Vis Comput
DOI 10.1007/s00371-015-1144-5

ORIGINAL ARTICLE

3D reconstruction for featureless scenes with curvature hints

Andrea Baldacci[1] · **Daniele Bernabei**[1] · **Massimiliano Corsini**[1] · **Fabio Ganovelli**[1] · **Roberto Scopigno**[1]

Abstract We present a novel interactive framework for improving 3D reconstruction starting from incomplete or noisy results obtained through image-based reconstruction algorithms. The core idea is to enable the user to provide localized hints on the curvature of the surface, which are turned into constraints during an energy minimization reconstruction. To make this task simple, we propose two algorithms. The first is a multi-view segmentation algorithm that allows the user to propagate the foreground selection of one or more images both to all the images of the input set and to the 3D points, to accurately select the part of the scene to be reconstructed. The second is a fast GPU-based algorithm for the reconstruction of smooth surfaces from multiple views, which incorporates the hints provided by the user. We show that our framework can turn a poor-quality reconstruction produced with state of the art image-based reconstruction methods into a high- quality one.

Keywords Image-based reconstruction · Image-based modeling, surface reconstruction · Depth maps fusion · Energy minimization on the GPU

1 Introduction

Image-based 3D reconstruction includes all techniques that employ images to infer the 3D shape of an object, e.g., shape-from-silhouette, shape-from-shading and multi-view stereo 3D reconstruction. These techniques have become a common tool for 3D object acquisition thus enabling a complex scene to be rapidly reconstructed from a set of digital images on a consumer PC with publicly distributed software to produce a dense reconstruction starting from a set of input images. Such software includes VisualSFM, which is a bundled application which includes several Open Source algorithms such as Bundler [37] and PMVS [17] to produce a dense reconstruction starting from a set of input images. This can be also be achieved also with online services such as the Autodesk 123D Catch Web application.

One weakness of most of the MVS algorithms is that the quality of the final reconstruction depends on some assumptions that are not always met. Incomplete or noisy reconstruction can be caused by various quite common conditions, such as a few overlaps between images, camera movements that provide insufficient parallax information, homogeneous color appearance (lack of texture) of the object to be reconstructed, hard shadows, and moving occluders such as cars or people. These unfavorable conditions often happen for man-made *functional* elements, such as buildings, streets, bridges, pipes and toys which often consist of smooth/flat surfaces of a homogeneous appearance.

To account for these and similar problems, many algorithms use different *shape priors* to increase the quality of

Electronic supplementary material The online version of this article (doi:10.1007/s00371-015-1144-5) contains supplementary material, which is available to authorized users.

✉ Massimiliano Corsini
massimiliano.corsini@isti.cnr.it

Andrea Baldacci
andrea.baldacci@isti.cnr.it

Daniele Bernabei
daniele.bernabei@isti.cnr.it

Fabio Ganovelli
fabio.ganovelli@isti.cnr.it

Roberto Scopigno
roberto.scopigno@isti.cnr.it

[1] ISTI-CNR, Pisa, Italy

Published online: 31 July 2015

the final reconstruction. For example, Sinha et al. and Gallup et al. [18,36] assume that the surfaces in the scene are flat, while Furukawa et al. [15] assume that adjacent surfaces are flat and form a quasi-right angle (the so-called *Manhattan world* assumption). More complex shape priors may be used, such as the swept surfaces adopted by Wu et al. [42] for the reconstruction of architectural buildings. Another method to improve the reconstruction of missing/incomplete parts specific for architectural buildings is the one of Chen et al. [44] which approximates the building surface with planar and curve surfaces in a robust fashion, taking into account that, in general, in this case many of the reconstructed points are inliers. However, there are cases where many different hypotheses for the missing surface may be consistent with the data. In these cases, additional geometric constraints are not sufficient and a *semantic* interpretation of the images is necessary. Bao et al. [3] introduce *semantic priors*, that is, high-level priors (e.g., a car) which are extracted and collected in a learning phase.

We propose an interactive framework to enable the user to provide high-level geometric information to improve the reconstruction. The user can specify with a few strokes the object of interest and the geometric constraints on one or more images, i.e., parts of the object that have one or more directions of zero curvature. These hints are not helpful where the reconstructed surface is something like a tree or a bas-relief, but they are very useful for man-made objects such as walls, pipes, panels and toys. For example, if a region is flat, the user can draw two orthogonal straight lines, if the region follows a cylindrical path, as in a pipe (see Fig. 4), the user can draw a single line along the pipe, and so on. The object selection is accurately propagated between all the images and on the reconstructed 3D points by a novel multi-view segmentation algorithm using a joint 2D–3D graph cut formulation. This algorithm can be seen as an extension of the GrabCut [33] algorithm. The final surface, expressed as a union of depth maps, one for each calibrated image, is obtained by recasting the segmentation and the curvature hints into an energy-based multi-view reconstruction problem, which is solved entirely in GPU. Each depth map is computed by means of minimizing an energy functional which takes into account the user indications, the initial reconstruction and the coherence among different overlapping depth maps. The multi-view segmentation and the soft constrained energy-based multi-view reconstruction algorithm of the framework are the main novel contributions of this work.

2 Related work

This paper contributes to two different fields, both of which have a substantial body of literature: Image Segmentation and Multi-View Reconstruction. In the following, we concisely review the state of the art on both fields, focusing on those algorithms that are more closely related to our work.

2.1 Image segmentation

Image Segmentation refers to partitioning the pixels of an image into groups that share similar characteristics. Here, we are interested in partitioning the image into *foreground pixels*, which represent the object of interest, and *background pixels*. One of the earliest improvements on the crude manual segmentation was Intelligent Scissors [29], where the user defines various anchor points along the silhouette of the object, and a minimization algorithm adjusts the contour to match the gradient change of the image. Active Contours (or Snakes) [23] also work by minimizing an energy function over a contour (that is, a snake) accounting for image gradient and contour bending. More recent approaches do not work on the parametric definition of the contour but on the foreground/background classification of the pixels. In their seminal paper, Boykov and Jolly [6] recast the segmentation problem as a graph problem. More specifically, an image is mapped onto a graph where each pixel is a node and is connected to neighbor pixels by arcs. There are also two special terminal nodes, one for the background and one for the foreground, to which all other nodes are connected. The cost of a pixel–pixel edge is set to penalize separation between *similar* pixels, while the cost of a pixel–terminal penalizes setting the pixel to the background or foreground (for example, on the basis of the initial manual pixel annotation by the user). With this formulation, any *cut* in the graph corresponds to a partition of the pixels into two sets: those connected to the foreground terminal and those connected to the background terminal. The solution thus has a cost which is the sum of the cost of the arcs in the cut, which can be minimized by min-cut max-flow algorithms. Graph Cut methods have become the de-facto standard for image segmentation [2], thanks to their conceptual simplicity and to very efficient polynomial time solvers [28]. Since their inception, the technique has been strengthened using shape prior information to disambiguate similar color areas [14], with an inclusion of Dijkstra's algorithm to preserve thin structures [40]. One of the most successful improvements is due to Rother et al. [33] who proposed an iterative algorithm where each iteration consists of solving the min-cut problem and re-assigning the cost functions on the basis of the solution found, until convergence. In addition, they introduced the Gaussian Mixture Model (GMM) to integrate color information in place of the simpler intensity histograms.

2.2 Multi-view object segmentation

Now that is common to have a set of calibrated images of an object, the problem of segmenting a single image has evolved

into how several images (of the same scene) can be segmented at the same time. Graph cuts can be naturally generalized to multiple images and thus many algorithms use them. All the approaches need to identify a way to connect pixels from different images. Sormann et al. [38] stack the images to form a 3D texture and use a preprocessing step to segment each image in clusters so as to reduce the size of the graph using one node per cluster and not per pixel. They assume a short baseline, that is, consecutive images in the stack do not change too much, so that the neighborhood among pixels of different images makes sense. Campbell et al. proposed the Volumetric Graph Cuts [10], where the scene is voxelized and node-voxels are added to the graph, an idea somewhat reminiscent of the voxel coloring approach [35]. They use the *fixation* hypothesis, that is, that all the images look towards the object, which consequently is located roughly at the center of each image. They can thus simultaneously initialize the foreground/background and define the bounding box of the scene to be voxelized. The voxel nodes are essentially a means to connect pixels from different images. In a subsequent work [9] by the same authors, the voxel grid was replaced by adding stereo correspondences and epipolar constraints to directly connect pixels from different images. Djelouah et al. [12] use a set of sample points uniformly distributed in the volume (under a similar hypothesis as in [10]) and consider the tuple of pixels defined by the projection of the sample point on the images. The key idea is that if all the elements of a tuple are classified as foreground, then the point belongs to the object's surface. Similar to [33], they use a GMM model and an iterative process for a posteriori estimation (MAP) of the classification variables. Sparse 3D samples are also used by Djelouah et al. [13], where the problem of multi-view segmentation is extended on the time dimension to support multi-view video segmentation and superpixels are used to reduce the computational complexity.

Other approaches, such as Bleyer et al. [5] and Kowdler et al. [1], assume that the objects in the scene can be approximated by planes, and that the baseline is so short that a reasonable depth map can be estimated. In this setting, the segmentation can be set at an object level and 3D spatial relations between objects are used.

2.3 Multi-view stereo matching

According to [34], the multi-view dense stereo reconstruction algorithms can be categorized depending on their properties, for example, depending on the surface representation used, on the reconstruction algorithm used, on the initial requirements, and initial hypotheses regarding the shape to reconstruct.

Many MVS reconstruction algorithms are based on segmentation. Typically, each image is segmented into the background and foreground (of the object of interest). One of the oldest of this class of methods is the *shape-from-silhouette*. Such methods estimate the *visual hull* of the object, that is the maximal surface consistent with the silhouette for all the views, by carving the volume of the object according to the silhouette in the different views. More recently, Yezzi and Soatto [43] explored the dual connection between the segmentation of an object in multiple images and 3D reconstruction of the underlying object. They employed a level set method, solved with a multiresolution scheme. Kolev et al. [25] reformulated the problem as a Bayesian estimation of the most probable shape that would yield the observed images, making the method more robust with respect to noise. In Sorman et al. [39], each image in the set is first clustered using mean-shift and then these clusters are segmented via GraphCut. However, segmentation happens sequentially, whereupon each segmentation provides a shape prior to be used in the subsequent one. Kolev et al. [26] deal with the image segmentation of all the views by projecting an evolving 3D surface. The problem is the setup in an energy minimization framework where the energy terms proposed take into account background and foreground terms plus a photo-consistency term. In a more recent work [27], the authors added an anisotropy term to this energy to also account for the orientation of the evolving surface. Jancosek et al. [22] compute an over-segmentation of the dataset as a first step to reduce the computational load and to provide priors for reconstructing flat areas of uniform colors. A recent hybrid (silhouette-based/correspondence-based) method capable to improve the reconstruction of objects with few visual features (e.g., uni-colored objects) has been proposed by Hoangminh et al. [31]. This method exploits the geometry reconstructed by means of standard SFM approaches to improve the automatic extraction of the silhouette. Another interesting paper to cite, even if not an MVS method, is the work of Liangliang et al. [30] which proposes a segmentation-based approach to complete the sparse reconstruction produced by scanned data.

Many other MVS algorithms work by estimating a depth map for each image and then integrating these depth maps into a unique surface. Goesele et al. [19] proposed a simple and effective algorithm to estimate the depth for each pixel by evaluating the photoconsistency (using NCC) of each 3D point estimated. Only pixels with high values of correlation are considered reliable. The sparse depth maps thus estimated are merged together by applying the volumetric surface reconstruction algorithm of Curless and Levoy [11]. Bradley et al. [7] developed a high-quality method by proposing a viewpoint adaptive window to drive the stereo matching between image pairs. The high-quality depth maps thus generated are then merged together using a lower dimensional triangulation algorithm [20]. Furakawa et al. [16] proposed one of the most general and accurate algorithms for 3D reconstruction from calibrated images. This algorithm is the core

of the PMVS software and is based on a patch representation of the surface. The initial oriented patches are estimated, then expanded to the nearby pixels, and filtered in an iterative way to produce a dense reconstruction.

Our approach is inspired by the multi-view segmentation-based methods but uses depth maps as evolving surfaces to obtain the final reconstruction. A depth map is estimated for each camera by minimizing an energy functional composed of three terms: a smoothness term, a term to account for surface coherency which imposes that overlapping depth maps must coincide, and a term that takes into account the curvature hints. The curvature hints drawn by the user as 2D curves are expressed per pixel by expanding them over the selected region (further details in Sect. 4). The advantage of this approach is that depth maps are intrinsically free of topological and geometrical inconsistencies (e.g., self-intersections), since the energy value and its gradient only depend on the depth values. In addition, the existing 3D reconstruction together with the curvature constraints reduce the ballooning effects typical of many energy-based methods. Finally, the GPU implementation guarantees a fast computation of the final surface.

3 Segmentation on calibrated images

Our approach is mostly a direct extension of Rother et el. [33] to the case of multi-view datasets. Unlike previous approaches, 3D points are not only a way to connect pixel of different images, but are also elements that are classified. Therefore, the user input is a selection of points or pixels, depending on which operation is easier on the given dataset. Let V be the set of reconstructed vertices. We know that each vertex in V corresponds to a pixel in two or more images, that is, the pixels that were matched to infer the 3D position of the vertex. So we indicate with $\mathrm{Corr}(v)$ the set of pixels corresponding to vertex v. Let $G = (V \cup P, E)$ be a undirected graph where V is the set of input vertices and P is the set of pixels of all images. The set of edges E is defined as:

$$\begin{aligned} E = {} & \{(p_i, p_j) | p_i \text{ adjacent } to\ p_j\} \cup \\ & \{(p_i, v_j) | p_i \in \mathrm{Corr}(v_j), v_j\} \cup \\ & \{(v_i, v_j) | \|v_i - v_j\| < \tau\} \end{aligned} \tag{1}$$

where we use p to refer to pixels and v for vertices. The first type of edge is the regular pixel–pixel edge used in the single image segmentation algorithm. The second type connects pixels that were matched to create a vertex with the vertex itself, thereby generating a 2D–3D connection. Finally, the third edge type connects vertices that are closer than a threshold in 3D space. This means that a selection made in one image can *propagate* through space and end up in other images. Choosing the weights for the pixel–vertex and the vertex–vertex edges entails taking into account the specific algorithm. Here, we use this general idea to extend the GrabCut [33] algorithm.

The GrabCut algorithm [33] employs two Gaussian Mixture Models (GMM), one for the foreground and one for the background, to model color distribution. These GMMs are mixtures of K full-covariance Gaussians. A vector $\alpha \in \{0, 1\}^N$ assigns to each element (pixel) a flag indicating foreground or background. In addition, Rother et al. [33] introduced the idea of using also a vector $\underline{\mathrm{k}}$ assigning an element to a specific component of the mixture model. The element themselves, that in our system can either be pixels or vertices, are indicated with vector $\underline{z} \in P \cup V$. The algorithm proceeds by globally minimizing a Gibbs energy function of the form

$$E_s(\alpha, \underline{\mathrm{k}}, \theta, \underline{z}) = U(\alpha, \underline{\mathrm{k}}, \theta, \underline{z}) + V(\alpha, \underline{z}) \tag{2}$$

where θ are the parameters of the Gaussian mixtures, i.e., the weights for the components, and the means and covariance matrices of the individual components. The data term U is computed as in the original formulation by evaluating the log-likelihood w.r.t the assigned Gaussian from the GMM, with the difference that our elements can be vertices in addition to pixels. More precisely:

$$U(\alpha, \underline{\mathrm{k}}, \theta, \underline{z}) = \sum_N (-\log p(\underline{z}_n | \alpha_n, \underline{\mathrm{k}}_n, \theta) - \log \pi(\alpha_n, \underline{\mathrm{k}}_n)) \tag{3}$$

The smoothness term is instead specified with respect to an augmented neighborhood system C, which takes into account 2D–2D links between adjacent pixels (the only type of link in the GrabCut), 3D–2D links between a vertex and its projection onto the images and 3D–3D links, between neighbor vertices in 3D space.

$$V(\alpha, \underline{z}) = \gamma \sum_{i,j \in C} \begin{cases} e^{-\beta \|z_i - z_j\|} & \text{if } i \in P,\ j \in P \\ \Gamma & \text{if } i \in P,\ j \in V \\ e^{-(\beta \|z_i - z_j\| \tau \|v_i - v_j\|)} & \text{if } i \in V,\ j \in V \end{cases} \tag{4}$$

Constants β is set as in the 2D case, that is:

$$\beta = 2\,(\llbracket z_i - z_j \rrbracket)^{-1} \tag{5}$$

where $\llbracket . \rrbracket$ indicates the expected value, and z is in CIElab color space, and constant τ for the 3D–3D links is found by extending the same idea as $\tau = 2\,(\llbracket v_i - v_j \rrbracket)^{-1}$ where v is the position in 3D space. Γ is a big constant that ensures that a pixel and its projected vertex do not get classified in different sets.

The background and foreground pixels marked by the user are used to initialize two Gaussian Mixture Models for the foreground color distribution and the background color distribution. This initialization is performed with a k-means algorithm. The algorithm then iteratively performs the following operations:

1. assign each unknown pixel and vertex either to the background or to the foreground GMM (estimate α)
2. assign a specific Gaussian within the assigned GMM to each pixel/vertex (estimate $\underline{k}$)
3. re-estimate mean and covariance for each Gaussian in the GMM based on assigned pixels/vertices (estimate θ)
4. solve minimization by GraphCut, estimating sink and source energies according to the GMMs (estimate α)
5. repeat from step 2 until convergence.

Figure 1 shows some results of our technique vs. the GrabCut approach. As expected, the multi-view version is more accurate and requires less precise user interaction. This is the obvious consequence of using multiple images in our setting. Additionally, we show in Fig. 2 a comparison with an automatic state of the art multi-view image segmentation method, i.e., the approach presented in [12], on a dataset where the object of interest, the car, is not fully visible in all of the images. It can be seen how, although both the algorithms produce acceptable results, our approach is able to correctly classify the car's pixel even behind vegetation. Moreover, the technique [12] applied to the Museum dataset (the one used in Fig. 1) cannot produce any usable result due to the fact that the object of interest (the tree) is not in the center of the images.

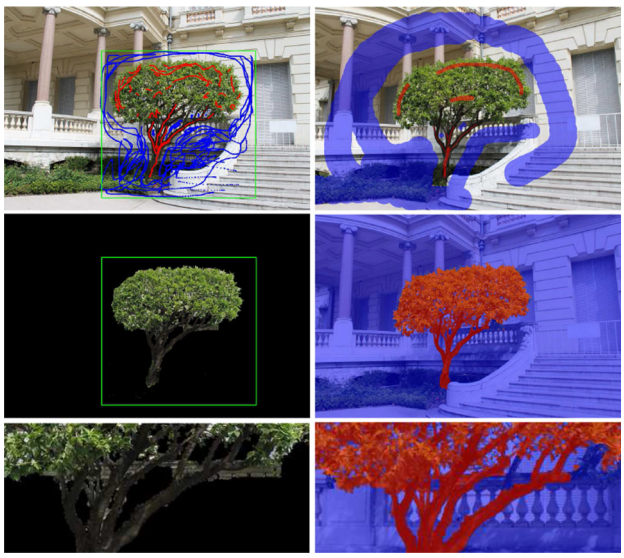


Fig. 1 The proposed algorithm vs. the standard GrabCut algorithm (*1st row* user input, *2nd row* result, *3rd row* details). Note that only 1 out of 27 images received input from the user

4 Defining curvature hint

The aim of the curvature hints provided by the user is to know, for a specific subregion of the image, that the directional curvature along certain directions is zero.

To better explain how these curvature hints work, we will refer to a practical example. Figure 3 shows a drawing of a gas pipe. The user wants to hint that the surface curvature is zero along the direction of the pipe and does so by drawing a line like the red one shown in the figure. The intended meaning is that on any location of the line the curvature along the tangent vector is zero. The other points of the region inherit the direction vector of their closest point on the line so as to obtain a vector field (described by blue segments in the figure). In other words, we infer a vector field for all the pixels of the region of interest starting from the input curvature constraints.

To achieve this, first, the input line L is sampled as $L_s = \{(l_0, t_0), \ldots, (l_k, t_k)\}$, where l_i is the position of the sample and t_i is the tangent vector at l_i. Then, the samples are projected onto the 3D surface obtaining $L'_s = \{(l'_0, n_0, t'_0), \ldots, (l_k, n_i, t_k)\}$ where l'_i is the projection of l_i, n_i is the surface normal at l'_i and t'_i is the projection of t_i on the tangent plane passing through l'_i, that is, the tangent plane at l'_i. So far, we have obtained the direction of the zero curvature for the sampled points. The next step is to propagate this information to the rest of the surface. We obtain the direction orthogonal to t'_i as $b_i = n_i \times t_i$ and then define a new grid node $(l'_{i,1}, n_{i,1}, t'_{i_1})$ where: $l'_{i,1}$ is the 3D point found by walking a distance Δ from $l'_{i,1}$ along b_i, $n_{i,1}$ is the normal at $l'_{i,1}$ and $t'_{i,1} = b_i \times n'_{i,1}$. We iterate this process along b_i and $-b_i$ for all i until we obtain a grid covering the projection of the selected region on the surface, where the direction of zero curvature and the distance from the point on the same line of the grid is associated with each node.

The final step is to interpolate the directions stored at the grid nodes for all the pixels. This can be simply done using rasterization, by rendering the grid as filled quads and letting attribute interpolation do the work. However, note that, unless the line is a straight one, pixels would be covered by more than one quads, while our goal is to get the value from the closest point on L'_s (see Fig. 4). This problem is solved using an idea proposed by Hoff et al. [21] to compute a voronoi diagram of points and lines. To each grid node, we assign a z coordinate (in view space) as its distance from the line L'_s. From the fragments with the same coordinates, the depth test will thus automatically return the one closest to the line.

5 Reconstruction

The reconstruction algorithm takes as input the reconstructed geometry, the selection as defined in Sect. 3, and the cur-

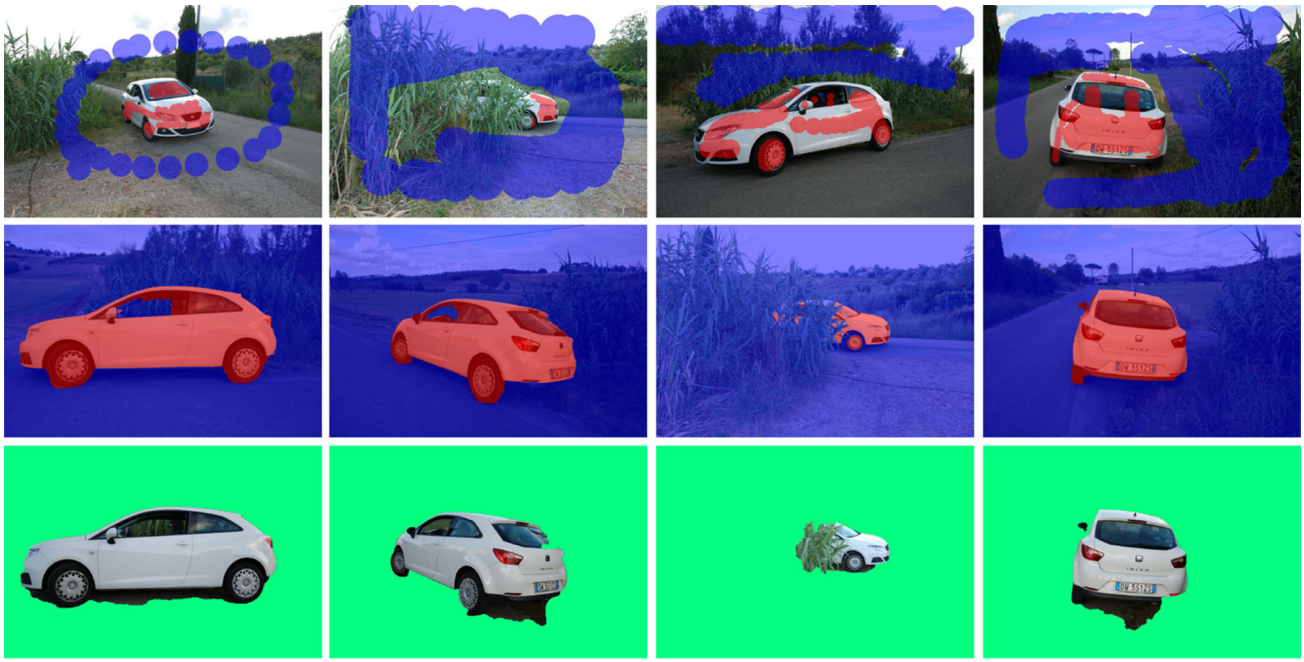


Fig. 2 Example of joint 2D–3D dataset segmentation. *Top row* the 4 images out of 55 where the user provided input for segmentation. *Second row* some of the output image; *third row* the same image with the technique introduced in [12]

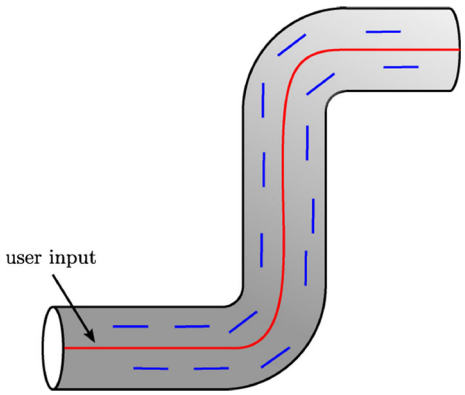


Fig. 3 Scheme of the curvature hint. The user input is shown as a *red line*, the inferred vector field with *blue short segments*

vature hints. Then, it gives in output a set of depth maps $Z = \{\mathbf{z}_h | h = 0 \dots N\}$, that is the depth values for the pixels of the regions r_h, which altogether form the final reconstructed surface.

The algorithm proceeds by minimizing a four term energy function $E(Z)$, defined as

$$E(Z) = \sum_{\forall h} S(\mathbf{z}_h) + F(\mathbf{z}_h) + R(\mathbf{z}_h) + C(\mathbf{z}_h) \tag{6}$$

$S(\mathbf{z}_h)$ is a term to ensure the smoothness of the surface, $F(\mathbf{z}_h)$ is a term to ensure that the surface approximates the original points, $R(\mathbf{z}_h)$ ensures that different depth maps agree on overlapping regions and $C(\mathbf{z}_h)$ accounts for the curvature hints given by the user.

The minimization is carried by iterative gradient descent. Since we want to leverage on the graphics hardware, we perform the gradient descent computation camera by camera, that is:

$$\begin{aligned} \mathbf{z}^{i+1}_{|h} &= \mathbf{z}^i + \alpha_{i,h} \nabla E(\mathbf{z}^i_{|h}), \quad h = 0, \dots, k \\ \nabla E(\mathbf{z}^i_{|h}) &= \nabla S(\mathbf{z}^i_{|h}) + \nabla F(\mathbf{z}^i_{|h}) + \nabla R(\mathbf{z}^i_{|h}) + \nabla C(\mathbf{z}^i_{|h}) \end{aligned} \tag{7}$$

We use an adaptive step size $\alpha_{i,h}$ which varies with the global energy as proposed in [4] and increases the convergence rate:

$$\begin{aligned} \alpha_{i,h} &= \frac{(\mathbf{z}^i_{|h} - \mathbf{z}^{i-1}_{|h}) \cdot \Delta^i}{\Delta^i \cdot \Delta^i} \\ \Delta^i &= \nabla E(\mathbf{z}^i_{|h}) - \nabla E(\mathbf{z}^{i-1}_{|h}) \end{aligned} \tag{8}$$

Note that the unknowns are the depth values of all the pixels in the ROI. We formulate these energy terms in such a way that the gradient $\nabla E(\mathbf{z}^i)$ can be evaluated in the GPU for each depth map. In the following, we define each of these terms, while we address the interested reader to the Appendix A for the complete algebraic derivations of the gradient term.

Please note that the formulas for energies involve derivations of the depth maps $\mathbf{z}_h$ along x and y in image space, more specifically second-order derivatives $\frac{\partial \mathbf{z}_{xx}(i,j)}{\partial z_{m,n}}$ $\frac{\partial \mathbf{z}_{xy}(i,j)}{\partial z_{m,n}}$ and $\frac{\partial \mathbf{z}_{yy}(i,j)}{\partial z_{m,n}}$, while we will need the gradient of these energies with respect to the z value, that is, the depth of each pixels, which are the unknown variables of the system.

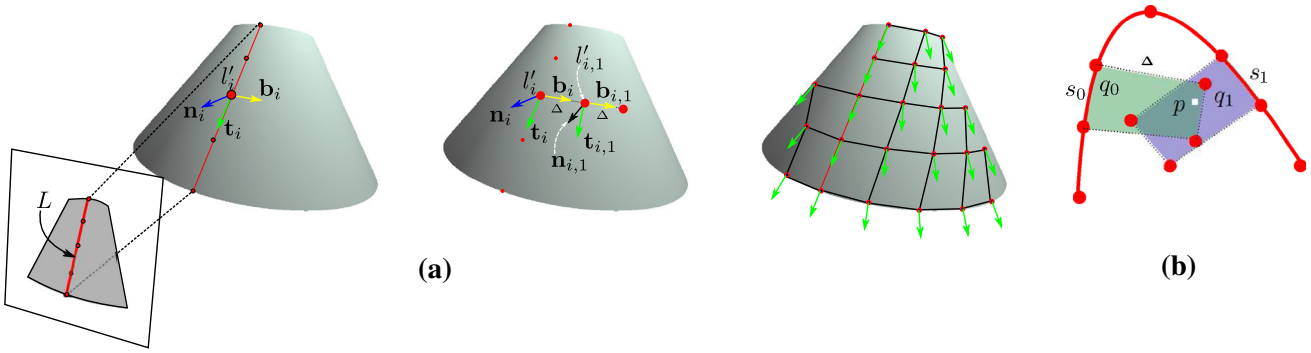


Fig. 4 Curvature hint propagation. **a** From a user-given line in image space to a regular sampling on the surface with inferred directions. **b** Inferring the directions in image space. Point p is included both in q_0 and in q_1, but it is closer to segment s_1 and thus the corresponding direction value is used

5.1 Smoothness term: $S(\mathbf{z}_h)$

The smoothness term is defined as the *thin plate* energy:

$$S(\mathbf{z}_h) = \sum_{i,j} \mathbf{z}^2_{xx}(i,j) + 2\mathbf{z}^2_{xy}(i,j) + \mathbf{z}^2_{yy}(i,j)\Delta x \Delta y \quad (9)$$

where we dropped the pedix h to simplify the notation.

5.2 Approximation term: $F(\mathbf{z}_h)$

The aim of the approximation is to make the final surface an approximation of the initially reconstructed one. We use the implicit Moving Least Squares formulation proposed in [32] to define the surface approximating the input points. With MLS, the surface is the zero set of a function F, $s = \{x | F(x) = 0\}$. At initialization time, we sample the value of ∇F in the neighborhood of the input points, storing the result in an octree. Note that this is the only term for which we can pre-compute and store the gradient because it does not depend on the depth values but only on the input points.

5.3 Coherence term: $R(\mathbf{z}_h)$

The coherence term forces two depth maps to coincide on their overlapping 3D region. In general, depth maps corresponding to the same portion of the real surface do not actually match. This is due to two factors: first, each depth map is obtained by interpolating a different set of vertices, although all of them are approximately on the real 3D surface (see also Sect. 5.5); second, even if the set of vertices were the same, the image space discretization of the depth maps would induce large aliasing effects, especially near the silhouette of each region where the surface is steep w.r.t. the camera point of view.

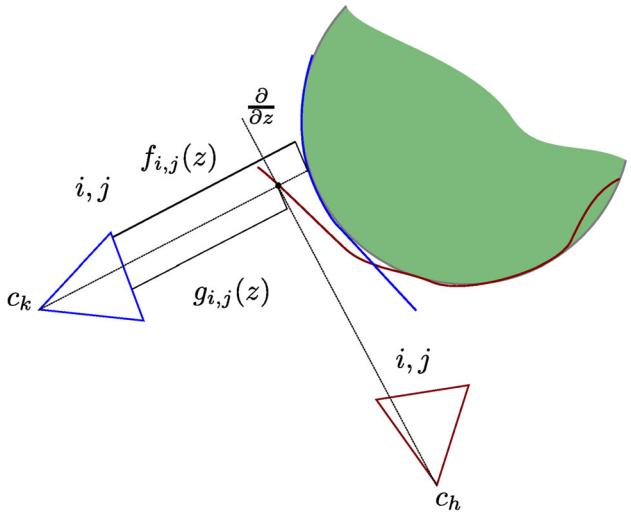


Fig. 5 Enforcing coherence between overlapping range maps

The coherence term is defined as:

$$R(\mathbf{z}_h) = \sum_{\forall k} \sum_{ij} [\phi(g_k(i,j,\mathbf{z}_{i,j}) - f_k(i,j,\mathbf{z}_{i,j}))]^2 \quad (10)$$

where $g_k(i,j,\mathbf{z}_{ij})$ gives the depth of the projection on the neighbor camera c_k of the unprojection of the triple $i, j, \mathbf{z}_{ij}$ of the camera h and $f_k(i,j,\mathbf{z}_{i,j})$ is the current depth value stored at the same location, as illustrated in Fig. 5. ϕ is a threshold function used to define when two depth values are close enough to enforce them to be the same. Note that the outer summation of Eq. (10) runs over the cameras that overlap with h. Typically, for each pixel in r_h, there are from 0 to 5 neighbors.

5.4 Curvature term: $C(\mathbf{z}_h)$

The specified directions of zero curvatures are indicated with $\mathbf{u} = [u, v]$ for the generic pixel and are obtained as explained in Sect. 5.4. To simplify the formula and avoid floating point divisions, instead of minimizing the actual curvature we use

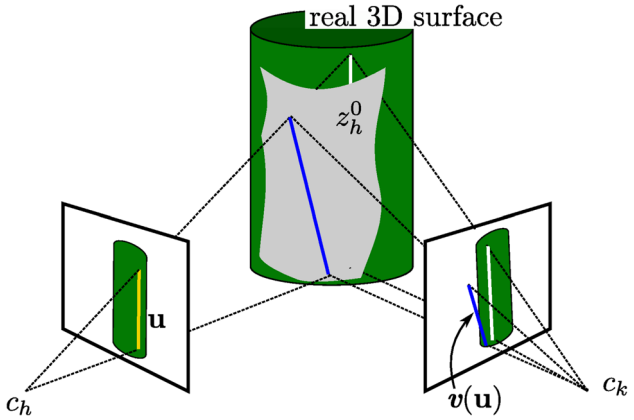


Fig. 6 Propagation of the user-given hint on the direction of zero curvature

the square of the directional second- order derivative of $\mathbf{z}_h$, which is:

$$C(\mathbf{z_h}) = \sum_{ij} (\mathbf{u}^T H(\mathbf{z_h}_{i,j})\mathbf{u})^2 \tag{11}$$

where H is the *Hessian* matrix of z_h.

Note that the direction vector has to be specified for all the cameras. However, it would be very tedious for the user to manually define the directions of zero curvature for each image and, worse, it is unlikely to be consistent for all the cameras. Therefore, we propagate the vector $\mathbf{u}_h$ on all the other cameras by projecting it in world space and hence re-projecting it on each camera.

This is achieved using the depth map z_h obtained by the initialization phase and therefore the approximation of the depth map will influence the projection (we recall that only the smoothing term is considered at the initialization phase). This means that the original vector and its propagation on the other cameras could be inconsistent on the real 3D surface, as shown in Fig. 6. To resolve this inconsistency, we perform the propagation of the vector at each iteration, so that it tends to converge along with the convergence of all the depth maps.

The full formula is written as:

$$C(\mathbf{z_h}) = \sum_{ij} (z_{xx}(i,j)u^2 + 2\, z_{xy}(i,j)uv + z_{yy}(i,j)u^2)^2 \tag{12}$$

5.5 Initialization

Since we are using gradient descent, it is crucial to find an initial solution, i.e., an initial depth map for each camera, which is close to the global minimum and at the same time approximates the input reconstructed points. If the projection of the input points on a given camera is dense (as can happen, for example, using the output of PMVS, see the Toy Car example in Fig. 9), we can simply triangulate the projections and obtain the depth map. On the other hand, if the starting point is a sparse reconstruction (as with the Pipe and the other models in Fig. 9), we minimize the energy term $S(\mathbf{z}_h)$ by imposing the input points as hard constraints, which is done by solving the resulting system $\nabla S(\mathbf{z}_h)$ as shown in [41].

5.6 Handling discontinuities in the depth maps

Note that using finite differences for approximating second-order derivatives always gives an expression of the following form:

$$\nabla(E(Z))_{ij} = \sum_{h,k=-2}^{i,j<2} w(h,k)z(i+h, j+k). \tag{13}$$

This means that the gradient at pixel (i, j) is a linear combination of values in a 5×5 kernel centered at (i, j). In the derivation of the energy terms shown so far, we have always employed central differences to express the partial second derivatives contained in the formula, i.e., $\frac{\partial E(z)}{\partial xx}, \frac{\partial E(z)}{\partial yy}, \frac{\partial E(z)}{\partial xy}$. Unfortunately, depth maps have borders, so we have to adjust the computation of second derivatives on border pixels. A pixel may be on a border in two cases: either because the adjacent pixels are not part of the ROI or because there are discontinuities in the depth maps. The latter case is detected by a check that is run at the beginning of each minimization step, by testing if the difference with one of their adjacent pixels is above a certain threshold. Typically, these discontinuities appear as the surface evolves, while at the beginning, triangulation alone or the thin plate energy tends not to create very steep surfaces.

To handle discontinuities, we write the energy term $E(z_h)$ so that differential quantities are discretized using central, forward or backward differences adaptively, depending on which adjacent pixels are available.

This is done by computing a bit code for each pixel, that is, a *tag*, indicating how each differential quantity must be computed. Listing 1 shows the algorithm , CxCx, FxFx and BxBx stand for central, forward and backward difference and NOxx means that it is not possible to compute the second-order derivative on that pixel.

Listing 1 Algorithm to create the code for the 2nd order derivative along x.

```
CodeDxDx(i,j)
{
  if( (i+1,j) in r_h && (i-1,j) in r_h )
    tag = CxCx;
  else if( (i+2,j) in r_h && (i+1,j) in r_h )
    tag = FxFx;
  else if( (i-1,j) in r_h && (i-2,j) in r_h )
    tag = BxBx;
  else
    tag = NOxx;
}
```

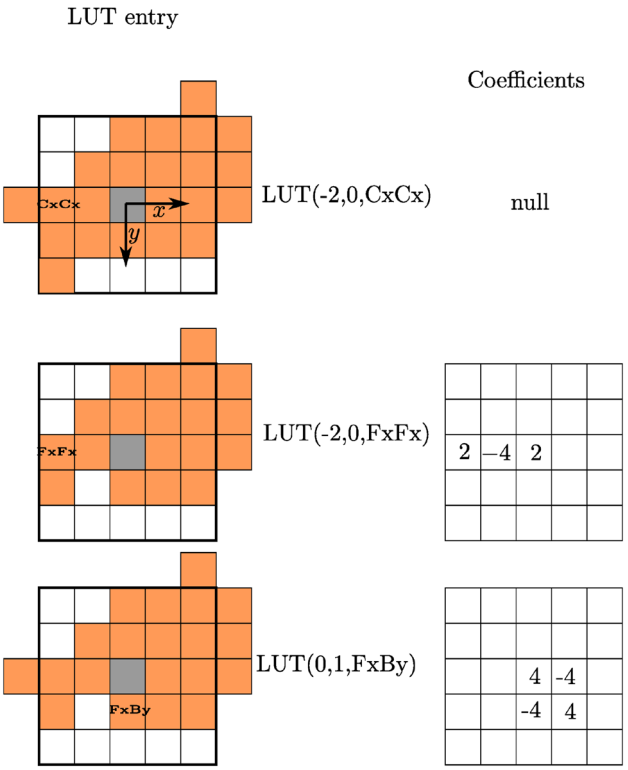


Fig. 7 Example LUT entries for computing the gradient of the smoothness term on a pixel (the central one)

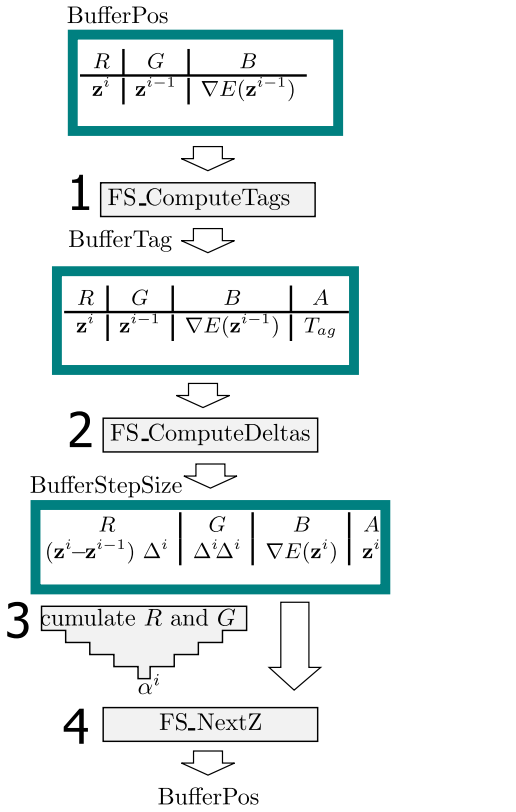


Fig. 8 The flow chart of the minimization algorithm

This means that the exact expression of $\nabla(E(Z))_{nm}$ depends on how the differential quantities in the neighbor pixels are computed. For each pixel of coordinates (i, j), we compute a code that says how $\mathbf{z}_{xx}$, $\mathbf{z}_{yy}$ and $\mathbf{z}_{xy}$ are computed. In a static LUT, for each code, we store the coefficients to be applied to the neighborhood pixels. Figure 7 shows an example of three entries of the LUT (see Appendix B for the derivation of these coefficients).

5.7 Performing the iterative minimization

Figure 8 shows a scheme of the iterative minimization algorithm for a generic camera. We use a fullscreen quad to enable the fragment shader to output a value for each single $z_{n,m}$. Each iteration of the minimization is performed in four steps. In the first pass, the tag values explained in Sect. 5.6 are computed and stored in the alpha channel of the target buffer `BufferTag`.

In the second pass, `BufferTag`, which contains the current and previous solution and the previous gradient, is bound as texture. The fragment shader `FS_ComputeDelta` computes the gradient $\nabla E(\mathbf{z}^i)$, passes along the value $\mathbf{z}^i$ and computes a *first part* of the Eq. (7) (that is, the components to be summed to obtain the dot product).

The third pass consists of summing all the values of the componentwise products to obtain the two factors of the fraction in Eq. (7). This is done by building a texture pyramid, as with mipmapping but summing the four texel values instead of averaging them. Then, the final 1×1 texture is readback in the main memory and α_i can be computed. The fourth and last step consists of bounding the `BufferStepSize` as texture and `BufferPos` as target and computing $\mathbf{z}^{i+1}$ and copying $\mathbf{z}^i$ and $\nabla E(\mathbf{z}^i)$.

Note that when computing $\mathbf{z}^{i+1}_{|h}$ we need to keep in video memory only the depth map h and the maps overlapping with h, which are normally less than five. The need for the overlapping depth map is due to the only term that is not separable over the depth maps, that is, the coherence term $R(\mathbf{z})$.

To speed up convergence rate, we also use a V-Cycle multigrid method (see [8]) on each camera. The multigrid approach consists of transferring the solution found for the original depth (grid) into a coarser grid (*restriction phase*), performing some minimization steps and then transferring back the solution to the finer grid (*interpolation phase*).

6 Results

We tested our system on several datasets, acquired from urban scenarios or daily life objects. Here, we highlight five examples: a pipe, a garbage bin, a van, a toy car and a plastic bath seat (for babies). Figure 9 shows on the first column a

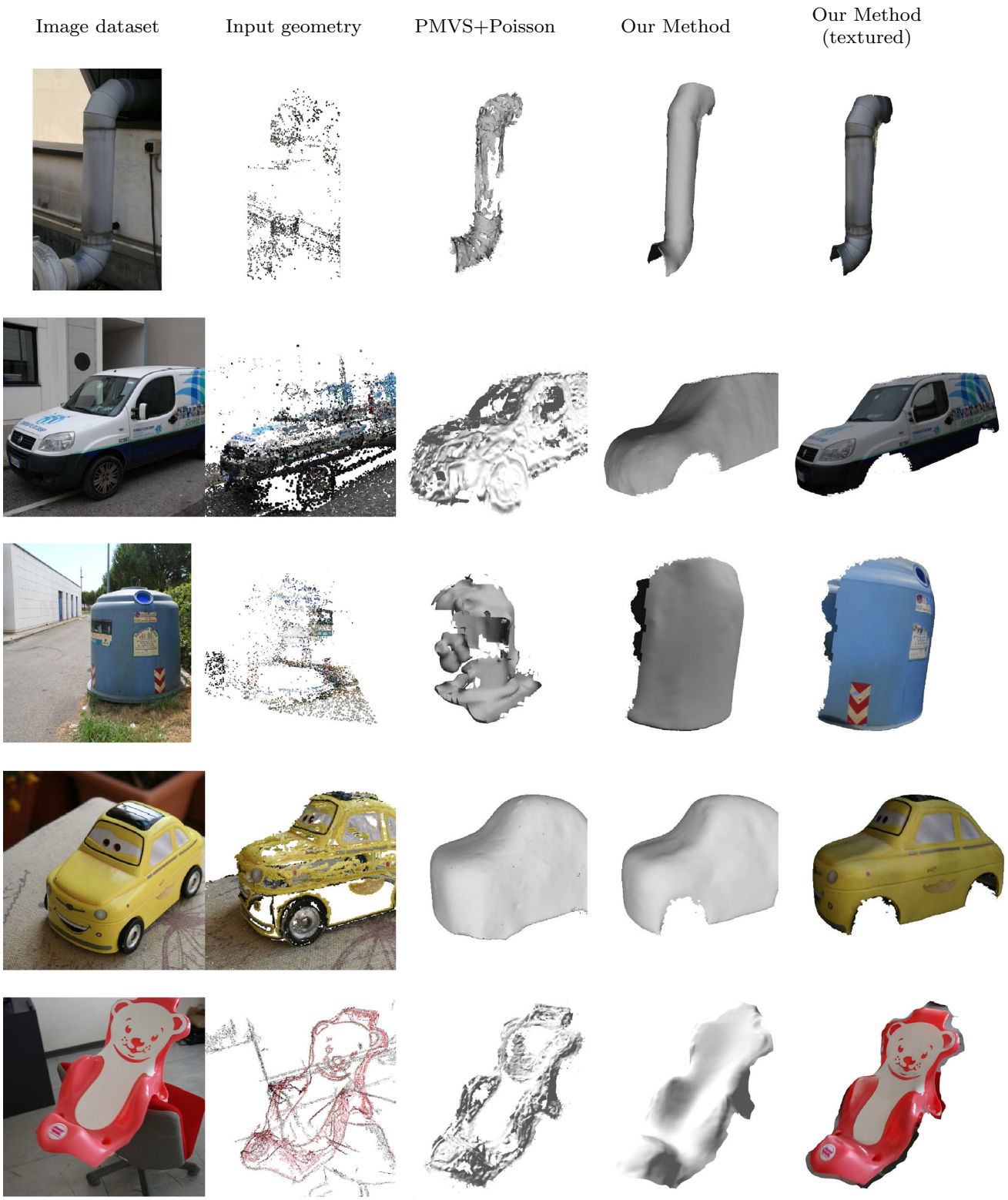


Fig. 9 Results of our experiments (*last two columns*) and comparison with reconstructions obtained by applying the Poisson surface reconstruction algorithm [24] to the output of the PMVS algorithm [17] (*3rd column*)

Table 1 Model reconstruction performance (time is in seconds)

Model	Camera (#)	Segm	Segm strokes	Curvature hint strokes	Init	Recons
Pipe	34	29.56	3; 12	3; 3	42.37	16.20
Van	50	65.95	8; 38	2; 4	184.13	27.63
Garbage bin	19	16.92	3; 10	2; 2	35.04	9.45
Toy car	75	75.63	10; 40	0; 0	1.1	141.32
Bath seat	28	33.205	5; 14	2; 2	829.37	59.76

sample image of the object, on the second the input used by our system. In these experiments, we used both dense and sparse reconstruction to demonstrate that the approach can work well in both cases. The input for the Pipe, the Garbage Bin, the Van are the points reconstructed by Bundler [37] after the camera calibration (note that these points are quite dense for the Van). Instead, for the Toy Car and the Bath Seat, we use the output of the PMVS algorithm as input. For comparison purposes, the images of the third row are produced by the Poisson reconstruction algorithm [24] run on the 3D points reconstructed using the PMVS [17]. Please note that the results obtained with the PMVS+Poisson reconstruction are of a poor quality w.r.t our final reconstruction (shown on the fourth row) also when the shape is quite complex like for the Bath Seat case. For the Pipe, the Garbage Bin and the Bath Seat, we needed only one constraint to specify the direction of zero curvature (see Table 1). Note also that, even if the input points are few and they are irregularly distributed (especially in the Pipe test), they are sufficient for a good initialization of our reconstruction algorithm.

Figure 10 shows two images of the reconstruction of the Pipe dataset. It can be easily seen how not imposing any constraint (left image) results in a deformed model where, although the smooth and overlapping energy terms are minimized, the final shape does not correspond to the one of the original pipe.

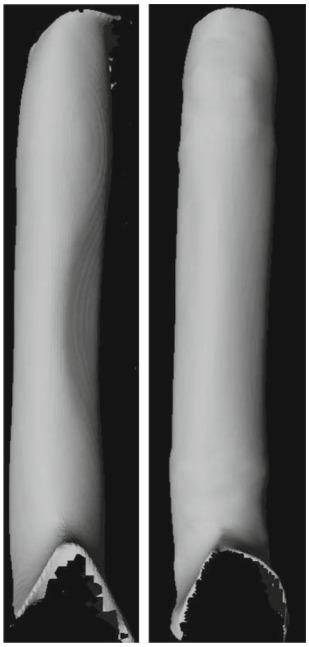


Fig. 10 Reconstruction of the Pipe without any curvature hint (*left*) and with curvature hint (*right*). Note that the reconstruction without constraints does not meet the cylindrical shape of the original pipe due to small reconstruction errors in the input points

6.1 Computation time

The lifecycle of a reconstruction consists of the following steps

1. The user selects the region of interest on one or more images.
2. The segmentation algorithm described in Sect. 3 and the initialization of range maps are run.
3. The user provides curvature hints as described in Sect. 4.
4. The reconstruction phase is run.

where steps 1–2 and 3–4 can be iterated to improve the final result.

Table 1 reports the time for the experiments run on a PC with Intel I7 4820k, 3.70 Ghz, equipped with 32 GB Ram, graphics board nvidia GeForce GTX 780 and the number of strokes provided by the user. The segmentation strokes are provided in the following way: n_1; n_2, where n_1 is the number of images annotated and n_2 is the total number of strokes on these images. The column "curvature hint strokes" reports the number of images annotated with the curvature constraints in the same way. For example, for the Garbage Bin, we put a vertical curvature constraint on its cylindrical part on two images; for the Pipe, the curvature constraint follows the pipe profile in three images. The Bath Seat requires only one curvature constraint in the center of its white lower part. The Van requires two strokes, one on the mirror and one on the hood to convey the right curvature of these two parts. It can be seen that all the times (segmentation, initialization and minimization) are roughly proportional to the number of photographs.

Note that the reconstruction time is fast (just a few seconds), while the segmentation and initialization times are quite slow. This is simply due to the fact that our system is still a prototype written in #F sharp. This limits the overall performance but not the reconstruction algorithm which is written entirely in GPU. The time for initialization is all due to the solution of the thin plate equation for each camera, except in the case of the Toy Car, where the initial depth map was obtained by triangulation in considerably less time. We point out that in the cases like the Bath seat, where the point's density is moderate–high both the approaches can be used. Hence, in this case, the initialization time can be reduced using the triangulation approach. Also, the number of strokes is very low in all cases, thanks to the technique shown in Sect. 4 which allows us to propagate the stroke done in one photo to the neighbor cameras.

7 Conclusions

We have proposed a framework for the user-assisted improvement of image-based 3D reconstruction of man-made objects. The framework is based on two novel techniques: a multi-view segmentation-based algorithm allowing a dataset of calibrated images to be efficiently segmented, and a GPU-friendly energy-based reconstruction algorithm with curvature constraints. A natural evolution of the proposed framework would be to add more possibilities for user hints, for example, to indicate sharp features or straight segments. In addition, we aim to modify the initialization phase of the reconstruction by integrating silhouette-based reconstruction methods to obviate the need for an initial sparse reconstruction.

Acknowledgments The research leading to these results was funded by EU FP7 project ICT FET Harvest4D (http://www.harvest4d.org/, G.A. no. 323567). The Museum Dataset is courtesy of Chaurasia et al. [45].

Appendix A: Algebraic derivations of the gradient of the energy terms

Smoothness term

The derivative of the smoothness term with respect to $z_{m,n}$ is simply:

$$\frac{\partial}{z_{m,n}} S(\mathbf{z}_h) = \sum_{i,j} 2\mathbf{z}_{xx}(i,j)\frac{\partial \mathbf{z}_{xx}(i,j)}{\partial z_{m,n}} + 4\mathbf{z}_{xy}(i,j)\frac{\partial \mathbf{z}_{xy}(i,j)}{\partial z_{m,n}} + 2\mathbf{z}_{yy}(i,j)\frac{\partial \mathbf{z}_{yy}(i,j)}{\partial z_{m,n}}\Delta x\Delta y \tag{14}$$

The unrolled formula, using central finite differences, is:

$$\nabla E_s(\mathbf{z})_{m,n} = \frac{\partial}{\partial z_{m,n}} \sum_{i,j} (z_{i+1,j} - 2z_{i,j} + z_{i-1,j})^2 + \frac{1}{8}(z_{i-1,j-1} - z_{i-1,j+1} - z_{i+1,j-1} + z_{i+1,j+1})^2 + (z_{i+1,j} - 2z_{i,j} + z_{i-1,j})^2 \tag{15}$$

which finally gives:

$$\nabla E_s(\mathbf{z})_{m,n} = 25\mathbf{z}_{m,n} + \frac{3}{2}(\mathbf{z}_{m,n+2} + \mathbf{z}_{m,n-2} + \mathbf{z}_{m+2,n} + \mathbf{z}_{m-2,n}) - 8(\mathbf{z}_{m+1,n} + \mathbf{z}_{m-1,n} + \mathbf{z}_{m,n+1} + \mathbf{z}_{m,n-1}) + \frac{1}{4}(\mathbf{z}_{m+2,n+2} + \mathbf{z}_{m+2,n-2} + \mathbf{z}_{m-2,n-2} + \mathbf{z}_{m-2,n+2}) \tag{16}$$

Coherence term

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{z}_{i,j}} R(\mathbf{z}_h) = \frac{\mathrm{d}}{\mathrm{d}\mathbf{z}_{i,j}}((g_k(i,j,\mathbf{z}_{i,j}) - f_k(i,j,\mathbf{z}_{i,j}))^2) = 2(g_k(i,j,z) - f_k(i,j,z)) \times \left(\frac{\mathrm{d}}{\mathrm{d}z} g_k(i,j,z) - \frac{\mathrm{d}}{\mathrm{d}z} f_k(i,j,z)\right) \tag{17}$$

So we need the derivatives of $g_k(i,j,z)$ and $f_k(i,j,z)$.

Since the cameras are calibrated, we know the matrix $\mathbf{R}_{h,k}$ that transforms the depth values from camera k to camera h:

$$\mathbf{R}_{h,k} = \mathbf{I}_k \mathbf{E}_k \mathbf{I}_h^{-1} \mathbf{E}_h^{-1} \tag{18}$$

where $\mathbf{I}$ and $\mathbf{E}$ are the intrinsic and extrinsic matrices of camera h and k.
$g_k(i,j,z)$ is defined as:

$$g_k(i,j,z) = s_w \cdot \mathbf{v}(z,i,j) = (\mathbf{r}_{30} i + \mathbf{r}_{31} j + \mathbf{r}_{33}) z + \mathbf{r}_{32} \tag{19}$$

where $\mathbf{s}_w$ is the row vector that selects component w, (i.e., $\mathbf{s}_w = [0\ 0\ 0\ 1]$) and the $\mathbf{r}_{ij}$ are the components of the $\mathbf{R}$ matrix. The derivative of g_k is then:

$$\frac{\mathrm{d}}{\mathrm{d}z} g_{i,j}(z) = (\mathbf{r}_{30} i + \mathbf{r}_{31} j + \mathbf{r}_{33}) \tag{20}$$

It is no surprise that the derivative does not depend on z, because function $g_k(i,j,z)$ simply returns the distance between a point along a line and a plane, which varies linearly.

For function $f_k(i,j,z)$, things are a little harder, because it describes the depth map z_k along the projection of the line on the image plane of camera k. Let us define the parametric function describing such a projection:

$$\mathbf{u}(z) : \mathbb{R} \to \mathbb{R}^2 = \frac{\mathbf{s}_{xy} \cdot \mathbf{v}(z)}{\mathbf{s}_w \cdot \mathbf{v}(z)} \tag{21}$$

Function f is then the composition of $z(x, y) : \mathbb{R}^2 \to \mathbb{R}$ with $\mathbf{u}$, i.e., $(z_k \cdot \mathbf{u}) : \mathbb{R} \to \mathbb{R}$. Therefore, the derivative of the composition is

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}z} f_k(i, j, z) = (z_k \cdot \mathbf{u}(z))' = \left[\frac{\partial z_k}{\partial x}, \frac{\partial z_k}{\partial y}\right] \\ \times (\mathbf{u}_x(z), \mathbf{u}_y(z)) \cdot \left[\frac{\mathrm{d}}{\mathrm{d}z}\mathbf{u}_x(z), \frac{\mathrm{d}}{\mathrm{d}z}\mathbf{u}_y(z)\right]\end{aligned} \tag{22}$$

We still need to define what $\frac{\mathrm{d}}{\mathrm{d}z}u_x(z)$ is (and, by symmetry, this will also yield its y-axis counterpart). This is the derivative

$$\frac{\mathrm{d}}{\mathrm{d}z}(\mathbf{s}_x \cdot \mathbf{v}(z)/\mathbf{s}_w \cdot \mathbf{v}(z)) \tag{23}$$

of a function with the form $\frac{\alpha z+\beta}{\gamma z+\delta}$ whose derivative is $\frac{\alpha\delta-\beta\gamma}{(\gamma z+\delta)^2}$. Therefore,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}z} f_k(i, j, z) = \left[\frac{\partial z_k}{\partial x}, \frac{\partial z_k}{\partial y}\right]\left[\frac{\mathbf{s}_{xy} \cdot \mathbf{v}(z)}{\mathbf{s}_w \cdot \mathbf{v}(z)}\right] \\ \times \left[\frac{\alpha_x \mathbf{r}_{32} - \mathbf{r}_{02}\gamma}{(\gamma z + \mathbf{r}_{32})^2}, \frac{\alpha_y \mathbf{r}_{32} - \mathbf{r}_{12}\gamma}{(\gamma z + \mathbf{r}_{32})^2}\right]\end{aligned} \tag{24}$$

where $\alpha_x = (\mathbf{r}_{00}i + \mathbf{r}_{01}j + \mathbf{r}_{03}) = \mathrm{d}\mathbf{v}_x$, $\alpha_y = (\mathbf{r}_{10}i + \mathbf{r}_{11}j + \mathbf{r}_{13}) = \mathrm{d}\mathbf{v}_y$ and $\gamma = (\mathbf{r}_{30}i + \mathbf{r}_{31}j + \mathbf{r}_{33}) = \mathrm{d}\mathbf{v}_w$. In conclusion, the gradient is:

$$\begin{aligned}\nabla R(z)_{m,n} &= 25\mathbf{z}^k_{m,n} \\ &+ \frac{3}{2}(\mathbf{z}^k_{m,n+2} + \mathbf{z}^k_{m,n-2} + \mathbf{z}^k_{m+2,n} + \mathbf{z}^k_{m-2,n}) \\ &- 8(\mathbf{z}^k_{m+1,n} + \mathbf{z}^k_{m-1,n} + \mathbf{z}^k_{m,n+1} + \mathbf{z}^k_{m,n-1}) \\ &+ \frac{1}{4}(\mathbf{z}^k_{m+2,n+2} + \mathbf{z}^k_{m+2,n-2} + \mathbf{z}^k_{m-2,n-2} + \mathbf{z}^k_{m-2,n+2}) \\ &+ 2(g_{m,n}(\mathbf{z}^k_{m,n}) - h_{m,n}(\mathbf{z}^k_{m,n}))\left(\frac{\mathrm{d}}{\mathrm{d}z} g_{m,n}(\mathbf{z}^k_{m,n})\right. \\ &\left. - \frac{\mathrm{d}}{\mathrm{d}z} h_{m,n}(\mathbf{z}^k_{m,n})\right)\end{aligned} \tag{25}$$

Curvature term

Proceeding as for the smoothness term:

$$\begin{aligned}\nabla C(\mathbf{z})_{m,n} = \frac{\partial}{\partial z_{m,n}} \sum_{i,j} \Big(z_{i+1,j} - 2z_{i,j} + z_{i-1,j}u^2 \\ + \frac{2\left(z_{i-1,j-1} - z_{i-,j+1} - z_{i+1,j-1} + z_{i+1,j+1}\right)uv}{4} \\ + z_{i+1,j} - 2z_{i,j} + z_{i-1,j}v^2\Big)^2\end{aligned} \tag{26}$$

which, after a trivial but tedious derivation, gives:

$$\begin{aligned}\nabla C(\mathbf{z})_{m,n} &= 2[24(u^4 + v^4) + 36u^2v^2)\ (z_{m,n}) \\ &- 16(u^4 - u^2v^2)(z_{m+1,n} + z_{m-1,n}) \\ &+ (4u^4 - 2u^2v^2)(z_{m+2,n} + z_{m-2,n}) \\ &- 16(v^4 - u^2v^2)(z_{m+1,n} + z_{m-1,n}) \\ &+ (4u^4 - 2u^2v^2)(z_{m+2,n} + z_{m-2,n} \\ &+ (4v^4 - 2u^2v^2)(z_{m,n+2} + z_{m,n-2} \\ &+ 8(u^3v + uv^3 + u^2v^2)(z_{m+1,n+1} + z_{m-1,n-1}) \\ &- 8(u^3v + uv^3 + u^2v^2)(z_{m-1,n+1} + z_{m+1,n-1}) \\ &+ u^2v^2(z_{m+2,n+2} + z_{m-2,n-2})]\end{aligned} \tag{27}$$

Appendix B: An example of handling discontinuities with the LUT table

In this section, we show how the coefficients of the LUT table are derived in a specific case. Let us consider, Eq. (14) for the gradient of the smoothness term, which is a weighted sum of second derivatives, and consider one of the terms of the sum:

$$A = \frac{\partial}{\partial z_{m,n}}\mathbf{z}^2_{xx}(n-2, m) = 2\mathbf{z}_{xx}(n-2, m)\frac{\partial \mathbf{z}_{xx}(n-2, m)}{\partial z_{m,n}} \tag{28}$$

If $\mathbf{z}_{xx}(n-2, m)$ is computed by central finite differences, we have:

$$\begin{aligned}\frac{\partial \mathbf{z}_{xx}(n-2, m)}{\partial z_{m,n}} &= \frac{\partial}{\partial z_{m,n}}(\mathbf{z}(n-3, m) \\ &-2\mathbf{z}(n-2, m) + \mathbf{z}(n-1, m)) \\ &= 0 \Rightarrow A = 0\end{aligned} \tag{29}$$

In other words, since $z_{m,n}$ does not appear in the computation of $\mathbf{z}_{xx}(n-2, m)$ the derivative on $z_{m,n}$, and thus A, is zero. Referring to Fig. 7, this is because the entry for this configuration (first row) is null.

On the other hand, if $\mathbf{z}_{xx}(n-2, m)$ is computed by forward finite differences, we have:

$$\begin{aligned}\frac{\partial}{\partial z_{m,n}}\mathbf{z}_{xx}(n-2, m) = \frac{\partial}{\partial z_{m,n}}(\mathbf{z}(n-2, m) \\ -2\mathbf{z}(n-1, m) + \mathbf{z}(n, m)) = 1\end{aligned} \tag{30}$$

and thus:

$$\begin{aligned}A &= 2\mathbf{z}_{xx}(n-2, m) \\ &= \mathbf{2z}(n-2, m) - \mathbf{4z}(n-1, m) + \mathbf{2z}(n, m)\end{aligned} \tag{31}$$

This gives the coefficients to apply as just shown in Fig. 7 (second row).

References

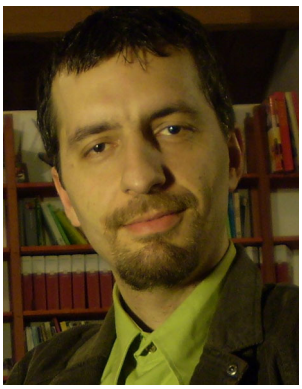
1. Adarsh Kowdle, S.N.S., Szeliski, R.: Multiple view object cosegmentation using appearance and stereo cues. In: European Conference on Computer Vision (ECCV 2012) (2012)
2. Alexe, B., Deselaers, T., Ferrari, V.: Classcut for unsupervised class segmentation. In: ECCV2010, pp. 8–10 (2010). http://www.springerlink.com/index/D62206186631X328.pdf
3. Bao, S.Y., Chandraker, M., Lin, Y., Savarese, S.: Dense object reconstruction with semantic priors. In: CVPR, pp. 1264–1271 (2013). doi:10.1109/CVPR.2013.167. http://www.cv-foundation.org/openaccess/CVPR2013.py
4. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA J. Numer. Anal. 8(1), 141–148 (1988). doi:10.1093/imanum/8.1.141. http://imajna.oxfordjournals.org/content/8/1/141.abstract
5. Bleyer, M., Rother, C., Kohli, P., Scharstein, D., Sinha, S.: Object stereo–joint stereo matching and object segmentation. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11, pp. 3081–3088. IEEE Computer Society, Washington (2011). doi:10.1109/CVPR.2011.5995581
6. Boykov, Y., Jolly, M.: Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In: ICCV 2001, pp. 105–112 (2001). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=937505
7. Bradley, D., Boubekeur, T., Heidrich, W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In: CVPR. IEEE Computer Society (2008). doi:10.1109/CVPR.2008.4587792
8. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia (2000)
9. Campbell, N., Vogiatzis, G., Hernandez, C., Cipolla, R.: Automatic object segmentation from calibrated images. In: Visual Media Production Conference, pp. 126–137 (2011). doi:10.1109/CVMP.2011.21
10. Campbell, N.D., Vogiatzis, G., Hernández, C., Cipolla, R.: Using multiple hypotheses to improve depth-maps for multi-view stereo. In: Proceedings of the 10th European Conference on Computer Vision: Part I, ECCV '08, pp. 766–779. Springer, Berlin (2008). doi:10.1007/978-3-540-88682-2_58
11. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96, pp. 303–312. ACM, New York (1996). doi:10.1145/237170.237269
12. Djelouah, A., Franco, J.S., Boyer, E., Clerc, F.L., Prez, P.: N-tuple color segmentation for multi-view silhouette extraction. In: ECCV (5) '12, pp. 818–831 (2012)
13. Djelouah, A., Franco, J.S., Boyer, E., Le Clerc, F., Perez, P.: Multi-view object segmentation in space and time. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 2640–2647 (2013). doi:10.1109/ICCV.2013.328
14. Freedman, D.: Interactive graph cut based segmentation with shape priors. In: CVPR'05, vol. 1, pp. 755–762 (2005). doi:10.1109/CVPR.2005.191
15. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Manhattan-world stereo. In: CVPR **2009**, 1422–1429 (2009). doi:10.1109/CVPR.2009.5206867. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206867
16. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R.: Towards internet-scale multi-view stereo. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
17. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE Trans. Pattern Anal. Mach. Intell. **32**(8), 1362–1376 (2010). doi:10.1109/TPAMI.2009.161
18. Gallup, D., Frahm, J., Pollefeys, M.: Piecewise planar and non-planar stereo for urban scene reconstruction. In: CVPR'10, pp. 1418–1425 (2010). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5539804
19. Goesele, M., Curless, B., Seitz, S.M.: Multi-view stereo revisited. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, CVPR '06, pp. 2402–2409. IEEE Computer Society, Washington (2006). doi:10.1109/CVPR.2006.199
20. Gopi, M., Krishnan, S., Silva, C.: Surface reconstruction based on lower dimensional localized delaunay triangulation. Comput. Graph. Forum **19**(3), 467–478 (2000). doi:10.1111/1467-8659.00439
21. Hoff III, K.E., Keyser, J., Lin, M., Manocha, D., Culver, T.: Fast computation of generalized voronoi diagrams using graphics hardware. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 277–286. ACM Press/Addison-Wesley, New York (1999). doi:10.1145/311535.311567
22. Jancosek, M., Pajdla, T.: Segmentation based multi-view stereo. In: Computer Vision Winter Workshop (2009)
23. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: active contour models. IJCV **1**(4), 321–331 (1988)
24. Kazhdan, M.M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Symposium on Geometry Processing, pp. 61–70 (2006)
25. Kolev, K., Brox, T., Cremers, D.: Robust variational segmentation of 3D objects from multiple views. Pattern Recognit. 688–697 (2006). http://www.springerlink.com/index/m68268261t8h0641.pdf
26. Kolev, K., Klodt, M., Brox, T., Cremers, D.: Propagated photoconsistency and convexity in variational multiview 3D reconstruction. In: Workshop on Photometric Analysis for Computer Vision, Rio de Janeiro (2007)
27. Kolev, K., Pock, T., Cremers, D.: Anisotropic minimal surfaces integrating photoconsistency and normal information for multiview stereo. In: ECCV'10, Heraklion (2010)
28. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 147–59 (2004). doi:10.1109/TPAMI.2004.1262177. http://www.ncbi.nlm.nih.gov/pubmed/15376891
29. Mortensen, E., Barrett, W.: Intelligent scissors for image composition. In: Proceedings of the 22nd Annual Conference, vol. 84602(801) (1995). http://dl.acm.org/citation.cfm?id=218442
30. Nan, L., Sharf, A., Chen, B.: 2D–3D lifting for shape reconstruction. Comput. Graph. Forum **33**(7), 249–258 (2014). doi:10.1111/cgf.12493
31. Nguyen, H., Wnsche, B., Delmas, P., Lutteroth, C., Zhang, E.: A robust hybrid image-based modeling system. Vis. Comput. 1–16 (2015). doi:10.1007/s00371-015-1078-y
32. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. Comput. Graph. Forum **28**(2), 493–501 (2009)
33. Rother, C., Kolmogorov, V.: Grabcut: interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. (2004). http://dl.acm.org/citation.cfm?id=1015720
34. Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: IEEE Computer Society Conference on Computer

Vision and Pattern Recognition (CVPR 2006), vol. 1, pp. 519–528 (2006). doi:10.1109/CVPR.2006.19
35. Seitz, S.M., Dyer, C.R.: Photorealistic scene reconstruction by voxel coloring. In: Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 1067–1073 (1997)
36. Sinha, S., Steedly, D., Szeliski, R.: Piecewise planar stereo for image-based rendering. In: ICCV, pp. 1881–1888 (2009). doi:10.1109/ICCV.2009.5459417
37. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3D. ACM Trans. Graph. **25**(3), 835–846 (2006). doi:10.1145/1141911.1141964
38. Sormann, M., Zach, C., Karner, K.: Graph cut based multiple view segmentation for 3D reconstruction. In: Third International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 1085–1092 (2006). doi:10.1109/3DPVT.2006.70
39. Sormann, M., Zach, C., Karner, K.: Graph cut based multiple view segmentation for 3D reconstruction. In: Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT '06), pp. 1085–1092 (2006). doi:10.1109/3DPVT.2006.70. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4155842
40. Vicente, S.: Graph cut based image segmentation with connectivity priors. In: CVPR '08, pp. 1–8 (2008). doi:10.1109/CVPR.2008.4587440
41. Wahba, G.: Spline Models for Observational Data. SIAM, Philadelphia (1990)
42. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Schematic surface reconstruction. In: CVPR, **2012**, 1498–1505 (2012). doi:10.1109/CVPR.2012.6247839
43. Yezzi, A., Soatto, S.: Stereoscopic segmentation. IJCV **53**(1), 31–43 (2003). http://www.springerlink.com/index/V812463066072825.pdf
44. Zhu, C., Leow, W.: Textured mesh surface reconstruction of large buildings with multi-view stereo. Vis. Comput. **29**(6–8), 609–615 (2013). doi:10.1007/s00371-013-0827-z
45. Chaurasia, G., Duchêne, S., Sorkine-Hornung, O., Drettakis, G.: Depth Synthesis and Local Warps for Plausible Image-based Navigation. ACM Trans. Graph. **32** (2013). http://www-sop.inria.fr/reves/Basilic/2013/CDSD13

Andrea Baldacci received a degree in Information Engineering from University of Pisa in 2012. In 2013, he joined the Visual Computing Laboratory of ISTI-CNR, Pisa, as a PhD student of Pisa University. His research interests are in the field of Computer Graphics and Computer Vision and include image-based reconstruction algorithm, real-time rendering, computational geometry and stereo image processing.

Daniele Bernabei received a degree in Computer Science (Laurea) from the University of Pisa and a PhD from the same University in 2012 working with the Visual Computing Laboratory of the ISTI-CNR in Pisa, Italy. His research interests are in advanced photorealistic rendering algorithms and image segmentation. Now, he works for The Foundry Visionmongers, a visual effects software development company based in London.

Massimiliano Corsini received a PhD degree in Information and Telecommunication Engineering from the University of Florence. Currently, he is a Researcher at the Visual Computing Laboratory of the ISTI-CNR in Pisa, Italy. His research interests are in the fields of Computer Graphics, Computer Vision and Image Processing and include 3D watermarking, perceptual metrics, visual appearance acquisition and modeling, registration algorithms and image-based relighting. He published about 50 papers in peer-reviewed international conferences and journals. He also collaborated in several National and International projects and served on numerous program committees.



Fabio Ganovelli is a Researcher at the Visual Computing Laboratory of the Istituto Scienza e Tecnologie dell'Informazione (ISTI), which is part of the Italian National Research Council (CNR). His research interests are in the field of Computer Graphics. He published around 50 papers in several topics including deformable objects, multiresolution rendering techniques, photorealistic rendering and geometry processing.

Roberto Scopigno is a Research Director with CNR-ISTI and leads the Visual Computing Lab. He graduated in Computer Science at the University of Pisa in 1984. He is engaged in research projects concerned with 3D scanning, surface reconstruction, multiresolution, scientific visualization, and cultural heritage. He published more than 150 papers in international refereed journals/conferences. Roberto has been the responsible person for CNR-ISTI in several EU projects and co-chaired several international conferences. He is member of the Eurographics Association, was awarded the EG "Outstanding Technical Contribution Award" in 2008, served as Chair of the Eurographics Association (2009–2010), Co-Editor in Chief of the Computer Graphics Forum journal (2001–2010) and member of the Editorial Board of the *ACM Journal on Computing and Cultural Heritage*.