Accurate, Dense, and Robust Multiview Stereopsis

Yasutaka Furukawa and Jean Ponce, *Fellow*, *IEEE*

Abstract—This paper proposes a novel algorithm for multiview stereopsis that outputs a dense set of small rectangular patches covering the surfaces visible in the images. Stereopsis is implemented as a *match, expand, and filter* procedure, starting from a sparse set of matched keypoints, and repeatedly expanding these before using visibility constraints to filter away false matches. The keys to the performance of the proposed algorithm are effective techniques for enforcing local photometric consistency and global visibility constraints. Simple but effective methods are also proposed to turn the resulting patch model into a mesh which can be further refined by an algorithm that enforces both photometric consistency and regularization constraints. The proposed approach automatically detects and discards outliers and obstacles and does not require any initialization in the form of a visual hull, a bounding box, or valid depth ranges. We have tested our algorithm on various data sets including objects with fine surface details, deep concavities, and thin structures, outdoor scenes observed from a restricted set of viewpoints, and "crowded" scenes where moving obstacles appear in front of a static structure of interest. A quantitative evaluation on the Middlebury benchmark [1] shows that the proposed method outperforms all others submitted so far for four out of the six data sets.

Index Terms—Computer vision, 3D/stereo scene analysis, modeling and recovery of physical attributes, motion, shape.

1 Introduction

Multiview stereo (MVS) matching and reconstruction is a key ingredient in the automated acquisition of geometric object and scene models from multiple photographs or video clips, a process known as image-based modeling or 3D photography. Potential applications range from the construction of realistic object models for the film, television, and video game industries, to the quantitative recovery of metric information (metrology) for scientific and engineering data analysis. According to a recent survey provided by Seitz et al. [2], state-of-the-art MVS algorithms achieve relative accuracy better than 1/200 (1 mm for a 20 cm wide object) from a set of low-resolution (640×480) images. They can be roughly classified into four classes according to the underlying object models: *Voxel-based* approaches [3], [4], [5], [6], [7], [8], [9] require knowing a bounding box that contains the scene, and their accuracy is limited by the resolution of the voxel grid. Algorithms based on *deformable polygonal meshes* [10], [11], [12] demand a good starting point —for example, a visual hull model [13]—to initialize the corresponding optimization process, which limits their applicability. Approaches based on *multiple depth maps* [14], [15], [16] are more flexible, but require fusing individual depth maps into a single 3D model. Finally, *patch-based* methods [17], [18] represent scene surfaces by collections of small patches (or *surfels*). They are simple and effective and often suffice for visualization purposes via point-based rendering technique [19], but require a postprocessing step to turn them into a mesh model that is more suitable for image-based modeling applications.[1]

MVS algorithms can also be thought of in terms of the data sets they can handle, for example, images of

- *objects*, where a single, compact object is usually fully visible in a set of uncluttered images taken from all around it, and it is relatively straightforward to extract the apparent contours of the object and compute its visual hull;
- *scenes*, where the target object(s) may be partially occluded and/or embedded in clutter and the range of viewpoints may be severely limited, preventing the computation of effective bounding volumes (typical examples are outdoor scenes with buildings, vegetation, etc.); and
- *crowded scenes*, where moving obstacles appear in different places in multiple images of a static structure of interest (e.g., people passing in front of a building).

The underlying object model is an important factor in determining the flexibility of an approach, and voxel-based or polygonal mesh-based methods are often limited to object data sets, for which it is relatively easy to estimate an initial bounding volume or often possible to compute a visual hull model. Algorithms based on multiple depth maps and collections of small surface patches are better suited to the more challenging scene data sets. Crowded scenes are even more difficult. Strecha et al. [15] use expectation maximization and multiple depth maps to

1. A patch-based surface representation is also used in [20] but in a context of scene flow capture.

- *Y. Furukawa is with Google Inc., 651 N. 34th St., Seattle, WA 98103. E-mail: furukawa@cs.washington.edu.*
- *J. Ponce is with the Ecole Normale Supérieure and the Willow project-team of LIENS, ENS/INRIA/CNRS UMR 8548, 45, rue d'Ulm, 75230 Paris Cedex 05, France. E-mail: Jean.Ponce@ens.fr.*

Manuscript received 22 Aug. 2008; revised 30 Mar. 2009; accepted 10 Aug. 2009; published online 19 Aug. 2009.
Recommended for acceptance by G.D. Hager.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-08-0560.
Digital Object Identifier no. 10.1109/TPAMI.2009.161.

Fig. 1. Overall approach. From left to right: A sample input image, detected features, reconstructed patches after the initial matching, final patches after expansion and filtering, and the mesh model.

reconstruct a crowded scene despite the presence of occluders, but their approach is limited to a small number of images (typically three) as the complexity of their model is exponential in the number of input images. Goesele et al. [21] have also proposed an algorithm to handle Internet photo collections containing obstacles and produce impressive results with a clever view selection scheme.

In this paper, we take a hybrid approach that is applicable to all three types of input data. More concretely, we first propose a flexible patch-based MVS algorithm that outputs a dense collection of small oriented rectangular patches, obtained from pixel-level correspondences and tightly covering the observed surfaces except in small textureless or occluded regions. The proposed algorithm consists of a simple *match, expand, and filter* procedure (Fig. 1): 1) *Matching*: Features found by Harris and difference-of-Gaussians operators are first matched across multiple pictures, yielding a sparse set of patches associated with salient image regions. Given these initial matches, the following two steps are repeated n times ($n = 3$ in all our experiments). 2) *Expansion:* A technique similar to [17], [18], [22], [23], [24] is used to spread the initial matches to nearby pixels and obtain a dense set of patches. 3) *Filtering:* Visibility (and a weak form of regularization) constraints are then used to eliminate incorrect matches. Although our patch-based algorithm is similar to the method proposed by Lhuillier and Quan [17], it replaces their greedy expansion procedure by iteration between expansion and filtering steps, which allows us to process complicated surfaces and reject outliers more effectively. Optionally, the resulting patch model can be turned into a triangulated mesh by simple but efficient techniques, and this mesh can be further refined by a mesh-based MVS algorithm that enforces the photometric consistency with regularization constraints. The additional computational cost of the optional step is balanced by the even higher accuracy it affords. Our algorithm does not require any initialization in the form of a visual hull model, a bounding box, or valid depth ranges. In addition, unlike many other methods that basically assume fronto-parallel surfaces and only estimate the depth of recovered points, it actually estimates the surface orientation while enforcing the local photometric consistency, which is important in practice to obtain accurate models for data sets with sparse input images or without salient textures. As shown by our experiments, the proposed algorithm effectively handles the three types of data mentioned above, and, in particular, it outputs accurate object and scene models with fine surface detail despite low-texture regions, large concavities, and/or thin, high-curvature parts. A quantitative evaluation on the Middlebury benchmark [1] shows that the proposed method outperforms all others submitted so far in terms of both *accuracy* and *completeness* for four out of the six data sets.

The rest of this paper is organized as follows: Section 2 presents the key building blocks of the proposed approach. Section 3 presents our patch-based MVS algorithm, and Section 4 describes how to convert a patch model into a mesh and our polygonal mesh-based refinement algorithm. Experimental results and discussion are given in Section 5, and Section 6 concludes the paper with some future work. The implementation of the patch-based MVS algorithm (PMVS) is publicly available at [25]. A preliminary version of this paper appeared in [26].

2 Key Elements of the Proposed Approach

The proposed approach can be decomposed into three steps: a patch-based MVS algorithm that is the core reconstruction step in our approach and reconstructs a set of oriented points (or *patches*) covering the surface of an object or a scene of interests; the conversion of the patches into a polygonal mesh model; and finally a polygonal-mesh based MVS algorithm that refines the mesh. In this section, we introduce a couple of fundamental building blocks of the patch-based MVS algorithm, some of which are also used in our mesh refinement algorithm.

2.1 Patch Model

A patch p is essentially a local tangent plane approximation of a surface. Its geometry is fully determined by its center $\mathbf{c}(p)$, unit normal vector $\mathbf{n}(p)$ oriented toward the cameras observing it, and a *reference* image $R(p)$ in which p is visible (see Fig. 2). More concretely, a patch is a (3D) rectangle, which is oriented so that one of its edges is parallel to the x-axis of the reference camera (the camera associated with $R(p)$). The extent of the rectangle is chosen so that the smallest axis-aligned square in $R(p)$ containing its image projection is of size $\mu \times \mu$ pixels in size (μ is either 5 or 7 in all of our experiments).

2.2 Photometric Discrepancy Function

Let $V(p)$ denote a set of images in which p is visible (see Section 3 on how to estimate $V(p)$ and choose the reference

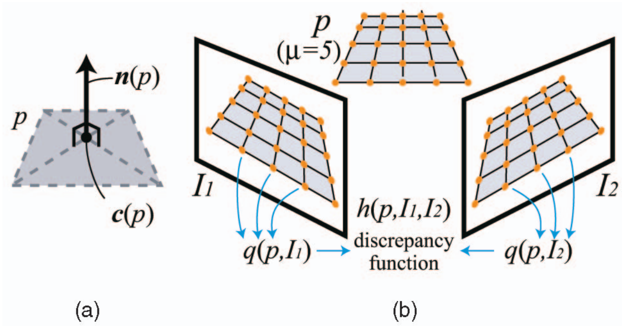


Fig. 2. (a) A patch p is a (3D) rectangle with its center and normal denoted as $\mathbf{c}(p)$ and $\mathbf{n}(p)$, respectively. (b) The photometric discrepancy $f(p, I_1, I_2)$ of a patch is given by one minus the normalized cross correlation score between sets $\mathbf{q}(p, I_i)$ of sampled pixel colors. See text for details.

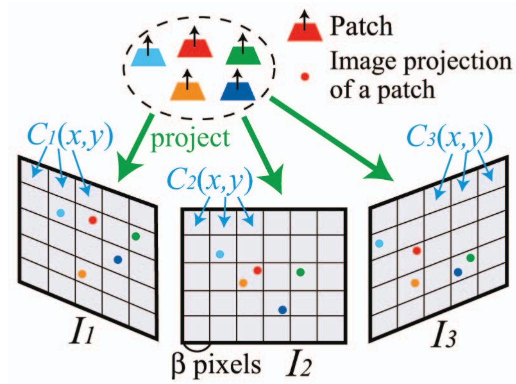


Fig. 3. We keep track of image projections of reconstructed patches in their visible images to perform fundamental tasks such as accessing neighboring patches, enforcing regularization, etc. See text for more details.

image $R(p) \in V(p)$). The photometric discrepancy function $g(p)$ for p is defined as

$$g(p) = \frac{1}{|V(p) \setminus R(p)|} \sum_{I \in V(p) \setminus R(p)} h(p, I, R(p)), \quad (1)$$

where $h(p, I_1, I_2)$ is, in turn, defined to be a pairwise photometric discrepancy function between images I_1 and I_2. More concretely (see Fig. 2), given a pair of visible images I_1 and I_2, $h(p, I_1, I_2)$ is computed by 1) overlaying a $\mu \times \mu$ grid on p; 2) sampling pixel colors $\mathbf{q}(p, I_i)$ through bilinear interpolation at image projections of all the grid points in each image I_i[2]; and 3) computing one minus the normalized cross correlation score between $\mathbf{q}(p, I_1)$ and $\mathbf{q}(p, I_2)$.[3]

We have so far assumed that the surface of an object or a scene is Lambertian, and the photometric discrepancy function $g(p)$ defined above may not work well in the presence of specular highlights or obstacles (e.g., pedestrians in front of buildings, as shown in Fig. 10). In the proposed approach, we handle non-Lambertian effects by simply ignoring images with bad photometric discrepancy scores. Concretely, only images whose pairwise photometric discrepancy score with the reference image $R(p)$ is below a certain threshold α are used for the evaluation (see Section 3 for the choice of this threshold):

$$V^*(p) = \{I | I \in V(p), h(p, I, R(p)) \le \alpha\}, \quad (2)$$

$$g^*(p) = \frac{1}{|V^*(p) \setminus R(p)|} \sum_{I \in V^*(p) \setminus R(p)} h(p, I, R(p)). \quad (3)$$

We simply replaced $V(p)$ in (1) with the filtered one $V^*(p)$ to obtain the new formula (3). Note that $V^*(p)$ contains the reference image $R(p)$ by definition. Also note that the new discrepancy function $g^*(p)$ still does not work if $R(p)$ contains specular highlights or obstacles, but our patch generation algorithm guarantees that this does not occur, as will be detailed in Section 3.1.2.

2.3 Patch Optimization

Having defined the photometric discrepancy function $g^*(p)$, our goal is to recover patches whose discrepancy scores are small. Each patch p is reconstructed separately in two steps: 1) initialization of the corresponding parameters, namely, its center $\mathbf{c}(p)$, normal $\mathbf{n}(p)$, visible images $V^*(p)$, and the reference image $R(p)$; and 2) optimization of its geometric component, $\mathbf{c}(p)$ and $\mathbf{n}(p)$. Simple but effective initialization methods for the first step are detailed in Sections 3 and 4, and we focus here on the second optimization step. The geometric parameters, $\mathbf{c}(p)$ and $\mathbf{n}(p)$, are optimized by simply minimizing the photometric discrepancy score $g^*(p)$ with respect to these unknowns. To simplify computations, we constrain $\mathbf{c}(p)$ to lie on a ray such that its image projection in one of the visible images does not change (see Section 3 for the choice of the image), reducing its number of degrees of freedom to one and solving only for a depth. $\mathbf{n}(p)$ is, in turn, parameterized by Euler angles (yaw and pitch), yielding an optimization problem within three parameters only, which is solved by a conjugate gradient method [28].

2.4 Image Model

The biggest advantage of the patch-based surface representation is its flexibility. However, due to the lack of connectivity information, it is not easy to just search or access neighboring patches, enforce regularization, etc. In our approach, we keep track of the image projections of reconstructed patches in their visible images to help performing these tasks. Concretely, we associate with each image I_i a regular grid of $\beta_1 \times \beta_1$ pixels cells $C_i(x, y)$ as in Fig. 3 ($\beta_1 = 2$ in our experiments). Given a patch p and its visible images $V(p)$, we simply project p into each image in $V(p)$ to identify the corresponding cell. Then, each cell $C_i(x, y)$ remembers the set of patches $Q_i(x, y)$ that project into it. Similarly, we use $Q_i^*(x, y)$ to denote the patches that are obtained by the same procedure but with $V^*(p)$ instead of $V(p)$. Please see the next section for how we make use of $Q_i(x, y)$ and $Q_i^*(x, y)$ to effectively reconstruct patches.

3 Patch Reconstruction

Our patch-based MVS algorithm attempts to reconstruct at least one patch in every image cell $C_i(x, y)$. It is divided into three steps: 1) initial feature matching, 2) patch expansion, and 3) patch filtering. The purpose of the initial feature

2. We have also tried bicubic interpolation but have not observed noticeable differences.

3. See [27] for an example of other photometric discrepancy functions.

matching step is to generate a sparse set of patches (possibly containing some false positives). The expansion and the filtering steps are iterated n times ($n = 3$ in our experiments) to make patches dense and remove erroneous matches. The three steps are detailed in the following sections.

3.1 Initial Feature Matching

3.1.1 Feature Detection

First, we detect blob and corner features in each image using the Difference-of-Gaussian (DoG) and Harris operators. Briefly, let us denote by G_σ a 2D Gaussian with standard deviation σ. The response of the DoG filter, at some image point, is given by $D = |(G_{\sigma_0} - G_{\sqrt{2}\sigma_0}) * I|$, where $*$ denotes the 2D convolution operator. The response of the Harris filter is, in turn, defined as $H = \det(M) - \lambda \text{trace}^2(M)$, where $M = G_{\sigma_1} * (\nabla I \nabla I^T)$ and $\nabla I = [\frac{\partial I}{\partial x} \frac{\partial I}{\partial y}]^T$. ∇I is computed by convolving the image I with the partial derivatives of the Gaussian G_{σ_2}. Note that $(\nabla I \nabla I^T)$ is a 2×2 matrix, and G_{σ_1} is convolved with each of its elements to obtain M. We use $\sigma_0 = \sigma_1 = \sigma_2 = 1$ pixel and $\lambda = 0.06$ in all of our experiments. To ensure uniform coverage, we lay over each image a coarse regular grid of $\beta_2 \times \beta_2$ pixels blocks and return as features the η local maxima with the strongest responses in each block for each operator (we use $\beta_2 = 32$ and $\eta = 4$ in all our experiments).

3.1.2 Feature Matching

Consider an image I_i and denote by $O(I_i)$ the optical center of the corresponding camera. For each feature f detected in I_i, we collect in the other images the set F of features f' of the same type (Harris or DoG) that lie within two pixels from the corresponding epipolar lines, and triangulate the 3D points associated with the pairs (f, f'). We then consider these points in order of increasing distance from $O(I_i)$ as potential patch centers and attempt to generate a patch from the points one by one until we succeed,[4] using the following procedure: Given a pair of features (f, f'), we first construct a patch candidate p with its center $\mathbf{c}(p)$, normal vector $\mathbf{n}(p)$, and reference image $R(p)$ initialized as

$$\mathbf{c}(p) \leftarrow \{\text{Triangulation from } f \text{ and } f'\}, \quad (4)$$

$$\mathbf{n}(p) \leftarrow \overrightarrow{\mathbf{c}(p)O(I_i)} / |\overrightarrow{\mathbf{c}(p)O(I_i)}|, \quad (5)$$

$$R(p) \leftarrow I_i. \quad (6)$$

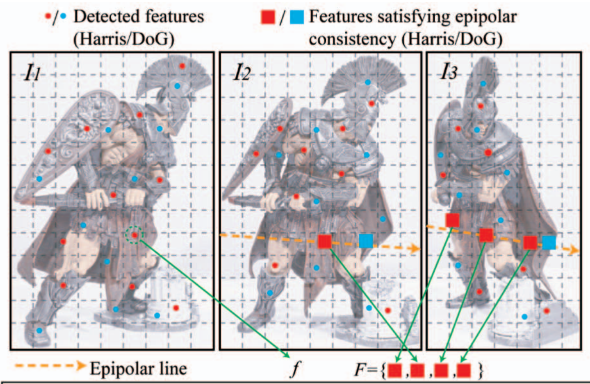
Since reconstructed patches are sparse with possibly many false positives in the initial feature matching step, we simply assume that the patch is visible in an image I_i when the angle between the patch normal and the direction from the patch to the optical center $O(I_i)$ of the camera is below a certain threshold ι ($\iota = \pi/3$ in our experiments)[5]:

$$V(p) \leftarrow \left\{ I | \mathbf{n}(p) \cdot \overrightarrow{\mathbf{c}(p)O(I)} / |\overrightarrow{\mathbf{c}(p)O(I)}| > \cos(\iota) \right\}. \quad (7)$$

4. Empirically, this heuristic has proven to be effective in selecting mostly correct matches at a modest computational expense.

5. In the next patch expansion step, where patches become dense and less erroneous, a simple depth-map test is used for visibility estimation.

(a)



(b)

```
Input: Features detected in each image.
Output: Initial sparse set of patches P.

P ← φ;
For each image I with optical center O(I)
  For each feature f detected in I
    F ← {Features satisfying the epipolar consistency};
    Sort F in an increasing order of distance from O(I));
    For each feature f' ∈ F
      // Test a patch candidate p;
      Initialize c(p), n(p) and R(p); // Eqs. (4, 5, 6)
      Initialize V(p) and V*(p); // Eqs. (2, 7)
      Refine c(p) and n(p); // (Sect.II-C)
      Update V(p) and V*(p); // Eqs. (2, 7)
      If |V*(p)| < γ
        Go back to the innermost For loop (failure);
      Add p to the corresponding Q_j(x,y) and Q*_j(x,y);
      Remove features from the cells where p was stored;
      Add p to P;
      Exit innermost For loop;
```

Fig. 4. Feature matching algorithm. (a) An example showing the features $f' \in F$ satisfying the epipolar constraint in images I_2 and I_3 as they are matched to feature f in image I_1 (this is an illustration only, not showing actual detected features). (b) The matching algorithm.

$V^*(p)$ is also initialized from $V(p)$ by (2). Having initialized all of the parameters for the patch candidate p, we refine $\mathbf{c}(p)$ and $\mathbf{n}(p)$ by the patch optimization procedure described in Section 2.3, then update the visibility information $V(p)$ and $V^*(p)$ with the refined geometry according to (7) and (2). During the optimization, $\mathbf{c}(p)$ is constrained to lie on a ray such that its image projection in $R(p)$ does not change. If $|V^*(p)|$ is at least γ, that is, if there exist at least γ images with low photometric discrepancy, the patch generation is deemed a success and p is stored in the corresponding cells of the visible images (update of $Q_i(x, y)$ and $Q_i^*(x, y)$). Note that we have used (2) to compute $V^*(p)$ before and after the optimization, and the threshold α in (2) is set to 0.6 and 0.3, respectively, because the photometric discrepancy score of a patch may be high before the optimization with its imprecise geometry. Also note that, in order to speed up the computation, once a patch has been reconstructed and stored in a cell, all of the features in the cell are removed and are not used anymore. The overall algorithm description for this step is given in Fig. 4. Finally, let us explain how this matching procedure is able to handle image artifacts such as specular highlights and obstacles successfully and guarantee that reference images do not contain such artifacts. If the matching procedure starts with a feature in an image containing artifacts, the image

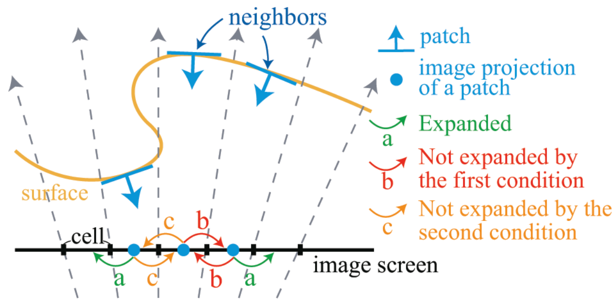


Fig. 5. (a) Given an existing patch, an expansion procedure is performed to generate new ones for the neighboring empty image cells in its visible images. The expansion procedure is not performed for an image cell (b) if there already exists a neighboring patch reconstructed there or (c) if there is a depth discontinuity when viewed from the camera. See text for more details.

becomes a reference and the patch optimization fails. However, this does not prevent the procedure starting from another image without artifacts, which will succeed.[6]

3.2 Expansion

The goal of the expansion step is to reconstruct at least one patch in every image cell $C_i(x, y)$, and we repeat taking existing patches and generating new ones in nearby empty spaces. More concretely, given a patch p, we first identify a set of *neighboring* image cells $\mathbf{C}(p)$ satisfying certain criteria, then perform a patch expansion procedure for each one of these, as detailed in the following sections.

3.2.1 Identifying Cells for Expansion

Given a patch p, we initialize $C(p)$ by collecting the neighboring image cells in its each visible image:

$$\mathbf{C}(p) = \{C_i(x', y') | p \in Q_i(x, y), |x - x'| + |y - y'| = 1\}.$$

First, the expansion is unnecessary if a patch has already been reconstructed there. Concretely, if an image cell $C_i(x', y') \in \mathbf{C}(p)$ contains a patch p', which is a *neighbor* of p, $C_i(x', y')$ is removed from the set $\mathbf{C}(p)$, where a pair of patches p and p' are defined to be neighbors if

$$|(\mathbf{c}(p) - \mathbf{c}(p')) \cdot \mathbf{n}(p)| + |(\mathbf{c}(p) - \mathbf{c}(p')) \cdot \mathbf{n}(p')| < 2\rho_1. \quad (8)$$

ρ_1 is the distance corresponding to an image displacement of β_1 pixels in the reference image $R(p)$ at the depth of the centers of $\mathbf{c}(p)$ and $\mathbf{c}(p')$. Second, even when no patch has been reconstructed, the expansion procedure is unnecessary for an image cell if there is a depth discontinuity viewed from the corresponding camera (see an example in Fig. 5).[7] Since it is, in practice, difficult to judge the presence of depth discontinuities before actually reconstructing a surface, we simply judge that the expansion is unnecessary due to a depth discontinuity if $Q_i^*(x', y')$ is not empty: If $C_i(x', y')$ already contains a patch whose photometric discrepancy score associated with I_i is better than the threshold α defined in (2).

6. Of course, this relatively simple procedure may not be perfect and may yield mistakes, but we also have a filtering step described in Section 3.3 to handle such errors.

7. This second selection criteria is for computational efficiency, and can be removed for simplicity because the filtering step can remove erroneous patches possibly caused by bad expansion procedure.

Input: Patches P from the feature matching step.
Output: Expanded set of reconstructed patches.

While P is not empty
  Pick and remove a patch p from P;
  For each image cell $C_i(x,y)$ containing p
    Collect a set $\mathbf{C}$ of image cells for expansion;
    For each cell $C_i(x',y')$ in $\mathbf{C}$
      // Create a new patch candidate p'
      $\mathbf{n}(p') \leftarrow \mathbf{n}(p), \quad R(p') \leftarrow R(p), \quad V(p') \leftarrow V^*(p');$
      Update $V^*(p')$; // Eq. (2)
      Refine $\mathbf{c}(p')$ and $\mathbf{n}(p')$; // (Sect.II-C)
      Add visible images (a depth-map test) to $V(p')$;
      Update $V^*(p')$; // Eq. (2)
      If $|V^*(p')| < \gamma$
        Go back to *For*-loop (failure);
      Add p' to P;
      Add p' to corresponding $Q_j(x,y)$ and $Q_j^*(x,y)$;

Fig. 6. Patch expansion algorithm. The expansion and the filtering procedure is iterated $n\ (= 3)$ times to make patches dense and remove outliers.

3.2.2 Expansion Procedure

For each collected image cell $C_i(x, y)$ in $\mathbf{C}(p)$, the following expansion procedure is performed to generate a new patch p': We first initialize $\mathbf{n}(p')$, $R(p')$, and $V(p')$ by the corresponding values of p. $\mathbf{c}(p')$ is, in turn, initialized as the point where the viewing ray passing through the center of $C_i(x, y)$ intersects the plane containing the patch p. After computing $V^*(p')$ from $V(p)$ by using (2), we refine $\mathbf{c}(p')$ and $\mathbf{n}(p')$ by the optimization procedure described in Section 2.3. During the optimization, $\mathbf{c}(p')$ is constrained to lie on a ray such that its image projection in I_i does not change in order to make sure that the patch always projects inside the image cell $C_i(x, y)$. After the optimization, we add to $V(p')$ a set of images in which the patch should be visible according to a depth-map test, where a depth value is computed for each image cell instead of a pixel, then update $V^*(p')$ according to (2). It is important to add visible images obtained from the depth-map test to $V(p')$ instead of replacing the whole set, because some matches (and thus the corresponding depth map information) may be incorrect at this point. Due to this update rule, the visibility information associated with reconstructed patches becomes inconsistent with each other, a fact that is used in the following filtering step to reject erroneous patches. Finally, if $|V^*(p')| \geq \gamma$, we accept the patch as a success and update $Q_i(x, y)$ and $Q_i^*(x, y)$ for its visible images. Note that, as in the initial feature matching step, α is set to 0.6 and 0.3, before and after the optimization, respectively, but we loosen (increase) both values by 0.2 after each expansion/filtering iteration in order to handle challenging (homogeneous or relatively texture-less) regions in the latter iterations. The overall algorithm description is given in Fig. 6. Note that when segmentation information is available, we simply ignore image cells in the background during initial feature matching and the expansion procedure, which guarantees that no patches are reconstructed in the background. The bounding volume information is *not* used to filter out erroneous patches in our experiments, although it would not be difficult to do so.

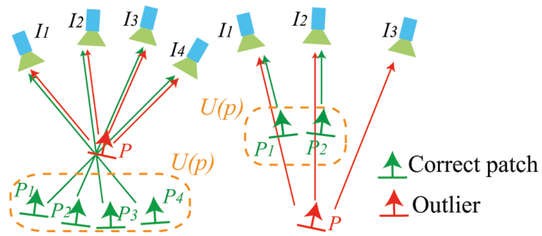


Fig. 7. The first filter enforces global visibility consistency to remove outliers (red patches). An arrow pointing from p_i to I_j represents a relationship $I_j \in V(p_i)$. In both cases (left and right), $U(p)$ denotes a set of patches that is inconsistent in visibility information with p.

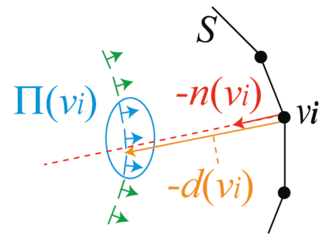


Fig. 8. In the iterative snapping algorithm, for each vertex $\mathbf{v}_i$ on the mesh model, we first collect $\pi\,(=10)$ patches $\Pi(\mathbf{v}_i)$ that are closest to the line defined by $\mathbf{v}_i$ and a surface normal $\mathbf{n}(\mathbf{v}_i)$ at the vertex. A (signed) distance $d(\mathbf{v}_i)$ from $\mathbf{v}_i$ to $\Pi(\mathbf{v}_i)$ is used to compute photometric discrepancy term. See text for details.

3.3 Filtering

The following three filters are used to remove erroneous patches. Our first filter relies on visibility consistency. Let $U(p)$ denote the set of patches p' that are inconsistent with the current visibility information—that is, p and p' are not neighbors (8), but are stored in the same cell of one of the images where p is visible (Fig. 7). Then, p is filtered out as an outlier if the following inequality holds

$$|V^*(p)|(1 - g^*(p)) < \sum_{p_i \in U(p)} 1 - g^*(p_i).$$

Intuitively, when p is an outlier, both $1 - g^*(p)$ and $|V^*(p)|$ are expected to be small, and p is likely to be removed. The second filter also enforces visibility consistency but more strictly: For each patch p, we compute the number of images in $V^*(p)$ where p is visible according to a depth-map test. If the number is less than γ, p is filtered out as an outlier. Finally, in the third filter, we enforce a weak form of regularization: For each patch p, we collect the patches lying in its own and adjacent cells in all images of $V(p)$. If the proportion of patches that are neighbors of p (8) in this set is lower than 0.25, p is removed as an outlier.

4 Polygonal Mesh Reconstruction

The reconstructed patches form an *oriented point*, or *surfel* model. Despite the growing popularity of this type of models in the computer graphics community [19], it remains desirable to turn our collection of patches into surface meshes for image-based modeling applications. In the following, we first propose two algorithms for initializing a polygonal mesh model from reconstructed patches, then a surface refinement algorithm, which polishes up a surface with explicit regularization constraints.

4.1 Mesh Initialization

4.1.1 Poisson Surface Reconstruction

Our first approach to mesh initialization is to simply use *Poisson Surface Reconstruction* (PSR) software [29] that directly converts a set of oriented points into a triangulated mesh model. The resolution of the mesh model is adaptive, and the size of a triangle depends on the density of the nearby oriented points: The denser the points are, the finer the triangles become. The PSR software outputs a closed mesh model even when patches are only reconstructed for a part of a scene. In order to remove extraneous portions of the mesh, we discard triangles whose average edge length is greater than six times the average edge length of the whole mesh since triangles are large where there are no patches.

4.1.2 Iterative Snapping

The PSR software produces high-quality meshes and is applicable to both object and scene data sets. However, it cannot make use of the foreground/background segmentation information associated with each input image that is often available for object data sets. Therefore, our second approach for mesh initialization is to compute a visual hull model from the segmentation information, which is then iteratively deformed toward reconstructed patches. Note that this algorithm is applicable only to object data sets with segmentation information. The iterative deformation algorithm is a variant of the approach presented in [12]. Concretely, the 3D coordinates of all the vertices in a mesh model are optimized by the gradient decent method while minimizing the sum of two per-vertex energy functions. The first function $E_s(\mathbf{v}_i)$ measures a geometric smoothness energy and is defined as

$$E_s(\mathbf{v}_i) = |-\zeta_1 \Delta \mathbf{v}_i + \zeta_2 \Delta^2 \mathbf{v}_i|^2 / \tau^2, \tag{9}$$

where Δ denotes the (discrete) Laplacian operator relative to a local parameterization of the tangent plane in $\mathbf{v}_i$, τ is the average edge length of the mesh model, and, with abuse of notation, $\mathbf{v}_i$ denotes the position of a vertex $\mathbf{v}_i$ ($\zeta_1 = 0.6$ and $\zeta_2 = 0.4$ are used in all our experiments). The second function $E_p(\mathbf{v}_i)$ enforces the consistency with the reconstructed patches (photometric discrepancy term) and is defined as

$$E_p(\mathbf{v}_i) = \max\left(-0.2, \min\left(0.2, \frac{d(\mathbf{v}_i) \cdot \mathbf{n}(\mathbf{v}_i)}{\tau}\right)\right)^2, \tag{10}$$

where $n(\mathbf{v}_i)$ is the outward unit normal of the surface at $\mathbf{v}_i$. $d(\mathbf{v}_i)$ is the signed distance between $\mathbf{v}_i$ and the reconstructed patches along $\mathbf{n}(\mathbf{v}_i)$, which is estimated as follows: For each patch p whose normal $n(p)$ is compatible with that of $\mathbf{v}_i$ (i.e., $\mathbf{n}(p) \cdot \mathbf{n}(\mathbf{v}_i) > 0$), we compute a distance between its center $c(p)$ and the line defined by $\mathbf{v}_i$ and $\mathbf{n}(\mathbf{v}_i)$, then collect the set $\Pi(\mathbf{v}_i)$ of $\pi = 10$ closest patches (see Fig. 8). Finally, $d(\mathbf{v}_i)$ is computed as the weighted average distance from $\mathbf{v}_i$ to the centers of the patches in $\Pi(\mathbf{v}_i)$ along $\mathbf{n}(\mathbf{v}_i)$:

$$d(\mathbf{v}_i) = \sum_{p \in \Pi(\mathbf{v}_i)} w(\mathbf{p})[\mathbf{n}(\mathbf{v}_i) \cdot (\mathbf{c}(p) - \mathbf{v}_i)],$$

where the weights $w(p)$ are Gaussian functions of the distance between $\mathbf{c}(p)$ and the line, with standard deviation ρ_1 defined as in Section 3.2.1, and normalized to sum to 1.[8] We iterate until convergence, while applying remeshing operations (edge split, contract, and swap [30]) to avoid self-intersections once every five gradient descent steps so that the edge lengths of the triangles on a mesh become approximately the same. After convergence, we increase the resolution of the mesh and repeat the process until the desired resolution is obtained, in particular, until image projections of edges of the mesh become approximately β_1 pixels in length.

4.2 Mesh Refinement

The mesh refinement is performed via an energy minimization approach similar to our iterative snapping procedure described in Section 4.1.2: The 3D coordinates of all of the vertices are optimized with respect to a sum of per-vertex photometric discrepancy and geometric smoothness energy functions. The smoothness function is the same as before (9). The photometric discrepancy energy is computed in the following two steps: 1) The depth and the orientation of a surface are estimated at each vertex independently for each pair of its visible images by using the patch optimization procedure; 2) the estimated depth and orientation information are combined to compute the energy function. More concretely, let $V(\mathbf{v}_i)$ denotes a set of images in which $\mathbf{v}_i$ is visible that is estimated from a standard depth-map test with a current mesh model. In the first step, for each pair (I_j, I_k) of images in $V(\mathbf{v}_i)$, we create a patch p on the tangent plane of the mesh at $\mathbf{v}_i$, namely, setting $\mathbf{c}(p) \leftarrow \mathbf{c}(\mathbf{v}_i)$ and $\mathbf{n}(p) \leftarrow \mathbf{n}(\mathbf{v}_i)$, then minimize the photometric discrepancy function $h(p, I_j, I_k)$ with respect to $\mathbf{c}(p)$ and $\mathbf{n}(p)$ as in Section 2.3.[9] Having obtained a set of patches $P(\mathbf{v}_i)$ after the patch optimization for pairs of images, the photometric discrepancy energy is computed as the sum of one minus (scaled) Gaussian function of the distance between each patch and the vertex:

$$E'_p(\mathbf{v}_i) = \zeta_3 \sum_{p \in P(\mathbf{v}_i)} 1 - \exp\left(-\left(\frac{d'(\mathbf{v}_i, p)}{\tau/4}\right)^2\right), \quad (11)$$
$$d'(\mathbf{v}_i, p) = n(p) \cdot (c(p) - \mathbf{v}_i).$$

$d'(\mathbf{v}_i, p)$ is the (signed) distance between the patch p and the vertex $\mathbf{v}_i$ along the patch normal, τ is the average edge length of the mesh, and ζ_3 is the linear combination weight. Note that we borrow the idea of occlusion robust photo-consistency proposed in [31], and first obtain multiple estimates of the depth and the orientation from pairs of visible images, instead of using all of the visible images at once to obtain a single estimate. Then, in the second step, multiple estimates are combined with Gaussian functions that are robust to outliers. Also note that the patches $P(\mathbf{v}_i)$ are computed only once at the beginning as preprocessing for each vertex, while the photometric discrepancy energy (11) is evaluated many times in the energy minimization procedure performed by a conjugate gradient method [28]. Fig. 9 illustrates how this refined photometric discrepancy energy handles outliers, or "bad" images and patches robustly and avoids false local minima. Although the fundamental idea has not changed from [31], there are a couple of differences worth mentioning. First, in addition to a depth value, a surface normal is incorporated in our framework, both in the patch optimization step and in the final formula (11). Second, we use a Gaussian (kernel) function to combine multiple hypothesis (patches) (11), while a box function is chosen in [31] with discretized voxels, which ends up simply casting votes to voxels.

8. $E_p(\mathbf{v}_i)$ has a form of the Huber function so that the magnitude of its derivative does not become too large in each gradient descent step to ensure stable deformation and avoid mesh self-intersections.

9. During the optimization, the patch center $\mathbf{c}(p)$ is constrained to lie on a ray passing through $\mathbf{v}_i$ in parallel to $\mathbf{n}(\mathbf{v}_i)$.

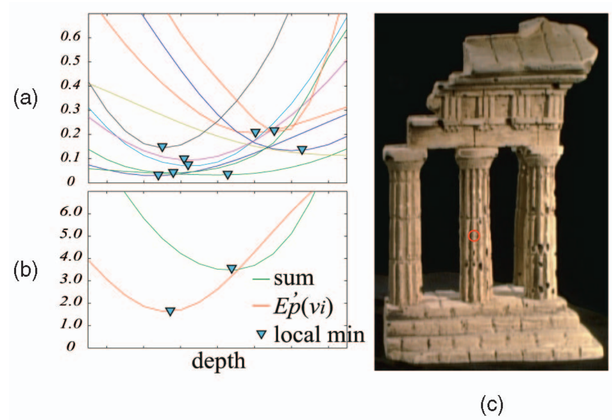


Fig. 9. (a) Pairwise photometric discrepancy scores $h(p, I_i, I_j)$ at a vertex of *temple* data set. For better illustration, among three degrees of freedom in the optimization (a depth and a normal), the discrepancy scores are plotted for different values of depths with a fixed normal. A triangle on each plot illustrates the location of a local minimum that is a depth value obtained from the patch optimization procedure for a pair of images. (b) The sum of all of the pairwise discrepancy scores giving an inaccurate local minimum location and our proposed photometric discrepancy energy (11). (c) An input image of *temple* with a red circle illustrating the location of the vertex.

5 Experiments and Discussion

5.1 Data Sets

Fig. 10 shows sample input images of all of the data sets used in our experiments. Table 1 lists the number of input images, their approximate size, the corresponding choice of parameters, the algorithm used to initialize a mesh model (either PSR software [29] or iterative snapping after visual hull construction, denoted as VH), and whether images contain obstacles (crowded scenes) or not. Note that all of the parameters except for μ, γ, and ζ_3 have been fixed in our experiments. The *roman* and *skull* data sets have been acquired in our lab, while other data sets have been kindly provided by S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski (*temple* and *dino*, see also [2]), S. Sullivan and Industrial Light and Magic (*face*, *face-2*, *body*, *steps-1*, and *wall*), and C. Strecha (*fountain*, *city-hall*, *brussels*, and *castle*). The *steps-2* data set has been artificially generated by manually painting a red cartoonish human in each image of *steps-1* images. To further test the robustness of our algorithm against outliers, the *steps-3* data set has been created from *steps-2* by replacing the fifth image with the third, without changing camera parameters. This is a particularly challenging example since the entire fifth image must be detected as an outlier.

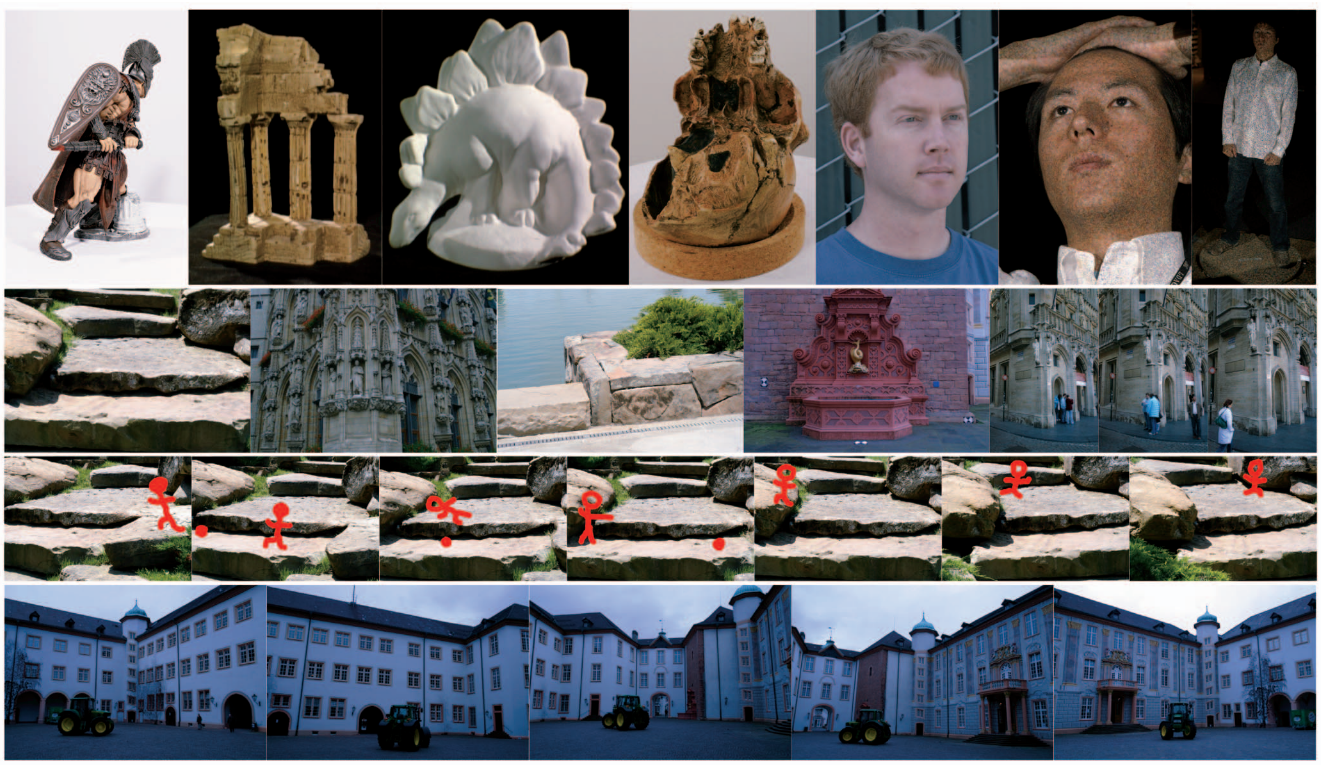


Fig. 10. Sample input images of all of the data sets used in our experiments. The top row shows the object data sets. From left to right: *roman*, *temple*, *dino*, *skull*, *face-1*, *face-2*, and *body*. The bottom three rows show the scenes and the crowded scenes data sets. From left to right and top to bottom: *steps-1*, *city-hall*, *wall*, *fountain*, *brussels*, *steps-{2,3}*, and *castle*. Multiple images are shown for *brussels*, *steps-{2,3}*, and *castle*.

5.2 Reconstructed Patches and Mesh Models

Reconstructed patches, texture-mapped using the reference image, are shown in Fig. 11. As illustrated by the figure, patches are densely covering reconstructed object and scene surfaces. *city-hall* is an interesting example because viewpoints change significantly across input cameras, and frontal statues are visible in some images in close-ups. Reconstructed patches automatically become denser for such places because the resolution of patches is controlled by that of input images (we try to reconstruct at least one patch in every image cell). The *wall* data set is challenging since a large portion of several of the input pictures consists of running water. Nonetheless, we have successfully detected and ignored the corresponding image regions as outliers. Obstacles such as pedestrians in *brussels* or cartoonish humans in *steps-{2,3}* do not show up in the texture mapped patches because our patch generation algorithm guarantees that they do not appear in reference images. Fig. 12 shows patches obtained from the initial feature matching step that are sparse, noisy, and erroneous. Fig. 13, in turn, shows patches that are removed in each of the three filtering steps. As illustrated by the figure, our filtering procedure is aggressive and removes a lot of patches possibly containing true-positives, but this is not a problem since the expansion and the filtering steps are iterated a couple of times in our algorithm. The number of the reconstructed patches at each step of the algorithm is given in Fig. 14.

TABLE 1
Characteristics of the Data Sets Used in Our Experiments

Name	Images	Image Size	μ	γ	ζ_3	Init	Crowded
roman	48	1800×1200	5	3	1	VH	
temple	15	640×480	5	2	1	VH	
dino	16	640×480	7	2	4	VH	
skull	24	1000×1000	5	3	1	VH	
face-1	13	1500×1400	5	3	1	VH	
face-2	4	1400×2200	5	2	1	PSR	
body	4	1400×2200	5	2	1	PSR	
steps-1	7	1500×1400	7	2	4	PSR	
steps-2	7	1500×1400	7	2	4	PSR	x
steps-3	7	1500×1400	7	2	4	PSR	x
city-hall	7	1500×1000	5	3	1	PSR	
wall	9	1500×1400	5	3	1	PSR	
fountain	11	1536×1024	7	3	1	PSR	
brussels	3	2000×1300	7	2	4	PSR	x
castle	19	1536×1024	7	3	1	PSR	

A visual hull model is used to initialize a mesh model before the iterative snapping procedure for all object data sets except *face-2* and *body* where viewpoints are limited and PSR software is used instead. The visual hull model is computed by using the EPVH software by Franco and Boyer [32] except for the *dino* data set, where an object is not fully visible in some images and a standard voxel-based visual hull algorithm is used instead (see Fig. 15). Mesh models before the refinement—that is, models obtained either by the visual hull construction followed by the iterative snapping (Section 4.1.2) or PSR software (Section 4.1.1) are shown for some data sets in the top row of Fig. 17.

Mesh models after the refinement step are shown in Fig. 16 for all of the data sets. Our algorithm has successfully reconstructed various surface structures such as the high-curvature and/or shallow surface details of *roman*, the thin

Fig. 11. Reconstructed patches. In the second last row, patches are shown for *steps-1*, *steps-2*, and *steps-3* from left to right. See text for the details.

cheekbone and deep eye sockets of *skull*, and the intricate facial features of *face-1* and *face-2*. The *castle* is a very interesting data set in that cameras are surrounded by buildings and its structure is "inside-out" compared to typical object data sets. Nonetheless, our algorithm has been directly used without any modifications to recover its overall structure. Finally, the figure illustrates that our algorithm has successfully handled obstacles in *crowded scene* data sets. The background building is reconstructed for the *brussels* data set, despite people occluding various parts of the scene. Reconstructed models of *steps-2* and *steps-3* do not look much different from that of *steps-1* despite many obstacles. As mentioned before, *steps-3* is a challenging example with a significant amount of outliers and some of the details are missing in the reconstructed model (also see Fig. 11), but this is simply because the corresponding surface regions are not visible in enough number of images. Close-ups of some of the reconstructions are shown in Fig. 17 before and after the mesh refinement step, qualitatively illustrating that the refinement step removes high frequency noise while retaining sharp structure.

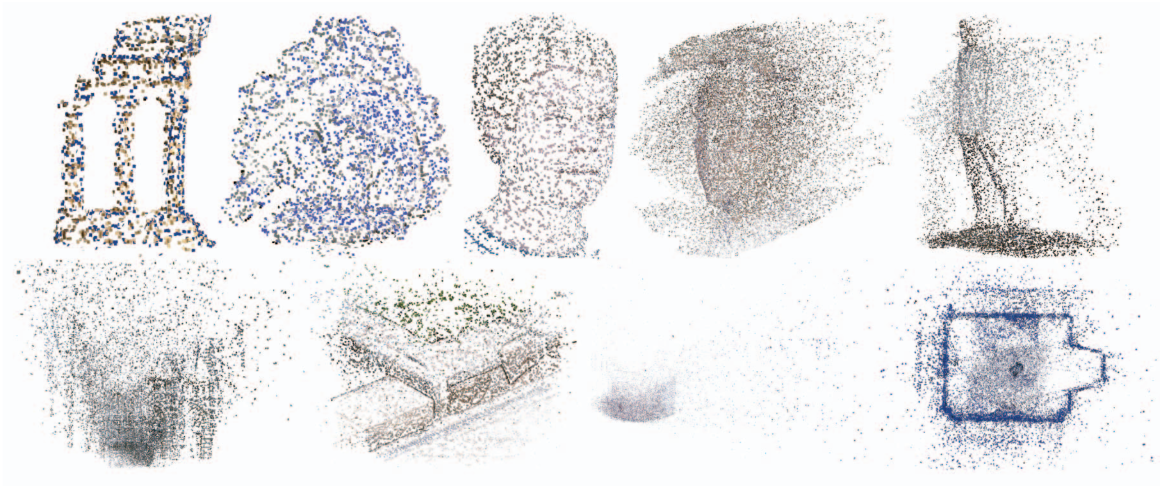


Fig. 12. Reconstructed patches after the initial feature matching step. Patches are sparse, noisy, and erroneous before the expansion and the filtering.

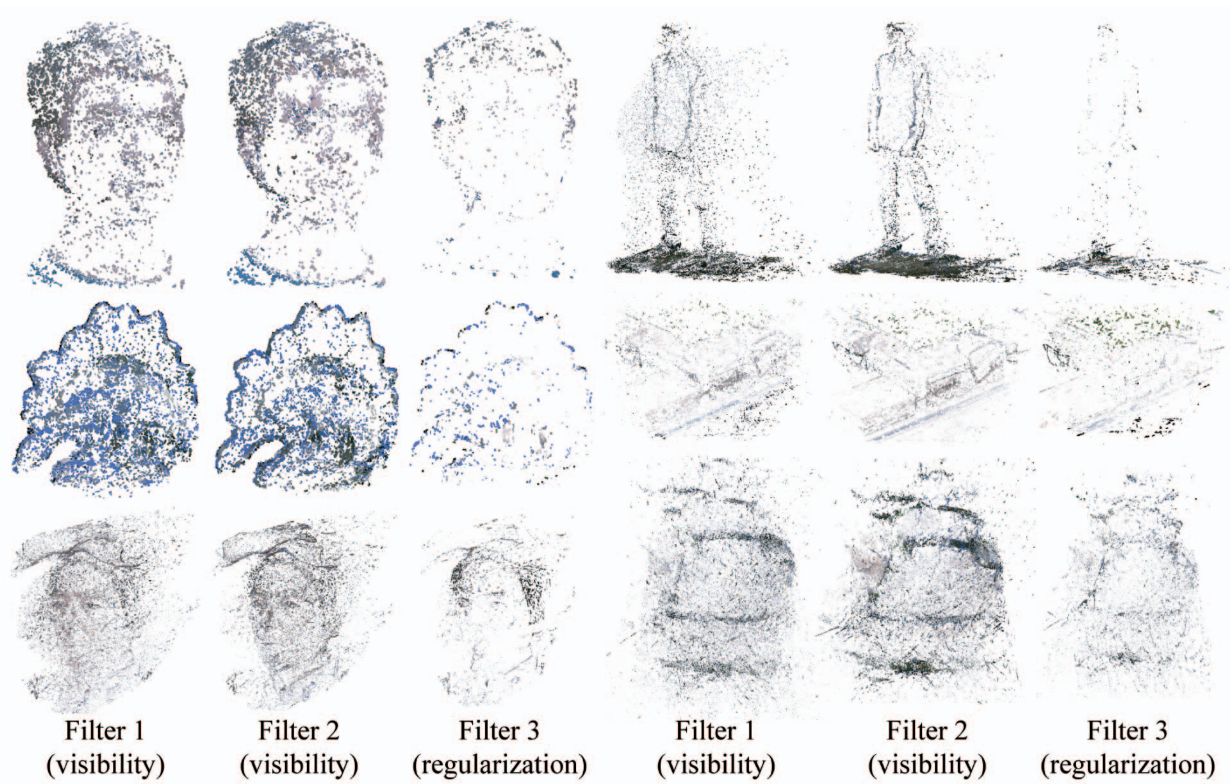
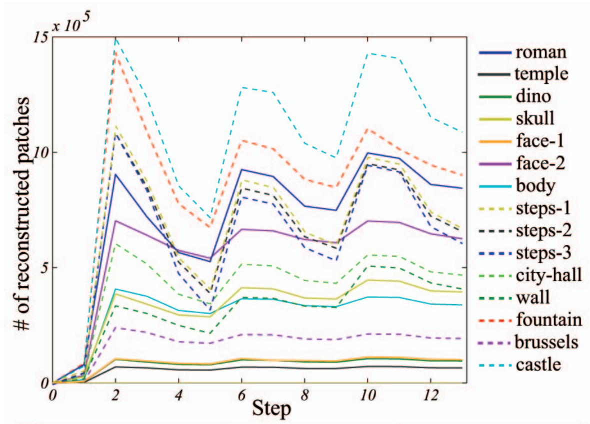


Fig. 13. Three filters are used to remove erroneous patches (Section 3.3). The first two filters enforce global visibility consistency and the last filter enforces weak regularization. Patches detected as outliers by each of the three filters are shown for the first iteration of our algorithm. Note that more patches are added during the subsequent iterations, leading to the results of Fig. 11.

5.3 Evaluations

Quantitative evaluations of state-of-the-art MVS algorithms are presented in [2] in terms of *accuracy* (distance d such that a given percentage of the reconstruction is within d from the ground truth model) and *completeness* (percentage of the ground truth model that is within a given distance from the reconstruction). The data sets consist of two objects (*temple* and *dino*), each of which, in turn, consists of three data sets (*sparse ring*, *ring*, and *full*) with different numbers of input images, ranging from 15 to more than 300. Note that the *sparse ring temple* and *sparse ring dino* data sets have been used in our experiments so far. Table 2 lists the evaluation results with other top performers in the main table provided at [2], and shows that our approach outperforms all of the other evaluated techniques in terms of both *accuracy* and *completeness* for four out of the six data sets (the intermediate *temple* and all the three *dino* data sets). Our approach also has the best *completeness* score for the *sparse ring temple* data set.[10] We believe that one reason why our results are among the best for these data sets is that we take into account surface orientation properly in computing

10. Rendered views of the reconstructions and all of the quantitative evaluations can be found at [2].



Step		
0: start		
1: after initial matching		
2: after expansion	6: after expansion	10: after expansion
3: after first filter	7: after first filter	11: after first filter
4: after second filter	8: after second filter	12: after second filter
5: after third filter	9: after third filter	13: after third filter
(first iteration)	(second iteration)	(third iteration)

Fig. 14. The graph shows the number of reconstructed patches at each step of the algorithm: after the initial feature matching, the expansion procedure and each of the three filtering steps. Note that three iterations of the expansion and the filtering have been performed.

photometric consistency, which is important when structures do not have salient textures or images are sparse and perspective distortion effects are not negligible.

Strecha et al. provide quantitative evaluations for two scene data sets, *fountain* and *herzjesu* [35], [36] (see Table 3).[11] A measure similar to *completeness* in [2] is used in their evaluation. More concretely, each entry in the table shows the percentage of the laser-scanned model that is within d distance from the corresponding reconstruction. σ is the standard deviation of depth estimates of the laser range scanner used in their experiments. For each column, the best and the second best completeness scores are highlighted in red and green, respectively. Note that the *herzjesu* data set is used only in Table 3 in this paper, and qualitative results (e.g., renderings of our reconstructed mesh model) are available in [36]. As the table shows, our method outperforms the others, especially for *herzjesu*. It is also worth mentioning that, as shown in Figs. 11 and 16, our method has been able to recover a building in the background for *fountain* that is partially visible in only a few frames, while none of the other approaches have been able to recover such structure, probably due to the use of the provided bounding box information excluding the building. Note that our algorithm does not require a bounding box or a visual hull model, valid depth ranges, etc., and simply tries to reconstruct whatever is visible in the input images, which is one of its main advantages.

Table 4 lists the running time of the proposed algorithms and numbers of triangles in the final mesh models. A standard PC with Dual Xeon 2.66 GHz is used for the experiments. The patch generation algorithm is very efficient, in particular, takes only a few minutes for *temple* and *dino*, in comparison to most other state-of-the-art techniques evaluated at [2] that take more than half an hour. It is primarily because the algorithm does not involve any large optimization (only three degrees of freedom for the patch optimization), and patches are essentially recovered independently. On the other hand, the iterative snapping algorithm is very slow, which is in part due to the iteration of the mesh deformation and the remeshing operations that are frequently applied to prevent self-intersections. The advantage of using the iterative snapping algorithm over PSR software is only that silhouette consistency can be enforced in initializing a mesh model. The mesh refinement step is also slow due to the energy minimization procedure by a conjugate gradient method, where the number of unknowns is three times the number of vertices in a high-resolution mesh model, which scales up to even more than 10 million for some data sets. To demonstrate that PSR software can be used instead to produce similar final results while saving a lot of computation time, for the five object data sets where the iterative snapping has been originally used we have run our algorithm again, but this time, with PSR software for the mesh initialization. Fig. 18 shows the reconstructed models and illustrates that noticeable differences (highlighted in red) appear only at a few places compared to results in Fig. 16, and the rest of the structure is almost identical.

11. In addition to multiview stereo, they also benchmark camera calibration algorithms. See their Web site [36] for more details.

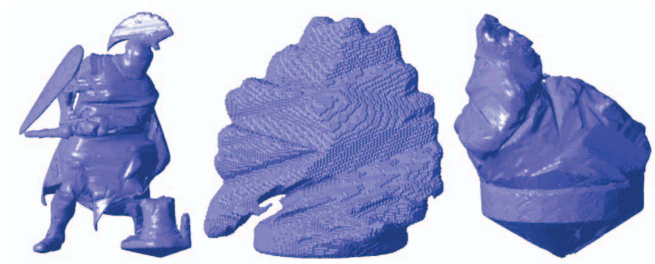


Fig. 15. Visual hull models for some object data sets where foreground/background segmentation information is available. See text for more details.

5.4 Shortcomings and Limitations

As mentioned earlier, one difference between the proposed method and many other MVS algorithms is that our method lacks strong regularization at the core patch reconstruction step, which helps in recovering complicated structure such as deep concavities but may cause a problem where image information is unreliable due to poor-texture surfaces or sparse input images. However, our experimental results, in particular quantitative evaluations on the Middlebury benchmark, show that these are in fact data sets where our approach significantly outperforms the others due to the surface normal estimation at the sacrifice of additional computation time. On the other end, because of the lack of regularization, our patch generation step reconstructs 3D points only where there is reliable texture information, and postprocessing is necessary to fill-in possible holes and obtain, for example, a complete mesh model. Another limitation is that, like many other MVS algorithms, our surface representation (i.e., a set of oriented points) and the reconstruction procedure do not work well for narrow-baseline cases, where the uncertainty of depth estimation is

Fig. 16. Final mesh models: From left to right and top to bottom: *roman*, *temple*, *dino*, *skull*, *face-1*, *face-2*, *body*, *city-hall*, *wall*, *fountain*, *brussels*, *steps-1*, *steps-2*, *steps-3*, and *castle* data sets. Note that the mesh models are rendered from multiple view points for *fountain* and *castle* data sets to show their overall structure.

high and disparities instead of depth values are typically estimated per image [38].

6 Conclusion and Future Work

We have proposed a novel algorithm for calibrated multiview stereo that outputs a dense set of patches covering the surface of an object or a scene observed by multiple calibrated photographs. The algorithm starts by detecting features in each image, matches them across multiple images to form an initial set of patches, and uses an expansion procedure to obtain a denser set of patches before using visibility constraints to filter away false matches. After converting the resulting patch model into a mesh appropriate for image-based modeling, an optional refinement algorithm can be used to refine the mesh, and achieve even higher accuracy. Our approach can handle a variety of data sets and allows outliers or obstacles in the images. Furthermore, it does not

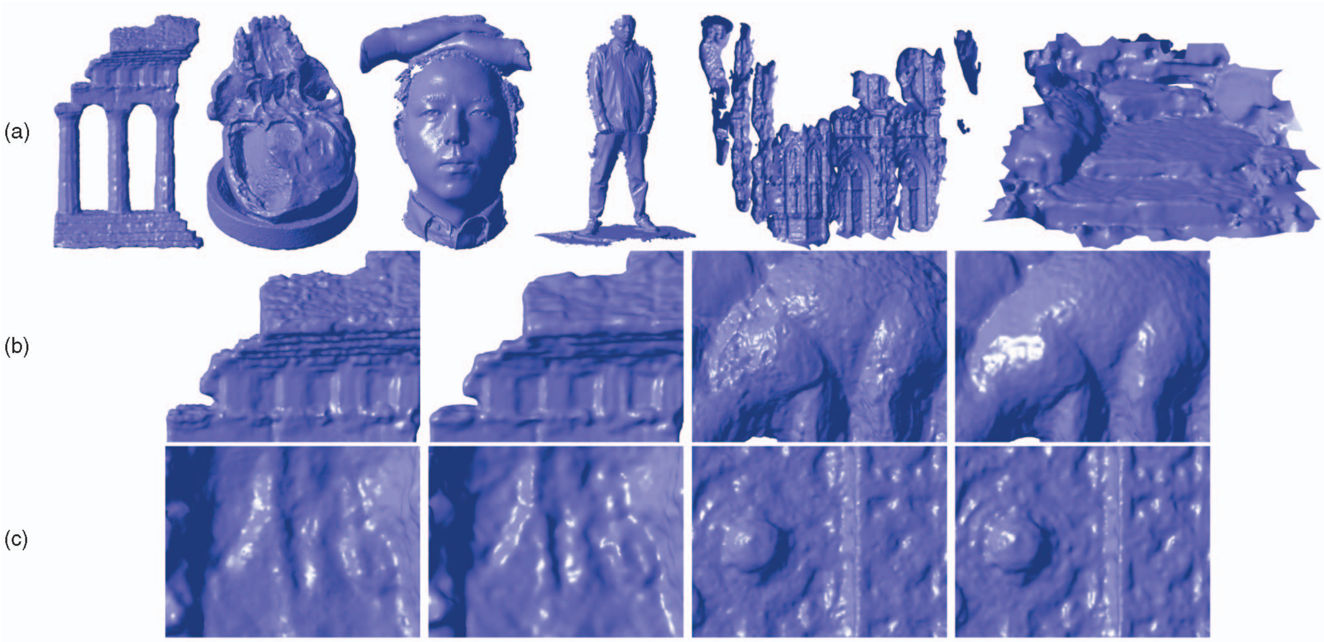


Fig. 17. (a) Mesh models before the refinement. (b) and (c) Comparisons of reconstructed models before (left) and after (right) the refinement.

TABLE 2
Quantitative Evaluations Provided at [2]

	temple						*dino*					
	Full		Ring		SparseRing		Full		Ring		SparseRing	
Proposed approach	0.49	99.6	0.47	99.6	0.63	99.3	0.33	99.8	0.28	99.8	0.37	99.2
Hernandez [10]	0.36	99.7	0.52	99.5	0.75	95.3	0.49	99.6	0.45	97.9	0.60	98.5
Vogiatzis [7]	1.07	90.7	0.76	96.2	2.77	79.4	0.42	99.0	0.49	96.7	1.18	90.8
Pons [5]			0.60	99.5	0.90	95.4			0.55	99.0	0.71	97.7
Bradley [16]			0.57	98.1	0.48	93.7			0.39	97.6	0.38	94.7
Zach [33]	0.51	98.8	0.56	99.0			0.55	98.7	0.51	99.1		
Vogiatzis [31]	0.50	98.4	0.64	99.2	0.69	96.9						
Campbell [34]	0.41	99.9	0.48	99.4	0.53	98.6						

For each data set and each algorithm, the table shows the accuracy (left) and the completeness (right) measures. The best result is highlighted in red. Tables are reproduced from [2] (courtesy of D. Scharstein).

require any assumption on the topology of an object or a scene and does not need any initialization, such as a visual hull model, a bounding box, or valid depth ranges that are required in most other competing approaches, but can take advantage of such information when available. Our approach takes into account surface orientation in photometric

TABLE 3
Quantitative Evaluations for Two Scene Data Sets

Completeness measures for the *fountain* dataset							
Threshold (*d*)	σ	2σ	3σ	4σ	5σ	6σ	7σ
Proposed	14.8	41.1	58.0	66.9	71.7	74.6	76.5
Strecha [15]	5.4	16.3	28.2	41.3	53.4	62.5	68.8
Strecha [37]	14.0	38.4	56.2	67.4	73.8	77.5	79.7
Zaharescu [11]	14.6	38.8	55.5	65.1	70.4	73.7	75.9
Completeness measures for the *herzjesu* dataset							
Threshold (d)	σ	2σ	3σ	4σ	5σ	6σ	7σ
Proposed	16.6	43.9	58.9	66.7	71.3	74.3	76.5
Strecha [15]	5.3	15.6	24.7	32.8	40.0	46.2	51.3
Strecha [37]	10.4	29.7	44.2	54.2	61.0	65.8	69.2
Zaharescu [11]	13.2	34.7	46.5	52.6	56.2	58.6	60.5

Tables are reproduced from [35] (courtesy of C. Strecha).

TABLE 4
Running Time of the Three Algorithms in Our Approach [min], and Numbers of Triangles in the Final Mesh Models

	Patch Generation	Mesh Init Snap	Mesh Init PSR	Mesh Refinement	Num of Triangles
roman	20	149		91	1549862
temple	3	89		49	192200
dino	2	97		69	524880
skull	6	120		46	2418140
face-1	9	123		56	2613318
face-2	4		3	36	2345849
body	2		2	38	1319878
steps-1	15		1	58	1416492
steps-2	15		1	65	1392204
steps-3	15		1	64	1391675
city-hall	16		3	26	2507103
wall	8		2	44	1830535
fountain	22		2	121	3522051
brussels	18		1	23	1079591
castle	36		3	278	1339905

Fig. 18. If segmentation information is not available or users choose not to extract such information from images, a visual hull model cannot be reconstructed, and hence, our iterative snapping algorithm, described in Section 4.1.2, is not applicable. However, PSR software [29] can still be used to produce similar results with only a few noticeable differences (illustrated with red circles), while saving a lot of computation time. Compare them to Fig. 16.

consistency computation, while most other approaches just assume fronto-parallel surfaces. Quantitative evaluations provided in [1] show that the proposed approach outperforms all of the other evaluated techniques both in terms of *accuracy* and *completeness* for four out of the six data sets. The implementation of the patch-based MVS algorithm (PMVS) is publicly available in [25]. Our future work will be aimed at better understanding the source of reconstruction errors and obtain even higher accuracy. For example, one interesting observation from Table 2 is that our results for the largest *full* data sets are worse than those for the intermediate *ring* data sets, which is in fact the case for some other algorithms. Further investigation of this behavior is part of our future work, together with the analysis of how parameter values influence results. It would also be interesting to study contributions of our mesh refinement step more quantitatively, which has mostly shown qualitative improvements in our results. Another possible extension is to model lighting and surface reflectance properties since MVS algorithms like ours typically assume that an object or a scene is *Lambertian* under constant illumination, which is of course not true in practice. Improving the mesh initialization algorithm by using a technique similar to [39] is also part of our future work.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant IIS-0535152, the INRIA associated team Thetys, and the Agence Nationale de la Recherche under grants Hfibmr and Triangles. The authors thank S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski for the *temple* and *dino* data sets and evaluations, S. Sullivan, A. Suter, and Industrial Light and Magic for *face*, *face-2*, *body*, *steps-1*, and *wall*, C. Strecha for *city-hall* and *brussels*, and finally J. Blumenfeld and S.R. Leigh for the *skull* data set.

References

[1] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "Multi-View Stereo Evaluation," http://vision.middlebury.edu/mview/, 2010.

[2] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[3] O. Faugeras and R. Keriven, "Variational Principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem," *IEEE Trans. Image Processing,* vol. 7, no. 3, pp. 336-344, Mar. 1998.

[4] S. Paris, F. Sillion, and L. Quan, "A Surface Reconstruction Method Using Global Graph Cut Optimization," *Proc. Asian Conf. Computer Vision,* http://artis.imag.fr/Publications/2004/PSQ04, Jan. 2004.

[5] J.-P. Pons, R. Keriven, and O.D. Faugeras, "Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score," *Int'l J. Computer Vision,* vol. 72, no. 2, pp. 179-193, 2007.

[6] S. Tran and L. Davis, "3D Surface Reconstruction Using Graph Cuts with Surface Constraints," *Proc. European Conf. Computer Vision,* 2006.

[7] G. Vogiatzis, P.H. Torr, and R. Cipolla, "Multi-View Stereo Via Volumetric Graph-Cuts," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

[8] A. Hornung and L. Kobbelt, "Hierarchical Volumetric Multi-View Stereo Reconstruction of Manifold Surfaces Based on Dual Graph Embedding," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2006.

[9] S. Sinha, P. Mordohai, and M. Pollefeys, "Multi-View Stereo via Graph Cuts on the Dual of an Adaptive Tetrahedral Mesh," *Proc. Int'l Conf. Computer Vision,* 2007.

[10] C. Hernández Esteban and F. Schmitt, "Silhouette and Stereo Fusion for 3D Object Modeling," *Computer Vision and Image Understanding,* vol. 96, no. 3, pp. 367-392, 2004.

[11] A. Zaharescu, E. Boyer, and R. Horaud, "Transformesh: A Topology-Adaptive Mesh-Based Approach to Surface Evolution," *Proc. Asian Conf. Computer Vision,* Y. Yagi, S.B. Kang, I.-S. Kweon, and H. Zha, eds., pp. 166-175, 2007.

[12] Y. Furukawa and J. Ponce, "Carved Visual Hulls for Image-Based Modeling," *Int'l J. Computer Vision,* vol. 81, pp. 53-67, Mar. 2009.

[13] B. Baumgart, "Geometric Modeling for Computer Vision," PhD dissertation, Stanford Univ., 1974.

[14] M. Goesele, B. Curless, and S.M. Seitz, "Multi-View Stereo Revisited," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2402-2409, 2006.

[15] C. Strecha, R. Fransens, and L.V. Gool, "Combined Depth and Outlier Estimation in Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2394-2401, 2006.

[16] D. Bradley, T. Boubekeur, and W. Heidrich, "Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[17] M. Lhuillier and L. Quan, "A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 3, pp. 418-433, Mar. 2005.

[18] M. Habbecke and L. Kobbelt, "Iterative Multi-View Plane Fitting," *Proc. 11th Fall Workshop Vision, Modeling, and Visualization,* 2006.

[19] L. Kobbelt and M. Botsch, "A Survey of Point-Based Techniques in Computer Graphics," *Computers & Graphics,* vol. 28, no. 6, pp. 801-814, 2004.

[20] R.L. Carceroni and K.N. Kutulakos, "Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape and Reflectance," *Int'l J. Computer Vision,* vol. 49, nos. 2-3, pp. 175-214, 2002.

[21] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz, "Multi-View Stereo for Community Photo Collections," *Proc. Int'l Conf. Computer Vision,* 2007.

[22] G.P. Otto and T.K.W. Chau, "'Region-Growing' Algorithm for Matching of Terrain Images," *Image Vision Computing,* vol. 7, no. 2, pp. 83-94, 1989.

[23] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Simultaneous Object Recognition and Segmentation by Image Exploration," *Proc. European Conf. Computer Vision,* 2004.

[24] A. Kushal and J. Ponce, "A Novel Approach to Modeling 3D Objects from Stereo Views and Recognizing them in Photographs," *Proc. European Conf. Computer Vision,* vol. 2, pp. 563-574, 2006.

[25] Y. Furukawa and J. Ponce, "PMVS," http://grail.cs.washington.edu/software/pmvs, 2010.

[26] Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multi-View Stereopsis," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[27] E. Tola, V. Lepetit, and P. Fua, "A Fast Local Descriptor for Dense Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[28] W. Naylor and B. Chapman, "Wnlib," http://www.willnaylor.com/wnlib.html, 2010.

[29] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson Surface Reconstruction," *Proc. Symp. Geometry Processing,* 2006.

[30] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," *Proc. SIGGRAPH,* pp. 19-26, 1993.

[31] G. Vogiatzis, C.H. Esteban, P.H.S. Torr, and R. Cipolla, "Multi-view Stereo Via Volumetric Graph-Cuts and Occlusion Robust Photo-Consistency," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 12, pp. 2241-2246, Dec. 2007.

[32] J.-S. Franco and E. Boyer, "Efficient Polyhedral Modeling from Silhouettes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 3, pp. 414-427, Mar. 2009.

[33] C. Zach, "Fast and High Quality Fusion of Depth Maps," *Proc. 3D Data Processing, Visualization and Transmission,* 2008.

[34] N. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla, "Multiple Hypotheses Depth-Maps for Multi-View Stereo," *Proc. European Conf. Computer Vision,* 2008.

[35] C. Strecha, W. von Hansen, L.V. Gool, P. Fua, and U. Thoennessen, "On Benchmarking Camera Calibration and Multi-View Stereo for High Resolution Imagery," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2008.

[36] C. Strecha, "Multi-View Evaluation," http://cvlab.epfl.ch/~strecha/multiview, 2010.

[37] C. Strecha, R. Fransens, and L.V. Gool, "Wide Baseline Stereo from Multiple Views: A Probabilistic Account," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2004.

[38] D. Scharstein and R. Szeliski, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *Int'l J. Computer Vision,* vol. 47, nos. 1-3, pp. 7-42, 2002.

[39] C.H. Esteban, G. Vogiatzis, and R. Cipolla, "Probabilistic Visibility for Multi-View Stereo," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

Yasutaka Furukawa received the BS degree in computer science from the University of Tokyo in 2001, and the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 2008. He then joined the Graphics and Imaging Laboratory at the University of Washington as a postdoctoral research associate. He joined Google Inc. in 2010. His research interests include computer vision and computer graphics focusing on 3D photography.

Jean Ponce is a professor at the Ecole Normale Supérieure (ENS) in Paris, France, where he leads a joint ENS/INRIA/CNRS research team, WILLOW, that focuses on computer vision and machine learning. Prior to this, he served for more than 15 years on the faculty of the Department of Computer Science and the Beckman Institute at the University of Illinois at Urbana-Champaign. He is the author of more than 150 technical publications, including the textbook *Computer Vision: A Modern Approach*, in collaboration with David Forsyth. He is a member of the editorial boards of *Foundations and Trends in Computer Graphics and Vision*, the *International Journal of Computer Vision*, and the *SIAM Journal on Imaging Sciences*. He was also editor-in-chief of the *International Journal on Computer Vision* (2003-2008), an associate editor of the *IEEE Transactions on Robotics and Automation* (1996-2001), and an area editor of *Computer Vision and Image Understanding* (1994-2000). He was program chair of the 1997 IEEE Conference on Computer Vision and Pattern Recognition and served as general chair of the year 2000 edition of this conference. In 2003, he was named an IEEE fellow for his contributions to computer vision, and he received a US patent for the development of a robotic parts feeder. In 2008, he served as general chair for the European Conference on Computer Vision.