A Variational Technique for Three-Dimensional Reconstruction of Local Structure

by

Eric Raphaël Amram

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1998

Author .....
Department of Electrical Engineering and Computer Science
November 9, 1998

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

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

A Variational Technique for Three-Dimensional Reconstruction of Local Structure

by

Eric Raphaël Amram

Submitted to the Department of Electrical Engineering and Computer Science
on November 9, 1998, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

The purpose of this thesis is to develop and implement techniques for retrieving depth and complex geometry, starting from near-planar areas like building facades, under solar illumination.

The approach used here consists of analyzing *multiple* color images taken from distinct points of view by a *calibrated* camera and working directly in 3D euclidean space for reconstruction. The final goal is to output a textured 3D polygonal model.

We use a variational approach based on level set methods for computation. The correlation between the camera images reprojected onto an evolving surface is optimized by driving the motion of the surface using the Euler-Lagrange equation of the correlation criterion. This leads to a global optimum for the reconstructed surface.

Thesis Supervisor: Seth Teller
Title: Associate Professor of Computer Science and Engineering

Contents

List of Figures

Chapter 1

Related work

1.1 Foundations of the problem

We will focus on the part of computer vision field called *stereovision* which tackle the question of reconstructing a 3D shape from two or more images taken by calibrated camera(s).

Input data

A *calibrated* camera (we know position, orientation and intrinsic parameters see [Fau93] for definition) takes pictures of a real scene under solar illumination. Several color images display a target area that we suppose 'almost planar', such as a facade of a building.

Our input data will be thus be

- Color images
- Camera calibration information for each image
- a 3D planar quadrilateral, which lies of some facade of interest, or a bounding box which surrounds the region of interest.

Reconstruction purpose

With at least two images, it is possible to triangulate corresponding points to get a 3D point, considering we know the camera position and the correspondence between points. The first condition is one of our hypotheses, but the second is one the problems computer vision has tried to solve for years without success.

Camera calibration provides the epipolar geometry [Fau93] between images, and thus reduces the corresponding search space of an image point from 2D to 1D in the other(s) image(s). However the matching process remains under-constrained, as color and intensity of pixels vary subject to reflective property of the material, change in lighting conditions, shadows, highlights and occlusions due to the 3D nature of the viewed scene.

Attempts to constrain this problem have been either to extract geometric features in the images - therefore simplifying the set of possible matching points - or to use small correlation windows in order to take pixel neighborhood information into account.

A few methods try to work directly in 3D, starting from correlation results, using voxels techniques or variational principles.

Most consider general 3D problem and not only *planar parallax*. This makes the problem harder and leads to a lot of troubles in the reconstruction process. We believe that the plane-based simplification – thanks to other work done in the city project (see [Coo98]) – improve current results on dense reconstruction methods. The interest of this approach has been studied in [Saw94a, Saw94b], [IA96] and [TDM96]. We do not explicitly formulate our evolution with respect to the plane, since our method works for general 3D objects, but this choice restricts our grid space and improve performance, as we will see.

1.2 Feature-based reconstruction

In order to simplify and speed up the matching process, features as corners or edges are extracted and matched between images. This does not provide dense depth in-

formation and suffer moreover from inaccuracy of the features: cylindric shapes give wrong edges correspondences between images for instance.

Despite the simplification, the correspondence problem remains. See (among many others) [BB82, OK85, Gri85]. This approach is not used anymore for dense depth maps retrieval.

1.3 Shape from shading techniques

I tried several techniques during my first semester ([ZTCS94, HB89, Hor90, Sze91, BP92, TS92, LB91, DO92, OD93]). Results were very disappointing and all shape from shading methods provide inaccurate results in term of depth value. Moreover the albedo of the material [Hor86] must be known before, which is not the case here. Eventually all techniques are very sensitive to the given light source direction.

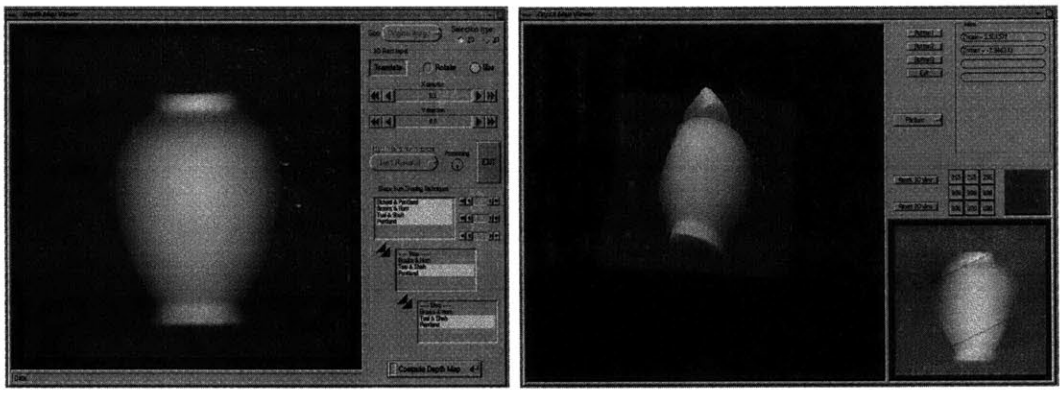

Figure 1-1: Example of shape from shading technique: image and result

1.4 Correlation-based methods

The idea of this approach is to correlate small windows around points to match and maximize the correlation score to find correspondences. Many good implementations with a lot of refinements have been achieved. I would mention as examples [ea93, CM92, Zha97], [DF94] for additional extraction of differential properties, [SS96] for Bayesian estimation technique, [OK93] for use of N cameras.

Due to variations in lighting and to the camera automatic gain control, most of the techniques use linear correlation between windows. Most of them assume that the plane of the correlation window is fronto-parallel, which is often not the case. The matching process is inevitably unreliable in case of changes in geometry, discontinuities, occlusions, highlights, shadows, etc.

Eventually all these methods do not retrieve genuine color and material, they do not deal with highlights, shadows, and it is hard to predict self-occlusion in the reconstruction process. Multiple camera handling is not natural since correlation is deeply pairwise.

I will present some results of this class of methods in the next part.

1.5 3D methods

As we want to retrieve 3D information, working directly is 3D space seems more natural, but it is also more complex since the only measurements we have come from the camera images.

- [Zit96] build clouds of 3D points from *possible* correspondences, and work in 3D to make the cloud consistent and accurate. They use several *aligned* cameras to eliminate false matches. In the end of the process they still have to build a mesh from this set of 3D points. This approach complies badly with our requirements and moreover suffer from being computationally expensive. Finally it does not provide a very dense depth map for the objects, in order to avoid false matches.

- [SD97, Sei97] use voxels and seem to have a very interesting and promising approach. However it still need a lot of texturing and preferably many colors. It realizes some sort of front pushing according to the number of consistent hits from image rays that go through a given voxel. For a uniform planar surface, we would expect as default result a plane, since we are dealing with urban environments. This method will give us unfortunately a bump instead.

- [FK96, FK97] choose a variational approach. This new technique using PDEs

has already proved to be efficient and I took it as a basis. I will develop it in the next parts. Improvements regarding discontinuities handling, color, reflective properties, removal of highlights and synthesis of a texture model getting rid of shadows and highlights still need to be made.

Chapter 2

Preliminary study

2.1 First stage correlation

Before going further into refinements regarding color, material, and light, a first stage comparison is necessary to develop as a basis of comparison. My implementation builds a depthmap, without looking for many refinements but giving a feeling of what to improve.

Dealing with *calibrated* images (modulo some small pose error), the epipolar geometry reduces the search space for correspondences from 2D to 1D (see [Fau93]). we can infer the 3D position of a point by triangulation through the retrieval of disparity between the points of the two images, that is to say the amount of displacement of one point with respect to the other along the epipolar line.

Several options are available to make the depth map consistent: simple correlation, relaxation algorithms like those developed by Marr-Poggio [MP79], Grimson [Gri81, Gri85] or Mayhew-Frisby [PMF85], and dynamic programming algorithms from Baker and Binford [BB82] or Ohta and Kanade [OK85]. The purpose of these algorithms is to diffuse disparity information extracted from one pixel to other pixels within one epipolar line and between epipolar lines.

2.1.1 Rectification

At the beginning, we rectify the images by warping them on the same facade and making the epipolar lines parallel; the scanning phase can then happen on horizontal screen lines (see fig. 2-1).

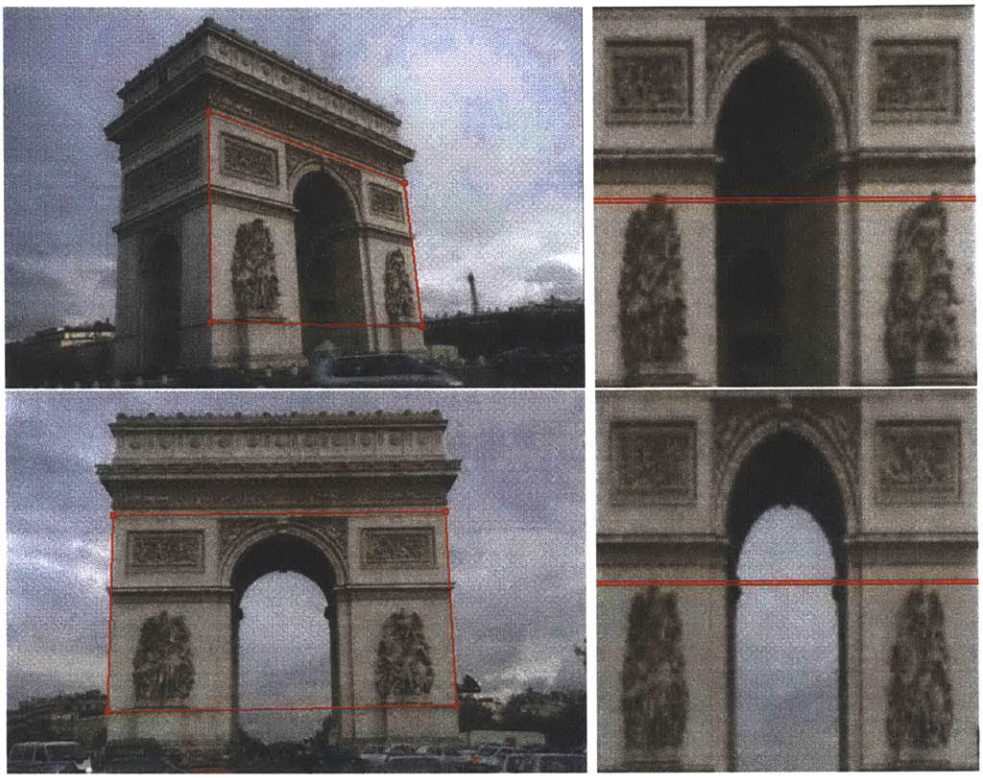

Figure 2-1: Rectification process

2.1.2 Matching process

To compute disparity, we correlate small windows around every point of corresponding image scanlines. I refined the correlation function to add color to linear correlation, which was previously computed on luminance only. This is done by weighting the elements by a function of the distance between the two pixels in the CIE xyY color space, as we will discuss in the next section. It avoids matching, for instance, a red pixel with a green pixel having the same intensity.

Figures 2-2 and 2-3 illustrate the correlation of the two scalines. The scanline of the upper warped image corresponds to the x axis of the correlation image, whereas

the scanline of the lower warped image corresponds to the y axis, oriented downwards.

If the 3D point belongs to the plane, then the two projected pixels of this point are located at the same position on the two scanlines – that is to say on the diagonal of the correlation array – and their disparity is zero. However if the point does not belong to the plane, then its projection on the two images is shifted from one image to another: the disparity is non null and the match occurs in the correlation array at a horizontal (or vertical) distance equal to the disparity.

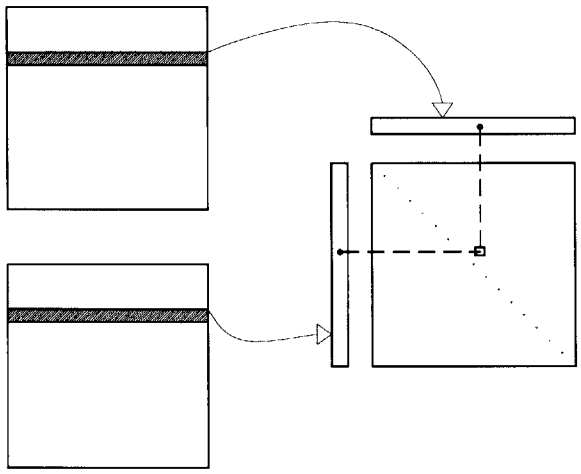

Figure 2-2: Correlation array between two epipolar lines

Therefore finding the crest of highest correlation score in the correlation array provides us with the best matches between pixels of the scanline, in the sense of the correlation criterion.

To enforce consistency, I use the following constraints:

- epipolar constraint
- ordering constraint: corresponding points appear in the same order on epipolar lines, that is, the disparity along an epipolar line is monotonic. This has to be made weaker, since it is not true for general situations like railings in front of an object.
- piecewise continuity

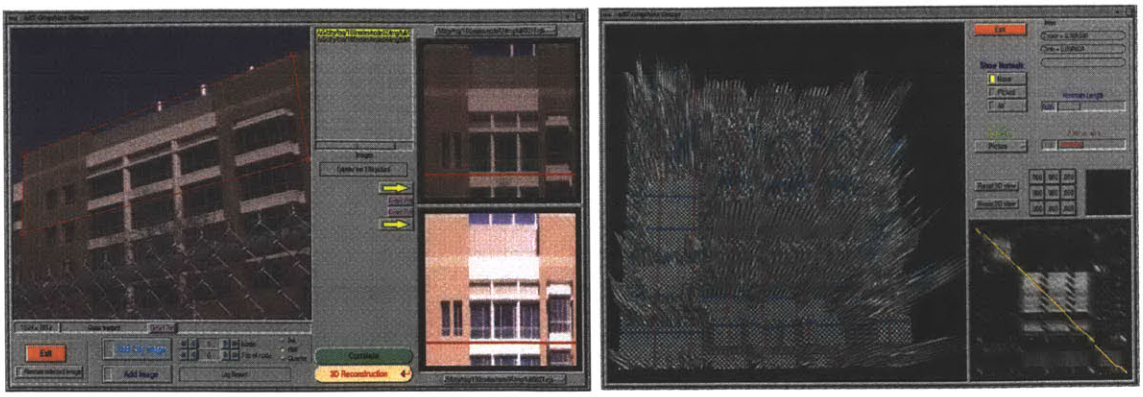

Figure 2-3: Matching pixels between the two scanlines

I designed an $O(N^2)$ algorithm for finding a path along an epipolar line using dynamic programming (brute force method is $O(N^4)$).

We first extract all the local maxima of the correlation array. Figure 2-4 shows the correlation array in grey level, white being the best correlation. The yellow triangles indicate the local maxima. Then we find the best path by dynamic programming, taking advantage of our constraints and the fact that the correlation values $correl(i, j)$ are all positive. The uniqueness constraint implies that we do not have two points of the path on the same horizontal or vertical line; the ordering constraint forces the path to be monotonic.

Let us denote $Opt(i, j)$ as the optimal path that is forced to go through (i, j); then we have

$$Opt(i,j) = \max \begin{cases} correl(i,j) + Opt(i-1, j-1) \\ Opt(i-1, j) \\ Opt(i, j-1) \end{cases}$$

We sweep the array line by line (left to right, up to bottom), computing $Opt(i, j)$ at every point. Finally we backtrack from the max value to get the optimal path. All this is achieved in $O(N^2)$.

2.1.3 Color correlation function

Linear correlation is well suited to grey level images but discards information when applied to color images. Indeed, it may perfectly match a green area with a red one

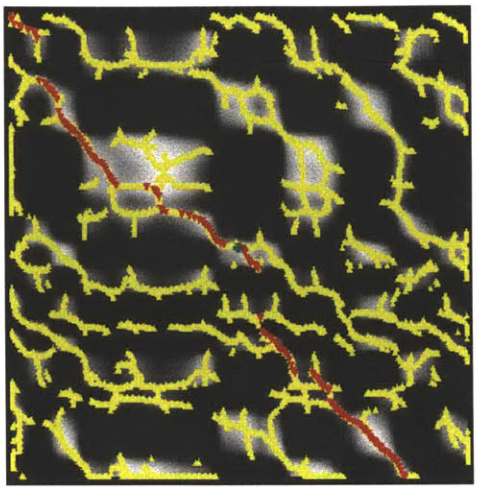

Figure 2-4: Path finding for pixel matching between epipolar lines

as long as they have the same luminance.

To correct this while still keeping the linear feature of the correlation – necessary for invariance to changes in illumination and automatic gain control of cameras – I weighted the elements of the sum by a function depending on the distance in color space between the two pixels.

The linear invariance property compels us to choose carefully the color space in which the distance is computed. HSV for instance does not satisfy the requirements: the hue and saturation are not invariant when intensity changes (consult [Poy97]). The Commission Internationale de l'Eclairage (CIE) defined in 1931 the xyY color space which we will use for our purpose. x and y are good indicators of the actual color, regardless of the luminance. We have nevertheless to deal with a new non-linearity in the transform between RGB and CIE xyY.

Denoting I_k the image intensity of camera k, we define the following dot product between two image quadrilaterals:

$$< I_1, I_2 >= \frac{1}{N} \sum_{m_1} [I_1(m_1) - \overline{I_1}] \; . \; [I_2(m_2) - \overline{I_2}] \; . \; (1 - D_{12}^2)$$

where N is the number of points, $m_2 = f(m_1)$ with f one-to-one function,

$$\overline{I_k} = \frac{1}{N} \sum_{m} I_k(m)$$

and

$$D_{12} = \alpha \; dist_{CIExyY}(\mathbf{I}_1^{RGB}(m_1), \mathbf{I}_2^{RGB}(f(m_1)))$$

2.1.4 Improvements

An improvement to this method would be to take edge information into account and do ordering and continuity constraint piecewise between unsplit regions.

Authors have developed many refinements of correlation technique, some of which were mentioned above.

However, I consider the PDE method to be more promising. The idea comes from the generalization in 3D of the work done for snakes in 2D.

The basis of my work, [FK96, FK97] is interesting for several reasons, as I will later explain in detail:

- it uses n images naturally ($n > 2$).
- it works directly in 3D and provides a real surface with orientation (not just a set of points or surface elements).
- the iterative process can be implemented using Sethian's level sets methods.
- the iterative process takes self-occlusion into account. This could be extended to self-shadows as well.
- the global optimization handles disconnected components.

Chapter 3

Stages of the reconstruction

3.1 Mathematical framework for front evolution

3.1.1 Interest of an evolving surface

Let us recall that the main goal of our approach is to retrieve the euclidean geometry of a surface in 3D space.

We have seen that current techniques for direct extraction break due to the lack of complexity of the underlying model (3D from disparity, or "2.5 dimensions"), or their inability to match the projected points correctly between images (voxel approach [SD97, Sei97] and 3D selection of correspondence candidates in 2D [Zit96]).

The idea developed by Keriven and Faugeras [FK96, FK97] is to make a surface evolve until a correlation criterion is minimized on the whole surface, which means that the images all agree when reprojected onto this surface. We will further discuss this variational approach and the criterion in the following sections.

Why is this approach more powerful? It is so for at least three fundamental reasons.

First we know at every step the *normal vector* of the surface and therefore the tangent plane (first order approximation). We can even compute higher order approximation or any *intrinsic property* of the surface if we want. This yields a better way to compute correlation at each point, since we can take into account the orientation

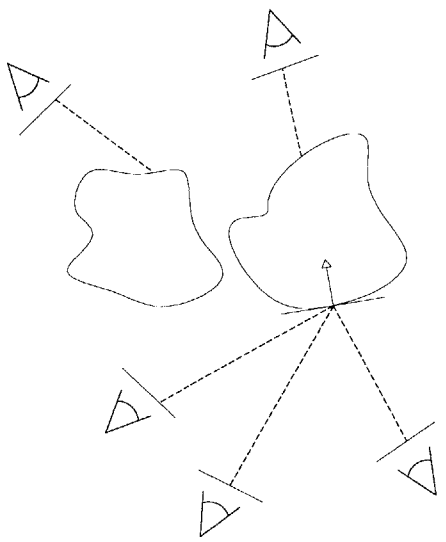

Figure 3-1: Interest of an evolving surface

of the surface and not assume it as fronto-parallel, as correlation techniques do.

It provides us also with the ability to compute reflected rays and therefore detecting *reflective* properties of surfaces.

For future development and highly curved sufficiently reflective surfaces – typically those that first-order correlation has trouble handling – highlights could be tracked. If we know the position of the light source, the evolving surface gives us a rough idea of where to look for highlights. These have the property of revealing accurate details about the surface, and should greatly refine areas where the correlation criterion is weak.

Second an evolving surface allows us to compute *self-occlusion* and consider only cameras that see the point (see figure 3-1). This is a real improvement compared to all other techniques (except the voxel approach). Note in particular that standard correlation techniques greatly suffer from points that do not have correspondences in the other images due to self-occlusion.

For future development, *shadows* could be also taken into account in the same way, as soon as we have a model of the lighting (sun position or model of cloudy sky). The linear correlation would adapt the illumination changes within the correlation window

at transition edges, and we could also check that the edges of shadows correspond.

Third, as we work directly in 3D and allow changes in topology, this technique handles *disconnected components* naturally. *Occlusions* should no longer be a problem, as long as they are seen by at least two cameras and the actual surface is also seen by several cameras. The evolving surface should tear and yield two new closed surfaces, one for the occluder and one for the occluded surface.

However this requires a fine enough grid, which is quite expensive in terms of memory and computational power.

This method starts from a simple surface and refines it progressively by iteration, which is intellectually more satisfying. It also naturally handles multiple cameras and the associated visibility problem.

3.1.2 Mathematical formulation

The typical evolution of a surface happens according to hyperbolic conservation law (same idea as the gradient descent methods), in which the temporal derivative of the surface is equal to a function depending on its position and intrinsic parameters:

$\frac{\partial \mathbf{S}}{\partial t} = \lambda_1 \mathbf{T}_1 + \lambda_2 \mathbf{T}_2 + \mu \mathbf{N}$, where $\mathbf{N}$ is the normal vector of the surface, $\mathbf{T}$ the tangent component and *lambda* and *mu* are speed functions of the position in space or intrinsic parameters of the surface.

In every case we have the property that there exists a new parameterization which removes the tangent component and yields the general expression

$$\frac{\partial \mathbf{S}}{\partial t} = \beta \, \mathbf{N} \tag{3.1}$$

Our task is therefore to find a speed β that globally optimizes our criterion to get the maximum correlation between images over the surface.

We have to deal with partial differential equations in 3D; let us now study a good way of computing this type of evolution.

3.2 Level set methods

Level set methods are a general framework for interface motion and front evolution. They are fully and clearly explained in [Set96]. The level set approach provides us with a robust implementation of the evolution of surfaces, for many reasons we will analyze.

This framework brings a true improvement because it considers a Eulerian point of view, as opposed to the Lagrangian point of view which follows elements on the surface. According to the Eulerian perspective, we compute values at fixed points in space instead of trying to follow moving points. This approach is generally preferred in physics because of its accuracy, and because we do not have to worry about the flow of particles as long as the grid is fine enough.

It handles splitting, merging and tearing naturally, without having to detect them, unlike the Lagrangian techniques. Level set methods solve problems like shocks and rarefaction fans, which are difficult to treat with sample points (see fig.3-2).

Finally it is designed to be robust to accuracy problems such as floating point roundoff errors (see [Set96]).

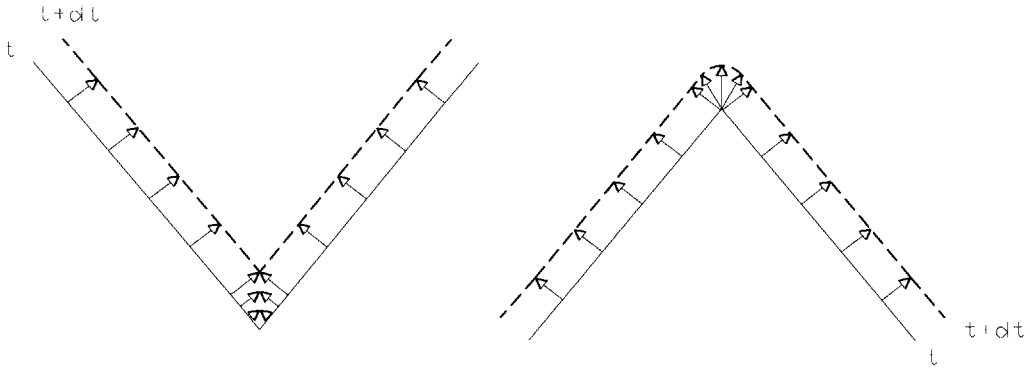

Figure 3-2: Shock and rarefaction fan in front evolution

3.2.1 Link between surface and level set

The level set u_0 of a scalar function $u(x, y, z, t)$ is the set of points at a given time which satisfy $u(x, y, z, t) = u_0$.

A set of points – in particular a surface – can be equivalently defined by a scalar function and a constant value over the space. If we know how to evolve the scalar function u in order to get a certain type of motion of the set of points, we can forget the point representation and deal only with u. Splitting, merging and tearing become suddenly totally natural thanks to this powerful equivalence.

For the type of evolution we are interested in, not only do we know the corresponding scalar function evolution, but its expression is quite similar to the one directly driving the points.

For a closed surface (in the sense that we can define an interior and an exterior), the level set approach allows a direct translation of $\frac{\partial \mathbf{S}}{\partial t} = \beta \mathbf{N}$ into

$$\frac{\partial u}{\partial t} = -\beta \, |\nabla u|$$

where $\mathbf{S}$ is a closed 3D surface depending on the time (for instance parameterized by two parameters, $\mathbf{S}(u, v, t)$), β denotes the speed along the normal and u is the level set function, which attaches a scalar to every point of space for a given time: $u : \Re^3 \times \Re \rightarrow \Re$

The level set function is defined by the fact that $u(S) = 0$, $u > 0$ outside the closed surface and $u < 0$ inside.

Starting from a surface, we can compute the level set function as the euclidean distance from each 3D point to the surface.

Conversely, we retrieve a surface from a level set function by extracting the points where $u = 0$.

3.2.2 Computation of the level set evolution

The main advantage of the level set method is that we no longer have to worry about the surface. We compute the speed at every point of the grid and update the level set function, and it will automatically handle changes in topology.

This supposes that we can extend the computation on the surface to a computation on *any* level set. This is easy in our case since the level set method provides us with normal vectors computed from the gradient of u. In the next sections we will discuss how to limit the number of updated points by computing only a narrow-band around the zero level set.

The reconstruction process requires that we periodically extract the points where $u = 0$. An efficient way of doing this is to use the *marching cubes* technique. The idea is to classify each voxel as outside, inside or at the interface and interpolate the vertices' values to approximate the interface by a triangulated mesh. The technique handles disconnected components as well.

3.2.3 Narrow-band methods

To avoid computing the evolution of u over the whole grid, we concentrate on our zone of interest, the zero level set. Its possible motion is represented by an "inner band" that we need to update. For each point of this inner band, we must compute some surface properties (such as the gradient) which compel to update a wider band, hatched on figure 3-3. The rest of the grid is simply frozen.

As soon as the zero level set crosses this hatched area, this narrow-band is no longer valid and we need to rebuild a new one by *reinitialization*, as the next section describes.

3.2.4 Reinitialization

When the motion of the zero level set causes it to exit the inner band, we have to renew the band. This means reinitializing the grid points with the distance to the surface.

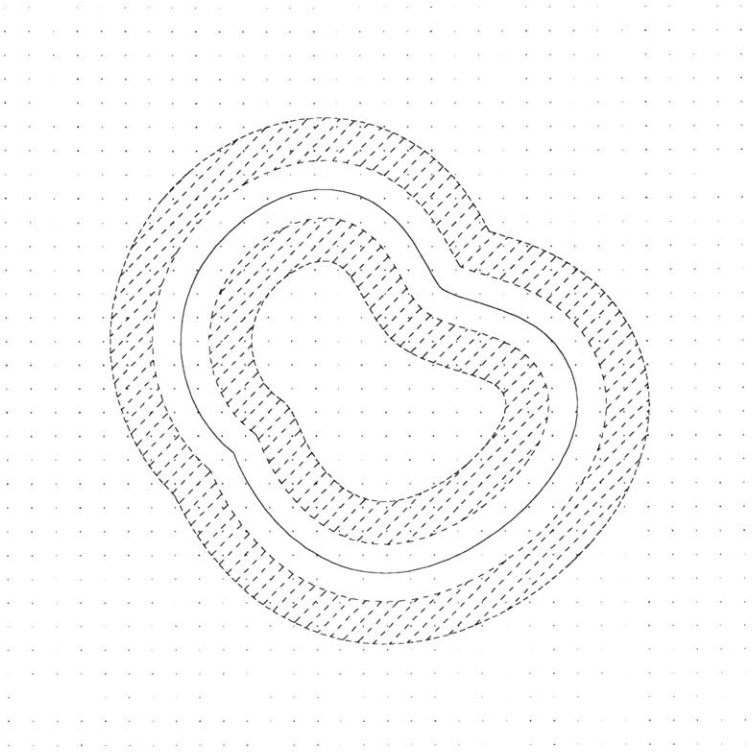

Figure 3-3: Narrow-band evolution

We need neither to reconstruct the zero level set nor to compute distance point by point.

The idea is to propagate the zero level set at speed one using the same techniques and calculate the crossing time at each point, i.e. the time at which a point value changes sign (the zero level set has reached it). This time corresponds to the distance between the point and the zero level set, since we made it propagate at speed 1.

This is fairly slow and the following *fast marching* technique is preferred for the purpose of reinitialization.

3.2.5 Fast marching methods

This technique is applicable if the normal speed depends on the position only (neither the orientation of the surface nor the time) and if it has constant sign. The evolution equation simplifies to the Eikonal equation that can be solved in an upwind fashion – that is to say in one pass instead of several iterations. Consult [Set96], chap. 9 for

more details.

The technique can be applied in particular to the reinitialization process during which the speed is equal to 1 (exterior) or -1 (interior).

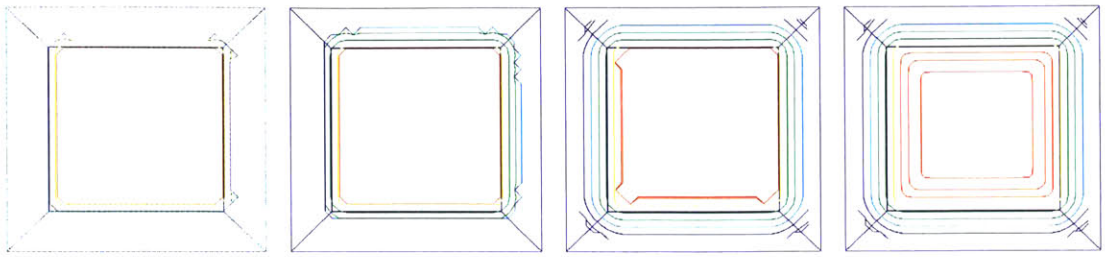

Figure 3-4: Fast Marching reinitialization

The fast marching method is a little less accurate, but far quicker than other iterated techniques.

The idea is to spread the information in the right direction, in an upwind fashion, starting from a small band around the zero level set, typically the nearest points.

This replaces the technique of iterating over the whole grid and is obviously much faster.

The technique does not seem applicable to the evolution itself. As Sethian himself writes, it is unclear how to update visibility properties during the evolution.

Moreover the speed is clearly not always in the same direction. We could imagine solving this single problem by adapting the time steps, taking care never to go beyond the extremum, as in some root finding techniques or the Levenberg-Marquardt technique. This is surely not straightforward.

The visibility problem prevents us from using the fast marching method for anything other than reinitialization anyway, taking into account if we want to avoid introducing more approximations in the computation.

We use in our case fast marching method for reinitialization purpose only.

3.3 Optimization criterion

The idea is similar to geodesic snakes (see [CKS95]). The geodesic active contours technique moves the surface according to a normal speed depending on image or volume gradient.

Here the camera images are our only source of measurement (stereovision). We will exploit them through the minimization of the correlation function.

Starting from a closed surface spanning our region of interest, we move the surface in a way that improves the correlation between images reprojected onto the surface.

The global optimization of a functional (integral of a function over the surface) leads to a Euler-Lagrange equation, since the functional depends on the position and the orientation (gradient) of the surface.

We will develop the mathematical expressions of this problem in the next chapter.

3.4 Reflective surfaces

Theoretically, we have the means to detect reflective surfaces by testing whether the correlation is better when we consider the surface as reflective. We try two hypotheses (but we could extend it to more) and select the best according to the correlation criterion.

As the computation is highly expensive in terms of memory, it is hard to keep a whole scene active, even if it is theoretically possible. If the grid is too coarse, then we lose the fine details and this cannot be retrieved at a finer resolution if we start from last results.

As it is impossible to store the whole model, we can use an approximation with environment maps – complete hemispherical images at every point of view (a node) – available from the City Scanning Project data. If the correlation function is higher at a voxel considered as reflective rather than lambertian, then we could "freeze" the evolution of this voxel: it will depend on its neighbors only (curvature dependent speed). We consider implicitly that the reflective surfaces are planar, which is mostly

true in urban environments and especially for buildings.

Detection of inter-reflection is theoretically possible too, if we check for self-intersection of the reflected rays.

3.5 Motivations for planar parallax

Despite the generality of the method, it is not practically applicable to the whole space.

It would probably require tens of gigabytes of memory to reconstruct an area covered by a hundred nodes.

Similarly, the computation cost is very high, involving several thousands of operations at each voxel for one step.

Multigridding

Multigridding allows us to focus on our zone of interest, without wasting computation time and memory.

It addresses the problem of memory by working recursively from a coarse to a fine grid. Note that if a region collapses and disappears totally at a given level it is almost impossible to retrieve it at a finer one. Moreover we have to deal carefully with the boundaries of the grid. The octree recursive subdivision must therefore overlap, in a way illustrated in 2D on the figure 3-5. This produces $8 + 6 + 1 = 15$ subspaces instead of 8 (in 2D, $4 + 4 + 1$ instead of 4).

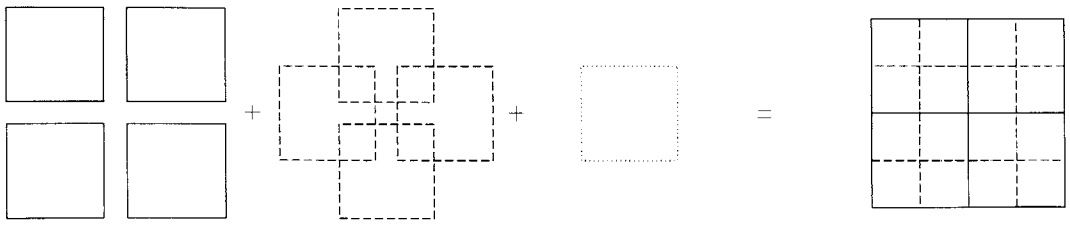

Figure 3-5: Overlap of regions for multigridding in 2D

Chapter 4

Normal speed computation

4.1 Geometric model

Our correlation criterion requires that we measure the correlation score of a grid point belonging to the level set narrow band, so we need to define some kind of neighborhood around this point. The normal vector gives us the first-order approximation of the surface, the tangent plane.

Keriven and Faugeras chose to consider the projection of the 3D point on the first image, define a rectangle around it and compute the affine approximation of the corresponding rectangle on the other image : the rectangle is reprojected onto the tangent plane and we take its image by the second camera with an affine approximation.

I thought it was more natural to define the correlation window in 3D and keep the full perspective, even if it introduces additional complexity, as we will see in the following sections.

Once visibility is checked, we still have the degree of freedom of rotation on the tangent plane. To exploit the scanline format of our images, we build the reference frame using the up vector of the first camera, so that the basis vectors match the horizontal and vertical axes of the first image. Note that this does not necessarily mean that the 3D rectangular correlation window will be projected as a rectangle, because of the full perspective projection.

Let $\mathcal{U}$ be the up vector of the first camera that sees the object. We use it to build

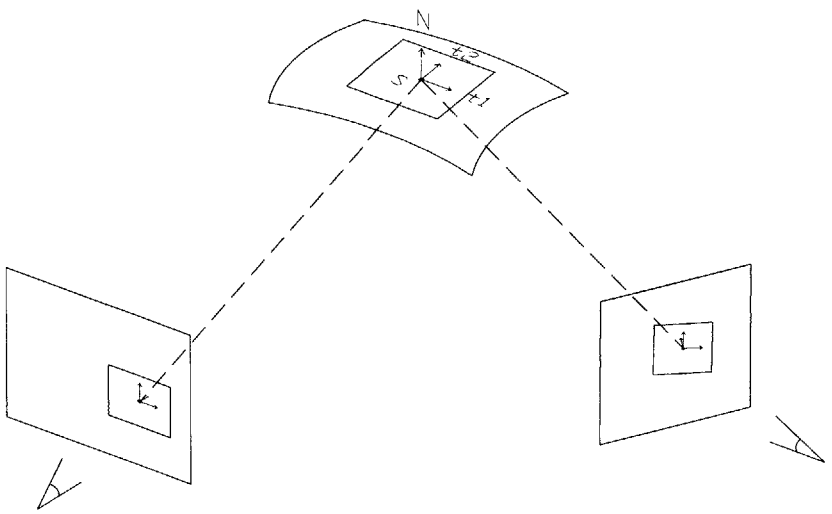

Figure 4-1: A pair of cameras and their projected correlation windows

an orthonormal basis on the tangent plane to the surface in the following way:

$$\mathbf{t}_1 = \frac{\mathcal{U} \times \mathbf{N}}{\|\mathcal{U} \times \mathbf{N}\|} \tag{4.1}$$

$$\mathbf{t}_2 = \mathbf{N} \times \mathbf{t}_1 \tag{4.2}$$

The 3D correlation window at the point $\mathbf{S}(x, y, z) = (x, y, z)^T$ and tangent to the surface $u = u(\mathbf{S}(x, y, z))$ is defined by

$$\mathbf{W}(\mathbf{S}, \mathbf{N}, i, j) = \mathbf{S} \ + \ i\lambda\mathbf{t}_1 \ + \ j\mu\mathbf{t}_2 \tag{4.3}$$

with $(i, j) \in [-p, p] \times [-q, q]$. Figure 4-2 illustrates the model.

The width and height of the correlation window depend on the geometry of the 3D grid (constants λ and μ), so that the rectangle roughly matches the grid spacing.

p and q are on the contrary functions of the resolutions of the images themselves. We keep the values of the smallest resolution, since oversampling is artificial: we wish to fully exploit the information available, but no more.

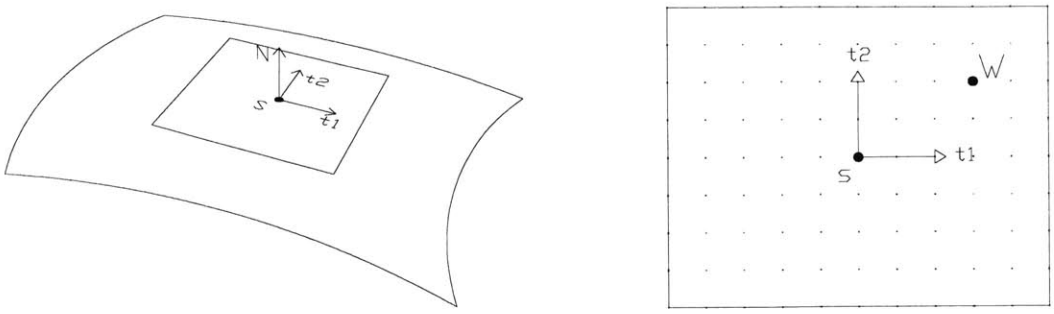

Figure 4-2: Tangent plane and 3D correlation window

We use in the implementation a Gaussian weighting of the points in the correlation window in order to increase the influence of its center. Remember that the correlation window is a first-order approximation of the surface. We will omit this weighting in the following sections to simplify the equations.

4.2 Correlation equation

We use as a criterion a modification of the correlation function that we studied in paragraph 2.1.3. The standard linear correlation function is enhanced with a color distance term to avoid matching different colors of the same luminance.

Denoting the camera images and their intensities as I_k, the projection matrices as $\tilde{\mathbf{P}}_k$, and the 4×1 vector $[\mathbf{W}^T \ 1]^T$ as $\tilde{\mathbf{W}}$, we define at the point $\mathbf{S}(x, y, z)$ of unit normal $\mathbf{N} = \frac{\nabla u}{\|\nabla u\|}$ the following dot product between two image quadrilaterals:

$$< I_1, I_2 > (\mathbf{S}, \mathbf{N}) = \frac{\lambda\mu}{4pq} \iint [I_1(\tilde{\mathbf{P}}_1\tilde{\mathbf{W}}) - \overline{I_1}] \ . \ [I_2(\tilde{\mathbf{P}}_2\tilde{\mathbf{W}}) - \overline{I_2}] \ . \ (1 - D_{12}^2) \ didj \quad (4.4)$$

where

$$\overline{I_k}(\mathbf{S}, \mathbf{N}) = \frac{\lambda\mu}{4pq} \iint I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) \ didj \quad (4.5)$$

and

$$D_{12}(\mathbf{S}, \mathbf{N}, i, j) = \alpha \ dist_{CIExyY}(\mathbf{I}_1^{RGB}(\tilde{\mathbf{P}}_1\tilde{\mathbf{W}}), \mathbf{I}_2^{RGB}(\tilde{\mathbf{P}}_2\tilde{\mathbf{W}})) \quad (4.6)$$

We recall that $\mathbf{W}$ depends on $\mathbf{S}$, $\mathbf{N}$, i and j. The norm associated with this dot product is $|I|^2 =< I, I >$

We can now define the correlation function as:

$$\Phi(\mathbf{S}, \mathbf{N}) = 1 - \frac{< I_1, I_2 >}{|I_1|.|I_2|} \tag{4.7}$$

Its values range between 0 (best correlation) and +2 (worst).

4.3 Euler-Lagrange equation

The proof of the speed formula that follows can be found in [FK96, FK97].

The error measure to minimize on the surface Γ is, as explained previously:

$$C = \sum_{views\ i \neq j} \iint_{\Gamma} \Phi_{ij}(\mathbf{S}, \mathbf{N})\ d\sigma \tag{4.8}$$

(i, j) designates a pair of views for which the patch $d\sigma$ is visible. In the following we focus (for calculus only) on the case of two cameras $\Phi = \Phi_{ij}$, since we can extend the results by simple summation.

We want to minimize this functional; the natural approach is to write the Euler-Lagrange equation which gives a necessary condition for an extremum.

Notice that it does not prove that this extremum is a minimum, nor that it is absolute and not local. Moreover it is not a sufficient condition.

However, as we will see, we will add a "balloon" term – a straight function of the correlation function – that highly increases our chances of reaching an absolute minima. Note that we may have several shapes which physically correspond to the same views.

In an urban environment, this is fortunately rare, and we look for shapes of minimal area (suited to planar surfaces).

From this Euler-Lagrange equation, we extract the component normal to the surface and plug it into a level-set evolution.

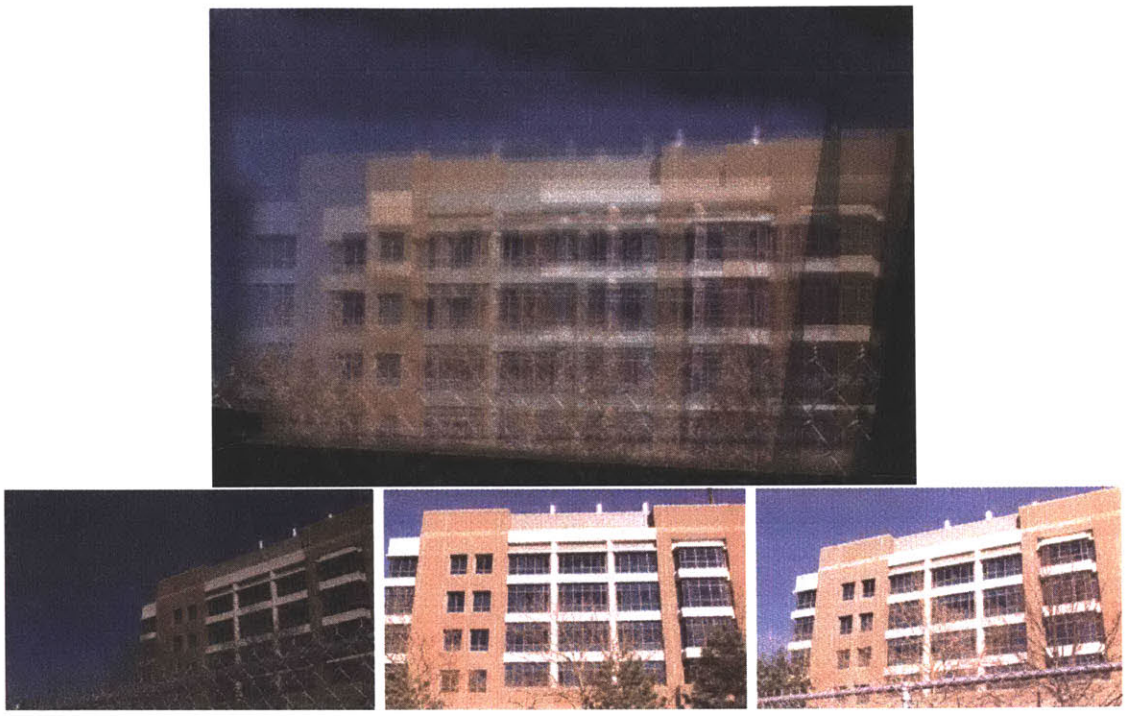

Figure 4-3: Uncorrelated reprojected images in case of wrong position

Consider $S_t = -\beta\ \mathbf{N}$, hyperbolic evolution of a surface. This equation translates directly into level-set evolution as $u_t = \beta\ |\nabla u|$ (see Sethian book, also simply chain rule from $u(\mathbf{S}, t) = 0$)

Like snake techniques and Faugeras-Keriven approach, we use the normal component of the Euler-Lagrange equation as β, so that the evolution stops at an extremum.

The normal component taken into account in the level set evolution is

$$\begin{aligned} \beta \ = \ & \frac{\partial \Phi}{\partial \mathbf{S}}\mathbf{N} \ + \frac{\partial \Phi}{\partial \mathbf{N}}\frac{d\mathbf{N}}{d\mathbf{S}}\mathbf{N} - 2H(\Phi - \frac{\partial \Phi}{\partial \mathbf{N}}\mathbf{N}) \ + \\ & \mathbf{t}_1^T \frac{\partial^2 \Phi}{\partial \mathbf{S} \partial \mathbf{N}}\mathbf{t}_1 + \mathbf{t}_2^T \frac{\partial^2 \Phi}{\partial \mathbf{S} \partial \mathbf{N}}\mathbf{t}_2 \ + Trace\left(d\mathbf{N} \circ (\frac{\partial^2 \Phi}{\partial \mathbf{N}^2})_{|Tan(\mathbf{S})}\right) \end{aligned} \tag{4.9}$$

where $H = \frac{1}{2} div(\frac{\nabla u}{||\nabla u||})$ is the normal curvature of the surface $u = u(\mathbf{S})$, Φ is the correlation function, $\frac{\partial \Phi}{\partial \mathbf{S}}$ and $\frac{\partial \Phi}{\partial \mathbf{N}}$ are 1×3 row vectors, $\frac{d\mathbf{N}}{d\mathbf{S}}$, $\frac{\partial^2 \Phi}{\partial \mathbf{S} \partial \mathbf{N}}$ and $\frac{\partial^2 \Phi}{\partial \mathbf{N}^2}$ are 3×3 matrices, all derivatives of the scalar function Φ. dN is a third-order tensor ($3 \times 3 \times 3$), differential of the Gauss map applied to the restriction of $\frac{\partial^2 \Phi}{\partial \mathbf{N}^2}$ to the tangent plane at the point $\mathbf{S}$.

This corresponds to the following level-set evolution:

$$u_t = \beta \, |\nabla u| \tag{4.10}$$

Therefore we need to compute the derivatives of Φ with respect to $\mathbf{S}$ and $\mathbf{N}$.

4.4 Derivatives of Φ

Let us first make clear that $\mathbf{S}$ and $\mathbf{N}$ are considered to be independent variables in the writing of the derivatives of Φ in the Euler-Lagrange equation. Indeed the relation between $\mathbf{N}$ and $\mathbf{S}$ is already taken into account in equation 4.9. So $\frac{\partial \Phi}{\partial \mathbf{S}}$ is the derivative of $\Phi(\mathbf{S}, \mathbf{N})$ with respect to its first variable and $\frac{\partial \Phi}{\partial \mathbf{N}}$ the derivative to its second variable, considered independent from the first one.

4.4.1 Derivatives of $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{W}$

We denote $[\mathcal{U}]_\times$ as the 3×3 antisymmetrical matrix such that $\mathcal{U} \times \mathbf{N} = [\mathcal{U}]_\times \mathbf{N}$.

The vector $\mathbf{t}_1$ is tangent to the surface and oriented to the right of the first camera:

$$\mathbf{t}_1 = \frac{\mathcal{U} \times \mathbf{N}}{\|\mathcal{U} \times \mathbf{N}\|}$$

so

$$\frac{\partial \mathbf{t}_1}{\partial \mathbf{S}} = 0$$

and

$$\begin{aligned} \frac{\partial \mathbf{t}_1}{\partial \mathbf{N}} &= \frac{[\mathcal{U}]_\times}{\|\mathcal{U} \times \mathbf{N}\|} - (\mathcal{U} \times \mathbf{N}) \frac{(\mathcal{U} \times \mathbf{N})^T}{\|\mathcal{U} \times \mathbf{N}\|^3} [\mathcal{U}]_\times \\ &= [\mathbf{I} - \frac{(\mathcal{U} \times \mathbf{N})(\mathcal{U} \times \mathbf{N})^T}{\|\mathcal{U} \times \mathbf{N}\|^2}] \, \frac{[\mathcal{U}]_\times}{\|\mathcal{U} \times \mathbf{N}\|} \\ &= [\mathbf{I} - \mathbf{t}_1 \, \mathbf{t}_1^T] \, \frac{[\mathcal{U}]_\times}{\|\mathcal{U} \times \mathbf{N}\|} \end{aligned} \tag{4.11}$$

The vector $\mathbf{t}_2$ is tangent to the surface and oriented according to the up direction

of the first camera:

$$\begin{aligned} \mathbf{t}_2 &= \mathbf{N} \times \mathbf{t}_1 = \frac{\mathbf{N} \times (\mathcal{U} \times \mathbf{N})}{\|\mathcal{U} \times \mathbf{N}\|} \\ &= \frac{\mathcal{U} - (\mathcal{U}^T\mathbf{N})\mathbf{N}}{\|\mathcal{U} \times \mathbf{N}\|} \end{aligned} \tag{4.12}$$

as $\mathbf{N}$ is a unit vector. So $\frac{\partial \mathbf{t}_2}{\partial \mathbf{S}} = 0$ and

$$\begin{aligned} \frac{\partial \mathbf{t}_2}{\partial \mathbf{N}} &= -\mathcal{U}\frac{(\mathcal{U} \times \mathbf{N})^T}{\|\mathcal{U} \times \mathbf{N}\|^3}[\mathcal{U}]_\times - \frac{\mathcal{U}^T\mathbf{N}}{\|\mathcal{U} \times \mathbf{N}\|}\mathbf{I} - \mathbf{N}\frac{d}{d\mathbf{N}}\left(\frac{\mathcal{U}^T\mathbf{N}}{\|\mathcal{U} \times \mathbf{N}\|}\right) \\ &= [(\mathcal{U}^T\mathbf{N})\mathbf{N} - \mathcal{U}]\frac{(\mathcal{U} \times \mathbf{N})^T}{\|\mathcal{U} \times \mathbf{N}\|^3}[\mathcal{U}]_\times - \frac{(\mathcal{U}^T\mathbf{N})\mathbf{I} + \mathbf{N}\mathcal{U}^T}{\|\mathcal{U} \times \mathbf{N}\|} \\ &= \frac{[(\mathcal{U}^T\mathbf{N})\mathbf{N} - \mathcal{U}]\ \mathbf{t}_1^T}{\|\mathcal{U} \times \mathbf{N}\|^2}[\mathcal{U}]_\times - \frac{(\mathcal{U}^T\mathbf{N})\mathbf{I} + \mathbf{N}\mathcal{U}^T}{\|\mathcal{U} \times \mathbf{N}\|} \\ &= -\frac{\mathbf{t}_2\mathbf{t}_1^T[\mathcal{U}]_\times + (\mathcal{U}^T\mathbf{N})\mathbf{I} + \mathbf{N}\mathcal{U}^T}{\|\mathcal{U} \times \mathbf{N}\|} \end{aligned} \tag{4.13}$$

We can consequently write the derivatives of $\mathbf{W}$, the 3D point within the correlation window on the tangent plane:

$$\mathbf{W}(\mathbf{S}, \mathbf{N}, i, j) = \mathbf{S} \ + \ i\lambda\mathbf{t}_1 \ + \ j\mu\mathbf{t}_2$$

which implies that

$$\frac{\partial \mathbf{W}}{\partial \mathbf{S}} = \mathbf{I} \tag{4.14}$$

and

$$\frac{\partial \mathbf{W}}{\partial \mathbf{N}} = i\lambda\frac{d\mathbf{t}_1}{d\mathbf{N}} + j\mu\frac{d\mathbf{t}_2}{d\mathbf{N}} \tag{4.15}$$

$\frac{\partial \mathbf{W}}{\partial \mathbf{S}}$ and $\frac{\partial \mathbf{W}}{\partial \mathbf{N}}$ are 3×3 matrices.

4.4.2 Derivatives of $I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$, $\overline{I_1}$ and D_{kl}^2

Let us study $I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$:

$$\frac{\partial}{\partial \mathbf{S}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) = \nabla I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})^T \cdot d\Theta(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) \cdot \mathbf{P}_k \tag{4.16}$$

$\tilde{\mathbf{P}}_k = [\mathbf{P}_k\ p_k]$ with $\mathbf{P}_k$ 3×3 matrix. We use the abusive notation $I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$ to simplify the obvious expression $I_k(\Theta(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}))$, where the intermediate function Θ is equal to $\Theta(m_x, m_y, m_z) = (m_x/m_z, m_y/m_z)$. This is needed to convert projective coordinates into euclidean ones. The derivative of Θ is equal to

$$\frac{d\Theta}{d\mathbf{m}} = \begin{pmatrix} 1/m_z & 0 & -m_x/m_z^2 \\ 0 & 1/m_z & -m_y/m_z^2 \end{pmatrix}$$

which means that

$$d\Theta = \frac{1}{m_z}\begin{pmatrix} 1 & 0 & -m_x/m_z \\ 0 & 1 & -m_y/m_z \end{pmatrix} d\mathbf{m} \tag{4.17}$$

Similarly,

$$\frac{\partial}{\partial \mathbf{N}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) = \nabla I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})^T \cdot d\Theta(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) \cdot \mathbf{P}_k \frac{\partial \mathbf{W}}{\partial \mathbf{N}} \tag{4.18}$$

and thus

$$\frac{\partial^2}{\partial \mathbf{S}\partial \mathbf{N}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}}) = 0 \tag{4.19}$$

- We know from (4.5) that

$$\overline{I_k}(\mathbf{S}, \mathbf{N}) = \frac{\lambda\mu}{4pq} \iint I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})\ didj$$

so $\frac{\partial \overline{I_k}}{\partial \mathbf{S}}$, $\frac{\partial \overline{I_k}}{\partial \mathbf{N}}$ and $\frac{\partial^2 \overline{I_k}}{\partial \mathbf{S}\partial \mathbf{N}}$ derive directly from $\frac{\partial}{\partial \mathbf{S}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$, $\frac{\partial}{\partial \mathbf{N}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$ and $\frac{\partial^2}{\partial \mathbf{S}\partial \mathbf{N}} I_k(\tilde{\mathbf{P}}_k\tilde{\mathbf{W}})$.

- The color transform from *RGB* color space to *CIE xyY* is not linear so we must take it into account in the differentiation. Let $\mathcal{T}$ be the linear transform from RGB color space to CIE XYZ modified as $XY\hat{Z}$ (with $\hat{Z} = Z + X + Y$, intermediate color space linearly transformed from RGB), we have $(x, y)^T = \Theta(\mathcal{T}(R, G, B)^T)$ and $D_{12}^2 = \|(x_1, y_1) - (x_2, y_2)\|^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$, so

$$\frac{\partial D_{12}^2}{\partial \mathbf{S}} = 2[\Theta(\mathbf{I}_1^{XY\hat{Z}}(\tilde{\mathbf{P}}_1\tilde{\mathbf{W}})) - \Theta(\mathbf{I}_2^{XY\hat{Z}}(\tilde{\mathbf{P}}_2\tilde{\mathbf{W}}))]^T [\frac{\partial \Theta(\mathbf{I}_1^{XY\hat{Z}}(\tilde{\mathbf{P}}_1\tilde{\mathbf{W}}))}{\partial \mathbf{S}} - \frac{\partial \Theta(\mathbf{I}_2^{XY\hat{Z}}(\tilde{\mathbf{P}}_2\tilde{\mathbf{W}}))}{\partial \mathbf{S}}]$$

which means

$$\begin{aligned}\frac{\partial D_{12}^2}{\partial \mathbf{S}} &= 2\ [\Theta(\mathbf{I}_1^{XY\hat{Z}}) - \Theta(\mathbf{I}_2^{XYZ})]^T \\ &\quad [d\Theta(\mathbf{I}_1^{XY\hat{Z}}) \cdot \mathcal{T} \cdot \frac{\partial}{\partial \mathbf{S}} \mathbf{I}_1^{RGB}(\tilde{\mathbf{P}}_1 \tilde{\mathbf{W}}) - \\ &\quad d\Theta(\mathbf{I}_2^{XY\hat{Z}}) \cdot \mathcal{T} \cdot \frac{\partial}{\partial \mathbf{S}} \mathbf{I}_2^{RGB}(\tilde{\mathbf{P}}_2 \tilde{\mathbf{W}})] \end{aligned} \tag{4.20}$$

$$\begin{aligned}\frac{\partial D_{12}^2}{\partial \mathbf{N}} &= 2\ [\Theta(\mathbf{I}_1^{XY\hat{Z}}) - \Theta(\mathbf{I}_2^{XYZ})]^T \\ &\quad [d\Theta(\mathbf{I}_1^{XY\hat{Z}}) \cdot \mathcal{T} \cdot \frac{\partial}{\partial \mathbf{N}} \mathbf{I}_1^{RGB}(\tilde{\mathbf{P}}_1 \tilde{\mathbf{W}}) - \\ &\quad d\Theta(\mathbf{I}_2^{XY\hat{Z}}) \cdot \mathcal{T} \cdot \frac{\partial}{\partial \mathbf{N}} \mathbf{I}_2^{RGB}(\tilde{\mathbf{P}}_2 \tilde{\mathbf{W}})] \end{aligned} \tag{4.21}$$

4.4.3 Derivatives of $< I_k, I_l >$

From (4.4) we know that

$$< I_1, I_2 > (\mathbf{S}, \mathbf{N}) = \frac{\lambda\mu}{4pq} \iint [I_1(\tilde{\mathbf{P}}_1 \tilde{\mathbf{W}}) - \overline{I_1}] \ . \ [I_2(\tilde{\mathbf{P}}_2 \tilde{\mathbf{W}}) - \overline{I_2}] \ . \ (1 - D_{12}^2)\ didj$$

Let us note to simplify

$$\Pi_k = I_k(\tilde{\mathbf{P}}_k \tilde{\mathbf{W}}) - \overline{I_k}$$

so that

$$< I_1, I_2 > \propto \iint \Pi_1 \ . \ \Pi_2 \ . \ (1 - D_{12}^2)$$

Then

$$d < I_1, I_2 > \propto \iint (d\Pi_1 . \Pi_2 + \Pi_1 . d\Pi_2) . (1 - D_{12}^2) - \Pi_1 . \Pi_2 . dD_{12}^2 \tag{4.22}$$

where d stands for $\frac{\partial}{\partial \mathbf{S}}$ or $\frac{\partial}{\partial \mathbf{N}}$.

4.4.4 Simplification of the summation

Consider the computation of $< I_1, I_2 >$. To avoid computing the mean first, we expand the terms in the discrete sum (over n points) to get the result in one pass. Let us recall that

$$\overline{I} = \frac{1}{n} \sum I$$

Consequently we can compute in a single loop, with $\omega = 1 - D_{12}^2$

$$\begin{aligned} < I_1, I_2 > &\propto \sum (I_1 - \overline{I_1}) \,.\, (I_2 - \overline{I_2}) \,.\, \omega \\ &\propto (\sum I_1.I_2.\omega) + \overline{I_1}.(\overline{I_2}.\sum \omega - \sum I_2.\omega) - \overline{I_2}.\sum I_1.\omega \quad (4.23) \\ |I|^2 &\propto (\sum I^2) - n\,\overline{I}^2 \quad (4.24) \end{aligned}$$

In the case of the differentiated expression, we get

$$\begin{aligned} d < I_1, I_2 > &\propto \sum (dI_1 - \overline{dI_1}).(I_2 - \overline{I_2}).\omega + \sum (I_1 - \overline{I_1}).(dI_2 - \overline{dI_2}).\omega - \\ &\quad \sum (I_1 - \overline{I_1}).(I_2 - \overline{I_2}).dD_{12}^2 \\ &\propto (\sum dI_1.I_2.\omega) + \overline{dI_1}.(\overline{I_2}.\sum \omega - \sum I_2.\omega) - \overline{I_2}.\sum dI_1.\omega + \\ &\quad (\sum I_1.dI_2.\omega) + \overline{I_1}.(\overline{dI_2}.\sum \omega - \sum dI_2.\omega) - \overline{dI_2}.\sum I_1.\omega \quad (4.25) \\ &\quad (\sum I_1.I_2.dD_{12}^2) - \overline{I_1}.(\overline{I_2}.\sum dD_{12}^2 - \sum I_2.dD_{12}^2) + \overline{I_2}.\sum I_1.dD_{12}^2 \end{aligned}$$

where d stands for $\frac{\partial}{\partial \mathbf{S}}$ or $\frac{\partial}{\partial \mathbf{N}}$.

4.4.5 Derivatives of Φ

$$\begin{aligned} \frac{\partial \Phi}{\partial \mathbf{S}} &= \frac{\partial}{\partial \mathbf{S}} \left(1 - \frac{< I_1, I_2 >}{|I_1|.|I_2|} \right) \quad (4.26) \\ &= \frac{-\frac{\partial}{\partial \mathbf{S}} < I_1, I_2 >}{|I_1|.|I_2|} + \frac{< I_1, I_2 >}{|I_1|.|I_2|} \left(\frac{\sum (I_1 - \overline{I_1}).(\frac{\partial I_1}{\partial \mathbf{S}} - \overline{\frac{\partial I_1}{\partial \mathbf{S}}})}{|I_1|^2} + \frac{\sum (I_2 - \overline{I_2}).(\frac{\partial I_2}{\partial \mathbf{S}} - \overline{\frac{\partial I_2}{\partial \mathbf{S}}})}{|I_2|^2} \right) \\ &= -\frac{\frac{\partial}{\partial \mathbf{S}} < I_1, I_2 >}{|I_1|.|I_2|} + \frac{< I_1, I_2 >}{|I_1|.|I_2|} \left(\frac{(\sum I_1.\frac{\partial I_1}{\partial \mathbf{S}}) - n\,\overline{I_1}.\overline{\frac{\partial I_1}{\partial \mathbf{S}}}}{|I_1|^2} + \frac{(\sum I_2.\frac{\partial I_2}{\partial \mathbf{S}}) - n\,\overline{I_2}.\overline{\frac{\partial I_2}{\partial \mathbf{S}}}}{|I_2|^2} \right) \end{aligned}$$

Similarly,

$$\frac{\partial \Phi}{\partial \mathbf{N}} = -\frac{\frac{\partial}{\partial \mathbf{N}} < I_1, I_2 >}{|I_1|.|I_2|} + \frac{< I_1, I_2 >}{|I_1|.|I_2|} \left(\frac{(\sum I_1 . \frac{\partial I_1}{\partial \mathbf{N}}) - n\ \overline{I_1} . \overline{\frac{\partial I_1}{\partial \mathbf{N}}}}{|I_1|^2} + \frac{(\sum I_2 . \frac{\partial I_2}{\partial \mathbf{N}}) - n\ \overline{I_2} . \overline{\frac{\partial I_2}{\partial \mathbf{N}}}}{|I_2|^2} \right)$$

4.4.6 Computation of $\frac{d\mathbf{N}}{d\mathbf{S}}$

$\frac{d\mathbf{N}}{d\mathbf{S}}$ is a 3×3 matrix. Its elements are the derivatives of the normal vector with respect to the space coordinates.

As $\mathbf{N} = \frac{\nabla u}{|\nabla u|}$, we have

$$\frac{d}{d\mathbf{S}} \left(\frac{\nabla u}{|\nabla u|} \right) = \left(\mathbf{I} - \frac{(\nabla u)(\nabla u)^T}{|\nabla u|^2} \right) \frac{\nabla^2 u}{|\nabla u|}$$

which means

$$\frac{d\mathbf{N}}{d\mathbf{S}} = \frac{1}{|\nabla u|} (\mathbf{I} - \mathbf{N}\mathbf{N}^T) \frac{d^2 u}{d\mathbf{S}^2} \tag{4.27}$$

We compute the term $\frac{d^2 u}{d\mathbf{S}^2}$ by one-sided upwind differences in the appropriate direction, according to the sign of each element by which it is multiplied.

For example, to compute $e_{xy} \frac{\partial u_x}{\partial y}$, we evaluate

$$\max(e_{xy}, 0)\ m(D^{-x-y}, D^{+x-y})\ +\ \min(e_{xy}, 0)\ m(D^{-x+y}, D^{+x+y})$$

where m is the switch function defined in [Set96], p.57:

$$m(x,y) = \left\{ \begin{array}{cc} \left\{ \begin{array}{ll} x & if\ |x| \leq |y| \\ y & if\ |x| > |y| \end{array} \right\} & xy \geq 0 \\ 0 & xy < 0 \end{array} \right\}$$

The difference approximation for the second-order derivatives of the level set function u are computed as illustrated in figure 4-4.

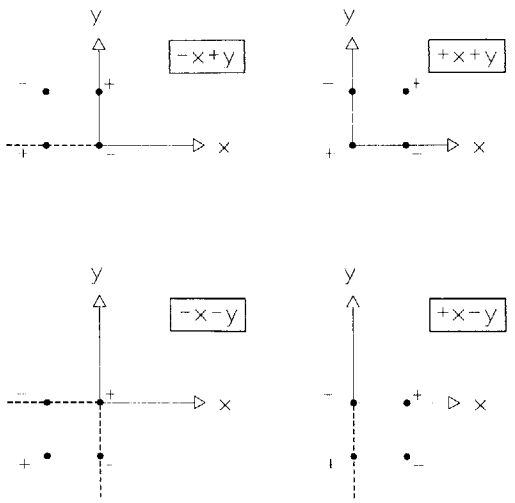

Figure 4-4: Approximation scheme for $\frac{d\mathbf{N}}{d\mathbf{S}}$ computation

4.5 Second order and approximation of the level set evolution

Let us recall the normal component of the speed seen in (4.9):

$$\begin{aligned}\beta \;=\; & \frac{\partial \Phi}{\partial \mathbf{S}}\mathbf{N} + \frac{\partial \Phi}{\partial \mathbf{N}}\frac{d\mathbf{N}}{d\mathbf{S}}\mathbf{N} - 2H(\Phi - \frac{\partial \Phi}{\partial \mathbf{N}}\mathbf{N}) + \\ & \mathbf{t}_1^T \frac{\partial^2 \Phi}{\partial \mathbf{S}\partial \mathbf{N}}\mathbf{t}_1 + \mathbf{t}_2^T \frac{\partial^2 \Phi}{\partial \mathbf{S}\partial \mathbf{N}}\mathbf{t}_2 + Trace\left(d\mathbf{N} \circ (\frac{\partial^2 \Phi}{\partial \mathbf{N}^2})_{|Tan(\mathbf{S})}\right)\end{aligned}$$

To limit the complexity of the calculus and speed up the computation, we need to approximate this expression to the dominant terms.

Clearly the last term can be neglected compared to the others: it is composed of components of the differential of the Gaussian map multiplied with second order derivatives. This is good news since it would have implied computing subtle intrinsic properties of the surface and dealing with third order tensors for second order differentials!

$\frac{\partial^2 I_k}{\partial \mathbf{S}\partial \mathbf{N}} = 0$ and $\frac{\partial^2 D_{i2}^2}{\partial \mathbf{S}\partial \mathbf{N}} = 0$ means that in the second order expression, we have only terms product of $\frac{\partial}{\partial \mathbf{S}}$ with $\frac{\partial}{\partial \mathbf{N}}$.

Chapter 5

Implementation

Please refer to appendix B for a detailed description of the interface.

5.1 Libraries and tools

The system was implemented on a Silicon Graphics O2 (*http://www.sgi.com*), as well as a Sun UltraSparc 2 (*http://www.sun.com*) and a PC running Linux. It is built on OpenGL, The Visualization Toolkit (VTK 2.0, see [SML98] or *http://www.kitware.com*) and XForms (*http://bragg.phys.uwm.edu/xforms*), which are all classical tools for 3D graphics and user interfaces.

I built a library for easy linking of OpenGL with XForms, for convenient image display and 3D interaction with a mouse. It is public domain, available for download at *http://graphics.lcs.mit.edu/~amram/Code/GLforms.tar*.

5.2 3D input region

- From the City Project (see [Coo98]),
- from interation with the user, who specifies two 2D quads on two images. The program reconstructs the closest 3D quad from these,
- or from line input, especially if larger than the images.

The program takes as input images with camera calibration information, displays them in 2D and 3D, and also renders a warped image as if a piece of the image (a 2D quad) was on a rectangular facade.

5.3 Level set visualization

The reader can refer to Appendix B for use of planar probes, zero level set extraction and fast marching method visualization.

The program performs the reinitialization phase either by the Fast Marching Method or by the crossing times method according to the user's choice.

5.4 Evolution and texture reprojection

Several shapes and several evolution are available. I used texture mapping and alpha blending to display the texture. Refer to Appendix B for details.

All computations are parallelized independently for each voxel, with tiny overhead. This means that a four-processor machine actually actually computes four times faster. The program can in the same way take advantage of any n processor machine.

5.5 Reconstruction

As we described in the previous chapters, the reconstruction relies on a correlation-based evolution. The visualization of the process is fully interactive, even if it is run in batch mode.

5.5.1 Choice of the pairs of images

The goal of choosing best pairs of images is to maximize the projected areas and optimize the angle between cameras. Once the best is found, the program selects the other pairs compared to it. This allows us to deal with distance and resolution of

images, and to take the best information available adaptatively. In some cases we have only little information available and we must use low detail images, while in others we have redundant information of varying precision: in these cases we must throw away the inaccurate information.

5.5.2 Visibility surfaces

The reconstruction makes heavy use of the marching cube algorithm (see [SML98]).

However this takes time so we have to avoid it at all costs. Theoretically, visibility of a grid point must be checked according to the level set of the current point. This produces too many surfaces to deal with and is not practical.

Therefore we approximate the problem by selecting only a few inner surfaces for the self-occlusion computation. A point always checks its visibility with respect to a level set of lower value.

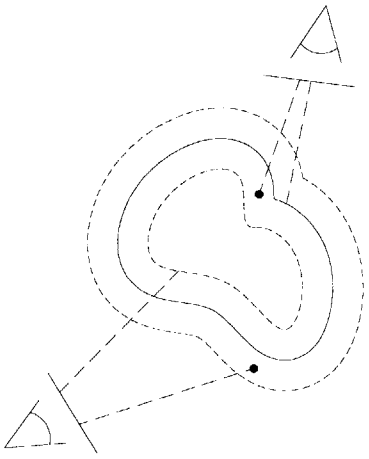

Figure 5-1: Visibility surfaces for the level set narrow band

The visibility computation itself is based on fast ray-triangle intersection along with a bucketing technique to split the grid space and avoid testing all triangles. We march the buckets along the ray efficiently (see [HB97], p.536–537) and check intersection with every triangle belonging to the bucket.

I preferred buckets to octrees because of the large overhead of octree subdivision, since our grids are relatively small and our set of triangles very specific.

5.5.3 Discretization of the normal speed computation

The discretized nature of the images introduces new challenges to compute the expression of the normal speed. In particular the evolution cannot be infinitesimal, which compels us to move the surface by discrete steps along the normal vector.

We use therefore a pyramid of images (of size 2^n) to approximate the gradient of the image, in order to take into account the interval of displacement on the image.

The step along the normal must be taken small enough to be consistent with the path it spans, but must be big enough to avoid local minima. This is a tradeoff, and we have chosen to adapt the step with respect to the maximum speed of points belonging to the zero level set.

The next observation is that only two terms are really useful to reach a coarse minimum of our functional $C = \sum_{views\ i \neq j} \iint_\Gamma \Phi_{ij}(\mathbf{S}, \mathbf{N})\ d\sigma$.

In our implementation we use $\frac{\partial \Phi}{\partial \mathbf{S}}\mathbf{N}$ and $-2H\Phi$ only. The $\frac{\partial}{\partial \mathbf{N}}$ terms can be reserved for the end of the evolution, saving a lot of computational power. The rest of the terms can be neglected for the 3D grids our computers allow us to treat currently.

The first two terms are clearly the most crucial. $\frac{\partial \Phi}{\partial \mathbf{S}}\mathbf{N}$ tends to make the surface rough at the beginning, whereas $-2H\Phi$ smoothes it in the end.

In order to reconstruct perfectly a right angle, the technique requires however a very fine grid (the surface is supposed to be C^2), or a fine grid and the $\frac{\partial}{\partial \mathbf{N}}$ terms. A multigrid approach avoids the computation of these terms which double (at least) the computation time.

We tried to add a balloon term on area of small curvature when the correlation score is high, in order to avoid local minima. Unfortunately the correlation function is not precise enough in some cases and the balloon term pushes sometimes the patch in a way that does not yield improvement in the correlation score.

As the term $-2H\Phi$ tends to minimize the surface, our region of interest need to be totally included in the initial surface. This curvature-dependent term allows us to get rid of small bumps on the surface, which would not have been the case in a voxel-based approach.

The computation stops when the speed of the zero level set is close to zero. All the parts seen by less than two cameras are removed.

5.5.4 Polygonal textured 3D model output

We keep all polygons seen by at least one camera, and output the result as an Open Inventor model.

Several simplification techniques can be envisioned, such as the VTK decimation method or some new techniques appeared during Siggraph'98.

We can also improve the output using median filtering (see [Coo98]) or voting algorithm to remove occluders from the final texture.

5.5.5 Results

The examples were run with grids of size $64 \times 32 \times 64$ or 64^3. The process took about 1 hour for the disk, 2 hours for the piece of the facade (2 images) and more than 12 hours for the whole sphere (22 images).

The disk and the sphere mapped with a checkboard texture are perfectly reconstructed, showing the power of the method on these synthetic images (figures 5-3 and 5-5). The illustrations show the superposition of the projected images using alpha blending. The surface evolves to make them correspond precisely with subpixel accuracy.

Observe how the program deformed the original box, how it splits sometimes small parts that correlate well locally and make them vanish. Note also some black spots due to the self-occlusion, which is taken into account and implemented by putting triangles into clusters and marching along rays.

Disk

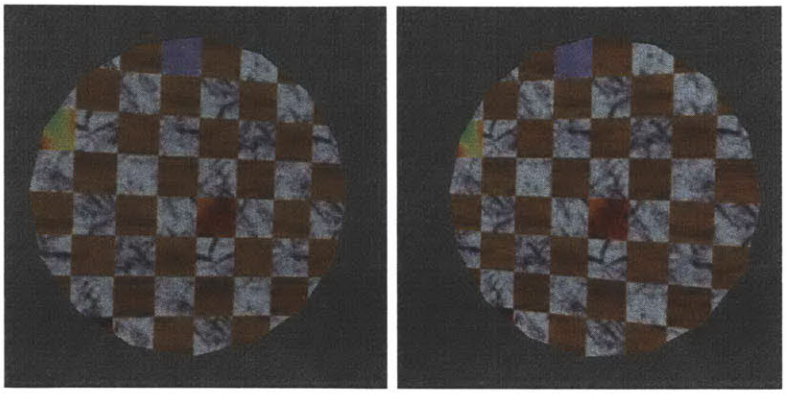

Figure 5-2: Disk - Starting images (512×512)

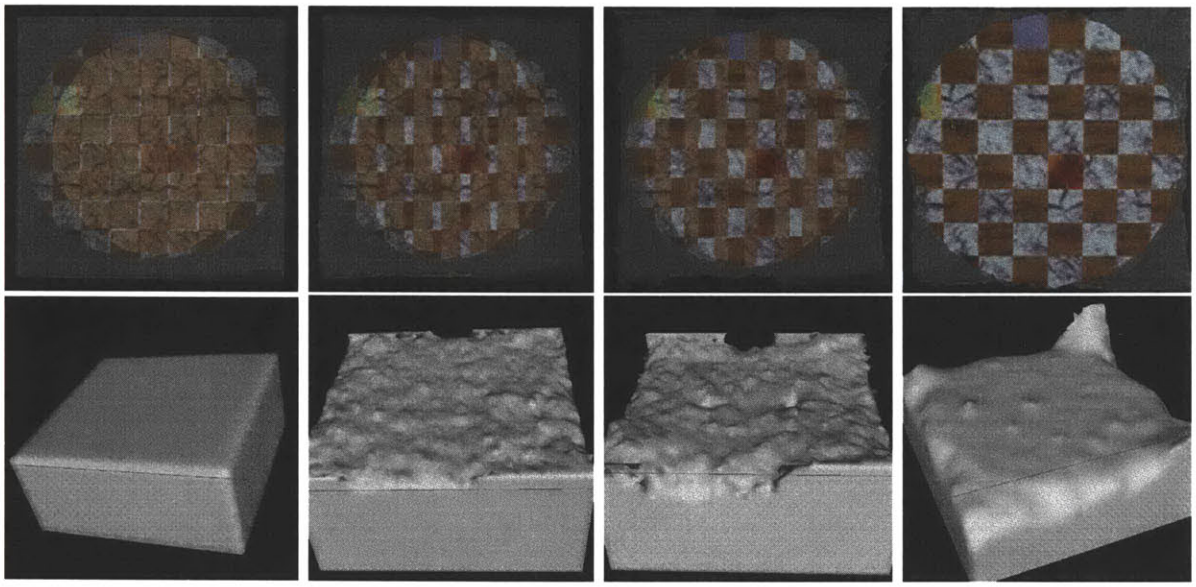

Figure 5-3: Reconstruction process example (Disk)

Sphere

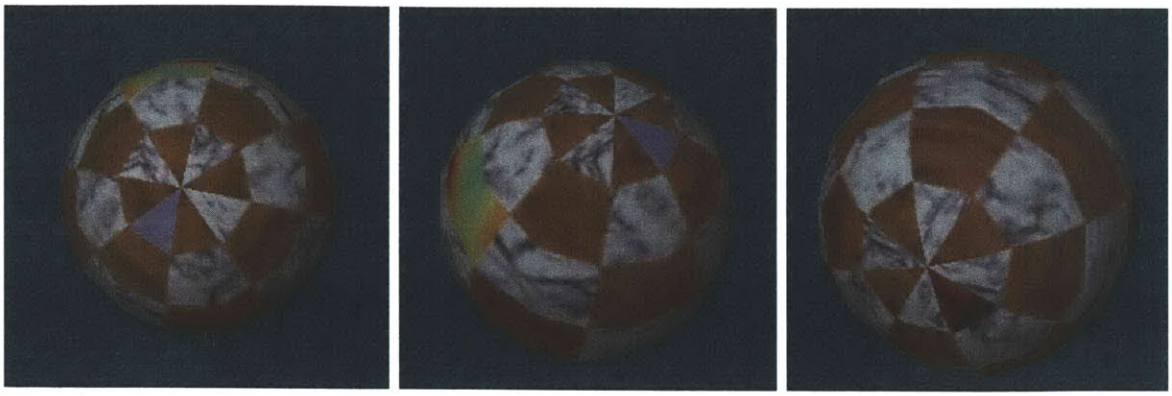

Figure 5-4: Sphere - Starting images (3 among 22, 256×256)

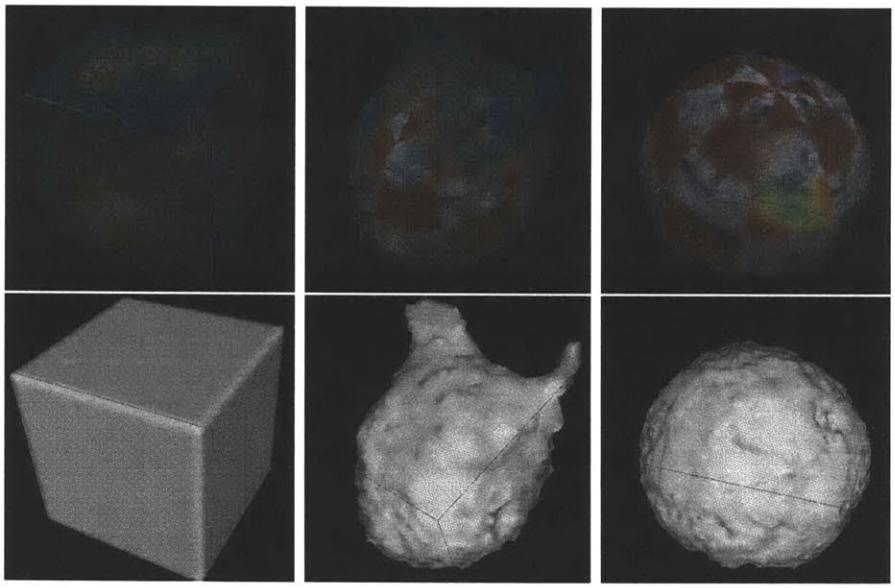

Figure 5-5: Reconstruction process example (Sphere)

5.5.6 Limitations

Light effects : the weaknesses of the correlation function

Reflections, shadows and highlights represent potential sources of failure for the reconstruction process. These are hard problems that the current method do not take into account. Extensions need to be developed to treat these cases properly. A transparent medium, such as glass, is also likely to be badly handled by the program.

Occlusions

Occlusions prevent images to match properly and give false information about the surface, since we do not have any a priori knowledge about them.

We observe in particular that the curvature-dependent term shrinks the corners of the initial surface, and often provoke a global collapse of the reconstructed surface. A solution to this problem is to apply a mask to the image, keeping the area of interest and filling the rest uniformly with black, so that the corners do not move during the evolution. This was done for the example of figure 5-7.

Impact of inaccurate calibration data

A building detail illustrates how crucial the precision on the calibration data is (see figure 5-7).

The horizontal and vertical bars of the window should correspond between the two images after evolution. However this can never happen, because of inaccurate camera pose information: the correct match does not belong to the epipolar line. Either the program aligns the vertical bars or the horizontal ones, but the crossing will never match; it becomes impossible to retrieve the exact shape.

Possible extension would be to allow the position of the camera to vary along the normal of the instantaneous epipolar plane and minimize the extrinsic parameters of the cameras in the same time.

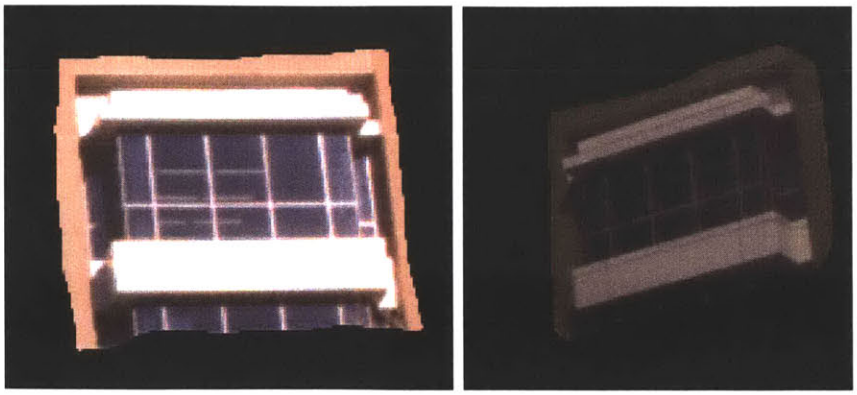

Figure 5-6: Starting images

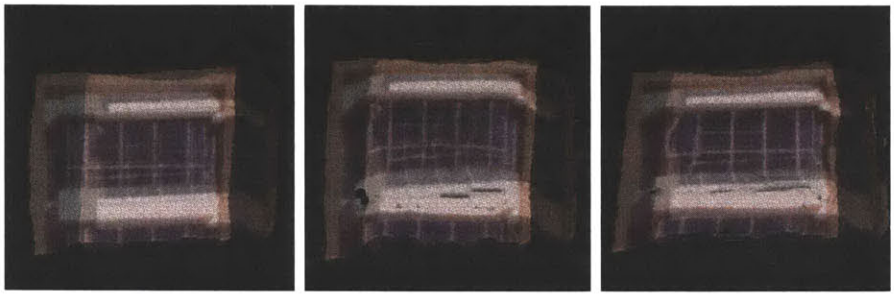

Figure 5-7: Importance of calibration data

Chapter 6

Extensions of this work

6.1 Reflectance models

To work on light and reflectance properties, we need a model of reflectance and illumination. We can distinguish several models, with different levels of refinement:

- Lambertian surface: $R = B\rho\vec{N}\cdot\vec{S}$, where R is the reflectance map of the surface, B the strength of the light source, ρ the albedo, $\vec{N}$ the surface normal and $\vec{S}$ the source direction.

- Specular surface: $R = B\delta(\theta_s - 2\theta_r)$, where B is the strength of the specular component, θ_s the angle between the light source and the viewing direction and θ_r is the angle between the surface normal and the viewing direction.

- Other models by Phong, Torrance-Sparrow [TS67], Healey-Binford [HB88], Nayar-Ikeuchi-Kanade [NIK91], Oren-Nayar [NO93], etc.

- The most complete model is the Bidirectional Reflectance Distribution Functions [Hor86].

6.2 Specularity and reflections

The simple detection of a fully reflective surface and the planar assumption of the reflective surface should be refined. Especially specularity in highly-curved region could complement the weakness of the correlation measurement.

Most techniques dealing with such issues often require us to know the shape of the surface as a hypothesis, which is not suitable here.

It would also be interesting to recover a simple material property description, which would not be the complete BRDF (too difficult) but a combination of Lambertian and Specular ideal surfaces.

A good basis of articles on the topic of using specularity to build shape can be selected among [NO95, ON95, SI96, YZOS97, ON97, BN95, BLL96, DS97, Sch94].

6.3 Shadows

If we have a good illumination model and the position of the sun (easy to acquire from time stamps and meteorological data), then we can predict self-shadows on our evolving surface and take advantage of them. Two methods are possible: correct the images for accurate linear correlation at the boundaries of shaded zones, or (more complex) track these boundaries to improve the accuracy of the geometrical reconstruction (see figure 6-1). We could take advantage of straight edges in urban environnement to look for straight edges in luminance space with continuous colors to detect shadows, then try to make them correspond to the images by evolving the surface. However we have to be careful about externally-cast shadows that come from other buildings and not the evolving surface itself.

Shadows contain information about the relief of the object but they are hard to distinguish. Two interesting articles on the topic, [YK96, FLB95]

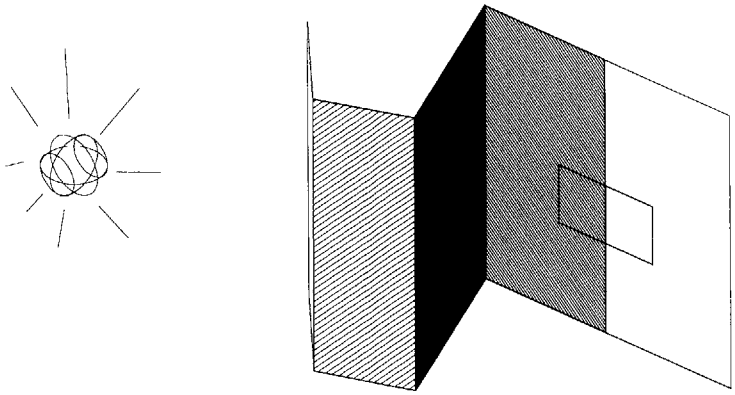

Figure 6-1: Shadow edge breaking the linear correlation

6.4 Highlights

The goal is to get information from them (therefore spot them) and remove them on the final texture when the shape of the object is retrieved (see [OSRW95]).

With a good reflectance model and some hypothesis on the nature of the surfaces seen, highlights can efficiently validate current shapes and give some accurate hints about highly curved areas, which would complement the fact that we locally approximate the surface by its tangent plane for image correlation.

Appendix A

Mathematical survival kit

A.1 Useful vector formulas

Lagrange identity:

$$\|\mathbf{a} \times \mathbf{b}\|^2 + (\mathbf{a} \cdot \mathbf{b})^2 = \|\mathbf{a}\|^2 \, \|\mathbf{b}\|^2$$

Gibbs formula:

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c}$$

$$(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{b} \cdot \mathbf{c})\mathbf{a}$$

Jacobi identity:

$$\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) + \mathbf{b} \times (\mathbf{c} \times \mathbf{a}) + \mathbf{c} \times (\mathbf{a} \times \mathbf{b}) = 0$$

Additionally,

$$\mathbf{a} \times (\mathbf{b} \times (\mathbf{c} \times \mathbf{d})) = (\mathbf{b} \cdot \mathbf{c})(\mathbf{d} \times \mathbf{a}) - (\mathbf{b} \cdot \mathbf{d})(\mathbf{c} \times \mathbf{a})$$

$$(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{c} \cdot \mathbf{a})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{d} \cdot \mathbf{a})(\mathbf{b} \cdot \mathbf{c})$$

The triple product is defined as

$$[\mathbf{a}\ \mathbf{b}\ \mathbf{c}] = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$$

from which we obtain

$$(\mathbf{a} \times \mathbf{b}) \times (\mathbf{c} \times \mathbf{d}) = [\mathbf{d}\ \mathbf{a}\ \mathbf{b}]\ \mathbf{c} - [\mathbf{a}\ \mathbf{b}\ \mathbf{c}]\ \mathbf{d} = [\mathbf{c}\ \mathbf{d}\ \mathbf{a}]\ \mathbf{b} - [\mathbf{b}\ \mathbf{c}\ \mathbf{d}]\ \mathbf{a}$$

More in Appendix A.2 of Robot Vision ([Hor86]).

A.2 Useful formulas for differentiation with respect to a vector

A.2.1 Differentiation of a scalar function

We recall that for $f : \Re^3 \to \Re$ and $\mathbf{r} = (x, y, z)^T$ vector of $\Re^3$

$$\frac{df}{d\mathbf{r}} = \nabla_{\mathbf{r}} f^T = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z})$$

A crucial observation is that we have defined a 1×3 vector (a linear form on $\Re^3$), which is consistent with the Jacobian matrix we will define below. In the general case, the derivative of an $n \times 1$ vector with respect to a $p \times 1$ vector is a $n \times p$ matrix.

Useful relations to know:

$$\frac{d}{d\mathbf{a}}(\mathbf{a} \cdot \mathbf{b}) = \frac{d}{d\mathbf{a}}(\mathbf{b}^T\mathbf{a}) = \mathbf{b}^T \quad \text{and} \quad \frac{d}{d\mathbf{b}}(\mathbf{a} \cdot \mathbf{b}) = \frac{d}{d\mathbf{b}}(\mathbf{a}^T\mathbf{b}) = \mathbf{a}^T$$

$$\frac{d}{d\mathbf{a}}\|\mathbf{a}\|^2 = 2\mathbf{a}^T\ ,\ \frac{d}{d\mathbf{a}}\|\mathbf{a}\| = \frac{\mathbf{a}^T}{\|\mathbf{a}\|} \quad \text{and} \quad \frac{d}{d\mathbf{a}}\frac{1}{\|\mathbf{a}\|} = -\frac{\mathbf{a}^T}{\|\mathbf{a}\|^3}$$

$$\frac{\partial}{\partial \mathbf{a}}\|\mathbf{F}(\mathbf{a}, ...)\| = \frac{\mathbf{F}^T}{\|\mathbf{F}(\mathbf{a}, ...)\|}\frac{\partial \mathbf{F}}{\partial \mathbf{a}} \quad \text{and} \quad \frac{\partial}{\partial \mathbf{a}}\frac{1}{\|\mathbf{F}(\mathbf{a}, ...)\|} = -\frac{\mathbf{F}^T}{\|\mathbf{F}(\mathbf{a}, ...)\|^3}\frac{\partial \mathbf{F}}{\partial \mathbf{a}}$$

$$\frac{d}{d\mathbf{a}}(\mathbf{a}^T\mathbf{M}\mathbf{b}) = \mathbf{b}^T\mathbf{M}^T \quad \text{and} \quad \frac{d}{d\mathbf{b}}(\mathbf{a}^T\mathbf{M}\mathbf{b}) = \mathbf{a}^T\mathbf{M}$$

In particular,

$$\frac{d}{d\mathbf{x}}(\mathbf{x}^T\mathbf{M}\mathbf{x}) = \mathbf{x}^T(\mathbf{M} + \mathbf{M}^T)$$

A.2.2 Differentiation of a vector function

For $\mathbf{F} : \Re^3 \to \Re^3$, the Jacobian matrix is

$$\frac{d\mathbf{F}}{d\mathbf{r}} = \begin{pmatrix} \frac{\partial F_x}{\partial x} & \frac{\partial F_x}{\partial y} & \frac{\partial F_x}{\partial z} \\ \frac{\partial F_y}{\partial x} & \frac{\partial F_y}{\partial y} & \frac{\partial F_y}{\partial z} \\ \frac{\partial F_z}{\partial x} & \frac{\partial F_z}{\partial y} & \frac{\partial F_z}{\partial z} \end{pmatrix}$$

Useful relations:

$$\frac{d}{d\mathbf{a}}(\alpha\mathbf{F}) = \alpha\frac{d\mathbf{F}}{d\mathbf{a}} + \mathbf{F}\frac{d\alpha}{d\mathbf{a}}$$

For any matrix $\mathbf{M}$, we have

$$\frac{d}{d\mathbf{a}}(\mathbf{M}\mathbf{a}) = \mathbf{M}$$

As

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \mathbf{b}$$

we have also (note the definition of $[\mathbf{a}]_\times$)

$$\frac{d}{d\mathbf{b}}(\mathbf{a} \times \mathbf{b}) = [\mathbf{a}]_\times = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} = -\frac{d}{d\mathbf{b}}(\mathbf{b} \times \mathbf{a})$$

Notice that $[\mathbf{a}]_\times^2 = \mathbf{a}\mathbf{a}^T - \|\mathbf{a}\|^2\mathbf{I}$.

A.3 Euler-Lagrange formulas

A.3.1 One function, one variable, first order

Problem: find an extremum of

$$I = \int_{x_1}^{x_2} F(x, f, f')dx$$

with $f(x_1) = f_1$ and $f(x_2) = f_2$ fixed.

If I is a extremum, then (necessary condition):

$$F_f - \frac{d}{dx} F_{f'} = 0$$

Without boundary conditions, we use the *natural boundary conditions*:

$$F_{f'} = 0 \quad \text{at } x = x_1 \text{ and } x = x_2$$

A.3.2 Higher order derivatives

Extremum of

$$I = \int_{x_1}^{x_2} F(x, f, f', f'', ...) dx$$

with $f(x_1) = f_1^0$, $f'(x_1) = f_1^1$,... and $f(x_2) = f_2^0$, $f'(x_2) = f_2^1$, etc. for all but the highest derivatives.

$$I \text{ extremum} \quad \Rightarrow \quad F_f - \frac{d}{dx} F_{f'} + \frac{d^2}{dx^2} F_{f''} - \cdots = 0$$

A.3.3 Several functions

Extremum of

$$I = \int_{x_1}^{x_2} F(x, f_1, f_2, ..., f_1', f_2', ...) dx$$

$$I \text{ extremum} \quad \Rightarrow \quad \forall i, \quad F_{f_i} - \frac{d}{dx} F_{f_i'} = 0$$

A.3.4 Several independent variables

Extremum of

$$I = \iint_D F(x, y, f, f_x, f_y) dx dy$$

with D simply connected region and $f(x, y)$ given on ∂D.

$$I \text{ extremum} \quad \Rightarrow \quad F_f - \frac{d}{dx}F_{f_x} - \frac{d}{dy}F_{f_y} = 0$$

The natural boundary conditions are in this case

$$F_{f_x}\frac{dy}{ds} = F_{f_y}\frac{dx}{ds} \quad \text{on } \partial D$$

where s is a parameter along the boundary.

Appendix B

User Manual

B.1 3D quad input

In order not to depend on output from other processes, the software has an interface to specify a 3D quad input. It provides us with a bounding box around our volume of interest for the next stage.

Figure B-1: Interface for 3D quad input and image loading

B.1.1 Mouse Input

Same model of interaction in all windows.

- *Left* button, no motion: picking.
- *Left* button, *motion*: trackball (3D) or dragging (2D).
- *Middle* button: translation.
- *Left+Middle* buttons: rotation about the viewing axis.
- *Right* button: zoom (move horizontally).

Every canvas has a *Reset* button, which resets the view to fit the viewport.

B.1.2 Image windows

The browser allows the user to switch between the listed images (click). The yellow arrow transfers the left image to the right window.

Move the vertices of the quads on two different images, taking care that at the end both images are displayed: a point by point triangulation yields a 3D quad. The right window displays either an image or the quad warped onto a square.

Ability is given to remove or add an image from the list loaded for the reconstruction process.

B.1.3 Camera files

Each image file must be associated with a camera file bearing the same name with the extension "`.cam`".

Easy loading of city scanning project images is possible through counters. The `-I` argument specifies the base directory of the city nodes and `-C` specifies the directory (from one level up) to examine if the camera file is not found in the current directory.

B.1.4 3D viewing

The 3D quad is reprojected onto the images by default (switch with *3D Preview* button), to get a taste of the accuracy of the triangulation. The user can then browse the other images to see its projection on them.

The 3D quad is updated only when a vertex of a 2D quad is moved. It requires of course a different image in the right window and a viable intersection (in front of both cameras).

The bottom-right window is a 3D visualization of the scene, with camera positions and orientations, and the relative geometry. The camera hilighted in red corresponds to the image we see in the left image window. The 3D quad – drawn in cyan – is surrounded by a bounding box when we launch the reconstruction process.

B.2 3d reconstruction

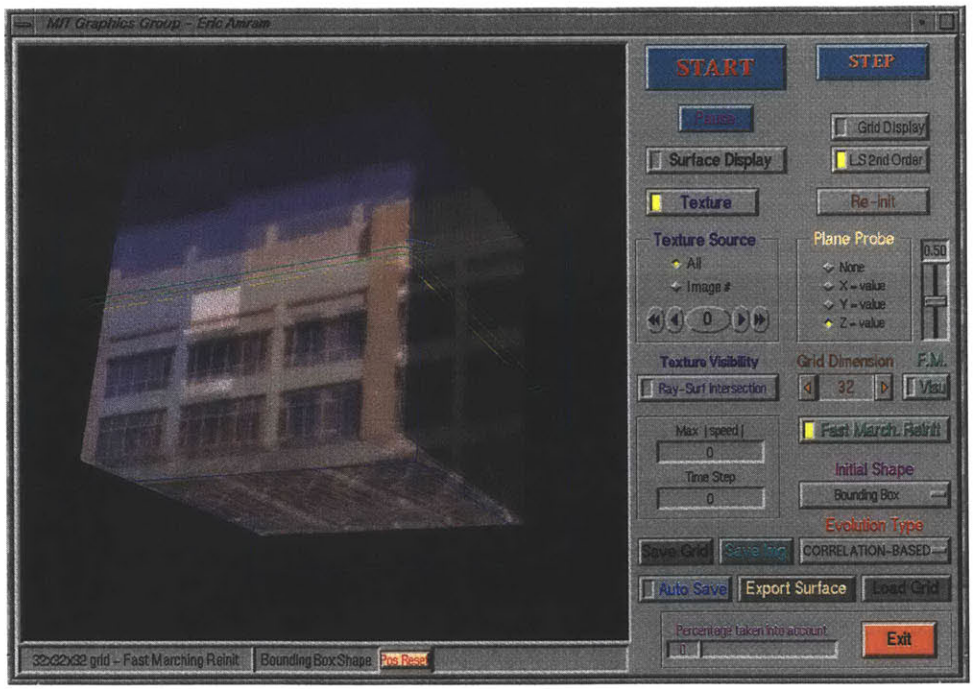

Figure B-2: Interface for surface evolution

B.2.1 Grid

Grid visualization

It is possible to display the grid as small colored segments at every grid point (button *Grid Display*). The color depends on the level set function value at that point.

Grid size

This counter allows the user to specify the resolution of the voxelized space. By default the size is $32 \times 32 \times 32$.

B.2.2 Initial shape and evolution type

Initial shape

The reconstruction usually starts with a bounding box, but nothing prevents us from using some other shape which includes the zone of interest.

Four initial shapes are available to better understand the evolutions in place and test different schemes : bounding box (default), sphere, two adjacent spheres, a star (7 spheres).

Evolution type

Correlation-based evolution reconstructs the 3D object by maximizing the correlation of reprojected camera images on its surface.

To illustrate the internal processes, some simpler evolutions are available, which all have constant speed: *shrink* makes the surface evolve with speed -1 along its normal vector, *grow* does the same with speed $+1$, *curvature* drives the surface with a speed equal to the opposite of its curvature (0 if planar).

B.2.3 Level Set related visualization capabilities

Planar probes

The level sets of the function are easily understood through the use of planar probes. In a plane that you specify by its normal axis (radio buttons) and one point (slider), you can visualize the zero level set (yellow) and integer level sets from -5 (red) to $+5$ (blue). According to the narrow-band technique, they are initialized by the distance function from the zero level set, and reinitialized periodically in the same way.

Surface

The zero level set is extracted using the marching cubes technique (set of triangles) and rendered as a polygonal surface.

Level set computation: second order and Fast Marching methods

You can switch on or off the use of second-order approximation for the level set function computation (button *LS 2nd order*). Nevertheless this has very little effect on the computation duration in the reconstruction process.

Note that a *percentage bar* indicates the quantity of voxels being processed during every step.

The *Fast March Reinit* button allows the user to choose between the Fast Marching method or crossing-times computation for the level set reinitialization (see section 3.2.4). The Fast Marching method is the default and is much faster. The user can visualize this technique by clicking on the button *F.M. Visu.*

B.2.4 Texturing

The images can be reprojected onto the current surface by switching on the *Texture* button. The user can either mix all images (by alpha blending) or choose to display only one thanks to the radio buttons and its associated counter *Texture Source.*

By default, all the vertices that are inside the frustum and facing the camera are displayed. To revome the parts hidden by the object itself, switch on the *Ray-Surf*

Intersection button.

The texture is recomputed at the end of a step only during the evolution.

B.2.5 Saving the results

The viewport, the grid, or an inventor model of the viewed surface can be saved to disk.

This can also be done automatically every n iterations.

B.3 Debugging window

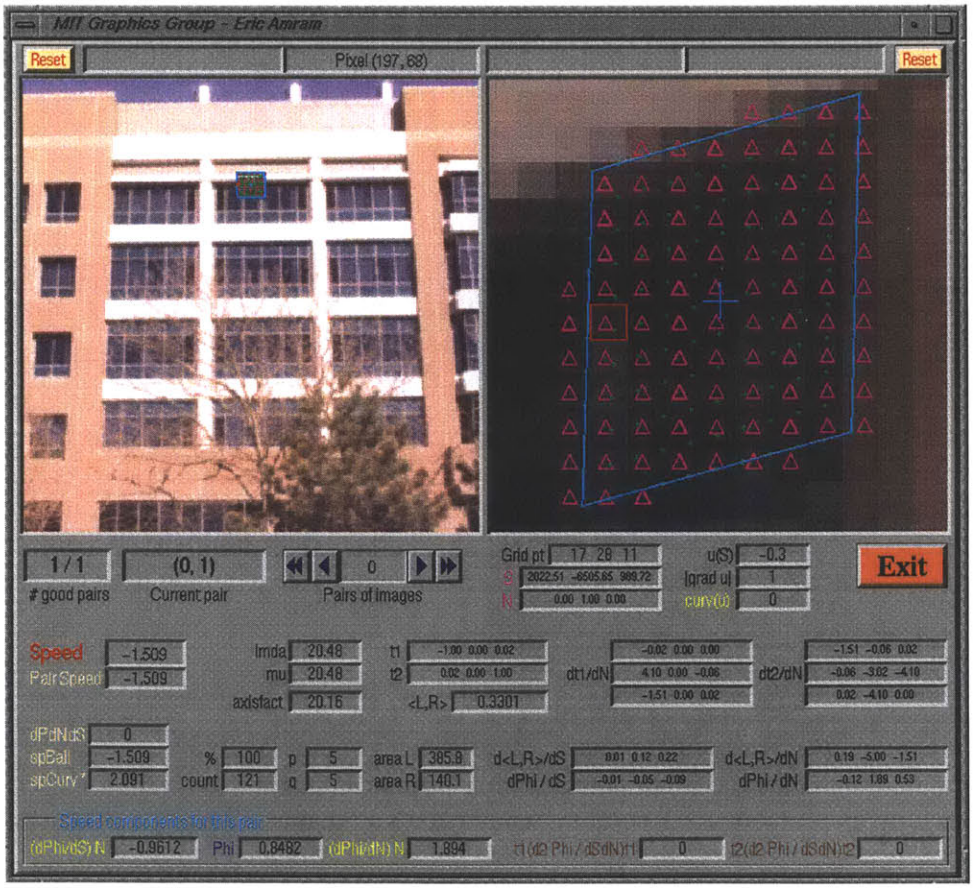

Figure B-3: Interface for internal process visualization

The debug window is activated by clicking on the surface or any displayed object. It visualizes the internal processing. Briefly, it especially renders the correlation

window reprojected on the images taken into account for the computation of the speed at this point.

B.4 Batch mode

The following options are available (`Recons3D-PDE -` shows this list).

`-I`	directory for city scanning project images.
`-C`	where to look for the camera file information if it is not in the current directory. Note that it is relative to one level up.
`-bb`	bounding box. Expects 6 numbers between two quotes, separated by spaces, in order to provide (Xmin,Xmax, Ymin,Ymax, Zmin,Zmax).
`-quad`	3D quad (12 numbers 'x1 y1 z1 x2...').
`-o`	basename for result export.
`-batch`	operates as a batch program.
...	provides the list of the images to load (words without `-`, shell extensions enabled).

Bibliography

[BB82] H.H. Baker and T.O. Binford. Depth from edge and intensity based stereo. In *Stanford AI*, 1982.

[BLL96] R.K. Bajcsy, S.W. Lee, and A. Leonardis. Detection of diffuse and specular interface reflections and inter-reflections by color image segmentation. *IJCV*, 17(3):241–272, March 1996.

[BN95] D.N. Bhat and S.K. Nayar. Stereo in the presence of specular reflection. In *ICCV95*, pages 1086–1092, 1995.

[BP92] M. Bichsel and A.P. Pentland. A simple algorithm for shape for shading. In *CVPR92*, pages 459–465, 1992.

[CKS95] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. In *ICCV95*, pages 694–699, 1995.

[CM92] S.D. Cochran and G.G. Medioni. 3-d surface description from binocular stereo. *PAMI*, 14(10):981–994, October 1992.

[Coo98] Satyan Coorg. *Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments.* PhD thesis, MIT Computer Graphics Group, September 1998.

[DF94] F. Devernay and O.D. Faugeras. Computing differential properties of 3d shapes from stereoscopic images without 3d models. In *CVPR94*, pages 208–213, 1994.

[DO92] P. Dupuis and J. Oliensis. Direct method for reconstructing shape from shading. In *CVPR92*, pages 453–458, 1992.

[DS97] L. Donati and N. Stolfi. Singularities of illuminated surfaces. *IJCV*, 23(3):207–216, 1997.

[ea93] Faugeras et al. Real time correlation-based stereo: algorithm, implementations and applications. Technical Report 2013, INRIA, August 1993.

[Fau93] O.D. Faugeras. Three-dimensional computer vision. *MIT Press*, 1993.

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

[FK97] O. Faugeras and R. Keriven. Level set methods and the stereo problem. In *ScaleSpace97*, 1997.

[FLB95] G. Funka-Lea and R. Bajcsy. Combining color and geometry for the active, visual recognition of shadows. In *ICCV95*, pages 203–209, 1995.

[Gri81] W.E.L. Grimson. A computer implementation of a theory of human stereo vision. *Royal*, B-292:217–253, 1981.

[Gri85] W.E.L. Grimson. Computational experiments with a feature based stereo algorithm. *PAMI*, 7(1):17–34, January 1985.

[HB88] G. Healey and T.O. Binford. Local shape from specularity. *CVGIP*, 42(1):62–86, April 1988.

[HB89] B.K.P. Horn and M.J. Brooks. Shape from shading. In *MIT Press*, 1989.

[HB97] Donald Hearn and M. Pauline Baker. *Computer Graphics, C version.* Prentice Hall, second edition, 1997.

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

[Hor90] B.K.P. Horn. Height and gradient from shading. *IJCV*, 5(1):37–76, August 1990.

[IA96] M. Irani and P. Anandan. Parallax geometry of pairs of points for 3d scene analysis. In *ECCV96*, pages 17–30, 1996.

[LB91] Y.G. Leclerc and A.F. Bobick. The direct computation of height from shading. In *CVPR91*, pages 552–558, 1991.

[MP79] D. Marr and T.A. Poggio. A computational theory of human stereo vision. *RoyalP*, B-204:301–328, 1979.

[NIK91] S.K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. *PAMI*, 13(7):611–634, July 1991.

[NO93] S.K. Nayar and M. Oren. Diffuse reflectance from rough surfaces. In *CVPR93*, pages 763–764, 1993.

[NO95] S.K. Nayar and M. Oren. Visual appearance of matte surfaces. *Science*, 267(5201):1153–1156, February 1995.

[OD93] J. Oliensis and P. Dupuis. A global algorithm for shape from shading. In *ICCV93*, pages 692–701, 1993.

[OK85] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *PAMI*, 7(2):139–154, March 1985.

[OK93] M. Okutomi and T. Kanade. A multiple-baseline stereo. *PAMI*, 15(4):353–363, April 1993.

[ON95] M. Oren and S.K. Nayar. Generalization of the lambertian model and implications for machine vision. *IJCV*, 14(3):227–251, April 1995.

[ON97] M. Oren and S.K. Nayar. A theory of specular surface geometry. *IJCV*, 24(2):105–124, September 1997.

[OSRW95] E. Ofek, E. Shilat, A. Rappoport, and M. Werman. Highlight and reflection-independent multiresolution textures from image sequences. Technical Report TR 95-1, Hebrew University, Institute of Computer Science, Jerusalem, Israel, Jan 1995.

[PMF85] S.B. Pollard, J.E.W. Mayhew, and J.P. Frisby. Pmf: A stereo correspondence algorithm using a disparity gradient limit. *Perception*, 14:449–470, 1985.

[Poy97] Charles A. Poynton. Frequently asked questions about color. http://www.inforamp.net/~poynton, March 1997.

[Saw94a] H.S. Sawhney. 3d geometry from planar parallax. In *CVPR94*, pages 929–934, 1994.

[Saw94b] H.S. Sawhney. Simplifying motion and structure analysis using planar parallax and image warping. In *ICPR94*, pages A:403–408, 1994.

[Sch94] H. Schultz. Retrieving shape information from multiple images of a specular surface. *PAMI*, 16(2):195–201, February 1994.

[SD97] S.M. Seitz and C.R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR97*, pages 1067–1073, 1997.

[Sei97] S.M. Seitz. *Image-Based Transformation of Viewpoint and Scene Appearance.* Ph.d. dissertation, University of Wisconsin, October 1997.

[Set96] J.A. Sethian. *Level Set Methods.* Cambridge University Press, 1996.

[SI96] Y. Sato and K. Ikeuchi. Reflectance analysis for 3d computer-graphics model generation. *GMIP*, 58(5):437–451, September 1996.

[SML98] Will Schroeder, Ken Martin, and Bill Lorensen. *The VISUALIZATION TOOLKIT.* Prentice-Hall, second edition, 1998.

[SS96] D. Scharstein and R. Szeliski. Stereo matching with non-linear diffusion. In *CVPR96*, pages 343–350, 1996.

[Sze91] R. Szeliski. Fast shape from shading. *CVGIP*, 53(2):129–153, March 1991.

[TDM96] C.J. Taylor, P.E. Debevec, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH96*, 1996.

[TS67] K.E. Torrance and E.M. Sparrow. Theory for off-specular reflection from roughened surfaces. *JOSA*, 57:1105–1114, 1967.

[TS92] P.S. Tsai and M. Shah. A fast linear shape from shading. In *CVPR92*, pages 734–736, 1992.

[YK96] D. Yang and J.R. Kender. Shape from darkness under error. In *ARPA96*, pages 1141–1148, 1996.

[YZOS97] J. Yang, D. Zhang, N. Ohnishi, and N. Sugie. Determining a polyhedral shape using interreflections. In *CVPR97*, pages 110–115, 1997.

[Zha97] Z.Y. Zhang. A stereovision system for a planetary rover: Calibration, correlation, registration, and fusion. *MVA*, 10(1):27–34, 1997.

[Zit96] C.L. Zitnick. Multi-baseline stereo using surface extraction. In *CMU-CS*, 1996.

[ZTCS94] R. Zhang, P.S. Tsai, J.E. Cryer, and M. Shah. Analysis of shape from shading techniques. In *CVPR94*, pages 377–384, 1994.