A Surface Reconstruction Method Using Global Graph Cut Optimization

Sylvain Paris[1]
MIT CSAIL
Cambridge, 02139 MA (USA)

François X. Sillion[2]
ARTIS, GRAVIR/IMAG-INRIA
38334 Saint Ismier (France)

Long Quan[3]
the department of Computer Science,
the Hong Kong University of Science and Technology
Hong Kong SAR (China)

[1] sparis@csail.mit.edu – Sylvain Paris has worked on this project during his PhD at ARTIS.

[2] francois.sillion@imag.fr

[3] quan@cs.ust.hk

Abstract

Surface reconstruction from multiple calibrated images mainly has been approached using local methods, either as a continuous optimization problem driven by level sets, or by discrete volumetric methods such as space carving. We propose a direct surface reconstruction approach which starts from a continuous geometric functional that is minimized up to a discretization by a global graph-cut algorithm operating on a 3D embedded graph. The method is related to the stereo disparity computation based on graph-cut formulation, but fundamentally different in two aspects. First, existing stereo disparity methods are only interested in obtaining layers of constant disparity, while we focus on high resolution surface geometry. Second, most of the existing graph-cut algorithms only reach approximate solutions, while we guarantee a global minimum. The whole procedure is consistently incorporated into a voxel representation that handles both occlusions and discontinuities. We demonstrate our algorithm on real sequences, yielding remarkably detailed surface geometry up to $1/10$th of a pixel.

Index Terms

Graph flow, graph cut, 3D reconstruction from calibrated cameras, discontinuities, self-occlusions, occlusions, global minimum

I. Introduction

In this paper, we consider the problem of the 3D reconstruction of a volumetric or surface representation of an object observed by several calibrated cameras. We target a classical scenario based on a short image set as shown in Figure 1. While providing a new and accurate solution to this practical problem, we also propose a general approach to several related issues.

Many researchers have been interested in this problem and have proposed different methods, including a geometric formulation solved by local optimization methods such as *space carving* [2] or *level sets* [3] and a discrete labeling formulation for computing stereo disparities solved by global *graph-cut* methods [1], [4]–[8]. These methods have their own strengths and weaknesses: Carving methods are intuitive but strongly rely on texture to achieve accurate results as shown by Kutulakos and Seitz [2]. Level sets propose a powerful geometric framework but the convergence of the process is not guaranteed and seems to depend on the starting point. Furthermore, disparity maps yield accurate contours but lack depth precision, as illustrated in Figure 1.

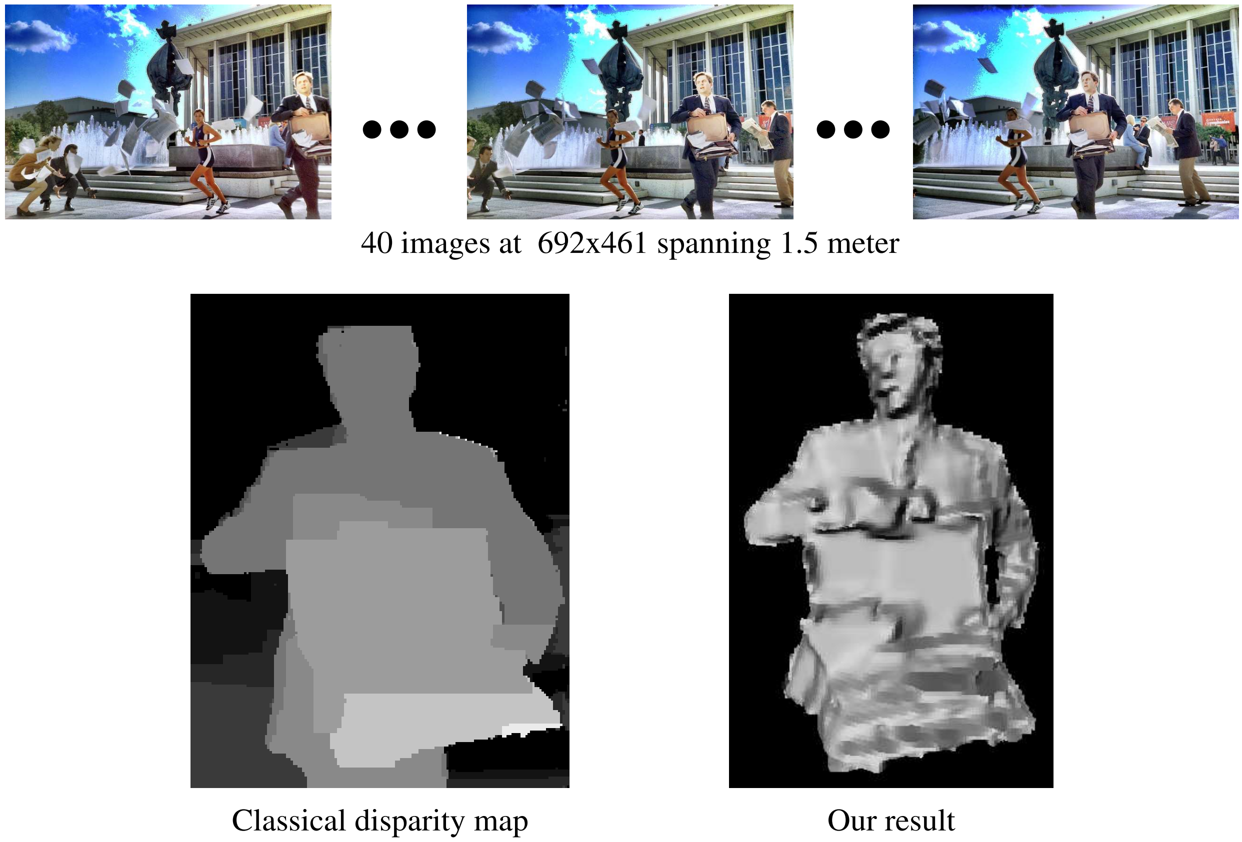

Fig. 1. This paper presents a method to build a 3D surface (bottom right) from an image sequence (top row). This result achieves a higher precision compared to a classical disparity map (bottom left) computed with an existing method [1] (the comparison is discussed later in the paper).

We introduce a new technique that uses a continuous geometric formulation that appears to be more suited to this problem as it is intrinsically related to the 3D space. A global optimization ensures a better object description as a whole rather than as a collection of small patches.

Most related methods are described in the following paragraphs.

Direct volumetric methods

The method of space carving, or voxel coloring [2], [9]–[12] directly works on discretized 3D space (or voxels) based on their image consistency and visibility. These techniques are characterized by their local treatment of the surface: Each voxel is considered separately. The methods differ in the way they update visibility. For efficiency purposes, some methods place constraints on the camera locations to simplify the visibility computation. A more detailed review can be found in [13]. Broadhurst *et al.* improve the technique using statistics to produce better images but the resulting surface still lacks details. Our method is different in that it is not purely local. Global constraints are added to handle the difficult situations of low texture surfaces and small baseline sequences.

Volumetric method based on level sets

Faugeras and Keriven [3] have proposed a variational approach solved with level sets. This approach naturally handles the topology and occlusion problems. Furthermore, they present a continuous formulation independent of the space discretization. Lhuillier and Quan [14] extend the method to handle various data types (3D points and silhouettes). Isodoro and Sclaroff [15] describe a technique which is not based on level sets but whose behavior seems similar. They guarantee that the silhouettes are invariant through the process. However it is not clear under what conditions these methods converge, since the proposed functional is highly non-convex.

Our method is different from this approach both in formulation and solution methods. We propose a highly modular technique that clearly separates the main problems (image consistency, topology, occlusion, surface localization) whereas the level-set method defines an all-in-one framework. Moreover, since we restrict our functional to a simpler expression, we guarantee that a global minimum is reached.

Disparity methods based on graph cuts

Originally, Roy and Cox [4], [5] have formulated graph-cut methods applied to 3D reconstruction as an extension of the existing approaches based on dynamic programming. The resulting surface is not clearly characterized by a functional but they have emphasized the robustness of the technique through its smoothing effect. An interpretation of the result as a labeling problem in the Markov Random Field framework is provided by [16]. Functionals with a concave smoothing term were introduced in [1], [8], [16]–[19]. This concave smoothing term handles surface discontinuities by bounding the influence of depth variation. Unfortunately, the resulting problem is NP-hard [17], and thus only an approximate solution can be reached in polynomial time. Ishikawa and Geiger [6], [20] follow a similar approach but restricted to a convex smoothing term which is proven to reach a global minimum in polynomial time [6], [21]. However, large depth variations can impair the results because they are not accounted for.

All these methods build disparity maps *i.e.* for each image point, they measure its 2D displacement between different images. This displacement is measured in pixels and each value is associated to a label – the problem is then to label each image point. Since the displacement is due to the perspective effect, it is directly linked to the depth of the corresponding 3D point. However, this approach is intrinsically image-based because it takes place in the image space and is limited to a discrete number of pixels. All these methods only handle a small number of labels (from 32 to 128) producing aliased results. Moreover, we show in Section III-C that most of these methods introduce spurious discontinuities because of the shape of their functional.

Also, most of these methods partially handle occlusions: If an object is occluded in some images but not in the others, it will not be reconstructed. Nevertheless, it seems reasonable to expect such a result because several cameras are unoccluded.

Geometric method based on graph cuts

The previous methods all rely on a disparity-based formulation. We believe that a geometric formulation is more natural and suitable to handle the volumetric reconstruction problem. For instance, it allows one to use geometric measurements such as the surface orientation which seems more meaningful than disparity in this specific context. Formally speaking, this geometric framework is similar to the one used in [3].

Recently, using such an approach, Boykov and Kolmogorov [22] have proposed a graph-cut method to compute geodesic surfaces for data segmentation. They use a Cauchy-Crofton formula that establishes a relationship between a surface and the 3D lines which intersect it. These intersected lines are naturally represented by the edges of the graph. The Cauchy-Crofton formula associated to the geometric framework leads to the edge capacities. The user places a *seed* inside each region to segment. The graph-cut technique then results in the segmentation of the data or equivalently in the surface separating the segmented data from the rest of the space. This surface is a global geodesic up to the chosen space and metric discretization. It also copes with partial topology changes: several seeds can merge in a single region but there cannot be a region without a seed or a seed without a region.

Despite different goals, some points are common with our method. Their graph is directly and fully embedded in the working geometric space and they also exactly solve a discretization of a continuous formulation. But the main departure between the two approaches is that their work is only useful for data segmentation whereas we target surface reconstruction. Although one may try to adapt their algorithm for 3D reconstruction, some non-trivial questions first need to be answered: where to put the *seeds* that initiate the segmentation and drive the topology, how to handle the visibility in a one-step process, how to account for discontinuities, etc. Furthermore their framework is defined for closed surfaces, whereas in many cases, we face an open-surface problem because the cameras are grouped on the same side of the scene (like in all the disparity-based techniques). We present a new graph-cut formulation which, like [22], has a strong geometric interpretation, but specifically, addresses the issues of a volumetric reconstruction.

Contributions

The main contributions of this paper can be summarized as follows:

- A new formulation of the surface reconstruction problem as a geometric optimization problem taking into account potential discontinuities of the object surface.
- A globally optimal graph-cut technique to solve this problem up to an arbitrary discretization.
- The integration with a voxel-based method to characterize object boundaries and account for visibility.
- A formal analysis of the continuity of the result leading to a method that avoids spurious discontinuities.

These improvements allow our method to reconstruct accurate 3D surfaces from real input images, even with very small baselines.

Overview

The rest of this paper is organized as follows. The motivation for a new functional is first discussed and defined in Section II. Then, a technique to compute a global minimum of the functional is detailed in Section III. Section IV details the algorithm that combines graph cut and voxels to achieve a surface reconstruction that accounts for occlusions and surface discontinuities. Section V discusses in depth the new formulation compared with existing ones. Finally, experimental results are presented in Section VI and we conclude with some perspectives on our approach in Section VII.

II. PROBLEM STATEMENT AND FORMULATION

Our goal is to create an object surface in 3D space. We first consider a general formulation that outlines the relations between our work and existing methods.

Let $(u,v) \mapsto \mathbf{X}(u,v) \equiv (x(u,v), y(u,v), z(u,v))$ be a regular parameterized surface representing the underlying object. Three-dimensional reconstruction can be cast as a variational problem of finding a suitable function or surface $\mathbf{X}$ that minimizes the functional:

$$\iint c(\mathbf{X})dudv,$$

where $c(\mathbf{X})$ is a positive function measuring the consistency of pixels in different images, often assuming Lambertian reflectance. Examples of such measurements are a function of pair-wise sums of squared difference (SSD, Zero-Mean SSD, or photo-consistency in color space) or of pair-wise cross-correlation function (CC or Zero-Mean Normalized CC). since reconstruction is ill-posed, this simple functional needs to be regularized to give smooth solutions.

Traditionally, the regularization terms are directly introduced for the parametric surface patch. The regularized problem can be formulated as a deformable surface model minimizing an energy [23]. One possibility is to consider the functional :

$$\iint \left(c(\mathbf{X}) + \alpha\, s\left(\frac{\partial \mathbf{X}}{\partial u}, \frac{\partial \mathbf{X}}{\partial v}, \frac{\partial^2 \mathbf{X}}{\partial u^2}, \frac{\partial^2 \mathbf{X}}{\partial v^2}, \frac{\partial^2 \mathbf{X}}{\partial u \partial v} \right) \right) dudv.$$

The second smoothing terms $s(\cdot)$ are

$$a\left|\frac{\partial \mathbf{X}}{\partial u}\right|^2 + b\left|\frac{\partial \mathbf{X}}{\partial v}\right|^2 + c\left|\frac{\partial^2 \mathbf{X}}{\partial u^2}\right|^2 + d\left|\frac{\partial^2 \mathbf{X}}{\partial u \partial v}\right|^2 + e\left|\frac{\partial^2 \mathbf{X}}{\partial v^2}\right|^2.$$

The minimization is solved by local methods *e.g.* a set of PDEs provided by the Euler-Lagrange equation. One common way is to use an iterative and steepest-descent method by considering a one-parameter family of smooth surfaces $\mathbf{X}(t) : (u,v,t) \mapsto (x(u,v,t), y(u,v,t), z(u,v,t))$ as a time-evolving surface $\mathbf{X}$ parameterized by the time t. The surface moves in the direction of the gradient of the functional p with the velocity $-\nabla p$, according to the flow $\frac{\partial \mathbf{X}(u,v,t)}{\partial t} = -\nabla p$. This is the Lagrangian formulation of the problem which describes how each point on the dynamic surface moves in order to decrease the weighted surface. The final surface is then given by the steady state solution $\frac{\partial \mathbf{X}}{\partial t} = 0$. The problem with this approach is well-known [24] in that it does not handle the changes in the topology.

In the approach developed in [3], the regularization is introduced intrinsically by considering the weighted minimal surface [25]. The weighted minimal surface is defined to be a minimizer of the functional

$$p(\mathbf{X}) = \iint c(\mathbf{X}) ds = \iint c(\mathbf{X}) \left\| \frac{\partial \mathbf{X}}{\partial u} \times \frac{\partial \mathbf{X}}{\partial v} \right\|_2 du\, dv.$$

The consistency function is a function of the correlation $\rho(\mathbf{X})$ between any pair of 2D images. Again the minimization is solved by an iterative steepest-descent method. However, further development shows that the formulation is intrinsic, *i.e.*, independent of any chosen parameterization, thus making the use of a level-set formulation possible. The level-set method [24], [26] regards the surface as the zero-level set of a higher dimensional function. The flow velocity is intrinsic *i.e.* dependent only on the surface curvature. Topological changes, accuracy, and stability of the evolution are handled by using the proper numerical schemes developed by Osher and Sethian [26]. Recently Boykov and Kolmogorov [22] showed that $\iint c(\mathbf{X})ds$ can also be minimized by a graph cut when $c(\mathbf{X})ds$ is a Riemannian metric. In contrast to the level-set method, thanks to its one-step design, this technique can cope with local minima but on the other hand it is impossible to adapt the functional according to the current estimate, which is a common way to handle occlusions.

The connection between these two different formulations, one with a multiplicative regularization term and another with an additive one, has been studied by many researchers [25]. In the case of 2D curves, these two formulations correspond respectively to geodesic active contours and classical snakes. These two formulations are equivalent [25]. It seems that their application to 3D surfaces is still an open question.

We propose the following functional

$$\iint \left(c(\mathbf{X}) + \left(\alpha_u(u,v) \left| \frac{\partial \mathbf{X}}{\partial u} \right| + \alpha_v(u,v) \left| \frac{\partial \mathbf{X}}{\partial v} \right| \right) \right) du\, dv.$$

Only first derivative terms are used for smoothing. Dropping the second derivative smoothing terms is primarily due to the optimization method we will introduce in the next section to solve the minimization problem. We also believe that the second derivative terms would lead to a complicated solution (for instance, the Euler-Lagrange solution of deformable models would contain fourth derivative terms) and to over-smoothed surfaces. Therefore, we believe the first derivative smoothing terms are sufficient, and even more desirable to capture fine geometric details, as demonstrated in our experiments. This formulation is also not intrinsic, and is therefore dependent on the parameterization. The $L1$ norm is used to fulfill the smoothing objective and leads to an efficient computation. However, our work can be extended to the $L2$ norm using the work of Ishikawa [6] with higher computation time. Note that α_u and α_v are not restricted to constants and thus make discontinuities possible because they allow local control of the smoothing.

Our approach lies between the geometric methods [3], [23] that are solved by local operators that only reach a local minimum of the functional, and the discrete techniques [1], [4]–[8], [16]–[18] driven by a graph-cut optimization. We keep the continuous geometric formulation as [3], [23] but guarantee to result in a global minimum as [6], [7]. In addition, the relationship between the continuous formulation and the discrete optimization process is clearly identified.

III. Global discrete solution

From the previously studied general framework, we set up a method that yields a discrete global solution to the object surface.

We propose a graph-cut technique to determine a solution to the discrete problem. The graph-cut technique solves the following general problem: given a water source linked to a sink through a pipe network, what is the maximum water flow that can reach the sink or, equivalently, where is the network bottleneck that limits the flow? Formally speaking, we have an oriented graph (the pipe network) with a *source* node and a *sink* node and each graph edge has two associated values: the *capacity* (the maximum flow through a pipe) and the *flow* (the actual flow in a pipe). A *cut* is defined as set of edges which separate the nodes into two connected components: one including the source and the other including the sink. Polynomial algorithms exist to find the *minimal cut* (the bottleneck), which is the cut with the lowest capacity from the source component to the sink component. The references [27], [28] give more details.

Link with surfaces

At first sight, the link between this "water flow problem" and a functional defined over a surface may not be obvious. To help the reader to be familiar with this concept, two simple examples are described. Neither of these examples are of practical use. They are only provided for illustrative purposes.

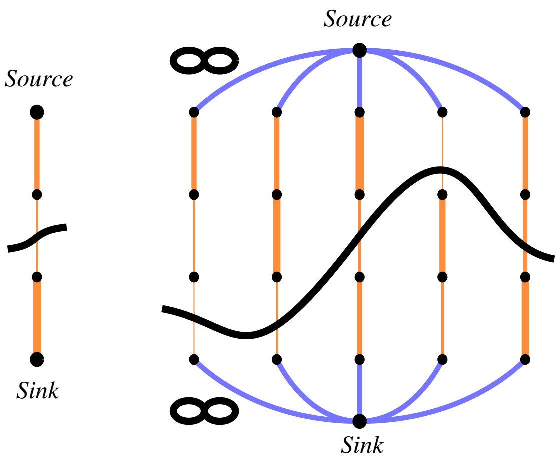

Fig. 2. Two simple examples: a graph to find the minimum value among 3 (left) and a graph to find the minimum values of 5 sets of 3 values (right).

Consider a set of n non-negative values. Let's use a graph cut to find the minimum of this set. Take n edges with capacities corresponding to the n values. Link these edges to form a *string*. The source and sink are the extreme vertices. The configuration is shown on Figure 2-left for $n = 3$. It is straightforward to see that the minimal cut (the bottleneck) is the edge with the smallest associated value.

Consider now m sets of n non-negative values. Let's use a graph cut to find the minimum of each set in a single run *i.e.* without using m times the previous algorithm. For each set, build a string as previously described. Then create two additional vertices: the source and the sink. For each string, use an edge with infinite capacity to link the source to one extremity and another such edge to link the other extremity to the sink. Figure 2-right shows the graph for $n = 3$ and $m = 5$. The minimal cut cannot cross an infinite edge, it therefore crosses an edge of each string. Again, it is straightforward to see that the minimal cut crosses the edge with the minimum value of each string. Notice that this cut "looks like" a curve.

The strategy of our graph-cut technique is to generalize this direction. Instead of considering a one-dimensional set of values that leads to a curve, we use a two-dimensional set that results in a surface. Adding edges between adjacent strings will account for the smoothing constraint.

Discretization

Without loss of generality, we assume that the 3D object space is described by (x,y,z) and the object surface is locally parameterized by a function, f: $\mathbf{X}(u,v,f(u,v))$, which can be seen as a depth function $z = f(x,y)$. Also let the domain on which f is defined be $\mathcal{D}$. This parameterization restricts the object being constructed to a single-valued depth field. If multiple depth values are needed, multiple functions $f_1, f_2, \dots$ could be used. Since x and y often play equivalent roles, the $x|y$ notation is used when a statement is applied to both x and y.

Our proposed functional consists of a consistency term $\mathcal{C}$ and a smoothing term $\mathcal{S}$:

$$\mathcal{C}(f) = \iint_{\mathcal{D}} c(x,y,f(x,y))dx\,dy, \tag{1}$$

$$\mathcal{S}(f) = \iint_{\mathcal{D}} \left(\alpha_x(x,y) \left| \frac{\partial f}{\partial x}(x,y) \right| + \alpha_y(x,y) \left| \frac{\partial f}{\partial y}(x,y) \right| \right) dx\,dy \tag{2}$$

Our solution strategy relies on approximating of equations (1) and (2) with a discrete formulation. Assuming the surface domain $\mathcal{D}$ is discretized as a regular rectangular grid, x, y, z have values $\{x_1, \dots, x_{n_x}\}$, $\{y_1, \dots, y_{n_y}\}$, $\{z_1, \dots, z_{n_z}\}$ separated by Δx, Δy, Δz. The extension to a general domain $\mathcal{D}$ with varying discretization steps is straightforward.

A discrete consistency term $\mathcal{C}^{\mathrm{d}}$ is obtained as:

$$\mathcal{C}^{\mathrm{d}}(f) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} c(x_i, y_j, f(x_i, y_j)) \Delta x \Delta y \tag{3}$$

and a discrete smoothing term $\mathcal{S}^{\mathrm{d}}$ as :

$$\begin{aligned} \mathcal{S}^{\mathrm{d}}(f) = & \sum_{i=1}^{n_x - 1} \sum_{j=1}^{n_y} \alpha_x(x_i, y_j) \left| f(x_{i+1}, y_j) - f(x_i, y_j) \right| \Delta y \\ & + \sum_{i=1}^{n_x} \sum_{j=1}^{n_y - 1} \alpha_y(x_i, y_j) \left| f(x_i, y_{j+1}) - f(x_i, y_j) \right| \Delta x. \end{aligned} \tag{4}$$

A. Building a first embedded graph

Our approach is based on a topologically embedded graph in the 3D geometric space; that is to say, it can be seen as a 3D geometric entity. Nodes and edges correspond to 3D points and 3D segments. Therefore, a cut is a real surface that crosses these segments. Moreover, edge capacities are fully expressed with geometric measurements. The main idea is to build the graph such that the cut capacity is equal to the surface functional. Then, a minimal cut will be a solution to the discrete problem. The graph is a 3D grid superimposed on the voxels with correspondence shown in Figure 3. All edges are bidirectional (*i.e.* made of two directional edges): for x-edges, the capacity is $\alpha_x(x_i, y_j)\Delta y \Delta z$; for y-edges, $\alpha_y(x_i, y_j)\Delta x \Delta z$; and for z-edges, $c(x_i, y_j, z_k)\Delta x \Delta y$. We add source and sink nodes out of this grid and for each (x_i, y_j) voxel column, the voxel with the minimum depth is linked to the source and the one with the maximum depth is linked to the sink. All source and sink edges have an infinite capacity. From a complexity point of view, there are three edges and one 6-connected node per regular voxel (*i.e.* not on a border). Thus, the graph complexity is proportional to the number of voxels.

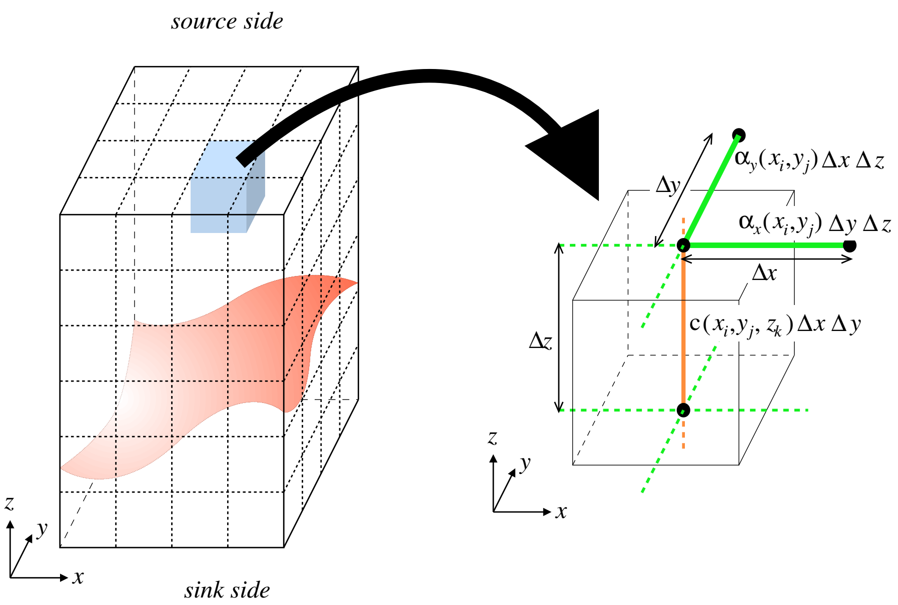

Fig. 3. A voxel grid with a surface inside (left). Correspondence between the voxel (x_i, y_j, z_k) and the graph (right).

Note that Roy [5] and Ishikawa [6] have also described embedded graphs with designs similar to ours. Roy associates each graph node to a disparity value although edges are the key entities in the graph-cut methods. One of the main reasons why the results of Roy's method are not well defined is that no functional or equivalently rigorous characterization is provided. Ishikawa embeds the graph in a generic Euclidean space but the links with the geometric space are unclear. In contrast to these approaches, our embedding is stronger: our graph is a 3D entity characterized by 3D quantities (*e.g.* geometric lengths) similar to [22].

B. Establishing a correspondence between a minimal cut and an object surface

We demonstrate a correspondence property which proves that our graph exactly computes a global minimum of the functional.

Correspondence property: There is an one-to-one correspondence between a subset of cuts called the *potential minimal cuts* and the surfaces defined by a function f. Moreover, the cut capacity is equal to the functional value of the corresponding surface.

The proof is based on necessary criteria for a cut to be minimal and a careful count of cut edges.

There are three necessary conditions for a cut to be minimal:

1) A minimal cut cannot cross an infinite edge;
2) It has to cross each (x_i, y_j) column at least once to separate the source from the sink;
3) It cannot cross a column more than once, otherwise the capacity would be higher than the single-crossing case.

A cut satisfying these three conditions is called a *potential minimal cut*. A direct one-to-one correspondence between such a cut and a surface exists: the cut is limited to the consistent voxel space (condition 1) and the single crossing point on the (x_i, y_j) voxel column (conditions 2 and 3) unambiguously determines $f(x_i, y_j)$.

The capacity of a potential minimal cut represented by f_{cut} can be computed as follows. The capacity of the crossed z-edges is exactly the consistency term $\mathcal{C}^{\text{d}}(f_{\text{cut}})$. Then, if we consider the two adjacent columns (x_i, y_j) and (x_{i+1}, y_j), the cut depths are $f_{\text{cut}}(x_i, y_j)$ and $f_{\text{cut}}(x_{i+1}, y_j)$. This implies that it crosses $\frac{1}{\Delta z}\left|f_{\text{cut}}(x_{i+1}, y_j) - f_{cut}(x_i, y_j)\right|$ x-edges. Thus, the total capacity of crossed x-edges is $\alpha_x(x_i, y_j)\left|f(x_{i+1}, y_j) - f(x_i, y_j)\right|\Delta y$. There is a similar result for the y-edges. The total capacity of all the x-edges and y-edges crossed by the whole cut is exactly $\mathcal{S}^{\text{d}}(f_{\text{cut}})$.

By adding these results together, we draw the expected conclusion: there is an exact correspondence between a surface with the discrete functional $\mathcal{C}^{\text{d}} + \mathcal{S}^{\text{d}}$ and a potential minimal cut with its capacity.

Therefore, we can solve for a global minimum in polynomial time using any minimal-cut algorithm.

C. Analyzing the smoothing term

Graph-cut techniques [1], [6]–[8] often yield flat and blocky results. This may not be fatal if we are only building a disparity map with limited precision (*e.g.*, 16 or 32 disparity values), but it becomes crucial if we target 3D shape reconstruction including small details and smooth slopes. We first elucidate the origin of this artifact, and then propose a solution by introducing a new smoothing term.

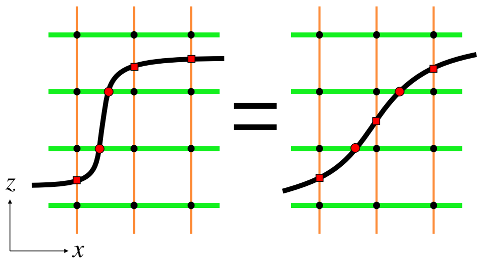

Fig. 4. A discontinuous variation and a continuous one are equivalent in a region with uniform consistency term: they cross the same number of x-edges (2 circles) and z-edges (3 squares). For clarity purpose, we present a 2D xz plane of the 3D grid.

Discontinuity artifacts appear in regions with depth variation and uniform smoothing and consistency terms. Consider the following simple example (restricted to 2D for clarity) where f is monotonic between x_A and x_B and $\alpha(x)$ is constant and equal to $\bar{\alpha}$:

$$\mathcal{S}_{\mathrm{2D}}(f) = \int_{x_A}^{x_B} \alpha(x) \left| \frac{\partial f}{\partial x}(x) \right| dx = \bar{\alpha}\, |f(x_B) - f(x_A)|$$

If $\alpha(x)$ is uniform then the smoothing term only depends on the extreme values of f ignoring its actual shape (Fig. 4). Therefore, if the consistency term is also uniform in that region, a continuous depth change and a discontinuous one yield the same functional value. This directly stems from the linear dependency of the smoothing term on the derivative. A convex term makes a continuous variation have a lower functional and a concave one makes it have a higher functional. There are three possible cases:

1) A convex smoothing term: it favors continuous variations but it may impede real surface discontinuities.
2) A concave smoothing term: it creates spurious discontinuities. Smoothing the surface would not be a solution because it would also remove the real discontinuities.

3) A linear smoothing term: it causes ambiguities because several surfaces have the same functional value. Unfortunately, it can be shown that, between several equivalent cuts, a graph-cut algorithm always "chooses" the most discontinuous one because it is the closest to the source or to sink. This leads to same artifacts than the concave case. (See Figure 13-a.)

This is summarized by the following property.

Smoothing term property: Concave and linear smoothing terms introduce spurious discontinuities on the surface. To overcome this artifact, the smoothing term must be strictly convex.

From this analysis, we do not use a concave smoothing term like [1], [29] nor a linear one like [7] because both introduce spurious discontinuities. We use a convex smoothing term while controlling it to allow real discontinuities to occur as described later in Section IV-B.

D. Building a second graph with a convex smoothing term

Applying the previous analysis to our first graph design (Fig. 3), the ambiguity inherent to the linear smoothing term is shown in Figure 4. This graph therefore needs to be adapted to incorporate a strictly convex component. We could have chosen the graph proposed in [6] but it introduces a constant in the smoothing term that can be difficult to handle in a multi-resolution context. We therefore propose a new graph based on the correspondence shown in Figure 5. We conserve the global 3D grid layout of the graph but there are now two kinds of $x|y$-edges: the $\alpha_{x|y}$-edges and the $\beta_{x|y}$-edges; the z-edges are split into 8 sub-edges. Nonetheless, the graph complexity is still proportional to the number of voxels since there are twelve edges, four 3-connected nodes and one 12-connected node per voxel. It adds an additional term $\mathcal{A}^{\mathrm{d}}$ in the functional:

$$\mathcal{A}^{\mathrm{d}}(f) = \sum_{i=1}^{n_x-1} \sum_{j=1}^{n_y} \beta_x(x_i, y_j) \left[\left| f(x_{i+1}, y_j) - f(x_i, y_j) \right| - \Delta z \right]^{+} \Delta y + \sum_{i=1}^{n_x} \sum_{j=1}^{n_y-1} \beta_y(x_i, y_j) \left[\left| f(x_i, y_{j+1}) - f(x_i, y_j) \right| - \Delta z \right]^{+} \Delta x \tag{5}$$

The correspondence property is still satisfied: The z-sub-edges are always cut by a group of four and therefore their total capacity is always equal to the consistency term $\mathcal{C}^{\mathrm{d}}(f_{\mathrm{cut}})$. Then, the $x|y$-axis, as the $\alpha_{x|y}$-edges correspond to the previous $x|y$-edges, always form $\mathcal{S}^{\mathrm{d}}(f_{\mathrm{cut}})$. Only the $\beta_{x|y}$-edges remain. Consider again two adjacent columns (x_i, y_j) and (x_{i+1}, y_j): there are $\frac{1}{\Delta z}\left|f_{\text{cut}}(x_{i+1}, y_j) - f_{cut}(x_i, y_j)\right|$ α_x-edges as previously shown and if this number is non-zero, because the cut can "go in the middle" of the z-sub-edges as seen in Figure 6, there is one less β_x-edge crossed. If we note $[\lambda]^+$ the value $\max(0, \lambda)$, the capacity of the crossed β_x-edges between the columns is $\beta_x(x_i, y_j)\left[\left|f(x_{i+1}, y_j) - f(x_i, y_j)\right| - \Delta z\right]^+ \Delta y$, which gives a strictly convex function. Hence, $\beta_{x|y}$-edges form $\mathcal{A}_{\text{d}}$.

As expected, with this convex term, the functional is lower for a continuous variation than for a discontinuous one as illustrated in Figure 6. This achieves our goal but the discrete $\mathcal{A}^{\text{d}}$ term (5) has no continuous counterpart like $\mathcal{C}^{\text{d}}$ with $\mathcal{C}$ and $\mathcal{S}^{\text{d}}$ with $\mathcal{S}$ because of the Δz term, which is directly linked to the discretization step. However, $\beta_{x|y}$ can be made small because we only need to penalize the step variations, no matter how important this penalty is. Therefore, $\mathcal{A}^{\text{d}} \approx 0$. Then, we can compute a close approximation of the discrete solution to the continuous functional while avoiding spurious discontinuities.

The last point to be observed is that crossing two z-sub-edges may lead to a lower capacity than crossing one β-edge. To avoid this situation, we can either add a constant on the z-sub-edge capacity (but it can be incompatible with a multi-resolution approach) or use *constraint edges* [6] (*i.e.* keep the same capacity for the oriented z-edges from the source to the sink and assign an infinite capacity to their reverse oriented edges) and then have no additional constant.

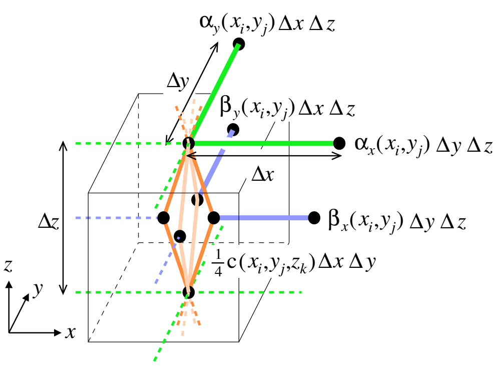

Fig. 5. Correspondence between the voxel (x_i, y_j, z_k) and the graph with a convex smoothing term. The 8 z-sub-edges have the same capacity.

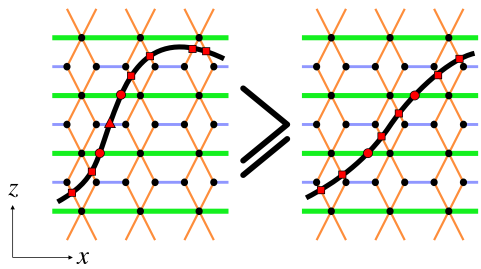

Fig. 6. The discontinuous variation crosses one more β_x-edge (the triangle) than the continuous variation. Thus, it has a higher functional value.

IV. Algorithm description

In the previous sections, we have described a general optimization tool. It has now to be adapted in a surface reconstruction framework. To demonstrate the capacity of this method, we have chosen a classical scenario which is well-known to raise precision difficulties and therefore challenges our technique. In the chosen configuration, we assume that there exists a separating plane: all the cameras are in the same half space and look toward the other half. This is a classical setup which is similar to many others [1], [7], [9], [18], [22]. This corresponds for instance to a short video sequence or to the situation in which it is impossible to go all around the scene. These two practical cases motivate this setup. We also assume that there is no moving object in the scene and since we are focusing on the reconstruction issue, the cameras are considered fully calibrated.

The proposed algorithm combines voxels and graph cut in a consistent way since both rely on the same space discretization. The advantage of this design is to deal with the visibility. Self-occlusions and object-by-object occlusions are handled: there are detected and if an object is partially hidden but still visible from a subset of cameras, it is reconstructed.

Before examining the details, we first give an overview of the algorithm illustrated by Figure 7.

Input: The algorithm uses a sequence of images, each image is given with its corresponding projection matrix which relates a 3D point to its projection in the image plane. The user also defines a bounding volume *e.g.* a 3D bounding box, containing the object of interest.

Initialization: Before entering the core of the process, the voxel space is built. The specific layout of the voxels ensure several properties including a new one which is proved. This allows a rigorous characterization of the surface boundaries.

Main loop: The algorithm is based on a multi-pass process. Each object is reconstructed one by one while visibility is progressively updated. Each pass first collects consistency information and localizes the potential discontinuities. Then the graph-cut optimization is run: it will be shown how to use it while accounting for self-occlusions. Before a new pass starts, visibility is updated (occluded regions are marked in images) so the next passes handle object-by-object occlusions.

Post-process: The resulting mesh is aliased because of the discrete representation of space. This artifact is removed with a specific PDE filter.

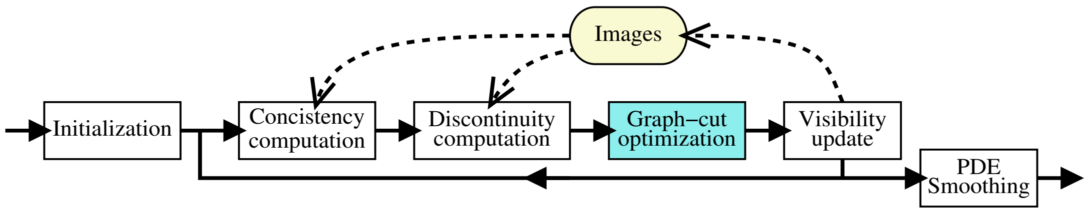

Fig. 7. Overview of the whole process. The consistency computation and the localization of the discontinuities are based on the input images in which the occluded regions are tagged at the end of each loop.

A. Initialization

The input bounding volume is discretized. This discrete space is seen in turn as a voxel space and a graph (see Figures 3 and 5). The space axes are determined as follows: the x-axis is defined by any two arbitrary cameras not orthogonal to the separating plane, the z-axis is orthogonal to x and going through the plane and $y = z \times x$. We then use a voxel configuration illustrated in Figure 8. The voxels are organized by planes $z = cst$: each plane is a scaled version of the $z = z_0$ plane. It fulfills the *constant footprint property* [12]: each voxel has the same projected area. There are other ways to achieve this property [10], [30] but this configuration also induces the *vertex coordinate property* [31] which characterizes the surface boundary $\mathcal{D}$. We here extend this property.

Vertex coordinate property: It characterizes the voxels near a surface of uniform color. As illustrated in Figure 9, a voxel near such a surface but not part of it can be consistent because all the lines of sight hit the surface and give the same color. Therefore, as shown in [2], surfaces of

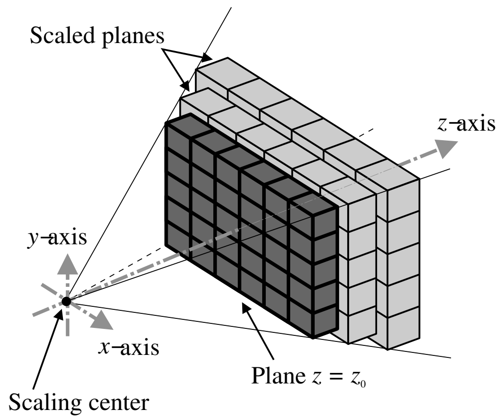

Fig. 8. Geometric configuration of our study case: cameras roughly lie on the x-axis and the voxel space is in the z direction.

uniform color[4] generate consistent regions which are bigger than the real surface. These regions go larger with smaller baselines. Therefore in our case, these regions are poor approximations of the real surface (Fig. 10-a). Nonetheless, these regions lead to useful properties.

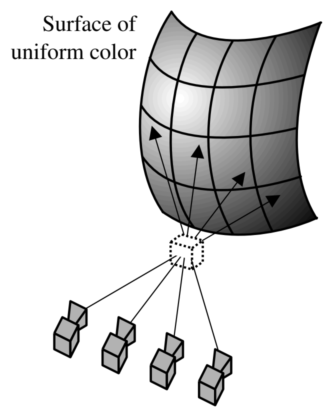

Fig. 9. A voxel that is consistent although it is not part of the surface.

In the following discussions, we assume that the surface color changes with its orientation relatively to the light and that the lighting is non-uniform. Conversely this implies that an uniform color surface is quite smooth and, for instance, cannot be peaked since it would appear

[4]To be precise, the surface must be uniform considering the consistency criterion used to define the c function. In many cases, this implies the color (*e.g.* photo-consistency under the Lambertian assumption).

of different colors because of the shading variations. We believe this hypothesis is always met by real scenes and therefore does not restrict our results.

Basic vertex coordinate property: For aligned cameras, if the scaling center used to build the voxels lies between the two extreme cameras, in an epipolar plane (*i.e.* a plane containing the optical centers) a surface of uniform color forms a region with consistent voxels. This region is a quadrilateral which vertices have one and only one extremal coordinate relatively to the voxel-grid coordinate system. Moreover, the left-most and right-most vertices belong to the surface.

Sketch of proof: The proof relies on the relationship between the camera positions, the angles between the lines of sight and the voxel boundaries, and the slopes of these lines of sight in the voxel coordinate system. This is illustrated in Figure 10-a. Then, the property is stated by examining these slopes. The complete proof can be found in [31].

This first property can be generalized to the case of non-aligned cameras as long as a separating plane exists. This results in the following property.

Extended vertex coordinate property: If the scaling center used to build the voxel grid lies on a segment linking any two cameras C_i and C_j, in the epipolar planes of C_i and C_j, the consistent voxels form 2D regions which left-most and right-most points belong to the surface.

Proof: Without loss of generality, consider two cameras C_1 and C_2 and name the C_1C_2 axis the x-axis, and then, as described previously, the z-axis is an orthogonal axis going through the separating plane and $y = z \times x$. The scaling center lies between C_1 and C_2. The configuration of an epipolar plane is illustrated in Figure 10-a.

We first restrict to this only two cameras. Using the basic vertex coordinate property, we know that the left-most and right-most vertices of the corresponding consistent region belong to the surface.

We consider now one more camera C_3 lying in a general 3D position. It makes a visual 3D cone with the surface of uniform color and then forms a 2D region in the epipolar plane as depicted in Figure 10-b. C_3 brings an additional constraint for the consistency, therefore the region consistent with C_1, C_2 and C_3 is the intersection between the initial quadrilateral and the new C_3 region. The resulting region may not be a quadrilateral or even a polygon anymore because C_3 may create a non-polygonal shape (Fig. 10-b). Nevertheless, since the previously characterized left-most and right-most points are part of the generating surface, they are consistent for any camera which

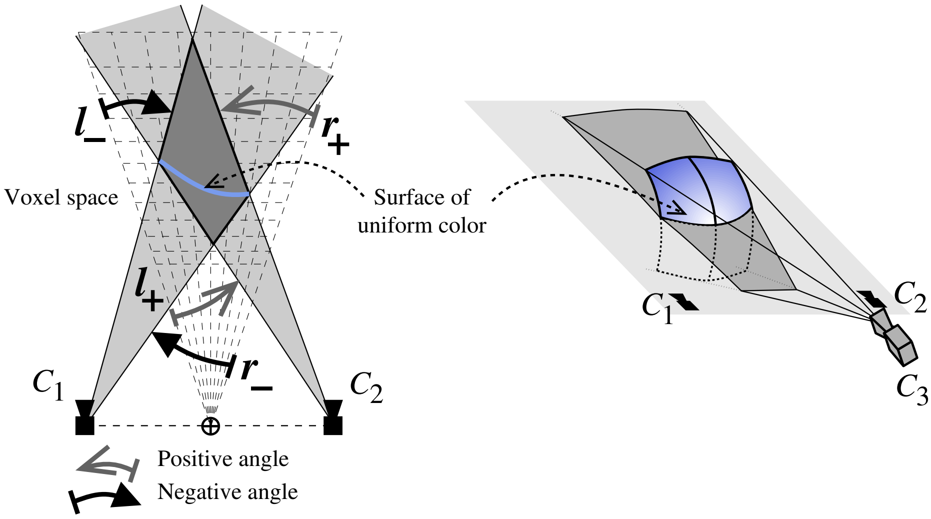

Fig. 10. (a) An epipolar plane of C_1 and C_2: the intersection of the two visual cones defines the consistent region. Note that the left-most and right-most vertices belong to the surface. (b) The consistent region in the epipolar plane due to a third camera C_3.

sees them. Hence they are part of the region consistent with C_1, C_2 and C_3 and because this region is only a restriction of the region for C_1 and C_2, they are still the left-most and right-most points. C_3 does not affect the property: the left-most and right-most points have not changed. This will therefore not change for any number of additional cameras. This demonstrates the extended property.

Thanks to this property, the optimization domain $\mathcal{D}$ is now defined. It is sufficient to chose two cameras C_i and C_j. The voxel space is then build as previously described: the scaling origin is set between C_i and C_j, the x axis is along C_iC_j, z goes through the separating plane (it is necessary to ensure the configuration of the angles in Figure 10-a) and $y = z \times x$. With this setup (Fig. 8), $\mathcal{D}$ is characterized by its left-most and right-most points in each epipolar planes (defined by C_iC_j).

B. Main loop

The algorithm has a multi-pass layout to reconstruct objects one by one, from the closest one to the most distant. The separating plane defines unambiguously this “distance”. It is a classical *front-to-back* approach [32] or it can also be seen as a *plane sweep* [2] in the z direction.

The iteration stops when no object remains unreconstructed in the scene.

Loop step 1: Consistency computation

For each voxel, consistency is computed. Then the voxels are thresholded to discard all the inconsistent voxels. Since we deal with real images, the resulting voxel set may contain some holes or isolated voxels. To overcome this point, the set is robustified with morphological operators as described in [31], [33]: a closure is first applied to fill in the holes and then an opening to remove isolated voxels.

In the rest of the pass, only the set of consistent voxels that is the closest to the cameras is considered.

Loop step 2: Discontinuities

Under the previously discussed and reasonable assumption that the aspect of a surface depends on its orientation, surface discontinuities result in image discontinuities. This remark is used to control the two functions α_x and α_y that appear in the smoothing term (eq. 2). These functions drive the continuity of the surface: If they are null, the surface is free to be discontinuous whereas for positive values, discontinuities are penalized.

We propose to define these functions as $\alpha_{x|y}(x,y) = A\chi_{x|y}$ where A is a constant which corresponds to the desired smoothing strength when there is no discontinuity and the functions $\chi_{x|y}$ is a *discontinuity factor* varying between 0 (discontinuity) and 1 (no discontinuity). $\beta_{x|y}$ is defined similarly using $B\chi_{x|y}$ with constant B such that $B \ll A$.

We now expose how to characterize potential surface discontinuities. The input images are used for this purpose: the color of each surface point is estimated relatively to the input images. But since this step occurs before the graph-cut optimization, we use the voxel approximation which a poor geometric approximation but is sufficient for this task. In practice, we compute the color of each voxel by averaging its color in each camera and the color of the surface at a point (x_i, y_j) is the average of the colors of all the voxels in the corresponding column. This estimation is reasonable because as discussed previously, depth uncertainty (*i.e.* several voxels in a (x_i, y_j) column) results from surfaces of uniform color. Therefore, only similar colors are averaged in this estimation.

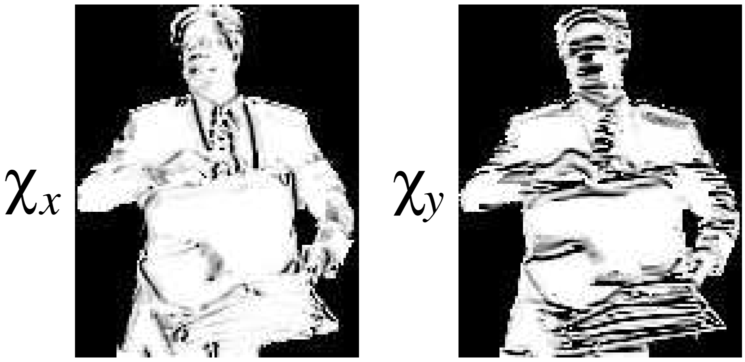

Fig. 11. The χ_x and χ_y functions (black = 0 and white = 1) for the briefcase man (Fig. 14).

Two *discontinuity maps* D_x and D_y are computed. D_x is related to the color difference between (x_i, y_j) and (x_{i+1}, y_j) (or (x_i, y_{j+1}) for D_y). As the estimation of the color discontinuities has to be independent of the local contrast (for instance, the estimation should not change between bright or dark lighting conditions), it is normalized by the standard deviation $\gamma_{\mathcal{N}_x^{ij}}$ of the color in a small neighborhood $\mathcal{N}_x^{ij}$ (*e.g.* 6×5 centered on $(x_{i+\frac{1}{2}}, y_j)$). Small values of $\gamma_{\mathcal{N}_x^{ij}}$ cause numerical instabilities and make detect spurious discontinuities. We therefore apply a threshold Γ according to the values of $\gamma_{\mathcal{N}_x^{ij}}$:

$$D_x(x_i, y_j) = \begin{cases} 0 & \text{if } \gamma_{\mathcal{N}_x^{ij}} < \Gamma \\ \frac{\text{distance}(\text{color}(x_i, y_j), \text{color}(x_{i+1}, y_j))}{\gamma_{\mathcal{N}_x^{ij}}} & \text{else} \end{cases} \tag{6}$$

We use the equivalent formula for $D_y(x_i, y_j)$ to get:

$$\chi_{x|y}(x_i, y_j) = \left[1 - D^2_{x|y}(x_i, y_j)\right]^+ \tag{7}$$

Remark: Our approach is different from existing methods. Some of them [3], [7] consider discontinuities as normal depth variations. This is obviously a drawback for important depth changes. Or the others handle discontinuities in their optimization process but without controlling their localization [1], [8], [16]–[18], [29] (*i.e.* the optimization process "decides" on its own where are the discontinuities). Our method adds this important issue of discontinuity localization according to the input images.

Loop step 3: Graph cut with self-occlusions

To account for self-occlusion, our graph-cut technique is adapted according to the geometric configuration. Adapting the functional is straightforward, it is sufficient to add a *visibility term*[5] $\mathcal{V}(f)$ which is infinite if f corresponds to a self-occluding surface and 0 in all the other cases. Obviously, no self-occluding surface can be minimal for this functional.

Then, *visibility edges* are added to the graph to ensure the correspondence property. Let's consider a voxel A occluded by a voxel B in an adjacent column (Fig. 12-a). An oriented edge with infinite capacity is added as shown in Figure 12-b. This edge is oriented from the sink to the source. Therefore any surface including both A and B crosses this edge with the orientation that makes the edge count in the cut value. Note that any surface including a voxel behind A and/or a voxel in front of B is self-occluding and crosses also this infinite edge as it should be. This proves that these edges correspond to the visibility term for adjacent columns. As the relationship "is occluded by" is transitive, this is sufficient to ensure the whole visibility

[5]This term is not the same as [1] but has an equivalent effect when considering one disparity map: self-occlusion cannot appear.

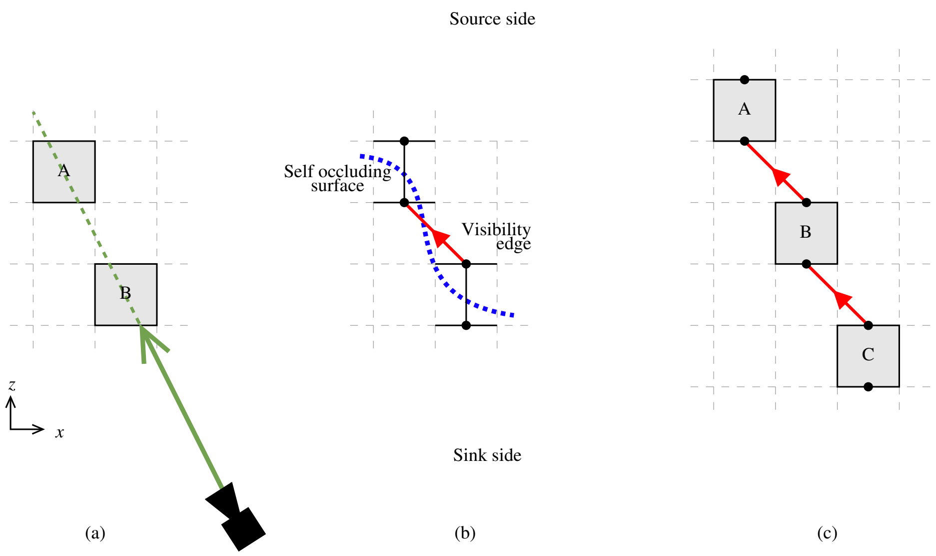

Fig. 12. For illustration purpose, only a *xz* plane is presented and the graph is the simple non-convex one (Fig. 3). (a) A is occluded by B. (b) The BA visibility edge and a surface containing both A and B. (c) The BA and CB edges are sufficient to handle the occlusion between C and A.

constraint as illustrated by the Figure 12-c: if A is occluded by B which is occluded by C then A is occluded by C and any surface included A and C must cross either the BA edge or the CB edge.

Hence, it is sufficient to add four visibility edges per voxel to have a graph that accounts for $\mathcal{V}(f)$. However, in practice the results we have obtained without these edges are never self-occluding. Therefore, it seems they can be omitted to alleviate the computation without any loss of quality.

Loop step 4: Visibility

After each pass, the lines of sight blocked by the previously reconstructed objects are computed and ignored in the following passes. In practice, the reconstructed objects are projected in the input images and the covered pixels are flagged "occluded".

C. Post-process: Mesh smoothing

As the whole process up to this step is purely discrete, it suffers from aliasing artifacts. A smoothing filter, inspired by PDE based image denoising [34] is well adapted to a geometric surface because it is driven by the principal curvatures κ_1 and κ_2. Also, because the potential discontinuity lines are already located, the PDE smoothing filter is adapted to avoid diffusing across these lines.

First, since $\chi_{x|y}$ evaluates the discontinuity between two adjacent columns, we define $\hat{\chi}_{x|y}$ a column-centered function $\hat{\chi}_x(x_i, y_j) = \frac{1}{2}\left(\chi_x(x_i, y_j) + \chi_x(x_{i-1}, y_j)\right)$ and equivalently for $\hat{\chi}_y$. To adapt to the directions θ_1 and θ_2 of the principal curvatures, we define a discontinuity factor $\hat{\chi}_\theta$ for the direction θ with the classical interpolation: $\hat{\chi}_\theta = \cos^2(\theta)\,\hat{\chi}_x + \sin^2(\theta)\,\hat{\chi}_y$

Thus, we can formulate the filter as a surface evolution driven by the following PDE with virtual time t:

$$\frac{\partial f}{\partial t} = \hat{\chi}_{\tilde{\theta}_1} g(\tilde{\kappa}_1)\kappa_1 + \hat{\chi}_{\tilde{\theta}_2} g(\tilde{\kappa}_2)\kappa_2 \qquad (8)$$

where g is a *stopping function* that controls diffusion to preserve the curvatures (its role is discussed in detail in [35]), $\tilde{\kappa}$ and $\tilde{\theta}$ are computed on a Gaussian filtered version of the surface, which leads to more robust estimations to control the filter [36]. Note that there are two controlling components: g driven by the surface geometry and $\hat{\chi}$ accounting for color discontinuity. These assure that both curvature and discontinuities are preserved.

V. Discussions

A discussion of comparisons of our method with existing methods is given in this section.

A. Global versus local minimum

The main difference between our approach and other existing methods [1], [3], [8], [16]–[18] is that we find an exact global minimum of the discrete functional. Other approaches use variational techniques: for level set methods, the variation is given by the Euler-Lagrange relation and for graph-cut methods it comes from an α-*expansion* [17]. These methods suffer from the same limitations:

- They are dependent on the starting point. Different results may be obtained from different starting points.
- They only reach a local minimum. At best, Boykov et al. [17] have demonstrated a highest bound of the error committed by their method on the functional. Currently, the geometric error is unknown.

We can determine this global minimum because we use a convex smoothing term coupled to an *a priori* discontinuity detection which results in a polynomial problem opposite to [1], [8], [16]–[18] which rely on a concave term to handle discontinuities. Moreover our discontinuity handling seems more consistent as it is driven by input images as discussed in Section IV-B.

(a) (b) (c) (d) (e)

Fig. 13. 3D reconstructions with various parameters (a) Without PDE filter and with a linear smoothing term: spurious discontinuities (on the shoulders and on the arms) (b) Without PDE filter and discontinuity maps: details are blurred (on the face and in the briefcase) (c) Without PDE filter: aliasing. (d) An input image. (e) Final result with all our contributions.

B. *Geometry versus labels*

Another important feature of our approach compared to the existing graph cut techniques [1], [6], [8], [16]–[18] is its geometric formulation. Up to now, graph-cut methods have mainly described the problem in terms of image disparity: different disparity values are selected and each image pixel has to be labeled with one of those values. Even if there is a link between the disparity and the depth, none of these methods has used it to characterize the functional. Therefore, the relation between the discretized resolution and the different terms of the functional is unclear: for instance, if one wants to work with a resolution twice coarser, it is not clear how to set the various values to ensure that the same underlying problem is solved. On the other hand, our approach makes explicit the measure functions dx, dy and dz whose relation with Δx, Δy and Δz is straightforward and then any change of resolution can be handled while solving the same underlying problem.

From this point of view, our method is quite similar to the level-set approach that first defines a continuous geometric functional before looking for a solution with a specific method. But our solution technique is purely discrete whereas the level-set is a local numerical method.

C. *Occlusions*

Our algorithm through the multi-pass design is able to handle strong occlusions for reconstructing partially hidden objects. From this point of view, these results are equivalent to those obtained with space-carving [2] or level-set methods [3]. They are more general than those obtained with disparity maps [1], [6]–[8] which detect occlusions but do not reconstruct hidden objects. Contrary to these methods, ours handles visibility to reconstruct objects which are only visible by a subset of the cameras.

D. *Open and closed surfaces*

The presented method is designed to reconstruct the front-facing parts of the objects because the viewpoints are grouped on the same side of the scene. Therefore, the surfaces we are dealing with are intrinsically open. This has to be compared with the disparity maps which are also open surfaces if one considers them as 3D entities. The main difference is that a disparity map builds a single surface representing the whole scene whereas our algorithm may results in several surfaces, one for each object. This difference is a key feature to handle occlusions and reconstruct partially hidden objects. This also explains is why a disparity map cannot "go behind" partial occlusions. The price for this feature is that the non-trivial boundary problem has to be solved. In this paper, a specific voxel setup and the general vertex coordinate property are proposed and demonstrated to characterize the surface boundaries. So the framework is complete and we can both determined the boundaries and the surfaces.

Then, this open-surface approach has also to be compared with the closed-surface techniques based on voxels [2], level sets [3] or graph cuts [22]. Theses methods have the obvious advantage to fully reconstruct the objects. However, this clearly implies that there are many views all around the scene to ensure no hidden part. Such a configuration is not always available and therefore the corresponding algorithms cannot be applied. Nonetheless, we consider in future work to develop and adapt our implementation to closed surface by working with a specific parameterization of the surface. This clearly deserves deeper studies.

VI. Experimental results

The implemented system has been demonstrated on many real examples. We show three typical sequences: a briefcase man, keyboard, and lantern sequences illustrated in Figures 14, 17 and 18. There are 40 frames for the street from Dayton Taylor's time-freezing setup with aligned and regularly spaced cameras. Except the time-freezing system which captures attractive "moving" scene, this setup is rather similar to short video sequence: baseline is short (40 images spanning 1.5 meter), image resolution is limited (about 40×40 for the face of the man), the images contain a significant amount of noise and an exposition change occurs throughout the sequence due to the back light. This sequence can be considered as representative of an input provided by a non-specialist user. It is calibrated by a commercial system with about 70 points manually extracted from the sequence. There are 11 frames of resolution 640×480 for the keyboard sequence and 23 frames of 800×600 for the lantern sequence captured by a hand-held digital camera. The geometry of the cameras for these sequences has been automatically computed using the system of Lhuillier and Quan [37]. To compute effective solution for results, there is a broad range for the definition of consistency. For simplicity, we use the *photo-consistency* [2], [9] in the *hue-saturation-value* color space for our current examples.

The space resolution ranges typically from 1 to 10 million voxels with 5 nodes and 12 double edges per voxel (Fig. 5). The precision of the reconstruction results is very high. We notice even the geometric details on the face of the man (Fig. 14), which comes from only a small patch of about 40×40 pixels. Almost every key on the keyboard is reconstructed and distinguishable (Fig. 17). We measured the physical size of the keyboard and the keys. This gives a 1/10 pixel accuracy.

Fig. 14. Man with briefcase reconstruction (right) 40 images at 692×461, face is about 40×40 (left). Notice that the discontinuities are preserved.

Figure 18 shows a case with strong occlusion. The folded chess-board is hidden by the lantern in half of the images. Nevertheless, our algorithm is able to exploit the remaining unoccluded views to rebuild it.

We have tested the behavior of the technique with decreasing number of cameras. We used the sequence of the man with briefcase that is a rather difficult input as discussed previously. With 20 images among a total of 40 (Fig. 15-middle), some details are slightly blurred away on the face and the briefcase but the overall accuracy is almost the same. With 10 images, (Fig. 15-left), some spurious geometry appears on the briefcase and the contours are less precise. However, the algorithm still performs well and reaches satisfying results. We have tested with only 5 cameras, the quality loss becomes unacceptable: Large spurious shapes appear due to the background and the silhouettes are too degraded. This result does not seem reasonably usable. So we have reached the limit of our technique. It shows that it works well for 10 cameras or more, even with non-perfect images. This number depends on the "quality" of the images. The input images used in Figure 15 are significantly noisy (outdoor scene, contrast has been raised and the time-freezing system is likely not to be perfectly synchronized). Therefore, it can be considered as a "worst

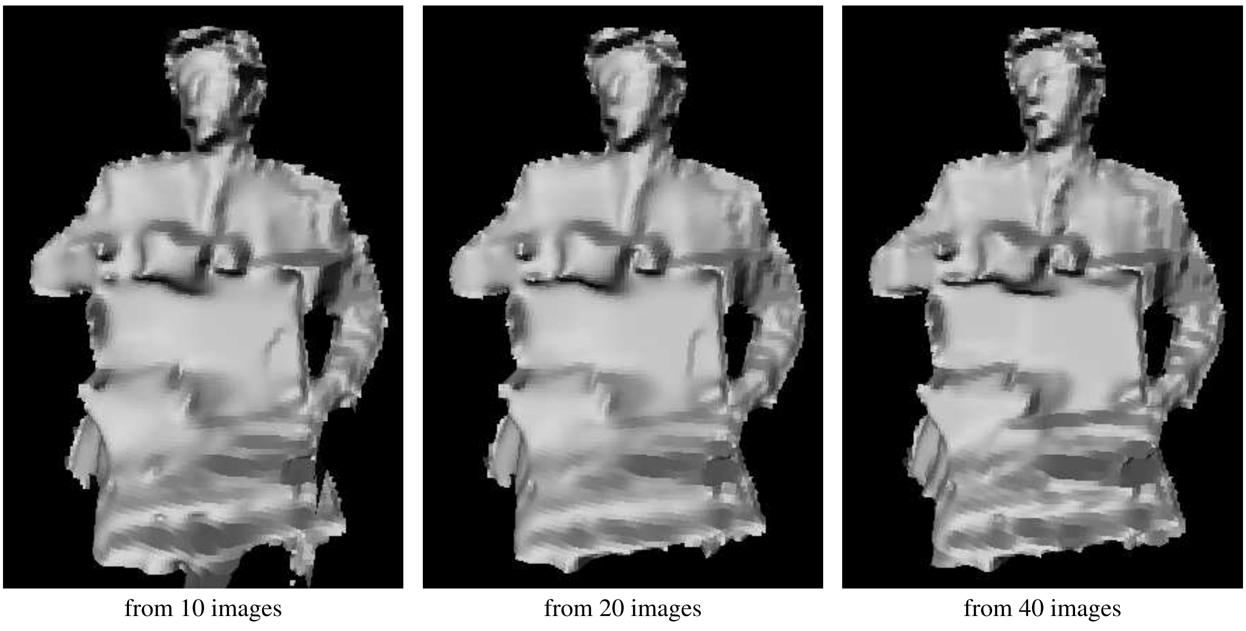

Fig. 15. Reconstruction from the same sequence as Figure 14 but with a varying number of images. The technique achieves precise results with only 10 images.

case" experiment. Hence 10 images are a safe threshold for any sequence and less images could be used if "better" pictures are available. This number has also to be related to occlusion: The chess-board in Figure 18 is occluded half of the time, so it requires 20 images or more to be sure to reach a satisfying result.

Comparison with disparity maps: We have run the algorithm of Kolmogorov et al [1][6] on the sequence of the man with a briefcase (Fig. 14). We have tried to match as close as possible our setup: we have used the same calibrated cameras with an equivalent bounding region. For the labels, we have placed 10 planes which span the whole depth of the subject, and 5 planes for the background in order to avoid spurious influence of background objects. As suggested in their paper, we have used the *robustified* L_1 *distance* to compute the smoothing term. Since the input values are limited to integers, we have tried all the relevant values for the smoothing term and selected the one leading to the best result. The obtained disparity map is shown in Figure 16.

Fig. 16. Disparity map computed with the method of Kolmogorov et al. [1] on the same sequence as Figure 14.

We can remark that contours are very precise but depth precision is limited: only 5 labels appear for the man (among 10 possible). This let us think that this method is not able to reach a higher precision on this sequence even if we allow more labels. Moreover, these labels introduce strong discontinuities in some regions where there is no clear reason to do so *e.g.* on the briefcase

[6]The code is available at `http://www.cs.cornell.edu/People/vnk/software.html`

and on the left arm. This clearly comes from the concave shape of the *robustified L_1 distance*. From the presented study, this point can be overcome with a convex smoothing term associated with an image-driven detection of the discontinuities.

A side-by-side comparison (Fig. 1) outlines the dramatic improvement of the depth precision brought by our technique.

Graph flow implementation: The graph-cut optimization process is time consuming. These examples took abuot 15 minutes to compute on an Intel Xeon 2.4GHz and they need between 300MB and 700MB of RAM. These values have to be evaluated considering the size of the graph ($\approx$ millions of vertices and edges). This has to be compared to the graphs used in [1] (at most 600 000 vertices and 4 millions directed edges). The huge graph-flow problem exceeds the existing graph-flow implementation. It has been made tractable through a careful implementation of the algorithm and heuristics presented by Cherkassky and Goldberg [38].

Then we add our own improvements. For memory space, the key point of our implementation is that it computes adjacency information on-the-fly instead of storing it. Then we ensure that there is no circular flow in double edges so that the flow in double edges can be stored in a single signed value instead of two. Note that the other classical implementation proposed by Boykov and Kolmogorov [39] is not suitable for our graphs because it is specifically designed for small graphs and they have shown that its higher complexity degrades performance for large graphs. This is a real drawback in our case.

Our code is available at `artis.imag.fr/Members/Sylvain.Paris/`

Complexity and timing: The complexity of the algorithm is dominated by the graph-cut computation which is theoretically polynomial of degree 2.5 [38] relatively to the number of voxels. It seems however to be almost linear in practice for grid-aligned graphs (degree 1.2 from [4]).

The graph cut is also the most time-consuming step with about 15 minutes. Among the other steps, the consistency computation is the only significant one with 5 minutes. The remaining treatments last a few seconds. These measures are made on an Intel Xeon 2.4GHz using code which could be further optimized.

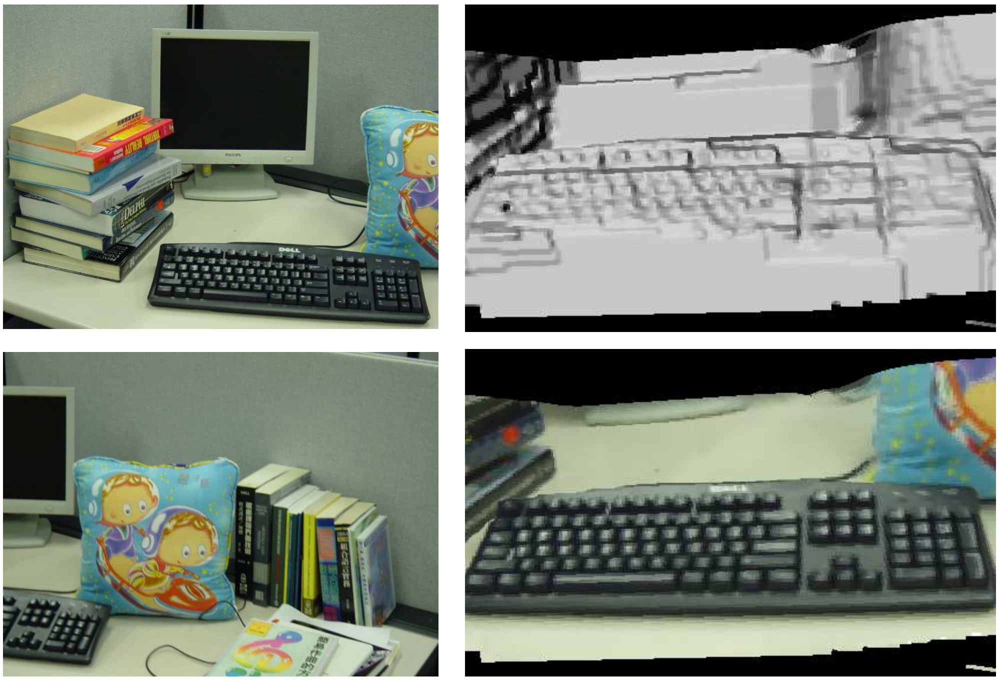

Fig. 17. Keyboard reconstruction (right) from 11 images at 640 $\times$ 480 (left). Notice that the keys of the keyboard are clearly distinguishable.

VII. Conclusions

We have described a new geometric formulation of the surface recovery problem. It is based on a functional that is simpler than those of the level set method or of other graph cut approaches. However, we believe that the geometric formulation is more meaningful than the labeling interpretation commonly proposed by graph-cut methods. Moreover, it explicitly takes into account discontinuities compared to the other methods which either do not handle discontinuities or consider them but have no control over their localization. We have integrated this into a voxel-based process to achieve complete reconstruction system. We demonstrate that this system handle self-occlusions and occlusions to reconstruct partially hidden objects. It also has the capacity to achieve very precise results even for complex configurations.

We believe that this new approach is promising and provides new solutions for interesting issues. The high robustness of the method coupled with a specific consistency evaluation may for instance lead to results for a more general reflectance model than the Lambertian model

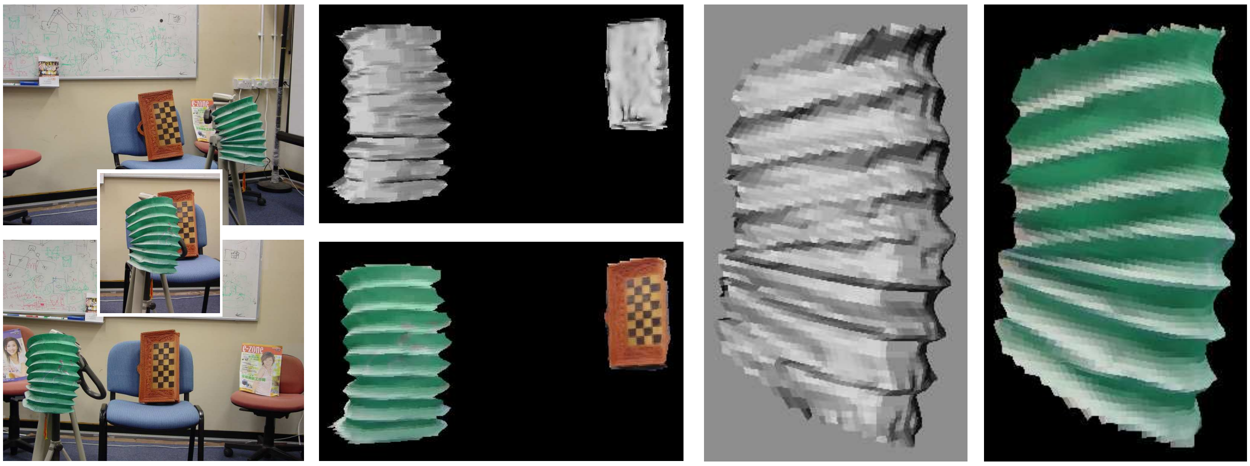

Fig. 18. Lantern and folded chess-board reconstruction (right) 23 images at 800×600 (left) The folded chess-board is occluded from images 4 to 15.

commonly used in Computer Vision. We also plan to study a generalization of the method to more general surfaces to make the link with other methods tighter.

References

[1] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *European Conference on Computer Vision*, 2002.

[2] K. Kutulakos and S. Seitz, "A theory of shape by space carving," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, 1999, pp. 307–314.

[3] O. Faugeras and R. Keriven, "Variational principles, surface evolution, PDE's, level set methods and the stereo problem," *Transactions on Image Processing*, 1998.

[4] S. Roy and I. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *IEEE International Conference on Computer Vision*, 1998.

[5] S. Roy, "Stereo without epipolar lines : A maximum-flow formulation," *Int. Journal of Computer Vision*, vol. 34, no. 2/3, pp. 147–162, August 1999.

[6] H. Ishikawa, "Global optimization using embedded graphs," Ph.D. dissertation, New York University, 2000.

[7] C. Buehler, S. Gortler, M. Cohen, and L. McMillan, "Minimal surfaces for stereo," in *European Conference on Computer Vision (ECCV 02)*, 2002.

[8] J. Kim, V. Kolmogorov, and R. Zabih, "Visual correspondence using energy minimization and mutual information," in *International Conference on Computer Vision*, 2003.

[9] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *IJCV*, 1999.

[10] H. Saito and T. Kanade, "Shape reconstruction in projective grid space from large number of images," in *Computer Vision and Pattern Recognition (CVPR-99)*, 1999, pp. 49–54.

[11] K. N. Kutulakos, "Approximate n-view stereo," in *European Conference on Computer Vision (ECCV 00)*, 2000.

[12] G. Slabaugh, T. Malzbender, and W. B. Culbertson, “Volumetric warping for voxel coloring on an infinite domain,” in *3D Structure from Images - SMILE 2000*, 2000.

[13] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer, “A survey of methods for volumetric scene reconstruction from photographs,” in *VolumeGraphics 01*, 2001.

[14] M. Lhuillier and L. Quan, “Surface reconstruction by integrating 3d and 2d data of multiple views,” in *Proc. of Int. Conf. on Computer Vision*. IEEE, 2003.

[15] J. Isidoro and S. Sclaroff, “Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints,” in *Proc. Int. Conf. on Computer Vision*. IEEE, 2003, pp. 1335–1342.

[16] O. Veksler, “Efficient graph-based energy minimization methods in computer vision,” Ph.D. dissertation, Cornell University, 1999.

[17] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2001.

[18] V. Kolmogorov and R. Zabih, “Computing visual correspondence with occlusions using graph cuts,” in *International Conference on Computer Vision*, 2001.

[19] ——, “What energy functions can be minimized via graph cuts?” in *European Conference on Computer Vision*, 2002.

[20] H. Ishikawa and D. Geiger, “Occlusions, discontinuities, and epipolar lines in stereo,” in *European Conference on Computer Vision (ECCV 98)*, 1998.

[21] H. Ishikawa, “Exact optimization for markov random fields with convex priors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2003.

[22] Y. Boykov and V. Kolmogorov, “Computing geodesics and minimal surfaces via graph cuts,” in *International Conference on Computer Vision*, 2003.

[23] D. Terzopoulos, A. Witkin, and M. Kass, “Constraints on deformable models: Recovering 3d shape and nonrigid motions,” *Artificial Intelligence*, 1988.

[24] J. Sethian, *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[25] V. Caselles, R., and G. Sapiro, “Geodesic active contours,” *International Journal of Computer Vision*, 1997.

[26] S. Osher and J. Sethian, “Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations,” *Journal of Computational Physics*, 1988.

[27] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton University Press, 1962.

[28] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[29] A. Blake and A. Zisserman, *Visual reconstruction*, ser. Artificial intelligence. MIT Press, Cambridge, 1987.

[30] R. Szeliski and P. Golland, “Stereo matching with transparency and matting,” in *International Conference on Computer Vision (ICCV 98)*, 1998.

[31] S. Paris and F. Sillion, “Robust acquisition of 3d informations from short image sequences,” *Graphical Models*, vol. 65, no. 4, pp. 222–238, July 2003.

[32] M. Ulvklo, H. Knutsson, and G. H. Granlund, “Depth segmentation and occluded scene reconstruction using ego-motion,” *SPIE Visual Information Processing*, 1998.

[33] K. Museth, D. Breen, R. Whitaker, and A. Barr, “Level set surface editing operators,” *ACM Transactions on Graphics (Siggraph 02)*, vol. 21, no. 3, pp. 330–338, July 2002.

[34] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*, ser. Applied Mathematical Sciences. Springer-Verlag, 2002, vol. 147.

[35] M. J. Black, G. Sapiro, D. Marimont, and D. Heeger, “Robust anisotropic diffusion,” *IEEE Transactions on Image Processing*, 1998.

[36] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll, “Image selective smoothing and edge detection by nonlinear diffusion,” *SIAM Journal on Numerical Analysis*, 1992.

[37] M. Lhuillier and L. Quan, “Quasi-dense reconstruction from image sequence,” in *European Conference on Computer Vision (ECCV 02)*, 2002.

[38] B. V. Cherkassky and A. V. Goldberg, “On implementing the push-relabel method for the maximum flow problem,” *Algorithmica*, vol. 19, no. 4, pp. 390–410, 1997.

[39] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2004.