation.* In Inst. of Computer Science-FORTH, Heraklion, Crete, Greece, 2006. 21

[Lowe 2003] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* IJCV, 2003. 10, 21

[Mao 1992] Jianchang Mao and Anil K. Jain. *Texture Classication and Segmentation Using Multiresolution Simultaneous Autoregressive Models.* Pattern Recognition, vol. 25, pages 173–188, 1992. 71, 83

[Metropolis 1949] Nicholas Metropolis and S. Ulam. *The Monte Carlo Method.* Journal of the American Statistical Association, vol. 44, no. 247, pages 335–341, 1949. 86

[MP: 2008] *Marc Pollefeys' homepage, University of North Carolina at Chapel Hill, www.cs.unc.edu/˜marc/.* 2008. 41

[Mueller 2006] P. Mueller, P. Wonka, S. Haegler, A. Ulmer and L. Van Gool. *Procedural Modeling of Buildings.* In Proc. SIGGRAPH, 2006. 67

[Mueller 2007] Pascal Mueller, Gang Zeng, Peter Wonka and Luc Van Gool. *Image-based procedural modeling of facades.* In Proc. SIGGRAPH, 2007. 58, 67, 68, 70, 73

[N. 2006] Snavely N., Seitz S. M. and Szeliski R. *Photo tourism: Exploring photo collections in 3D.* In ACM Transactions on Graphics (SIGGRAPH Proceedings), volume 25(3), pages 835–846, 2006. 7

[OSG 2008] *OpenSceneGraph: an open source high performance 3D graphics toolkit, 2008. http://www.openscenegraph.org/.* 2008. 40

[P. J. Rousseeuw 1987] A. M. Leroy P. J. Rousseeuw. Robust regression and outlier detection. John Wiley & Sons, Inc, 1987. 17, 19

[P.A. Beardsley 1997] D. W. Murray P.A. Beardsley A. Zisserman. *Sequential Updating of Projective and Affine Structure from Motion.* International Journal on Computer Vision, vol. 23, pages 235–259, 1997. 20

[P.A. 1997] Beardsley P.A., Zisserman A. and Muray D.W. *Sequential Updating of Projective and Affine Structure from Motion.* IJCV, vol. 23(3), pages 235–259, 1997. 7

[Petkov 1997] N. Petkov and P. Kruizinga. *Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli: bar and grating cells.* Biological Cybernetics, vol. 76, no. 2, pages 83–96, 1997. 74

[Pollefeys 2004] Marc Pollefeys. *Visual Modeling With A Hand-Held Camera.* IJCV, vol. 59(3), pages 207–232, 2004. 7, 20, 23, 35, 37

[Ponce 2005] J. Ponce, T. Papadopoulo, M. Teillaud and B. Triggs. *On the Absolute Quadric Complex and its Application to Autocalibration.* In CVPR, 2005. 7

[Prusinkiewicz 1991] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. Springer Verlag, 1991. 58

[Randen 1999] Trygve Randen and John H. Husoy. *Filtering for texture classification: a comparative study.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, pages 291–310, April 1999. 83

[Ripperda 2008] Nora Ripperda. *Determination of Facade Attributes for Facade Reconstruction.* In International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2008. 67, 68, 86

[Scharstein 2002] D. Scharstein and R. Szeliski. *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms.* International Journal of Computer Vision (IJCV), vol. 47(1), pages 7–42, 2002. 35

[Seitz 1997] S. M. Seitz and C. R. Dyer. *Photorealistic Scene Reconstruction by Voxel Coloring.* In Computer Vision and Pattern Recognition Conf., pages 1067–1073, 1997. 43, 45, 48

[Shreiner 2003] D. Shreiner, M. Woo, J. Neider and T. Davis. Addison-Wesley, 2003. 36

[Stiny 1972] G. Stiny and J. Gips. *Shape Grammars and the Generative Specification of Painting and Sculpture.* Inf. Proc., vol. 71, pages 1460–1465, 1972. 58

[Sturm 1996] Peter Sturm and Bill Triggs. *A Factorization Based Algorithm for Multi-Image Projective Structure and Motion.* In ECCV, 1996. 7

[Sun 2005] Jian Sun, Lu Yuan, Jiaya Jia and Heung-Yeung Shum. *Image Completion with Structure Propagation.* In SIGGRAPH, 2005. 76

[Szelinski 1993] R. Szelinski. *Rapid Octree Construction From Images Sequences.* In CVGIP, pages 23–32, july 1993. 48

[Trendall 2000] C. Trendall and A. J. Steward. *General Calculation Using Graphics Hardware, with Application to Interactive Caustics.* In Eurographics Workshop on Rendering, pages 287–298. Springer, 2000. 44

[Triggs 1996] Bill Triggs. *Factorization Methods for Projective Structure and Motion.* In CVPR, 1996. 7

[VGG 2008a] *Visual Geometry Group Home Page, University of Oxford, www.robots.ox.ac.uk/˜vgg/.* 2008. 41

[VGG 2008b] VGG. *Visual Geometry Group Dataset.* www.robots.ox.ac.uk/˜vgg/data/data-mview.html, 2008. 44

[Viola 2002] Paul Viola and Michael Jones. *Robust real-time object detection.* International Journal of Computer Vision, 2002. 70, 74, 78

[W. R. Mark 2003] K. Akelet M.J. Kilgard W. R. Mark R. S. Glanville. *Cg: A System for Programming Graphics Hardware in a C-like Language.* In Proceedings of SIGGRAPH, 2003. 44

[Wang 2006] L. Wang, M. Liao, M. Gong, R. Yang and D. Nister. *High Quality Realtime Stereo using Adaptive Cost Aggregation and Dynamic Programming.* In Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT), 2006. 37

[Werner 2002] Tomas Werner and Andrew Zisserman. *Model Selection for Automated Architectural Reconstruction from Multiple Views.* In British Machine Vision Conference, 2002. 70

[Wonka 2003] Peter Wonka, Michael Wimmer, Francois Sillon and William Ribarsky. *Instant Architecture.* In Proc. SIGGRAPH, 2003. 58

[Xiao 2008] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek and Long Quan. *Image-based Facade Modeling.* In Proc. SIGGRAPH Asia, 2008. 67, 68

[Yang 2002] R. Yang, G. Welch and G. Bishop. *Real-Time Consensus-Based Scene Reconstruction Using Commodity Graphics Hardware.* In Proceedings of Pacific Graphics 2002, pages 225–234, 2002. 35

[Zach 2003] C. Zach, A. Klaus and K. Karner. *Accurate Dense Stereo Reconstruction using Graphics Hardware.* In Eurographics 2003, pages 227–234, 2003. 35

[Zhang 1995] Z. Zhang. *Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting.* Rapport technique, INRIA - Sophia Antipolis, 1995. 12, 14

[Zhang 1996] Z. Zhang. *Determining the Epipolar Geometry and its Uncertainty: A Review.* Rapport technique, INRIA - Sophia Antipolis, 1996. 12, 14, 16, 17, 18, 19

[Zissermann 2003] A. Zissermann and R. Hartley. Multiple view geometry in computer vision. Cambridge University press, 2nd édition, 2003. 48

Towards Large-Scale Urban Environments Modeling from Images

Abstract: The main goal of this thesis is to develop innovative and practical tools for the reconstruction of buildings from images. The typical input to our work is a set of facade images, building footprints, and coarse 3d models reconstructed from aerial images. The main steps include the calibration of the photographs, the registration with the coarse 3d model, the recovery of depth and sematic information, and the refinement of the coarse 3d model.

To achieve this goal, we use computer vision, pattern recognition and computer graphics techniques. Contributions in this approach are presented on two parts.

In the first part, we focused on multiple view reconstruction techniques with the aim to automatically recover the depth information of facades from a set of uncalibrated photographs. First, we use structure from motion techniques to automatically calibrate the set of photographs. Then, we propose techniques for the registration of the sparse reconstruction to a coarse 3d model. Finally, we propose an accelerated multi-view stereo and voxel coloring framework using graphics hardware to produce a textured 3d mesh of a scene from a set of calibrated images.

The second part is dedicated to single view reconstruction and its aim is to recover the semantic structure of a facade from an ortho-rectified image. The novelty of this approach is the use of a stochastic grammar describing an architectural style as a model for facade reconstruction. we combine bottom-up detection with top-down proposals to optimize the facade structure using the Metropolis-Hastings algorithm.

Keywords: 3d reconstruction, image-based modeling, procedural modeling, shape grammars, architecture.