Texture and Relief Estimation from Multiple Georeferenced Images

by

Franck Taillandier

Engineer of the École Polytechnique

DEA Algorithmique, Filière: Géométrie Formes et Images

at

Écoles doctorales EDITE (UPMC), Jussieu-Chevaleret, ENS-Cachan, Paris-Sud et École Polytechnique

September 2000

Author .....
DEA Algorithmique, Filière: Géométrie Formes et Images
September 7, 2000

Certified by .....
Jean-Daniel Boissonnat
Professor of Computer Science
INRIA
Thesis Supervisor

Certified by .....
Seth Teller
Associate Professor of Computer Science and Engineering
MIT Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Jean-Marc Steyaert
Chairman of the DEA
École Polytechnique

Contents

List of Figures

Acknowledgments

I would like to thank my advisor, Seth Teller, for his support, his excellent suggestions and his constant availability, Pascal Fua and Richard Lengagne for their support in the development of the mesh generation, their patience and their willing to cooperate, Jean-Daniel Boissonnat for his reassuring presence and Rachid Deriche for his interest in my research and his human qualities.
Thanks also to Gernot, Mike, Max, Byong-Mok and Fredo and to all my Graphics Group mates with whom I shared precious moments. I would like also to thank Neel and Adel for their amazing knowledge and their patience to answer my questions. Finally, I would like to especially thank Lilla for checking my English and for her support during all the hard times.

Warning

This thesis describes a system and emphasizes an end-to-end process. The focus is not on a special theory or technique. Instead, the objective is to find solutions in order to get a working process. Some solutions may seem simple but the emphasis was on finding a simple solution to validate the approach rather than a complex, noisy solution. Later on each step of the process could be studied in order to find better and more general solutions.

Chapter 1

Introduction

1.1 Thesis Overview

Extracting and rendering 3D environments is an increasingly important problem in computer vision and computer graphics and has numerous applications in civil engineering, military simulation, resource management and architecture for instance. Due to the development of hardware, software and algorithms making it easy for the user to visualize and interact with the data, the need for accurate and realistic 3D models of an urban environment is becoming more and more important.

The goal of this thesis is to describe a robust set of algorithms for two important steps of this process that consist of generating *realistic textures* and *relief for building facades*. The techniques presented toward achieving this goal are based on the following ideas:

- We use as input *pose imagery*, that is, images annotated with camera pose (position and orientation in a global coordinate system).
- The texture estimation uses all the available information and strives to iteratively refine a Consensus Texture, eliminating most occlusion, lighting variations and blur due to residual camera pose errors.
- The relief estimation uses geometric constraints and periodicity assumptions derived from 2D image analysis to enforce consistency on stereo computations and improve surface approximation.

Most of the ideas developed here are robust and general. However some simplifications have been made to get an end-to-end process in order to validate the approaches presented here.

The rest of the chapter describes the upstream components in the City Scanning Project pipeline and the input data used everywhere in this thesis. Chapter 2 describes the algorithm developed to obtain an accurate and realistic texture for a facade from all available pose images. Chapter 3 focuses on a general scheme to derive a good relief estimation from image-based and geometric-based information. The techniques presented here have been implemented and tested on a large dataset (four thousand images) of a real urban environment.

1.2 General Overview of the City Scanning Project

Despite several years of research and although architectural modeling has been addressed for long now ([Deb96],[SNW+99]), the acquisition and rendering of a 3D texture-mapped model for a large urban environment remains a tedious task whose main bottleneck lies in the need of human intervention for most of the steps of the process, which prevents it from being scalable for a large dataset. The goal of the City Scanning Project (CSP) is to build an automatic system to recover and output a realistic (texture-mapped) 3D model of an urban environment from real-world imagery [Tel97].

Whereas many other proposed schemes, in architectural modeling as well as in other vision fields, require a lot of user intervention (e.g: definition of relations between primitives in [DTM96], explicit correspondences for camera parameters computations in [Fau93]) , the objective of this system is to "accelerate and improve the population process for complex geospatial database, surmounting the scaling problem and human dependencies posed by existing semi-automatic systems" [Tel97]. I will first review the main steps of this 3D pipeline and give a brief description of results obtained from the upstream process as well as of the data used in this pipeline that I will refer to hereafter.

In the following chapters, I will then focus on texture estimation and relief estimation for a facade building and describe new algorithms that reinforce the realism of a 3D urban model.

1.2.1 The Pipeline

As shown in Figure 1-1 after [Coo98], the automatic 3D pipeline consists of several distinct steps: Acquisition, Pose Refinement (Spherical Mosaicing and Mosaic registration) and Model Extraction (Vertical Facade Extraction and Texture and Relief Estimation).

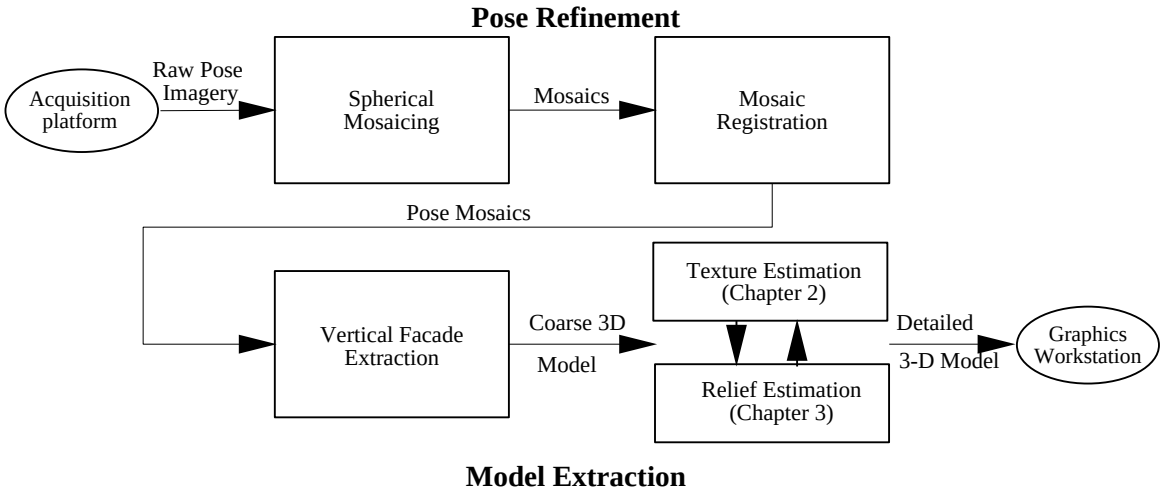


Figure 1-1: Overview of the 3D pipeline of the CSP

1.2.2 Acquisition

The first step is the acquisition platform. Although a lot of research has addressed the problem of 3D recovery from multiple images, very few physical probes exist that automatically acquire sets of images: an example is [SNW+99] whose device integrates a laser range scan but has a lack of mobility that makes it uneasy to handle in urban environment. The CSP uses a semi-autonomous robot called "Argus" ([BdCT00], Figure 1-2) featuring a differential GPS sensor, relative motion sensor and a rotative CCD camera. Data is acquired in nodes that are sets of images (in this dataset, 47 [CMT98]) taken from a single position of Argus and tiling a hemispherical space around the optical center. Each image is annotated with camera pose, position and orientation given in Earth coordinates through the use of GPS.

The nodes have significant baselines (10m) and are acquired under various illumination conditions. This produces strong perspective parallax, significant occlusion, strong lighting variations and specularity effects. These are the main difficulties we face in our task of estimating surface texture and relief.

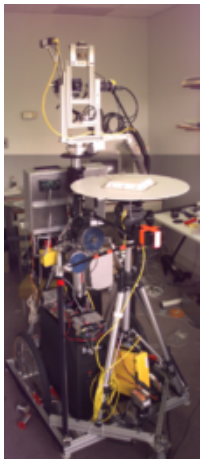
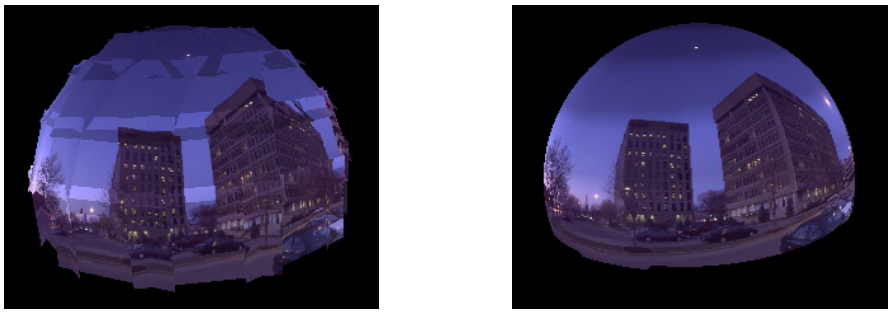


Figure 1-2: View of Argus, a semi-autonomous probe used to capture data.

1.2.3 Pose Refinement

At this stage, the first process enables us to get the intrinsic camera parameters by registering the images in **one** node. A spherical mosaicing procedure can thus be performed, which maps the images onto a sphere surrounding the camera. Another optimization process refines the camera position across the **entire set of nodes**, using feature correspondences to solve the registration among the nodes ([Coo98], [AT00]).



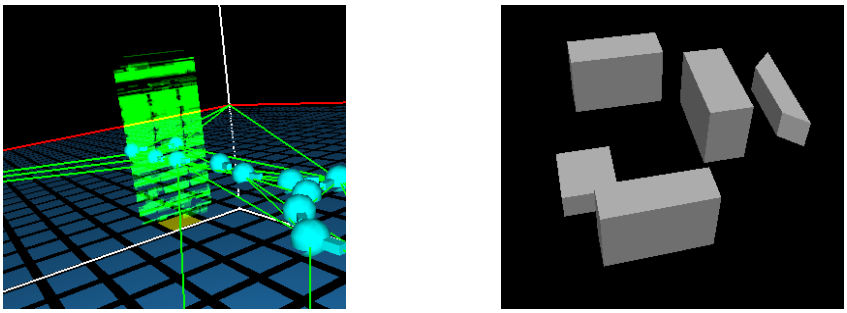
(a) (b)

Figure 1-3: (a): hemispherical tiling used for node acquisition; (b): textured hemispherical mosaic.

1.2.4 Facade Extraction

The next step of the pipeline consists of extracting a vertical facade through the whole set of images located at different positions that see it. The "sweep-plane" algorithm, about which further details can be found in [Coo98], is mainly based on the idea of sweeping a plane along different locations. By using a Hough-like transformation, horizontal edges of

each image are projected on the plane. By observing the peak of correlation while sweeping the plane, one can estimate the position of the facade. Further algorithms filter small false-positives, close the recovered facades into solids, and add rooftops, thus giving a good 3D model of the environment.



(a) (b)

Figure 1-4: (a): the sweep-plane algorithm; (b): the recovered model (not textured).

1.2.5 Input data and Objectives

I have presented the general upstream steps of the 3D pipeline of the CSP. Many refinements are still in progress, especially to retrieve camera information with more accuracy ([AT00]). For our algorithms (texture as well as relief estimation), we assume we have as input:

- a set of images annotated with camera intrinsic parameters and also a reasonably accurate estimate of camera pose and orientation (the 6 degrees of freedom).
- a coarse geometric model, mainly the orientation and the position of the planes that support the facades whose texture and relief have to be computed.

The algorithms presented in the following chapters produce as output:

- a realistic Consensus Texture for each facade, computed from the set of available images and avoiding occlusions, clutter and illumination variations.
- a depth mesh approximating the facade relief onto which the texture is mapped to finally display a realistic model of the environment.

Chapter 2

Texture Estimation

The focus of this chapter is the estimation of a Consensus Texture for a building facade from many observations taken at different locations and under various illumination conditions. The challenge mainly lies in the handling of these lighting variations as well as the removal of all kinds of occlusions that may occur. The algorithm presented here can successfully deal with global illumination variations, general occlusions due to other buildings or smaller ones such as trees, cars or pedestrians. It can also handle distortion effects due to perspective projection and blur due to residual camera pose errors. The algorithm starts out by normalizing the luminance values among the warped images in order to eliminate global illumination variations. It then uses a simple average scheme in which we introduce, in the form of weights, high-level and low-level information to overcome occlusion, clutter, noise and perspective distortion. We call these weights the Obliqueness Mask, Environment Mask and Correlation Mask.

2.1 Introduction

2.1.1 Motivation

As already mentioned, the goal of the CSP is to develop efficient techniques for fully automated 3-D reconstruction of extended high-fidelity, textured and realistic urban landscapes from ground-level imagery.

Here we describe one component of the system which takes place after the imagery has been georeferenced and refined and after a coarse geometric model of the buildings

in the environment has been extracted through a technique described elsewhere ([Coo98]). This component enables to obtain a realistic texture for a facade, avoiding occlusions and handling illumination variations.

2.1.2 Related Work

Occlusions, obliqueness, illumination variations are, among others, problems that arise when one wants to obtain a consensus color for each texel.

Occlusions as well as distortion effects make it difficult to choose one "best node" and the alternative methods proposed by [DTM96, BB95] to interpolate or sample through different observations according to the viewpoints also have the same drawbacks.

Interactive systems require the user to identify occluded pixels [DTM96]. These do not fit the requirements of our project and are obviously impractical in our case, when dealing with a large dataset.

Many other approaches have relied on the estimation of a reflectance model. These techniques consist of the estimation of the parameters of a reflectance model for an object seen by several images as in [SWI97]. In the project Realise for example, the Flux team models the Bidirectional Reflection Distribution Function (BRDF) of a rough surface. However, such algorithms are unable to handle occlusions automatically and are much less effective when dealing with significant illumination variations.

Other recent algorithms developed on the basis of the "Inpainting" technique can fill in occlusions or restore damaged pictures [BSCB00]. Although these techniques could be used in our examples to fill in occluded parts of the facades, their drawback is that we need to know in advance the position of occluded or noisy texels.

Another method, introduced previously in the City Scanning Project ([Coo98]) relies on a median-based technique to get rid of outliers (illumination variations, occlusions) present in different observations. Although this technique is successful in the removal of occlusions or illumination effects (assuming that fewer than half of the observations of each texel are occluded), it introduces speckle, characteristic of any median-based technique. Thus, the Consensus Texture produces patches of various colors and furthermore, the boundaries of any object (such as windows) tend to be blurred or disrupted, which is the result of selecting a peculiar color through images altered by projective errors.

2.1.3 Overview

The facade extraction algorithm (Section 1.2.4), in the upstream process of the CSP, supplies us with a 3D model of the environment consisting of the boundaries and the orientation of each planar surface (facade) detected in the area. The algorithm takes as input the location of the facade whose texture should be synthesized and a set of images along with their camera pose (translation and orientation with the respect to some origin).

The algorithm is mainly based on a weighted average scheme. We believe that this scheme is much smoother and more robust than the median-based one. However we need more information to discard occlusion and to handle illumination variations. New information is introduced in the scheme as weights whose importance and computation are explained in further detail in the following sections. The main advantage of this algorithm is to utilize in the computation high-level information in order to discard occlusions and perspective errors. The general computation works as follows:

- The images are first rectified onto the facade (initially assumed to be planar).
- A global occlusion mask, the Environment Mask (EM), is then computed for each node seeing the facade, taking into account any other modeled buildings that may occlude the view. Also, for each node, an "Obliqueness Mask" (OM) whose pixels have values inversely proportional to the obliqueness of the corresponding texel on the facade is estimated.
- A initial "consensus image" is then computed using these two masks embedded in the weighted average scheme as explained in Section 2.2.2. This first estimation will serve as basis for the following loop.
- The main loop described in detail in Section 2.2.3 inputs the Consensus Texture estimation and computes a Correlation Mask (CM) between the actual Consensus Texture (CT) and each rectified image. The CM models clutter such as trees, cars, pedestrians and is used as a new weight to refine the Consensus Texture. Another step embedded inside the loop deblurs the Consensus Texture by getting rid of the residual camera pose errors through a Levenberg Marquardt-like method.

The general computation is summed up in Flowchart 2-1. In this flowchart CM stands for Correlation Mask, EM for Environment Mask, OM for Obliqueness Mask and CT for

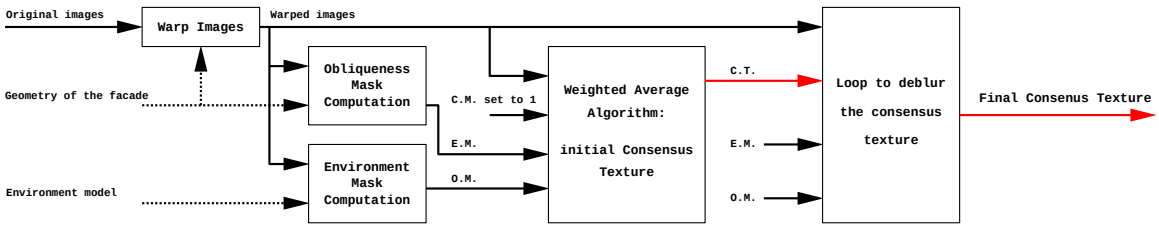


Figure 2-1: Flowchart of the general algorithm

Consensus Texture. All these notations will be reused afterwards and the distinct steps are explained in further details in the next section.

2.2 Details

2.2.1 Global Illumination

Taken in very different lighting conditions, the images may include strong illumination variations. These variations make it difficult to synthesize a Consensus Texture. In our case, however, the surfaces are mainly matte and their illumination can be thus well approximated by a Lambertian diffuse model [FvDFH90]. Furthermore, the main lighting source (the sun) is predominantly white and in practice [Coo98] the RGB color components scale linearly in different lighting conditions. Therefore, to exploit this property, we decorrelate the chrominance values and the luminance value by using the normalized CIE xyY color representation [Poy96]. Although this representation introduces a new non-linearity in the transformation RGB-xyY, it enables us to work, in most of the steps only on the Y channel (luminance) since under various (nearly white) lighting conditions, only this value should vary.

In order to normalize the luminance values among the warped images, each Y channel

is rescaled as follows:

$$Y^{\tau}_{i,j} \leftarrow Y^{\tau}_{i,j} * \frac{\overline{Y}}{\overline{Y^{\tau}}} \tag{2.1}$$

where $Y^{\tau}_{i,j}$: Y value of pixel i,j for image τ

$\overline{Y^{\tau}} = \frac{1}{\sum pixels} \sum_{i,j} Y^{\tau}_{i,j}$: average luminance of image τ

$\overline{Y} = \frac{1}{\sum nodes} \sum_{\tau} \overline{Y^{\tau}}$: average luminance of all images

Thus, each image is normalized to have the same mean luminance. This allows to model the global illumination variations among the images. Results (see Section 2.3) have proven to be more realistic due to this normalization.

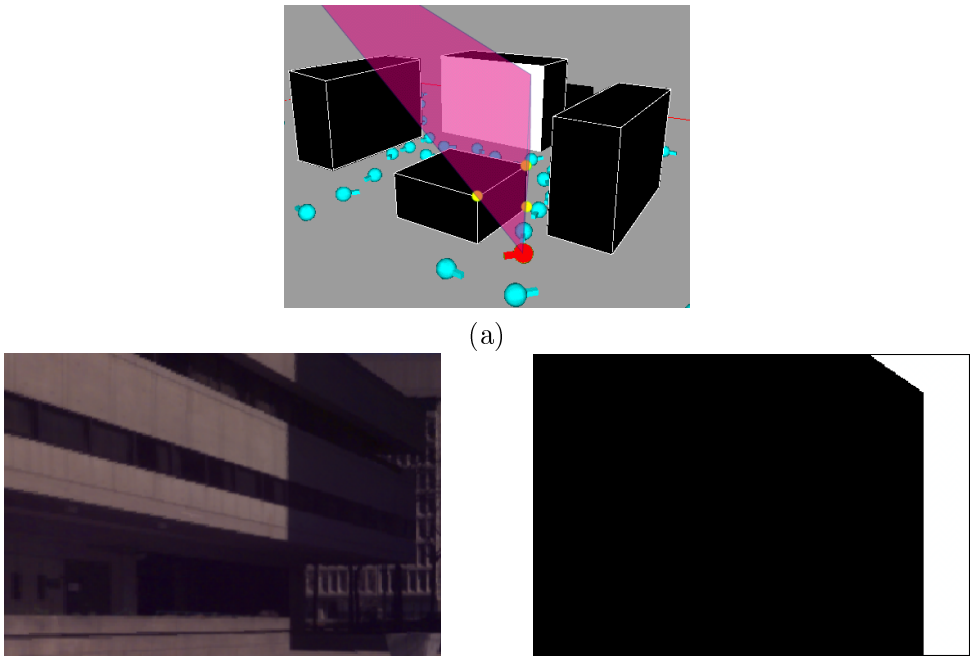
2.2.2 Weighted Average Algorithm

Environment Mask

Although previous algorithms provide us with a set of nodes that see the facade (taking into account distance and orientation of the facade compared to the node), they do not handle partial or total occlusions that may occur due to the known 3D environment. A good way to easily introduce this information in the weighted average algorithm is to use a binary "environment mask" (EM) image whose pixels have value 0 (black) if a building occludes and 1 otherwise (Figure 2-2). For every node, each facade, rendered as a black polygon in the 3D environment, is projected on the current working facade by a classical ray-cast algorithm. To speed up the process, the algorithm skips the black facade if it is further from the node than the current working facade. The EM, or binary image produced by this step, is then introduced into the loop.

Obliqueness Mask

Nodes that view a texel obliquely generally yield poor observations especially due to projective errors as shown in Figure 2-3. This effect (not taken into account in any of the previous work on aerial images that are, in most cases, fronto-parallel to the landscape) must be handled in our algorithm because the facade can be seen from very oblique viewpoints. To downweight these nodes, a new greyscale weight image is computed. Each pixel



(a) (b) (c)

Figure 2-2: (a): the position of the camera (red) and of the facade (white) in the modeled environment. The building in the foreground is projected onto the facade; (b): a color image showing the warped image; (c): the binary corresponding Environment Mask (EM)

takes the following value:

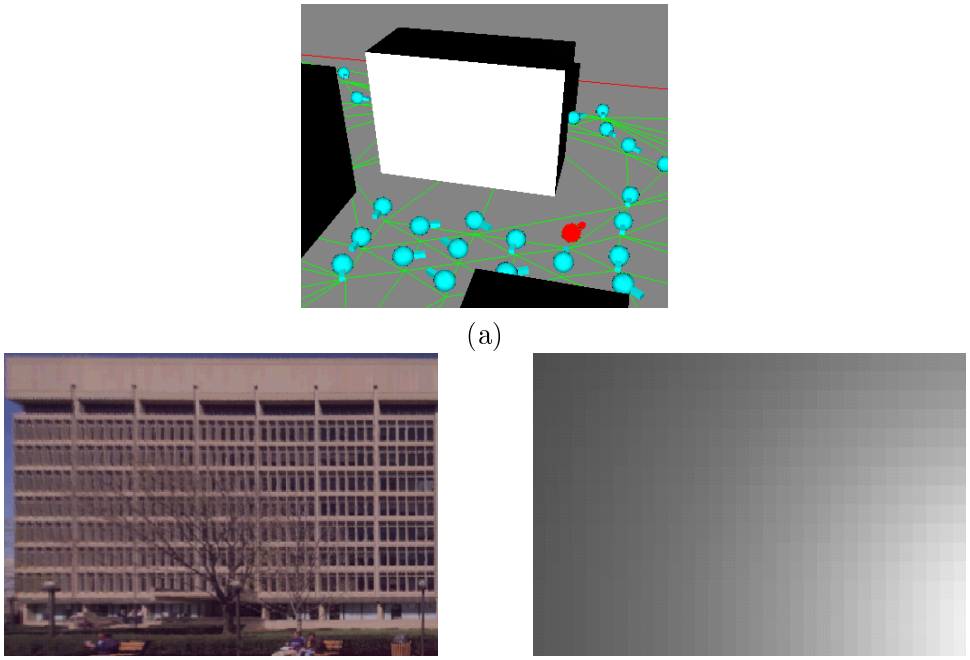
$$OM_{i,j} = \mathbf{N}.\mathbf{V_{i,j}} \tag{2.2}$$

where $\mathbf{N}$: facade's normal (normalized)

$\mathbf{V_{i,j}}$: vector direction from the texel to the center of the node (normalized)

Correlation Mask

Although the Environment Mask models "known" occlusions, those due to other buildings, we need additional information to discard smaller "clutter" occlusions caused by trees, pedestrians, cars and fences for instance. In order to achieve this, a Correlation Mask (CM) is computed for each node. We use a straightforward normalized linear correlation function on the luminance value between the actual consensus image and the warped image related to the node.



(a) (b) (c)

Figure 2-3: (a): the position of the camera (red) and of the facade (white) in the modeled environment; (b): a color image showing low effective resolution at the left of the image due to projective distortion; (c): the corresponding Obliqueness Mask (OM), greyscale.

Each pixel of the CM has thus the value given by:

$$CM^{\tau}_{i,j} = \frac{\sum_{x=-k}^{x=+k} \sum_{y=-k}^{y=+k} Y^{\tau}_{i+x,j+y} Y^{CT}_{i+x,j+y}}{\left[\sum_{x=-k}^{x=+k} \sum_{y=-k}^{y=+k} Y^{\tau}_{i+x,j+y}\right] \left[\sum_{x=-k}^{x=+k} \sum_{y=-k}^{y=+k} Y^{CT}_{i+x,j+y}\right]} \tag{2.3}$$

where $Y^{\tau}_{i,j}$ stands for the luminance value of the pixel (i,j) in the image τ and $Y^{CT}_{i,j}$ for the luminance value of the pixel (i,j) in the Consensus Texture. The parameter k defines the size of the correlation window around the point (we use $k = 8$).

The result is then low-pass filtered to keep only real occlusions such as trees or cars, and avoid bad correlation values close to the boundaries of objects such as windows and doors, which are due to small residual camera errors. As shown in Figure 2-4, the Correlation Mask retains well the occlusions and this information used in the weighted average scheme could also be applied for other purposes.

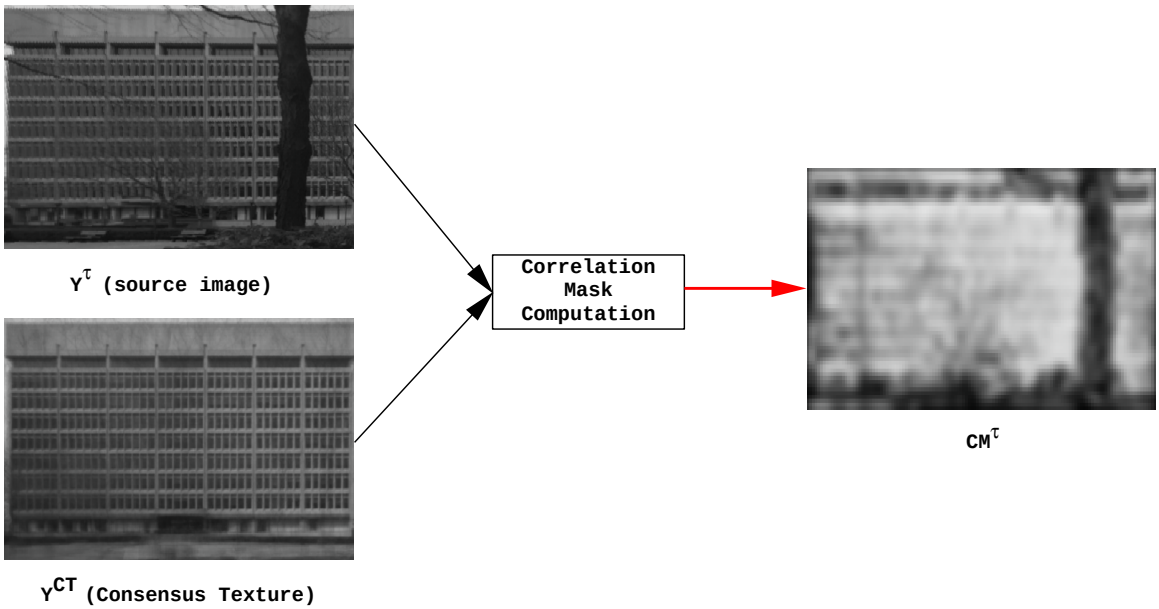


Figure 2-4: Correlation Mask computation, using the original warped image and the current Consensus Texture.

Weighted Average Scheme

As indicated above, the median-based technique introduces speckle noise whose removal is difficult without sacrificing the sharpness of the texture. We thus use a new weighted average process that takes advantage of all available information to remove occlusion and projective errors. This scheme produces much smoother results because it is based simply on the weighted average of all available images. Each pixel of the Consensus Texture CT takes the following value:

$$\left(x_{i,j}^{CT}, y_{i,j}^{CT}, Y_{i,j}^{CT}\right) = \frac{\sum_\tau EM_{i,j}^\tau \cdot OM_{i,j}^\tau \cdot CM_{i,j}^\tau (x_{i,j}^\tau, y_{i,j}^\tau, Y_{i,j}^\tau)}{\sum_\tau EM_{i,j}^\tau \cdot OM_{i,j}^\tau \cdot CM_{i,j}^\tau} \tag{2.4}$$

In this equation, $EM_{i,j}^\tau$ designates the value of the Environment Mask for the pixel (i,j) in the image τ, $OM_{i,j}^\tau$ is the Obliqueness Mask, and $CM_{i,j}^\tau$ the Correlation Mask. As usual with the CIE convention, x,y,Y stand for, respectively, the chromaticity x and y and the luminance Y (superscripts designate either the Consensus Texture [CT] or the image τ).

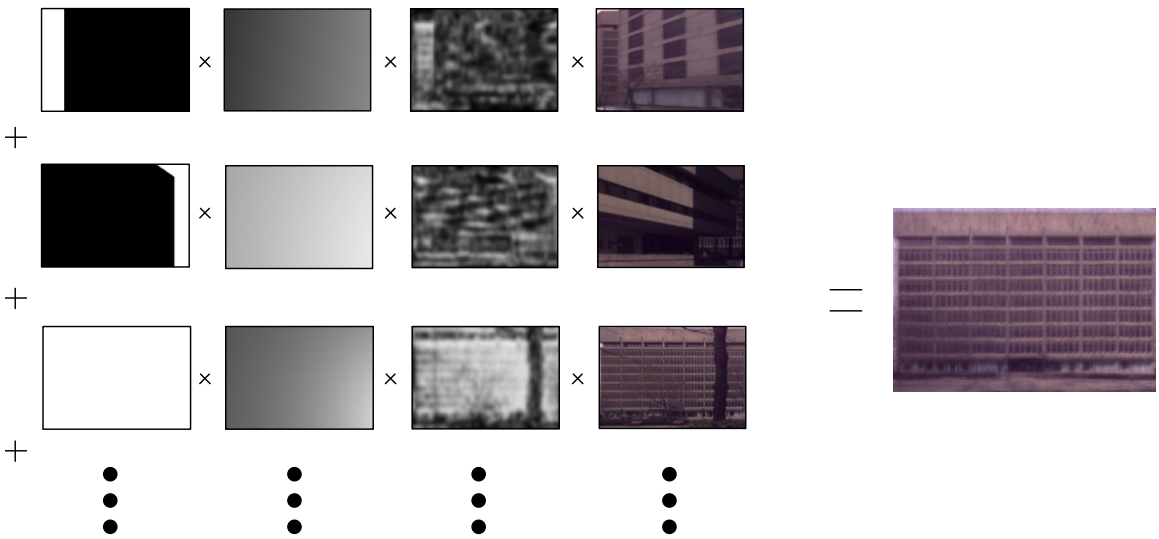


Figure 2-5: Examples for the weighted average scheme

2.2.3 Deblurring Loop

Deblurring the Consensus Texture

The previous scheme produces a good Consensus Texture with few occlusion or illumination effects. However, some blur usually persists, due to residual errors in camera pose after the refinement process. We present here a technique to eliminate these errors.

This optimization process uses a technique similar to the warp-based mosaicing optimization described in [Sze96], that optimizes a correlation function between two images. The principle is to re-warp each source-image onto the Consensus Texture. In general, for any point projected on an image by perspective projection, we can write in homogeneous coordinates [Fau93]:

$$\tilde{\mathbf{m}} \cong [\mathbf{AR}, \mathbf{AT}]\, \tilde{\mathbf{M}} \tag{2.5}$$

where $\mathbf{A}$ is the 3x3 internal camera parameter matrix, $\mathbf{R}$ is the 3x3 rotation matrix and $\mathbf{T}$ is the 3x1 translation matrix (the matrices $\mathbf{R}$ and $\mathbf{T}$ are referred to as external camera parameters).

The two images were taken from a fixed optical center (but with different orientations),

Equation 2.5 reduces to [Sze96, Har97]:

$$\tilde{\mathbf{m}} \cong \mathbf{ARM} \tag{2.6}$$

The relation between 2 corresponding pixels in 2 different images is thus given by

$$\tilde{\mathbf{m_2}} \cong \mathbf{AR_2R_1^{-1}A^{-1}}\tilde{\mathbf{m_1}} = \mathbf{P}\tilde{\mathbf{m_1}} \tag{2.7}$$

where the matrix $\mathbf{P}$ is called the 2-D projective transformation. Since Equation 2.7 is valid up to a scale factor, 8 parameters are needed to describe the matrix $\mathbf{P}$, therefore called the *8-parameter warp* (in practice we set $\det(\mathbf{P})$=1).

In our case, for each node, we want to compute a refined warping matrix that re-warps a given source onto the Consensus Texture. Starting from the identity matrix, the basic idea is to compute a warp that minimizes the sum-of-squared differences in luminance [Sze94]:

$$E = \sum_{u,v} \left[Y_{u,v}^{CT} - Y_{u\prime,v\prime}^{\tau}\right]^2 \tag{2.8}$$

where

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} u\prime \\ v\prime \\ 1 \end{bmatrix} \tag{2.9}$$

However, we introduce in this sum some higher level information derived from the previous computations. It is natural to give more weight in the evolution to a pixel that has more weight in the averaging algorithm. Thus, instead of using the straightforward sum-of-squared differences, we minimize the weighted sum-of-squared differences given by the equation:

$$E = \sum_{u,v} OM_{u\prime,v\prime}^{\tau} \cdot EM_{u\prime,v\prime}^{\tau} \cdot CM_{u\prime,v\prime}^{\tau} \left[Y_{u,v}^{CT} - Y_{u\prime,v\prime}^{\tau}\right]^2 \tag{2.10}$$

We can draw a parallel between this weight embedded in the minimization and the usual standard deviation or measurement error introduced generally in any statistical model. This global weight $OM_{u\prime,v\prime}^{\tau} \cdot EM_{u\prime,v\prime}^{\tau} \cdot CM_{u\prime,v\prime}^{\tau}$ gives the degree of confidence we have in any

particular pixel.

We consider single terms of the previous sum (Equation 2.10):

$$e_{u,v}^2 = OM_{u',v'}^{\tau} \cdot EM_{u',v'}^{\tau} \cdot CM_{u',v'}^{\tau} \left[Y_{u,v}^{CT} - Y_{u',v'}^{\tau}\right]^2$$

The overall gradient with respect to $\mathbf{P}$ remains [Pa92]:

$$\mathbf{G} = \sum_{u,v} e_{u,v} \frac{\partial e_{u,v}}{\partial \mathbf{P}}$$

and similarly, the (approximate) Hessian is:

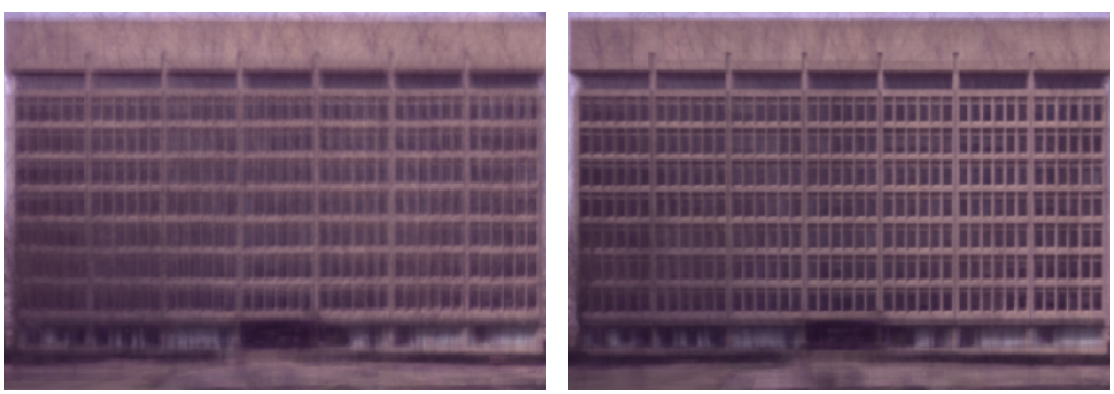
$$\mathbf{h} = \sum_{u,v} \frac{\partial e_{u,v}}{\partial \mathbf{P}} \left(\frac{\partial e_{u,v}}{\partial \mathbf{P}}\right)^T$$

The optimization increments the value of $\mathbf{P}$ by:

$$\Delta \mathbf{P} = -(\mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{G}$$

where λ is reduced to 0 as the procedure converges (see [Pa92] for further details).

Figure 2-6 shows the result of this deblurring process. Note the blur on the boundaries of the windows in the first image and the improvement in the second image. The result is clearly sharper and although computed from many different images, the delimitations of most elements are straight and sharp.



(a) (b)

Figure 2-6: (a): the initial Consensus Texture without deblurring (b): the Consensus Texture after deblurring.

Loop

The previous deblurring method is embedded in a loop that iteratively computes the Correlation Mask and deblurs the Consensus Texture through the updating of the warps. Since the Environment Mask and Obliqueness Mask are based only on geometric considerations, they are computed at the first stage and never updated. (Remember that the updating of the warping values updates only the homography for one node and thus the vector $V_{i,j}$ remains unchanged for each pixel.)

Figure 2-7 summarizes the whole process. As before, CM stands for Correlation Mask, EM for Environment Mask, OM for Obliqueness Mask and CT for Consensus Texture.

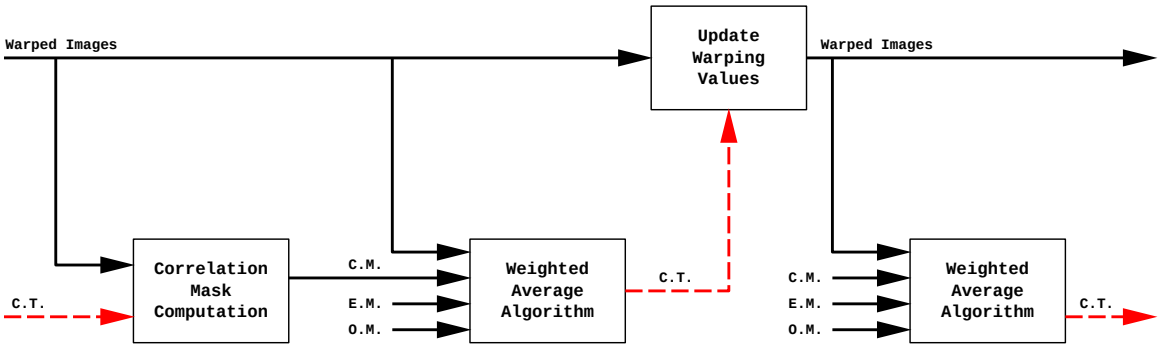


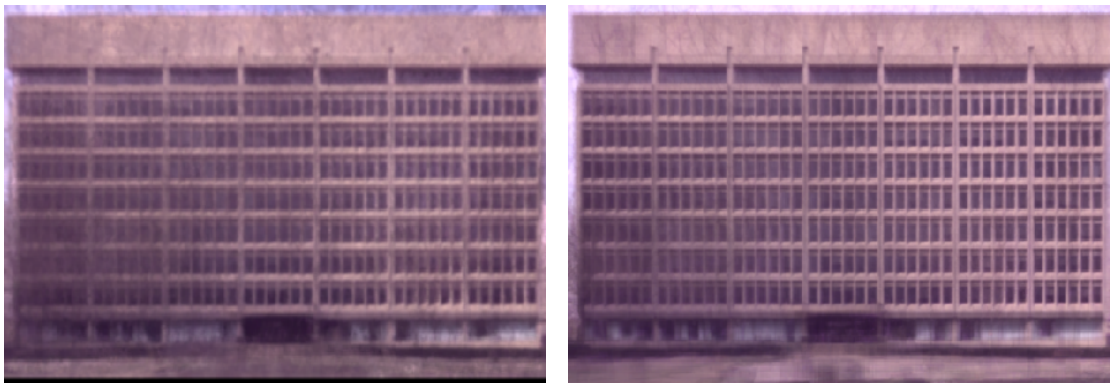
Figure 2-7: The main loop used to iteratively deblur the Consensus Texture.

Convergence is ensured by computing differences between two successive consensus textures at the output of the loop. Once the normalized difference is lower than a certain threshold, the loop is stopped. In practice, convergence is ensured after two iterations, a number that has been kept to avoid intense and long computations.

2.3 Results

The scheme presented here provides a smoother and more reliable result than the median-based technique, which was previously applied [Coo98]. Figure 2-8 demonstrates the difference between the two techniques. Note the significant improvement in the quality of the texture with our algorithm, both in the smoothness of the texture and in the removal of blur. The straight lines close to the windows are clearly sharper.

The algorithm removes most occlusion and illumination variations by introducing high-level (Obliqueness Mask, Environment Mask) as well as low-level (Correlation Mask) in-



(a) (b)

Figure 2-8: (a): result with median algorithm (b): result with improved algorithm

formation. Furthermore, the information computed in this process may be reused in any downstream process as it provides clues about where occlusions are and how reliable the pixels are.

Figures 2-9 and 2-10 show the results of the algorithm on two different facades. The thumbnails are the rectified images used in the algorithm to compute the Consensus Texture. Note the important differences in lighting and the strong occlusions that the algorithm faces. The algorithm eliminates occlusions as well as illumination variations and the luminance is reasonably approximated on the surface. Eventually, most of the changes in luminance across the views have been removed (shadows, reflection effects). Figure 2-11 shows a general view of the resulting textured model. Roofs have been added procedurally to give more realism [CT99].

2.4 Future Work

The scheme presented here has proven to be robust, and able to eliminate most occlusion. Furthermore, the deblurring procedure gives good results and significantly improves the sharpness of the texture. However, some blur does persist, due to non-planar elements in the facade (windows, doors, etc). The following step, presented in the next chapter, consists of augmenting this process with a Self-occlusion Mask whose values are computed after a depth computation-regularization process and a straightforward ray-cast projection.

Due to the nature of the Correlation Mask (which takes into account a small neighborhood of each pixel), only occlusions larger than the correlation window are detected. The mask

detects, with lower confidence, smaller occlusions such as branches or foliage. Although the averaging algorithm tends to eliminate them, some residuals do persist. We tried a PDE's technique[1] described in [TD00] to diffuse color in the Consensus Texture without blurring it. Unfortunately, it did not work as well as we expected: we need higher-level information to eliminate residual occlusion. One idea is to use the edges projected on the facade and to classify them according to the vanishing points detected through another process [AT00]. Due to the regularity of the boundaries on a building, the edges that do not belong to a vanishing direction are likely to be part of occlusions. By expanding them as black values (using a factor of expansion related to the neighborhood in the Correlation Mask), we could make use of this information to remove residual occlusions.

[1]Many thanks to D. Tschumperlé and R. Deriche for their experimentation on our images.

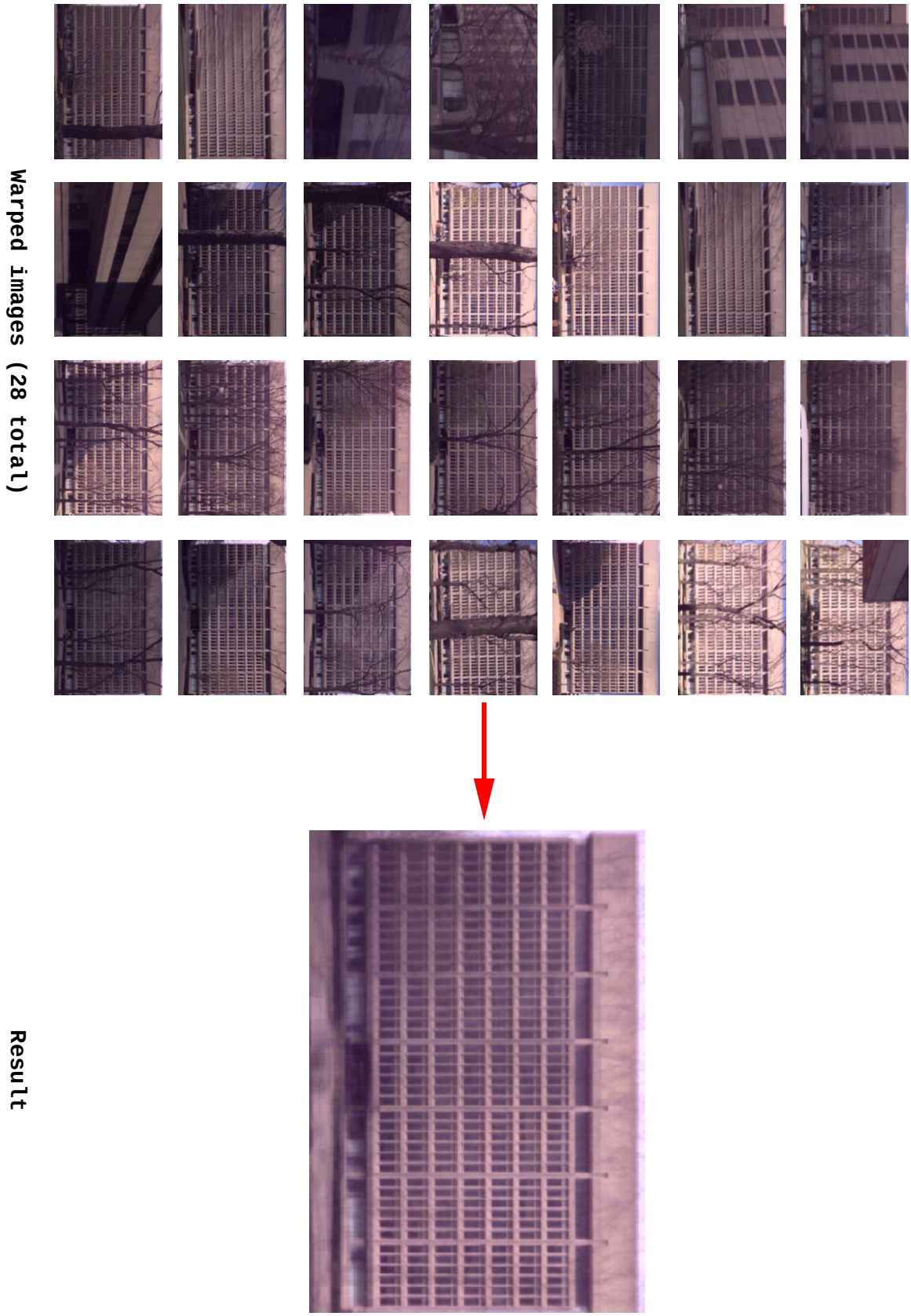


Figure 2-9: Warped images and result after the process of the algorithm for one facade.

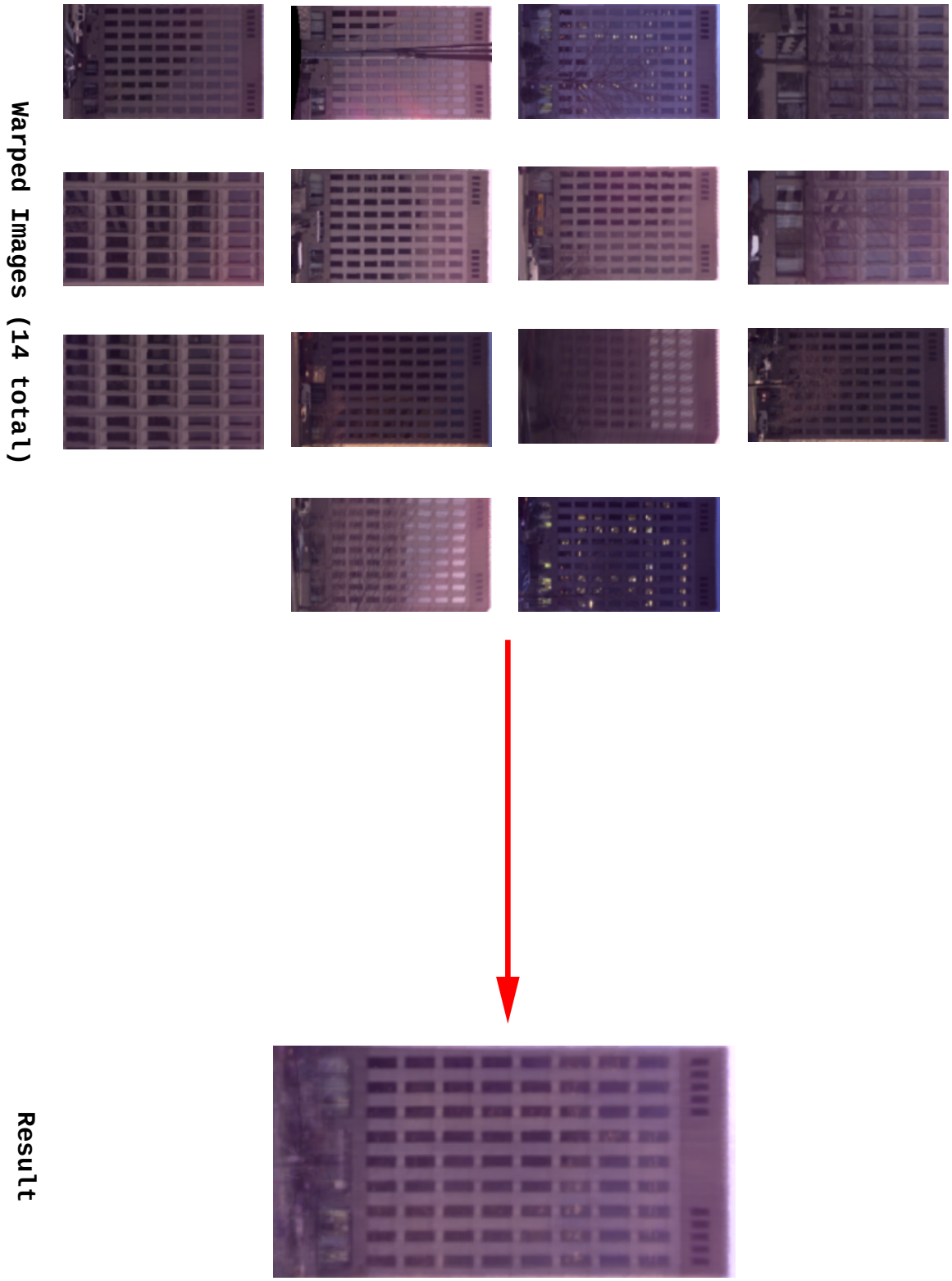


Figure 2-10: Warped images and result after the process of the algorithm for another facade.

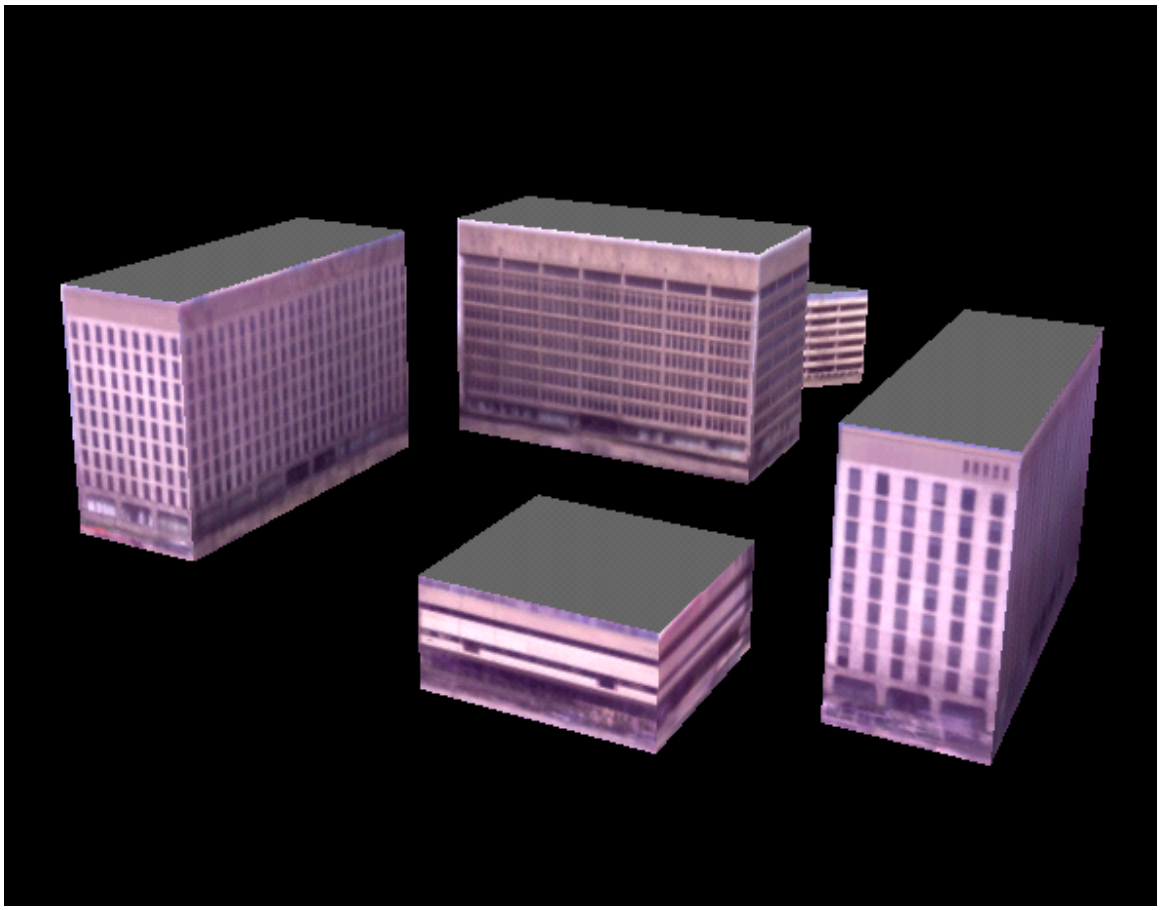


Figure 2-11: The whole textured model, with procedural roof polygons.

Chapter 3

Relief Estimation

3.1 Related Work

Relief estimation from a set of images has been widely researched. Several techniques have been used. Some are based on the retrieval of shape from information on reflectance properties (Section 3.1.1). Other use epipolar constraints to search for correspondences, either directly on the images (Section 3.1.2) or between features (Section 3.1.4). The latest methods try to integrate model-based constraints in order to take advantage of higher level data or to use a priori information (Section 3.1.5).

3.1.1 Shape from Shading

Shape from shading methods estimate the shape of an object by analyzing the light coming from it. In our case, these techniques are not immediately applicable since we do not know the albedo of the material a priori, which is one of the hypothesis of these techniques [Hor86]. Furthermore, it seems hard to use them in our case where illumination variations are so strong across the images, and where the *conditions of lighting and shadows* are not known. Also, these techniques are very sensitive to the position of the light source, which is unknown in our system (due to clouds and occlusion).

3.1.2 2D Correlation-based Methods

With two images and camera information, it is possible to get a 3D position from 2 corresponding points in the images. Unfortunately, the problem of automatically finding corre-

spondences between images is one of the hardest that Computer Vision has tried to solve for years. Epipolar constraints ([Fau93]) enable us to reduce the search for correspondences from 2D to 1D between a stereo pair of images. This led to the development of correlation-based (or area-based) methods. Roughly, one searches for points to match by maximizing the correlation of small windows around the points in both images. In these algorithms the choice of the size of the windows is critical and is the result of the tradeoff between criteria such as desired smoothness and handling of perspective distortion. Many refinements have been developed to make this method consistent with ordering constraints or piecewise continuity. Other relaxation techniques make use of Bayesian [SS98] or PDE processes [ADSW00] to propagate constraints on the results. This approach, although the matching process remains tedious due to occlusions or shadows, is often used as the first step of other more complex methods.

3.1.3 3D Methods

Some authors try to work directly in 3D. [ZK98] works for instance with voxels to iteratively get a cloud of 3D points (obtained initially through a regular window-based correlation on luminance values), using maximum likelihood method. Although it seems very promising and does detect some occlusions, this method suffers from being discrete (a mesh must be generated at the end of the process) and computationally intense.

[FK96] uses a PDE to express the stereo problem. This method has the great advantage that it works in the continuous case, avoiding the tricky generation of the mesh from a set of points. Unfortunately, this method when tried in the CSP proved to be too sensitive to noise, especially to camera pose errors that lead to misregistration [Amr98]. Furthermore, it starts from a 3D structure (like a cube) and the generated surface often collapses. Finally, it is difficult to embed in the same expression higher level information to constrain the evolution.

3.1.4 Feature-based Methods

In order to make the depth recovery process easier, edges or points correspondences are sought in images (see for instance [Gri85]). These methods are not used alone anymore (no dense map is generated and the correspondence problem remains), but they are used to enforce constraints on correlation-based techniques. Other more recent very promising

approaches are centered on plane surface extraction, using both photometric and geometric (epipolar) constraints [VT95]. However, in these methods, the correspondences between regions in two images are assumed, which avoids the problem of occlusion and does not comply with our requirement that the system be automated. Furthermore, their work uses only two images as in the usual stereo problem whereas we want to use all available information.

3.1.5 Model-based Approaches

More recent approaches make full use of model-constraints to constrain the stereo-problem. They can, thus, use higher-level information.In the project Realise, for instance, the user can interactively draw some edges, which enables the system to obtain the geometry of the cameras. In a more automatic process a team at INRIA uses parallel lines to get the geometry of the N cameras [FLR+95]. Unfortunately, in this case, which is the only one that fits our requirement, the aerial images are close in time so there are very few illumination variations and since they are aerial taken under good weather conditions, there is little occlusion. Finally, the scene to reconstruct is quasi fronto-parallel since the images are nadir views (thus there are few perspective distortions).
[Deb96] proposes another interesting approach where the reconstruction process is constrained through relations between blocks in the urban scene. The user-intervention is however very restrictive and therefore limits the use of the system to small datasets.

3.1.6 Mesh Generation

The approach developed by Fua and Leclerc appears to be the most promising for our objectives. Their mesh can indeed handle image-based correlation information and in the same minimization process it is easy to include geometric constraints, derived for example from correspondences found between edges as explained in Section 3.1.4. The main characteristics and advantages of this technique are

- It is based on a minimization function in which geometric constraints can be added (no need for another drastically different method).
- It can handle any number of images and features.
- It can handle self-occlusion during the reconstruction process.

- *It starts from a plane*, which is better suited to our case and thus is much less sensitive to noise that would tend to collapse another surface as in the PDE approach.
- It can also include some shape from shading clues, information that can be added afterwards.

In Section 3.3, I will briefly review some of the important aspects of this mesh and present early results obtained through this method.

3.2 Overview of the General Algorithm

I believe that any of the methods proposed before is not accurate enough to give a good representation of a building, given our data. Therefore, we designed a very general scheme whose main outline is as follows:

- First, some geometric constraints are defined to constrain the depth generation as well as to improve the surface representation.
- Then, a 3D mesh is used to model the relief of the surface. Geometric constraints are embedded in it and it uses all available information from the source images.
- The next step consists of making a polygonal approximation of the surface. This type of representation suits well the geometry of the buildings. This regularization step is necessary because the surface generation is too noisy to be used directly.
- Finally, a Self-occlusion Mask is computed for each source image. This new mask is introduced into the texture estimation, iteratively, to refine the Consensus Texture.

Figure 3-1 reviews these different steps. The texture estimation has been described in the previous chapter. The other steps will be described in further details in the next sections. Be aware, however that, despite the generality of the flowchart, some simplifications have been made to enable an end-to-end process.

3.3 3D Mesh

The 3D mesh used in our project is based on Fua and Leclerc's algorithm [FL94]. It starts out with a plane surface and evolves by iteratively minimizing a objective function through a

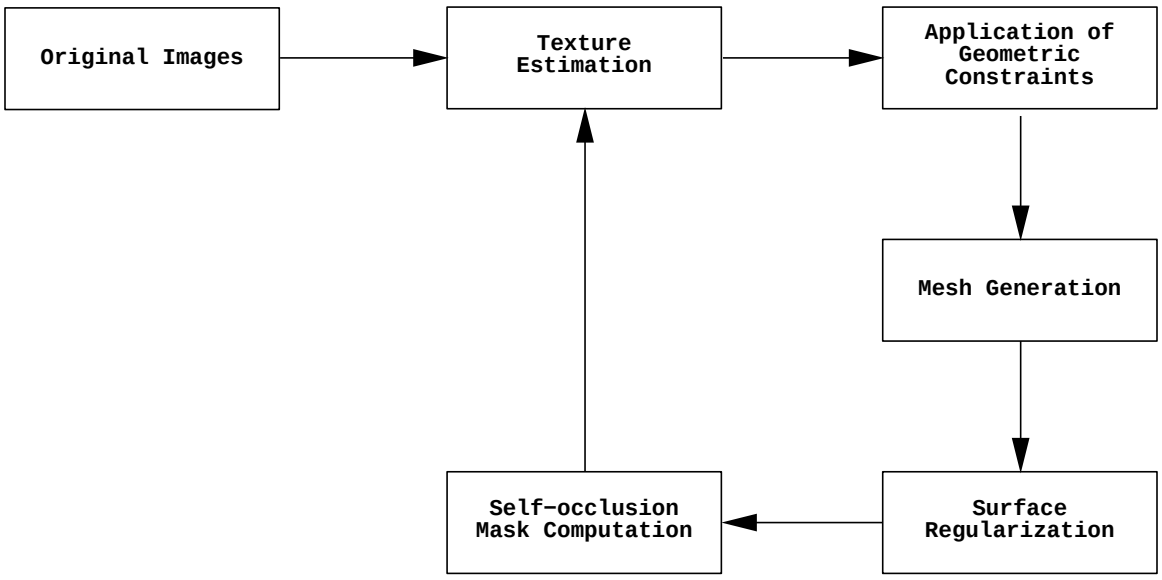


Figure 3-1: Flowchart of the general algorithm with embedded surface extraction.

"snake-like" optimization process. In the objective function, we introduce image-based and geometry-based information. A smoothness term controls how much the surface deviates from a plane and forces the convergence of the minimization.

3.3.1 Minimization of an Objective Function

As already observed, the information from a single source seems to be insufficient to obtain an accurate shape. Fua and Brechbuhler had already studied the use of constraint snakes in imagery [FB96] and Fua alone presents a model-based optimization based on the generalization of these snakes and on the introduction of user constraints [Fua97]. In general, mesh representations for 3D modeling have been widely searched especially in medical imagery (see, among others at INRIA EPIDAURE team [Del94]). However, Fua and and Leclerc ([FL95, FL94]) are the only ones to present a unified framework for 3D shape reconstruction that allows, in a mesh generation process, the combination of image-based and geometry-based information. I will briefly recall the main principles of this technique (please refer to [FL94] for further details).

The principle is to "deform a 3D representation of the surface as to minimize an objective function" [FL94]. The surface $\mathcal{S}$ is represented by a hexagonally connected set of vertices organized into triangular elements called facets. The fact that the mesh is hexagonal will be explained in Section 3.3.2. The objective function used in our application can be described

by Equation 3.1 (compared to Fua and Leclerc's approach, we do not yet use either the shading term nor the silhouette component).

Their approach seeks to minimize:

$$\mathcal{E}(\mathcal{S}) = \lambda_D \mathcal{E}_D(\mathcal{S}) + \lambda_{ST} \mathcal{E}_{ST}(\mathcal{S}) + \lambda_G \mathcal{E}_G(\mathcal{S}) \tag{3.1}$$

In this equation, $\mathcal{E}_D(\mathcal{S})$ refers to a smoothness term that controls how much the surface deviates from a plane. In our case, when dealing with mostly flat surfaces, this parameter is essential and prevents the mesh from collapsing into an undesirable state. $\mathcal{E}_{ST}(\mathcal{S})$ refers to the "stereo" component, described in Section 3.3.3. Finally, the last term $\mathcal{E}_G(\mathcal{S})$ introduces the geometric constraints added in the general scheme. The λ parameters control how much influence stereo, smoothness or geometric constraints have in the minimization. They are chosen at the beginning. We choose to give the same importance to geometric constraints and stereo evolution, the smoothness component remains the most important one. On a scale so that they sum to 1, we would get $\lambda_D = 0.4$, $\lambda_{ST} = 0.3$ and $\lambda_G = 0.3$.
The global function $\mathcal{E}(\mathcal{S})$ can then be minimized through a "snake-like" optimization technique because the $\mathcal{E}_D(\mathcal{S})$ has a quadratic expression: $1/2\mathcal{S}^t K_{\mathcal{S}} \mathcal{S}$ where $K_{\mathcal{S}}$ is a sparse stiffness matrix. The parameters of this optimization are the coordinates of the vertices of the mesh. Roughly, we embed the curve in a viscous medium and iteratively solve the dynamics equation:

$$\frac{\partial \mathcal{E}}{\partial \mathcal{S}} + \alpha \frac{d\mathcal{S}}{dt} = 0 \tag{3.2}$$

α is the viscosity in this expression. Due to the quadratic expression of $\mathcal{E}_D(\mathcal{S})$, its derivatives are linear with respect to S and can be written as:

$$\frac{\partial \mathcal{E}_D}{\partial \mathcal{S}} = K_{\mathcal{S}} \mathcal{S} \tag{3.3}$$

Equation 3.2 can therefore be discretized as:

$$\lambda_{\mathcal{S}} K_{\mathcal{S}} \mathcal{S}_t + \alpha(\mathcal{S}_t - \mathcal{S}_{t-1}) = -\lambda_{ST} \frac{\partial \mathcal{E}_{ST}}{\partial \mathcal{S}}\bigg|_{\mathcal{S}_{t-1}} - \lambda_G \frac{\partial \mathcal{E}_G}{\partial \mathcal{S}}\bigg|_{\mathcal{S}_{t-1}} \tag{3.4}$$

The derivatives of $\mathcal{E}_D(\mathcal{S})$ are decoupled with respect to the coordinates x,y and z and Equation 3.4 can be rewritten as a set of three differential equations in x,y and z. This set of equations can then be computed using LU decomposition and back-substitution. The optimization is iterative and goes on as long as the total energy decreases. If the total energy increases, then the previous result is kept, α is increased in order to decrease the step size and the iteration continues. As far as implementation is concerned, we use the heuristics mentioned by Fua and Leclerc [FL94]. We first fix x and y and let z evolve. The last step lets all three coordinates vary. We also use several levels of mesh size, gradually reducing the size, starting from a coarse mesh to finally obtain a fine result, while speeding the computation.

3.3.2 Smoothness Component

This component measures the deviation of the mesh from a plane. Its general quadratic expression makes it act as a regularization term, essential to forcing the convergence of the algorithm. Using a regular mesh can be seen as a restriction but it enables us to use a finite-difference scheme. In the case of a hexagonally connected mesh, the smoothness term can be rewritten as [Fua97]:

$$\begin{aligned}
\mathcal{E}_D(\mathcal{S}) &= \frac{1}{2}(X^T K X + Y^T K Y + Z^T K Z) \\
&= \sum_{i=1}^{N}\sum_{k=1}^{6}(2x_i - x_{i,k} - x_{i,k+3})^2 + \\
&\quad (2y_i - y_{i,k} - y_{i,k+3})^2 + \\
&\quad (2z_i - z_{i,k} - z_{i,k+3})^2
\end{aligned} \tag{3.5}$$

where X,Y and Z are the vectors of the (x_i, y_i, z_i) coordinates of the vertices, K is the stiffness matrix, which is sparse and banded, and (i, k) denotes the $k - th$ neighbor of vertex i if the six neighbors of i are ordered clockwise.

3.3.3 Stereo Component

The stereo component $\mathcal{E}_{ST}$ is based on the sum of squared differences in normalized luminance from all the images. Each facet of the mesh is sampled at regular intervals and the squared difference for a given sample-point $\mathbf{x}$ is computed using the luminance value of

the projections of $\mathbf{x}$ into the source images. This projection, as stated in [Fua97], is more accurate when dealing with slanted surfaces. The previous result is refined by introducing the occlusions of the surface as mentioned in [Fua97]. The mesh can therefore deal with self-occlusion by discarding pixels that are not seen in a given image.

Taking advantage of the texture computation, we augment the meshing scheme with a measure of the reliability for each facet. First, we use the Occlusion Mask and threshold it to keep only the most reliable pixels. The thresholded mask gives us a binary image that is multiplied by the given luminance image to obtain a greyscale image where all the portions in which we have little confidence are set to 0. This image is then multiplied by the Environment Mask. By discarding, for each given sample-point, each image for which the projection gives a 0 value, we model the reliability of each pixel. The sum of squared differences in luminance from all the images at a given sample-point thus becomes

$$\begin{aligned} \mu(\mathbf{x}) &= \frac{\sum_{\tau=1}^{n_i(\mathbf{x})} v_\tau(\mathbf{x}) Y_\tau(\mathbf{P}_\tau(\mathbf{x}))}{\sum_{\tau=1}^{n_i(\mathbf{x})} v_\tau(\mathbf{x})} \\ \sigma^2(\mathbf{x}) &= \frac{\sum_{\tau=1}^{n_i(\mathbf{x})} v_\tau(\mathbf{x})(Y_\tau(\mathbf{P}_\tau(\mathbf{x})) - \mu(\mathbf{x}))^2}{\sum_{\tau=1}^{n_i(\mathbf{x})} v_\tau(\mathbf{x})} \end{aligned} \tag{3.6}$$

where v_τ is 0 if the sample-point is not visible, 1 otherwise; $n_i(\mathbf{x})$ stands for the number of meaningful images related to the sample-point $\mathbf{x}$ (images that have values different from 0 at the location of the projection of $\mathbf{x}$, see Figure 3-2); $\mathbf{P}_\tau$ is the perspective projection matrix from 3D to 2D such that a point $\mathbf{x}$ in space is projected into a point $\mathbf{u}$ in the image τ with $\mathbf{u} = P_\tau(\mathbf{x})$; and $Y_\tau(\mathbf{u})$ is the luminance value in the image at position $\mathbf{u}$. The global stereo energy is then given by summing this squared difference over all sample points:

$$\mathcal{E}_{ST} = \sum_{k=1}^{n_s} \sigma^2(\mathbf{x}_k) \tag{3.7}$$

where n_s is the total number of sample points.

3.3.4 Geometric Constraints

One major advantage of the mesh developed by Fua and Leclerc is its ability to handle geometric constraints. These constraints are introduced as mesh edge attractors in the general

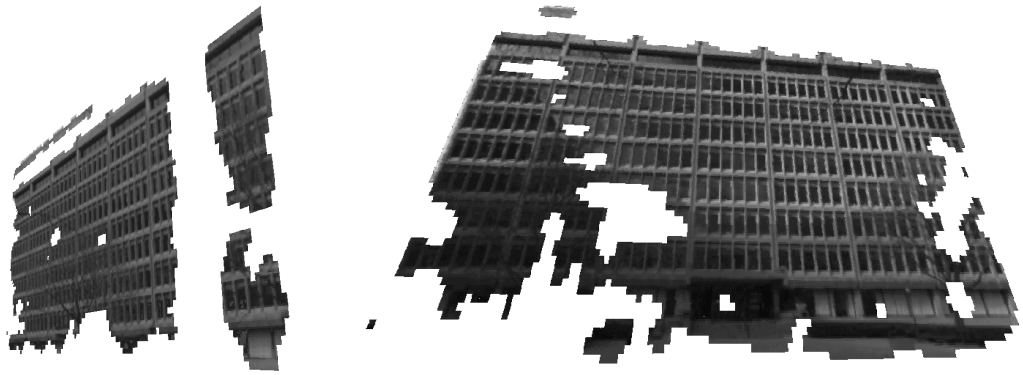


Figure 3-2: Two examples of the generated masked images. The white patches correspond to the pixels that are not reliable enough and have been set to 0 by using the Correlation and Environment Masks.

optimization process. 3D linear features are considered to be collections of attractors. Each attractor attracts the surface and is modeled by an energy term:

$$e_g = 1/(2d_g^2) \tag{3.8}$$

where d_g is the orthogonal distance of the attractor to the closest facet. See [FL94] for a description of a way of computing d_g as well as "the closest facet". The overall Energy is then defined by summing over all attractors.

3.3.5 Results

Due to lack of time, Geometric constraints have not been implemented yet. The results depicted in Figure 3-3 are the straightforward mesh using the stereo component implemented with the Environment Mask and the Correlation Mask as explained n Section 3.3.3. These results are noisy but we can guess the positions of the windows in the general view. I believe that these results could be improved significantly by using constraints defined as explained in Section 3.4. It shows also that the first step consisting in the definition of the constraints is very important to get a good estimation of the relief.

3.4 Defining Constraints with Periodicity Assumptions

A good way to introduce 3D linear features in our scheme would be to compute edge correspondences between images as in the feature-based correlation methods. However,

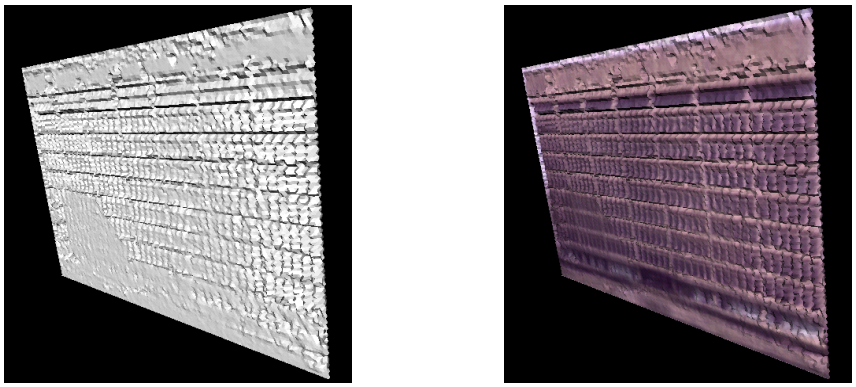


Figure 3-3: The generated mesh: shaded and textured

in order to obtain an end-to-end algorithm and to take advantage of work done in the CSP, we have decided to develop another approach based on repetitive elements found on a building facade. This section describes an algorithm to detect rectangular windows on a facade. These windows can then be placed on the surface of the building as constraints for the generation of the mesh. These are in fact "fake" 3D constraints that model well the reality, however, since the windows lie on the facade. This algorithm is first initialized by a region-growing algorithm working on the Consensus Texture [WH97]. Then, some processing eliminates false positives from the result and fills in the missing windows. Each "filled in" window is then checked against edges detected in the original source images. Figure 3-4 summarizes the outline of the process. Each step is described in further detail in the following sections.

3.4.1 Finding a Pattern

The region-growing algorithm provides us with the coordinates of the center of the windows as well as their height and width (on the consensus texture) [WH97]. However, some false windows are detected and some are missing as well. We first have to find the general pattern corresponding to a window. To find the most general pattern, the algorithm starts out by sorting the windows by height. Then it uses a given tolerance interval (whose size depends on the parameters used in the previous "region-algorithm") and finds out the position on the height scale for which the number of windows whose height is included in this interval tolerance is maximum. This is done through a "sliding rectangular window"; by taking the maximum of this convolution process, we retain the most current height. The process is repeated on the widths of the windows extracted in the first pass. The windows whose

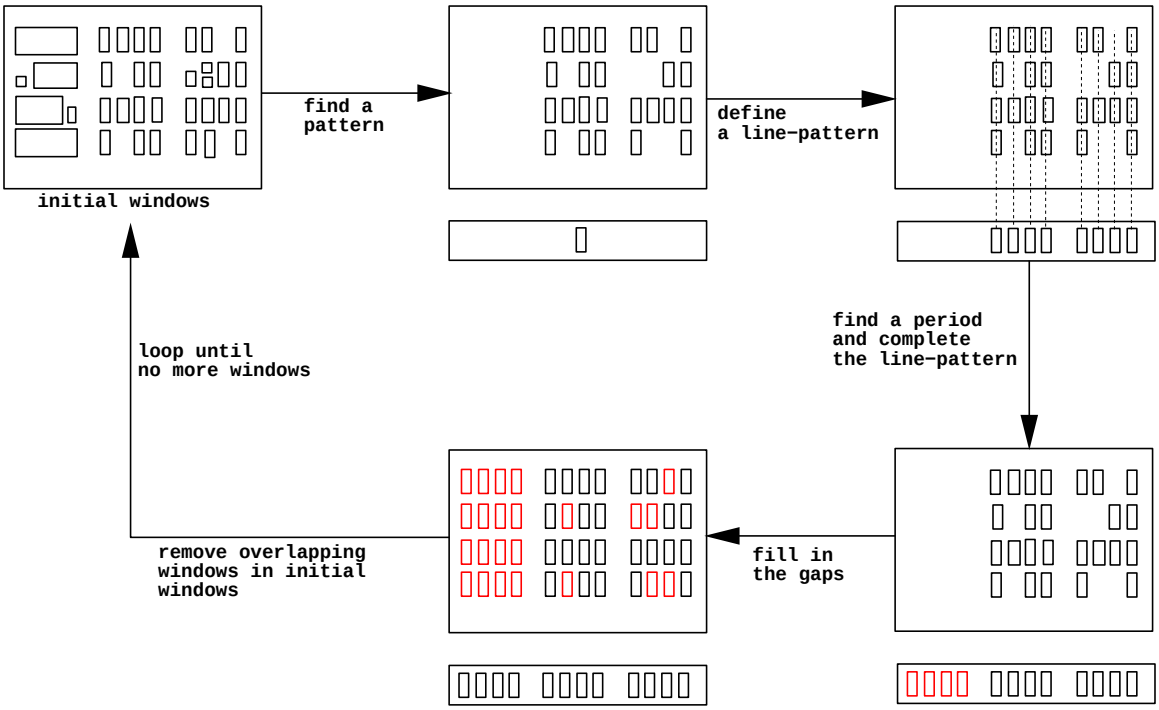


Figure 3-4: Flowchart of the algorithm to detect rectangular windows on a facade

height and width fit in the tolerance box are kept. An average on both height and width enables us to get the general pattern on which the algorithm will work.

3.4.2 Finding a Period

The algorithm works under the assumption that *windows are organized in rows*. A matrix of windows is created and a line-pattern must thus be defined. We take the average of each x-position for the center, in each column.

Once the line-pattern has been created we could use this general information to "fill in" the matrix where there are some missing windows as explained in the Section 3.4.3. Unfortunately, the line-pattern may be incomplete due to persisting blur in the consensus texture that may cause the region-growing algorithm to find bigger windows or no windows at all in all rows of a specific column. Therefore, we must interpolate and/or extrapolate to get a plausible line-pattern.

First, the algorithm finds the most likely period in the line-pattern. To achieve this, we use the general definition of the period t, assuming that the line-pattern contains, after the

initial average process, n windows:

$$\exists d \in \mathbb{R} \quad \forall i \in [1..n-t] \quad x(i+t) - x(i) = d \tag{3.9}$$

For each possible period $p \in [1..n/2]$ we compute the array of differences $a[i] = x(i+p) - x(i)$. The algorithm reuses, then, the same method as in finding the general pattern: the $a[i]$ are sorted and we keep the position of a virtual "sliding window" (size the same as before) for which the number of $a[i]$ whose value is in the tolerance interval is maximum. The number of $a[i]$ in this interval is called R_p. It gives an index of the reliability of the possible period. In order to be able to compare the R_p, we have to rescale them to fit between 0 and 1. We will thus use the value $R_p \leftarrow R_p/(n-p)$. The probable period is then taken as

$$t = pmax \quad \text{where} \quad R_{pmax} = \max_p(R_p) \tag{3.10}$$

We can then threshold the result to avoid noise and uncertain period. Once the period has been found, the line pattern is completed with the windows that were previously missing and extrapolation is made until the limit of the image. Each row is then compared to the line-pattern. As soon as a window is classified as missing, its presence is checked with the algorithm explained in Section 3.4.3.

3.4.3 Filling the Gaps

Although the line pattern may give the representation of a raw "general" row, we must check whether the window that might be missing in a given row should be added. We could use a classic directional gradient image on the consensus texture. However, due to the blur that has persisted in this image, the region-growing algorithm has failed and thus the limit is not straight or clear enough. I decided to go back to the source images. I project all the edges from the source images that do not belong to a black region in the Environment Mask (occlusion due to other buildings) onto the facade and classify them according to their direction (threshold on the slope): vertical or horizontal. All other edges are discarded. For each window, we check the probability that the four edges are present. To achieve this, for each of these edges whose position is known, we make the correlation with all the possible projected edges coming from the source images. This correlation is made as in [Coo98] by "blurring" the edges into rectangles of half-width σ_l where l is the length of the edge (I use

$\sigma_l = 0.1$). The overlap region of the two edges is then computed and normalized with both areas. The correlation function for two edges blurred into two rectangles R_1 and R_2 is thus:

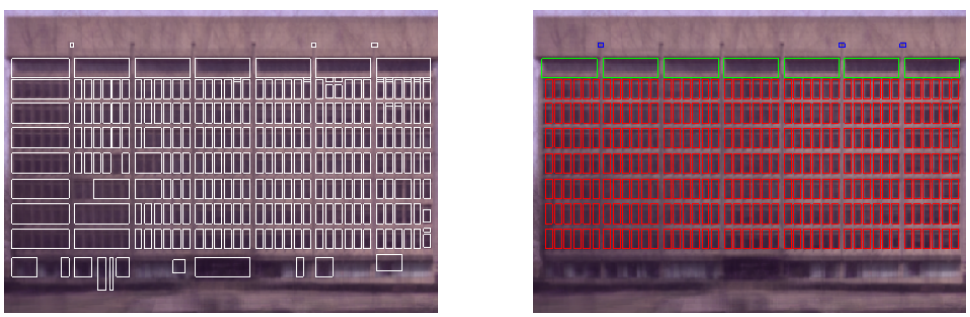
$$C = \frac{\int\int R_1(u,v) \cdot R_2(u,v) dudv}{\left(\int\int R_1(u,v) dudv\right)\left(\int\int R_2(u,v) dudv\right)} \tag{3.11}$$

where $R_i(u,v) = 1$ if $(u,v) \in R_i$ and 0 otherwise.

The correlation functions are summed for each of the four window edges and the result is divided by the number of nodes (or images) available: the result is thus between 0 and 4. We can add the windows whose correlation value is above a certain threshold defined at 0.1 after some experimentation.

3.4.4 Results

Starting from the initial windows, we process once the whole algorithm, remove all the overlapping windows and redo the process so as to find several patterns and to be able to give a good overview of the building. For each pattern, we average through the depth map and find for each window belonging to a certain pattern the average depth. This result will be used afterwards as well to display the building relief and as another embedded step in the texture estimation process. Figure 3-5 displays the initial windows as given by the region-growing algorithm and the result of the whole process for a complete facade.



(a) (b)

Figure 3-5: (a): initial windows as given by the region-growing algorithm (b): result of the whole process

3.5 The Whole Relief Generation Algorithm

Section 3.4 has shown how to generate constraints in order to constrain the mesh evolution. We can thus get a good estimation of the depth for the detected windows. In the following sections, I present the Self-occlusion Mask, a good way of introducing this new information about the non-planar elements on the facade in the texture estimation. I also show how to output the results in order to obtain an interactive modeled environment.

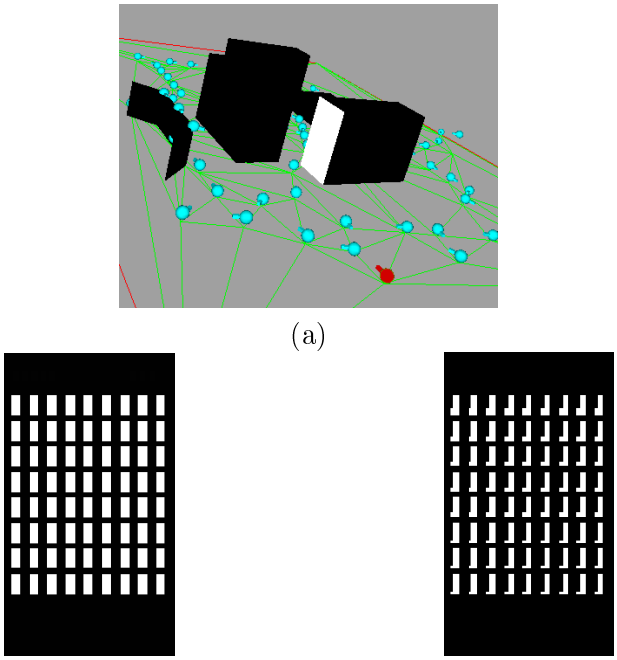
3.5.1 Self-occlusion Mask

The constraint mesh gives a noisy approximation of the surface of the facade. The regularization process averages the depth for each type of window detected with the previous algorithm. Each window, according to its pattern is thus assigned an average depth.
This work gives us a good approximation of the facade with embedded relief. The idea is then to use this information to compute for each node a Self-occlusion Mask. This is, once again, a binary mask whose pixel has value 1 if it is visible from the viewpoint, 0 if not.
This Self-occlusion Mask is computed using the depth generated from the regularized mesh and a general ray-cast algorithm. An example is shown in Figure 3-6. As in the previous chapter, some blur in the texture persists, due in part to some non-planar elements. Thus, this new mask is introduced in the general weighted average algorithm to refine again the consensus texture.

3.5.2 Results

Figure 3-1 summarizes the general process for the depth computation as well as the new loop introduced in the texture estimation. We believe that this scheme is fairly general. In our case, the study made a lot of hypotheses (for instance: mainly planar surface, only repetitive elements) to obtain an end-to-end algorithm. Later on, each specific part can be elaborated with more general techniques.

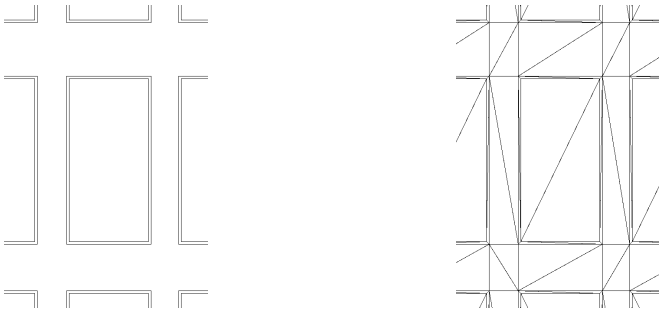
Interactive display is generally a problem when one wants to interact with complex mesh data. In our case, several displays can be used. The first one made of triangle strips, can represent any depth map. Although very general, it is very slow and limits any manipulation. We have found that a constrained Delaunay triangulation can be used on



(a) (b) (c)

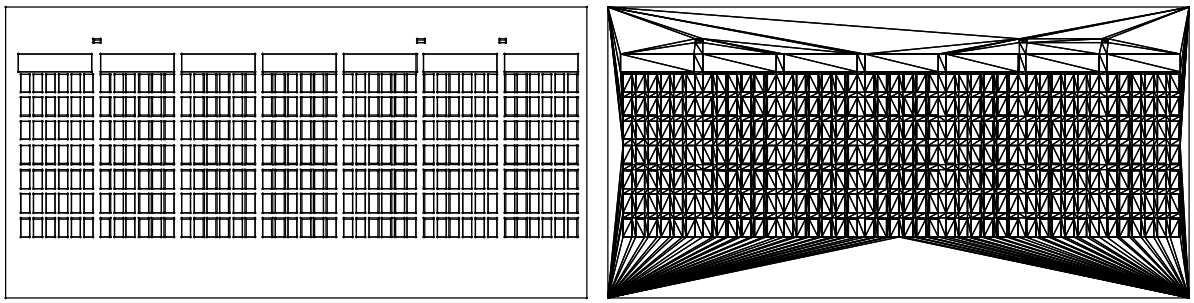
Figure 3-6: (a): the position of the camera (red) and of the facade (white) in the modeled environment; (b): the windows detected in the environment shown in inverted colors (c): the corresponding binary Self-occlusion Mask (SM) shown in inverted colors.

all the edges detected on our facades. Each window is segmented into two rectangles, one inside the other. The separation between the rectangles is very small. All the edges as well as the boundaries of the facade are then triangulated (in 2D) as shown in Figure 3-7. Then, the inner rectangles corresponding to the windows are pushed away according to their depth. This small trick (to define a window as an inner part and an outer part) enables us to represent depth while keeping a 2D texture. Although we could compute for each side of the window a correct texture (each side would be seen as a "facade"), this would be drastically slower. In general, a constrained Delaunay triangulation can be used as soon as we have edges with their depth. We first compute the 2D Delaunay triangulation and then push away the edges according to their depth. It is a good way of representing a polygonal surface with a minimum number of triangles. Figure 3-8 shows an example of the Delaunay traingulation applied to the edges found thanks to the window-grid algorithm.



(a) (b)

Figure 3-7: (a): the inner and outer part of a window; (b): the resulting constrained triangulation on a window.



(a) (b)

Figure 3-8: (a): the edges used for the constrained triangulation; (b): the resulting constrained triangulation.

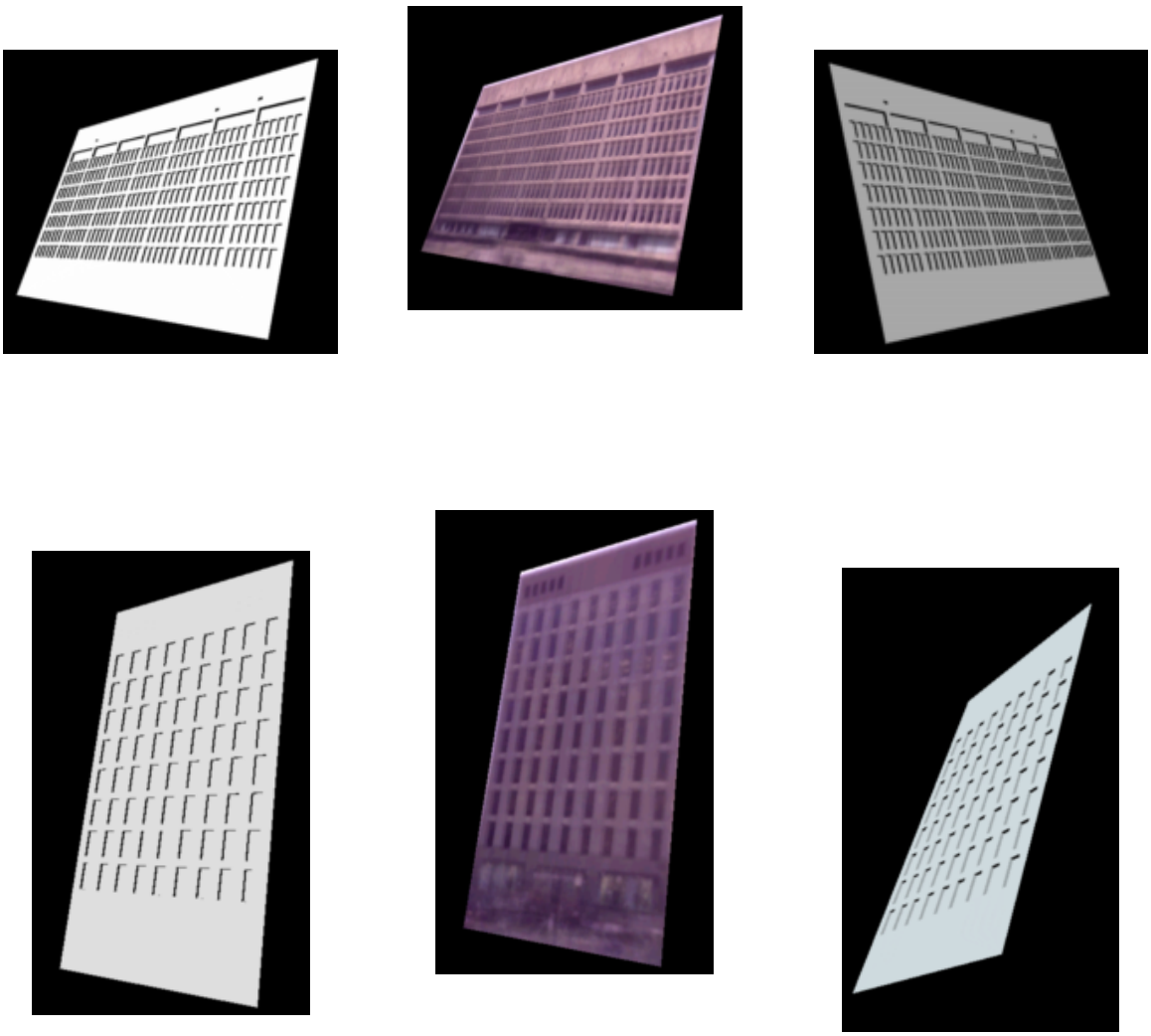


Figure 3-9: Results on two examples: shaded (no texture) with different orientations and lighting directions and textured

Chapter 4

Future Work and Conclusions

4.1 Implementation

The work presented in this thesis has been implemented and tested on a model of 19 facades, using a dataset of about 4000 images. The running process takes about four hours on four MIPS R10000 processors at 250Mhz for the whole textured model without relief estimation. It takes about 12 hours for the general texture and relief estimation algorithm, assuming that the region-growing algorithm has been processed before. The region-growing algorithm itself takes about 10 hours.

I believe that the scheme presented in this thesis is very general and well-suited for any texture and relief extraction method. However, there is clearly some lack of generality in some of the developments proposed here, which is due to the emphasis on an end-to-end process. Further work could significantly improve the results shown in this thesis.

4.2 New Triangulation

The "hexagonal" triangulation is clearly not well-suited to our objectives. First it can not match the boundaries of the building and we must thus use another representation afterwards. Second the edges of the triangles do not necessarily lie on the regular edges detected on the facade. Third, the process is very time-consuming. Finally, the number of triangles is obviously not optimized.

An adaptative triangulation would suit better our purposes. For example, in [LMF97][1],

[1] We are currently in touch with Richard Lengagne in order to adapt his mesh to our challenging data.

the triangulation proposed is very general and can also adapt to the curvatures through the optimization process. The results presented for ground imagery seem promising. This mesh has a lot of advantages: it can adapt itself to changes in curvatures of the ground (or terrain), and can, as the previous method, integrate stereo constraints. A lot of work must be done in order to start with a triangulation whose main edges are defined through the geometric constraints found on the facade (windows or else) and to integrate attractors to model the mesh according to 3D features.

4.3 3D Constraints and Better Surface Approximation

Although the geometric constraints used in the mesh are well suited for an industrial environment, they are too restrictive and suffer clearly from a lack of generality. A good way to produce consistent 3D constraints is to register 3D features among the images as in [Gri85]. Another limitation is the polygonal approximation of the mesh. We need to estimate a polygonal surface from this mesh, which could fit our requirements in our urban area. The model used until now is too simplistic (simple windows) but it is a good start to enforce constraints on the mesh as well as to help the approximation process.

4.4 Improved Texture Estimation through Propagation

The main algorithm for texture estimation is very general and produces good results. However, some blur can persist where the images are too much occluded: we miss information. We could use the repetitive elements to obtain a better texture: we could use an "inpaint" algorithm to extrapolate what the texture could be [BSCB00] or a texture synthesis algorithm as the one stated in [EL99]. We have rough information on how reliable each window is (presence or not, union of the several weights). We could identify areas that are the least reliable, and modify local texture from the vicinity of the most reliable areas of the Consensus Texture.

4.5 Conclusions

We presented in this thesis a general scheme to estimate a Consensus Texture and a Relief for a building facade from a large set of ground images. Our method eliminates most occlusion,

clutter and illumination variations. Furthermore, the mesh used in this work is well suited for any nearly planar surface and so for most building facades. Future investigations in collaboration with Richard Lengagne at EPFL may pursue more general meshing strategies that will fit our requirement even better.

Bibliography

[ADSW00] Luis Alvarez, Rachid Deriche, Javier Sánchez, and Joachim Weickert. Dense disparity map estimation respecting image discontinuities: a PDE and scale-space based approach. Technical Report RR-3874, INRIA, January 2000.

[Amr98] Eric R. Amram. A variational technique for three-dimensional reconstruction of local structure. Master's thesis, MIT, September 1998.

[AT00] Matthew E Antone and Seth Teller. Automatic recovery of relative camera rotations for urban scenes. In *CVPR'00 Conference Proceedings*, pages 282–289, June 2000.

[BB95] S. Becker and V.M. Bove. Semiautomatic 3-d model extraction from uncalibrated 2-d camera views. In *Proc. SPIE Images Synthesis*, pages 447–461, 1995.

[BdCT00] Michael Bosse, Douglas de Couto, and Seth Teller. Eyes of Argus. *GPSworld*, pages 20–30, April 2000.

[BSCB00] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. In *SIGGRAPH'00 Conference Proceedings*, 2000.

[CMT98] Stayan Coorg, Neel Master, and Seth Teller. Acquisition of a large pose-mosaic dataset. In *Proceedings of CVPR*, pages 872–878, 1998.

[Coo98] Satyan Coorg. *Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments*. PhD thesis, Dept. of Electrical Engineering and Computer Science, September 1998.

[CT99] Satyan Coorg and Seth Teller. Extractin textured vertical facades from controlled close-range imagery. In *Conference on Vision and Pattern Recognition*, June 1999.

[Deb96] Paul E. Debevec. *Modeling and Rendering Architecture from Photographs*. PhD thesis, Dept. of Computer Science Berkeley, 1996.

[Del94] Hervé Delingette. Simplex meshes: a general reprentation for 3d shape reconstruction. Technical Report RR-2214, INRIA, March 1994.

[DTM96] P.E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry and image-based approach. In *SIGGRAPH '96 Conference Proceedings*, pages 11–20, August 1996.

[EL99] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV'99*, 1999.

[Fau93] Olivier Faugeras. *Three-Dimensional Computer Vision, a Geometric Viewpoint*. The MIT Press, 1993.

[FB96] P. Fua and C. Brechbuhler. Consistent site modeling: Imposing hard constraints on deformable models. In *IUW 96*, pages 1077–1093, 1996.

[FK96] Olivier Faugeras and Renaud Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. Technical Report RR-3021, INRIA, October 1996.

[FL94] P. Fua and Y.G. Leclerc. Using 3-dimensional meshes to combine image-based and geometry-based constraints. In *ECCV94*, pages B:281–291, 1994.

[FL95] P. Fua and Y.G Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. In *IJCV*, pages 35–56, 1995.

[FLR+95] Olivier Faugeras, Stephane Laveau, Luc Robert, Gabriella Csurka, and Cyril Zeller. 3-D reconstruction of urban scenes from sequences of images. Technical Report RR-2572, INRIA, June 1995.

[Fua97] P. Fua. *RADIUS: Image Understanding for Intelligence Imagery*, chapter Model-Based Optimization: An Approach to Fast, Accurate, and Consistent Site Modeling from Imagery. Morgan Kaufmann, 1997.

[FvDFH90] James D. Foley Foley, Andries van Dam, Steven K Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice*. Addison Wesley, Reading, Massachussets, second edition, 1990.

[Gri85] W.E.L Grimson. Computational experiments with a feature based stereo algorithm. In *PAMI*, pages 17–34, January 1985.

[Har97] Richard Hartley. Self-calibration of stationary cameras. *IJCV*, 22(1):5–23, February 1997.

[Hor86] B.K.P. Horn. *Robot Vision*. MIT Press, 1986.

[LMF97] Richard Lengagne, Olivier Monga, and Pascal Fua. Using differential constraints to reconstruct complex shapes from stereo. In *Conference On Computer Vision and Pattern Recognition*, pages 1081–1086, June 1997.

[Pa92] Press and al. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[Poy96] Charles Poynton. *A Technical Introduction to Digital Video*. John Wiley & Sons, 1996.

[SNW+99] V. Sequeira, K. C. Ng, E. Wolfart, J.M.C Gonalves, and D.C. Hogg". Automated 3d reconstruction of interiors with multiple scan-views. In *Proceedings of SPIE, Electronic Imaging '99, IS&T/SPIE's 11th Annual Symposium*, San Jose Convention Center, San Jose, California, USA, January 1999.

[SS98] Daniel Scharstein and Richard Szeliski. Stereo matching with nonlinear diffusion. *IJCV*, 28(2):155–174, 1998.

[SWI97] Yoichi Sato, M. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH'97 Conference Proceedings*, pages 379 – 387, August 1997.

[Sze94] Richard Szeliski. Image mosaicing for tele reality, applications. Technical report, Digital Equipment Corporation, Cambridge Research Lab, May 1994.

[Sze96] Richard Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(2):22–30, March 1996.

[TD00] D Tschumperle and R Deriche. Restauration d'images vectorielles par edp. In *12ème congrès RFIA*, Paris, February 2000.

[Tel97] Seth Teller. Automatic acquisition of hierarchical, textured 3d geometric models of urban environments: Project plan. In *Proceedings of the Image Understanding Workshop*, 1997.

[VT95] Jean-Marc Vezien and Jean-Philippe Tarel. A generic approach for planar patches stereo reconstruction. Technical Report RR-2507, INRIA, February 1995.

[WH97] X. Wang and A. Hanson. Extracting surface textures and microstructure from mutiple aerial images. In *IEEE CVPR*, 1997.

[ZK98] Charles Zitnick and Takeo Kanade. A volumetric iterative approach to stereo matching and occlusion detection. Technical Report CMU-RI-TR-98-30, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1998.