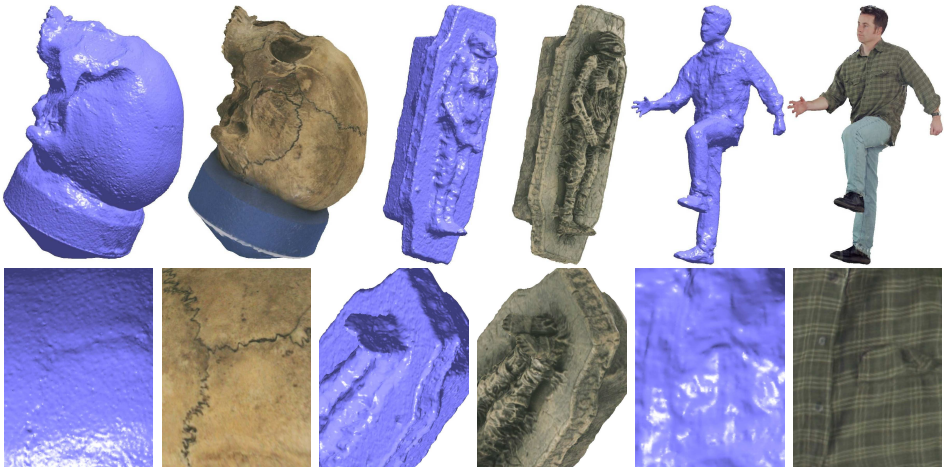

FIGURE 19.12: Shaded and texture-mapped renderings of carved visual hulls, including some close-ups. *Reprinted from "Carved Visual Hulls for Image-Based Modeling," by Y. Furukawa and J. Ponce, International Journal of Computer Vision, 81(1):53–67, (2009a). © 2009 Springer.*

Figure 19.12 shows some more 3D models obtained using this method. Note that some of the surface details are not recovered accurately. In some cases, this is simply due to the fact that the surface is not visible from any cameras; see for example the bottom part of the skull stand. In other cases, missing details correspond to failure modes of the algorithm: for example, the eye sockets of the skull are simply too deep to be carved away by graph cuts or local refinement. The person is a particularly challenging example, because of the extremely complicated folds of the cloth, and its high-frequency stripe patterns. Nonetheless, the algorithm performs rather well in general, correctly recovering minute details such as the fin undulations for the toy dinosaur, with corresponding height variations well below 1mm, or the bone junctions for the skull.

19.2 PATCH-BASED MULTI-VIEW STEREOPSIS

The approach to image-based modeling and rendering presented in the previous section is effective, but best suited for controlled situations where image silhouettes can be delineated accurately, through background subtraction for instance. For more general settings—for example, when using hand-held cameras in outdoor environments—it is tempting to revisit the stereopsis techniques presented in Chapter 7 in a context where thousands of images taken from very different viewpoints may be available. Two key ingredients of several of the techniques presented in that chapter are how they compare image brightness or color patterns in the neighborhood of potential matches, and how they enforce spatial consistency among pairs of these correspondences. As shown in Chapter 7, these are easily generalized to multiple images for *narrow-baseline* scenarios, where the cameras are close to each other, and can be assumed to share the same neighborhood structure—that is, if

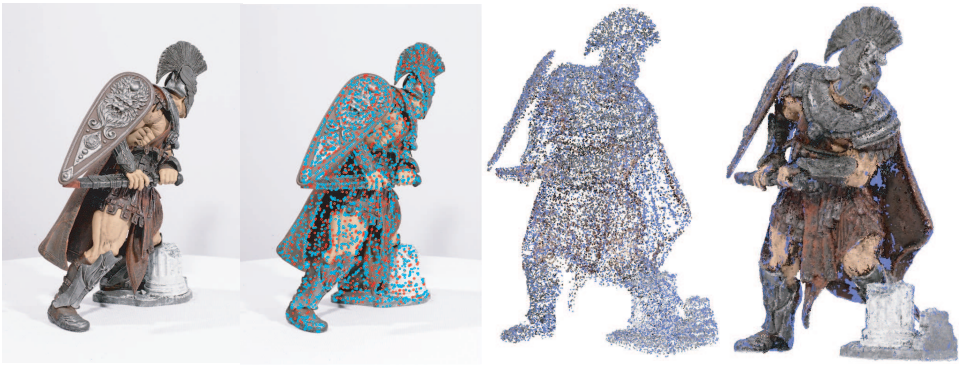

FIGURE 19.13: The PMVS approach to image-based modeling and rendering, illustrated using 48 $1,800 \times 1,200$ images of a Roman soldier action figure. From **left** to **right**: A sample input image; detected features; reconstructed patches after the initial matching; and final patches after expansion and filtering. *Reprinted from "Accurate, Dense, and Robust Multi-View Stereopsis," by Y. Furukawa and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8):1362–1376, (2010).* © *2010 IEEE.*

pixels are adjacent in some reference picture, so are their matches in the others. In the context of image-based modeling and rendering, the observed scene can in this case be reconstructed as a depth map, where the grid structure (or some triangulation) of the reference image provides a mesh whose vertices have coordinates in the form $(x, y, z(x, y))$, then can be rendered with classical computer graphics technology.

In the *wide-baseline* case, cameras may be positioned anywhere—all around an object, for example, or perhaps scattered over a large area. This case is much more challenging. Each image encodes part of the scene connectivity, but hides some of it as well, due in part to occlusion phenomena. Although various heuristics for stitching partial reconstructions obtained from a few views into a single mesh structure are available (see Chapter 14 for the case of range data), optimizing both the correspondences and the global mesh structure of the reconstructed points today remains, as far as we know, an open problem. This section thus abandons a full mesh model of the reconstructed scene in favor of small patches tangent to the surface, using the image topology as a proxy for their connectivity. This information is not used for rendering purposes, but instead to enforce spatial consistency and handle the visibility constraints (is some patch visible in the input images given other patch hypotheses?) that are crucial in wide-baseline stereopsis.

This technique, dubbed *PMVS* for *Patch-Based Multi-View Stereo* (Furukawa and Ponce 2010), has proven quite effective in practice. After an initial feature-matching step aimed at constructing a sparse set of *photoconsistent* patches, in the sense of the previous section—that is, patches whose projections in the images where they are visible have similar brightness or color patterns—it divides the input images into small square cells a few pixels across, and attempts to reconstruct a patch in each one of them, using the cell connectivity to propose new patches, and visibility constraints to filter out incorrect ones (Algorithm 19.4). The overall

process is illustrated in Figure 19.13.

In practice, the expansion and filtering steps are iterated $K = 3$ times.

1. **Matching (Section 19.2.2):** Use feature matching to construct an initial set of patches, and optimize their parameters to make them maximally photoconsistent.
2. Repeat K times:
   (a) **Expansion (Section 19.2.3):** Iteratively construct new patches in empty spots near existing ones, using image connectivity and depth extrapolation to propose candidates, and optimizing their parameters as before to make them maximally photoconsistent.
   (b) **Filtering (Section 19.2.4):** Use again the image connectivity to remove patches identified as outliers because their depth is not consistent with a sufficient number of other nearby patches.

Algorithm 19.4: The PMVS Algorithm.

19.2.1 Main Elements of the PMVS Model

As in the previous section, we assume throughout that n cameras with known intrinsic and extrinsic parameters observe a static scene, and respectively denote by O_i and I_i $(i = 1, \ldots, n)$ the optical centers of these cameras and the images they have recorded of the scene. The main elements of the PMVS model of multi-view stereo fusion and scene reconstruction are small rectangular patches, intended to be tangent to the observed surfaces, and a few of these patches' key properties—namely, their geometry, which images they are visible in and whether they are photoconsistent with those, and some notion of connectivity inherited from image topology. Before detailing in Sections 19.2.2 to 19.2.4 the different stages of Algorithm 19.4, let us give concrete definitions for these properties.

Patch Geometry. We associate with every rectangular patch p some reference image $\mathcal{R}(p)$. We will see in the following sections how to determine this picture but, intuitively, p should obviously be visible in $\mathcal{R}(p)$, and preferably nearly parallel to its retinal plane. As illustrated by Figure 19.14 (left), p is defined geometrically by its center $\boldsymbol{c}(p)$; its unit normal $\boldsymbol{n}(p)$, oriented towards the cameras observing it; its orientation about $\boldsymbol{n}(p)$, chosen so one of the rectangle's edges is aligned with the rows of $\mathcal{R}(P)$; and its extent, chosen so its projection into $\mathcal{R}(p)$ fits within a square of size $\mu \times \mu$ pixels. As in the case of the correlation windows used in narrow-baseline stereopsis, this size is chosen to capture a sufficiently rich description of the local image pattern, yet remain small enough to be robust to occlusion. Taking $\mu = 5$ gives good results in practice.

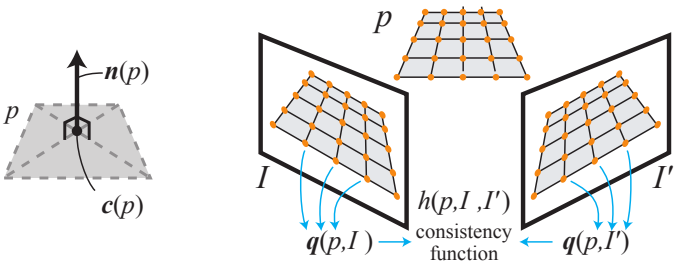

FIGURE 19.14: A patch p (**left**) and its projection into two images I and I' (**right**). The photoconsistency of p, I, and I' is measured by the normalized correlation between the sets $\boldsymbol{q}(p, I)$ and $\boldsymbol{q}(p, I')$ of interpolated pixel colors at the projections of the patch's grid points. *Reprinted from "Accurate, Dense, and Robust Multi-View Stereopsis," by Y. Furukawa and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8):1362–1376, (2010). © 2010 IEEE.*

Visibility. We say that a patch p is *potentially visible* in an image I_i when it lies in the field of view of the corresponding camera and faces it—that is, the angle between $\boldsymbol{n}(p)$ and the projection ray joining $\boldsymbol{c}(p)$ to O_i is below some threshold $\alpha < \pi/2$. Let us denote by $\mathcal{V}(p)$ the set of images where p is potentially visible. We also say that the patch p is *definitely visible* in an image I_i of $\mathcal{V}(p)$ when its center $\boldsymbol{c}(p)$ is the closest to O_i among all patches potentially visible in I_i.

Photoconsistency. In narrow-baseline stereo settings, photoconsistency is typically measured by the normalized correlation between fixed-sized image patches whose brightness or color patterns are naturally sampled on the corresponding image grids. As shown in Chapter 7, this might become problematic in the presence of foreshortening, which can be quite severe in wide-baselines scenarios. In this context, it is more natural to overlay a $\nu \times \nu$ grid on the rectangle associated with a patch p, and measure its photoconsistency with two images I and I' as the normalized cross-correlation $h(p, I, I')$ between (bilinearly) interpolated pixel values at the projections of the grid points in the two images (Figure 19.14, right). It is also natural to take $\mu = \nu$ because this ensures that cells on the patch grid roughly correspond to pixels in the reference image. The photoconsistency of a patch p with some image I in $\mathcal{V}(p)$ can now be defined as

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

and we say that p is *photoconsistent* with I when $g(p, I)$ is above some threshold β. Note that the overall photoconsistency of a patch p can be measured as $f(p) = g(p, \mathcal{R}(p))$. This measure can be used to select potential matches between image features. More interestingly, it can also be used to refine the parameters of a patch p to make it maximally photoconsistent along the corresponding projection ray of $\mathcal{R}(p)$: in practice, the simplex method (Nelder and Mead 1965) is used to (locally) maximize $f(p)$ with respect to the two orientation parameters of $\boldsymbol{n}(p)$ and the depth

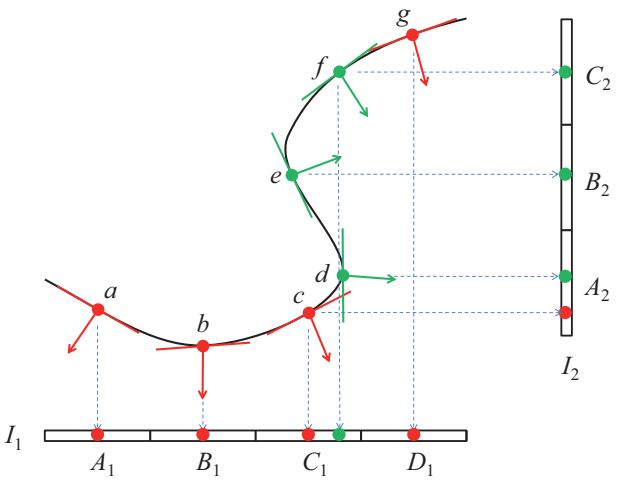

FIGURE 19.15: A toy 2D example with seven patches a to g and two orthographic input images I_1 and I_2. I_1 is divided into four cells, A_1 to D_1, and serves as the reference image for the patches a, b, c, g. I_2 is divided into three cells A_2, B_2, C_2 and serves as reference image for d, e, f. Here we have, for example, $\mathcal{C}_1(b) = B_1$ and $\mathcal{C}_2(e) = B_2$. Also note that, although the projections of the patches d and e into I_1 fall inside the cell C_1, these two patches don't belong to $\mathcal{P}(C_1) = \{c, f\}$ because the angles between their normals and the direction of projection into I_1 is larger than $\alpha = \pi/3$ (e actually faces away from I_1). Indeed, $\mathcal{V}(d) = \mathcal{V}(e) = \{I_2\}$. On the other hand, we have, for example, $\mathcal{V}(f) = \{I_1, I_2\}$. Although the patches c and f are potentially visible in I_1, only c can be said to be definitely visible in this image. Likewise, d is definitely visible in I_2, but c can only be ascertained to be potentially visible in that image.

of $\boldsymbol{c}(p)$ along the ray, but any other nonlinear optimization technique could be used instead.

Connectivity. As noted earlier, the image topology can be used as a proxy for the connectivity of reconstructed surface patches. Concretely, one can overlay on each picture a regular grid of small square cells a few pixels across (potentially up to one cell per pixel, although 2×2 cells are used in all experiments presented in this chapter), and associate with any patch p and image I_i in $\mathcal{V}(p)$ the cell $\mathcal{C}_i(p)$ where it projects, and with any cell A_i of some image I_i the list $\mathcal{P}(A_i)$ of patches p such that I_i belongs to $\mathcal{V}(p)$ and $\mathcal{C}_i(p) = A_i$ (Figure 19.15). This allows us to define the *potential neighbors* of a patch p as the patches p' that belong to $\mathcal{P}(A'_i)$ for some image I_i and some cell A'_i adjacent to $\mathcal{C}_i(p)$. A potential neighbor p' of p is a *definite neighbor* of this patch when p and p' are consistent with a smooth surface—that is, on average, the center of each patch lies close enough to the plane of the other one, or

$$\frac{1}{2}\left(|[\boldsymbol{c}(p') - \boldsymbol{c}(p)] \cdot \boldsymbol{n}(p)| + |[\boldsymbol{c}(p) - \boldsymbol{c}(p')] \cdot \boldsymbol{n}(p')|\right) < \gamma$$

for some threshold γ.

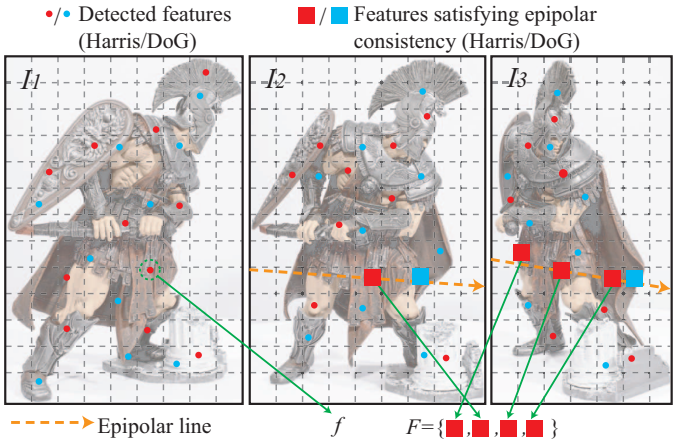

FIGURE 19.16: Feature-matching example showing the features f' in F satisfying the epipolar constraint in images I_2 and I_3 as they are matched to feature f in image I_1. (This is an illustration only, not showing actual detected features.) *Reprinted from "Accurate, Dense, and Robust Multi-View Stereopsis," by Y. Furukawa and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8):1362–1376, (2010). © 2010 IEEE.*

The heuristic nature of the definitions given in this section is obvious, and somewhat unsatisfactory, because they require hand-picking appropriate values for the parameters α, β, and γ. In practice, however, default values give satisfactory results for the vast majority of situations. In particular, Furukawa and Ponce (2010) always use a value of $\pi/3$ for α, and use values of $\beta = 0.4$ before patch refinement and $\beta = 0.7$ afterwards in the initial feature-matching stage of Algorithm 19.4, loosening (decreasing) these thresholds by a factor of 0.8 after each expansion/filtering iteration to gather more patches in challenging areas. Likewise, when deciding whether two patches p and p' are neighbors, γ is automatically set to the lateral distance between the preimages of the corresponding cell centers at the depth of the mid-point between $\boldsymbol{c}(p)$ and $\boldsymbol{c}(p')$.

19.2.2 Initial Feature Matching

In the first stage of Algorithm 19.4, Harris and DoG interest points are matched to construct an initial set of patches (Figure 19.16). The parameters of these patches are then optimized to make them maximally photoconsistent. Consider some input image I_i, and denote as before by O_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. Each pair (f, f') defines a 3D point and an initial patch hypothesis p centered at that point $\boldsymbol{c}(p)$ with a normal $\boldsymbol{n}(p)$ aligned with the corresponding projection ray. These hypotheses are examined one by one in increasing depth order from O_i until either one of them leads to the creation of a photoconsistent patch or their list is exhausted. This simple heuristic gives good

results in practice for a modest computational cost. Given some initial patch hypothesis p with center $\boldsymbol{c}(p)$ and normal $\boldsymbol{n}(p)$, let us now define $\mathcal{R}(p) = I_i$. The extent and orientation of p are easily computed from these parameters, and $\mathcal{V}(p)$ is then determined using the threshold β. The optimization procedure described in the previous section can then be used to refine p's parameters and update $\mathcal{V}(p)$. When p is found to be visible in at least δ photographs (in practice, taking $\delta = 3$ yields good results), the patch generation procedure is deemed a success, and p is stored in the corresponding cells of the images in $\mathcal{V}(p)$. The overall procedure is given in Algorithm 19.5.

This outputs an initial list P of patch candidates.
$P \leftarrow \emptyset$.
For each image I_i with optical center O_i and for each feature f detected in I_i do

1. $F \leftarrow$ {Features satisfying the epipolar constraint}.
2. Sort F in increasing depth order from O_i.
3. For each feature f' in F do
   - **(a)** Initialize a patch p by computing $\boldsymbol{c}(p)$, $\boldsymbol{n}(p)$, and $\mathcal{R}(p)$.
   - **(b)** Initialize $\mathcal{V}(p)$ with $\beta = 0.4$.
   - **(c)** Refine $\boldsymbol{c}(p)$ and $\boldsymbol{n}(p)$.
   - **(d)** Update $\mathcal{V}(p)$ with $\beta = 0.7$.
   - **(e)** If $|\mathcal{V}(p)| \geq \delta$, then
     - **i.** Add p to $\mathcal{P}(\mathcal{C}_i(p))$.
     - **ii.** Add p to P.

Algorithm 19.5: The Feature-Matching Algorithm of PMVS.

19.2.3 Expansion

Patch expansion is an iterative procedure that repeatedly tries to generate new patches in "empty" cells $\mathcal{E}(p)$ adjacent to the projections of existing patches p in the input images. The new patches are initialized by extrapolating the depth of the old ones, and their parameters are then optimized as before to make them maximally photoconsistent. Let us first define $\mathcal{D}(p)$ as the set of cells adjacent to $C_i(p)$ for all images I_i in $\mathcal{V}(p)$ (Figure 19.17). These are candidates for expansion, but some of them must be pruned because they are already consistent with p—that is, they contain one of its definite neighbors—or with I_i—that is, they contain a patch p' photoconsistent with this image. The latter case typically corresponds to occlusion boundaries, where the observed surface folds away from camera j between the patches p and p'. The set $\mathcal{E}(p)$ of empty cells adjacent to p thus consists of the elements of $\mathcal{D}(p)$ that are neither consistent with p nor with I_i (Figure 19.17).

For each image cell A_i in $\mathcal{E}(p)$, a depth extrapolation procedure is performed to generate a new patch p', initializing $\boldsymbol{c}(p')$ as the point where the viewing ray

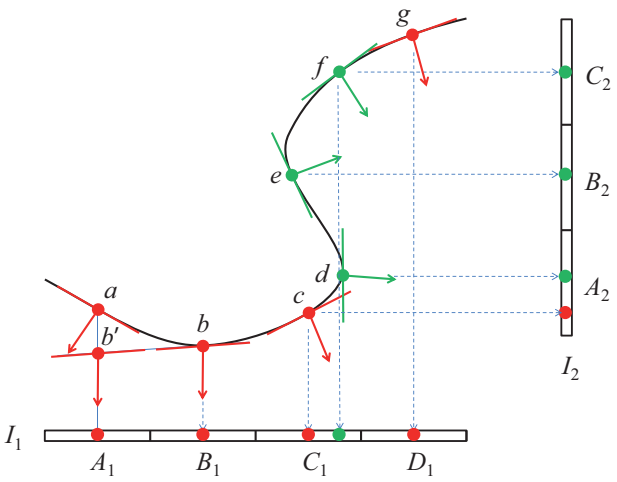

FIGURE 19.17: Candidate cells for expansion. In this example, $\mathcal{D}(c) = \{B_1, D_1, B_2\}$ and $\mathcal{D}(b) = \{A_1, C_1\}$. Assume that all patches except a have been constructed, that they are consistent with the two images, and that b and c are neighbors. In this case, $\mathcal{E}(c)$ is empty because b is a neighbor of c, thus B_1 must be eliminated, and g and e are respectively consistent with I_1 and I_2, thus D_1 and B_2 must be eliminated as well. On the other hand, $\mathcal{E}(b) = \{A_1\}$ because A_1 is (so far) empty. During the expansion procedure, the patch b' is generated in the unique cell A_1 of $\mathcal{E}(b)$, and it is then refined into the patch a.

passing through the center of A_i intersects the plane containing p. The parameters $\boldsymbol{n}(p'), \mathcal{R}(p')$, and $\mathcal{V}(p')$ are then initialized with the corresponding values for p, and $\mathcal{V}(p')$ is pruned using the threshold β to eliminate extraneous pictures. After this step, $\boldsymbol{c}(p')$ and $\boldsymbol{n}(p')$ are refined as before (Figure 19.17). After the optimization, we add to $\mathcal{V}(p')$ additional images where p is deemed definitely visible. A visibility test with a tighter threshold ($\beta = 0.7$) is then applied as before to filter out extraneous images. Finally, p' is accepted as a new patch when $\mathcal{V}(p')$ contains at least δ images, and $\mathcal{P}(\mathcal{C}_k(p'))$ is updated for all images I_k in $\mathcal{V}(p')$. The procedure is detailed in Algorithm 19.6.

19.2.4 Filtering

This stage of the algorithm again exploits image connectivity information to remove patches identified as outliers because their depth is not consistent with a sufficient number of other nearby patches. Three filters are used for this task. The first one is based on visibility consistency constraints: two patches p and p' are said to be inconsistent when they are not definite neighbors in the sense of Section 19.2.1, yet are stored in the same cell for one of the images (Figure 19.18). For each reconstructed patch p, if U denotes the set of patches inconsistent with p, p is discarded as an outlier when

$$|\mathcal{V}(p)|g(p) < \sum_{p' \in U} g(p').$$

It takes as input the candidate patches P from Algorithm 19.5 and outputs an expanded set of patches P'.
$P' \leftarrow P$.
While $P \neq \emptyset$ do

1. Pick and remove a patch p from P.
2. For each cell A_i in $\mathcal{E}(p)$ do
   - **(a)** Create a new patch candidate p', with $\boldsymbol{c}(p')$ defined as the intersection of the plane containing p and the ray joining O_i to the center of A_i.
   - **(b)** $\boldsymbol{n}(p') \leftarrow \boldsymbol{n}(p), \quad \mathcal{R}(p') \leftarrow \mathcal{R}(p), \quad \mathcal{V}(p') \leftarrow \mathcal{V}(p)$.
   - **(c)** Update $\mathcal{V}(p')$ with $\beta = 0.4$.
   - **(d)** Refine $\boldsymbol{c}(p')$ and $\boldsymbol{n}(p')$.
   - **(e)** Add images where p' is definitely visible to $\mathcal{V}(p')$.
   - **(f)** Update $\mathcal{V}(p')$ with $\beta = 0.7$.
   - **(g)** If $|\mathcal{V}(p')| \geq \delta$ then
     - **i.** Add p' to P and P'.
     - **ii.** Add p' to $\mathcal{P}(\mathcal{C}_k(p'))$ for all I_k in $\mathcal{V}(p)$.

Algorithm 19.6: The Patch-Expansion Algorithm of PMVS.

Intuitively, when p is an outlier, both $g(p)$ and $|\mathcal{V}(p)|$ are expected to be small, and p is likely to be removed.

The second filter also enforces visibility constraints by simply rejecting all patches that are not definitely visible, in the sense of Section 19.2.1, in at least δ images. Finally, the third filter enforces a weak form of smoothness: For each patch p, we collect the patches lying in its own and adjacent cells in all images of $\mathcal{V}(p)$. If the proportion of patches that are neighbors of p in this set is lower than 25%, p is removed as an outlier.

19.2.5 Results

Figure 19.19 shows some results using four datasets, with 48 input images for the Roman soldier figurine, 16 for the dinosaur, 24 for the skull, and 4 for the face. Like the number of these photographs, their resolution varies with the dataset, from $1,800 \times 1,200$ pixels for the Roman soldier to 640×480 for the dinosaur. The top part of the figure shows one image per dataset, and its central part shows two views of each reconstructed model. Although the models might look like texture-mapped meshes, they are just—rather dense, to be sure—sets of floating patches, each rectangle being painted with the mean of the interpolated pixel values used to reconstruct it. Finally, the bottom part of the figure shows shaded views of meshes fitted to the patch models using the method presented in Furukawa and Ponce (2010). This procedure takes as input an outer approximation of the model, such as a visual hull of the observed scene if silhouette information is available, or

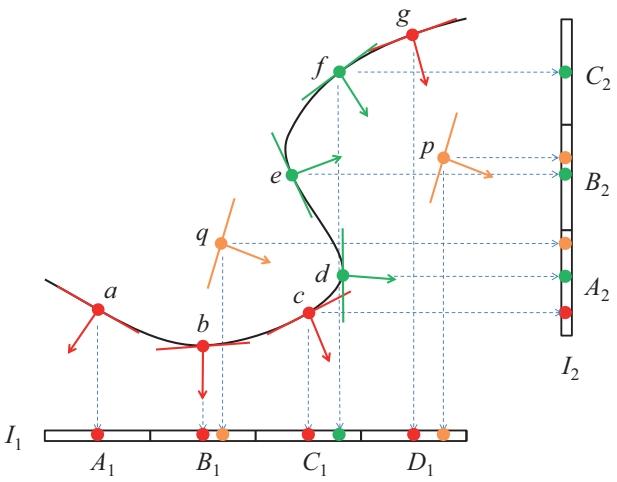

FIGURE 19.18: Filtering outliers. The patch p is rejected as an outlier by the first filter, granted that the photoconsistency scores of e and g are high enough because these two patches project into the same cells (respectively B_2 and D_1) and are inconsistent with p—that is, $U = \{e, g\}$. The patch q is eliminated by the second filter because it is not definitely visible in any image.

the convex hull of the reconstructed patches otherwise, and iteratively deforms the corresponding mesh to fit it to these patches under both smoothness and photo-consistency constraints. The reader is refered to Furukawa and Ponce (2010) for the details of this algorithm, which are beyond the scope of this book.

FIGURE 19.19: From **top** to **bottom**: Sample input images, reconstructed patches, and final mesh models. *Reprinted from "Accurate, Dense, and Robust Multi-View Stereopsis," by Y. Furukawa and J. Ponce, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8):1362–1376, (2010). © 2010 IEEE.*

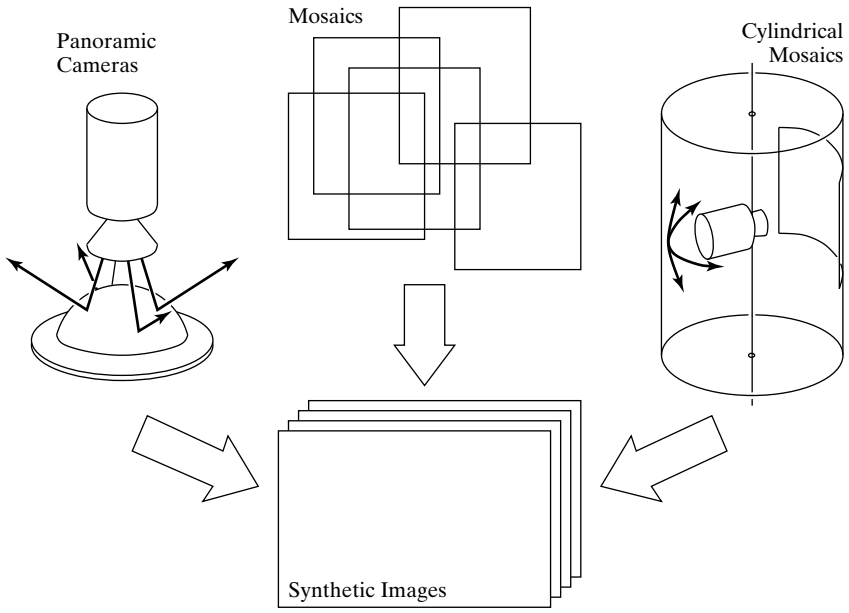

FIGURE 19.20: Constructing synthetic views of a scene from a fixed viewpoint.

19.3 THE LIGHT FIELD

This section discusses a totally different approach to image-based modeling and rendering, that entirely forsakes the construction of a three-dimensional object model, yet is capable of synthesizing realistic new views of scenes with arbitrarily complex geometries. To show that this is possible, let us consider, for example, a panoramic camera that optically records the radiance along rays passing through a single point and covering a full hemisphere (Peri and Nayar 1997). It is possible to create any image observed by a virtual camera whose pinhole is located at this point by mapping the original image rays onto virtual ones. This allows a user to arbitrarily pan and tilt the virtual camera and interactively explore his or her visual environment. Similar effects can be obtained by stitching together close-by images taken by a hand-held camcorder into a mosaic (see Shum and Szeliski [1998] and Figure 19.20, middle), or by combining the pictures taken by a camera panning (and possibly tilting) about its optical center into a cylindrical mosaic (see Chen [1995] and Figure 19.20, right).

These techniques have the drawback of limiting the viewer motions to pure rotations about the optical center of the camera. A more powerful approach can be devised by considering the *plenoptic function* (Adelson and Bergen 1991) that associates with each point in space the (wavelength-dependent) radiant energy along a ray passing through this point at a given time (Figure 19.21, left). The *light field* (Levoy and Hanrahan 1996) is a snapshot of the plenoptic function for light traveling in a vacuum in the absence of obstacles. This relaxes the dependence of the radiance on time and on the position of the point of interest along the corresponding ray (since radiance is constant along straight lines in a nonabsorbing medium) and

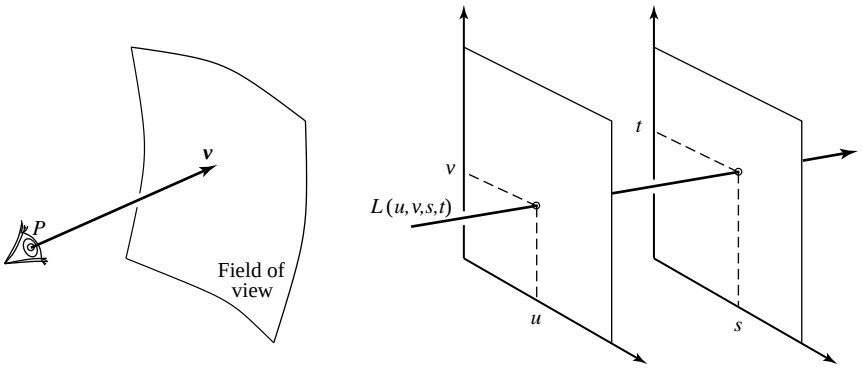

FIGURE 19.21: The plenoptic function and the light field. **Left:** The plenoptic function can be parameterized by the position P of the observer and the viewing direction $\boldsymbol{v}$. **Right:** The light field can be parameterized by the four parameters u, v, s, t defining a light slab. In practice, several light slabs are necessary to model a whole object and obtain full spherical coverage.

yields a representation of the plenoptic function by the radiance along the four-dimensional set of light rays. In the image-based rendering context, a convenient parameterization of these rays is the *light slab*, where each ray is specified by the coordinates of its intersections with two arbitrary planes (Figure 19.21, right).

The light slab is the basis for a two-stage approach to image-based rendering. During the learning stage, many views of a scene are used to create a discrete version of the slab that can be thought of as a four-dimensional lookup table. At synthesis time, a virtual camera is defined, and the corresponding view is interpolated from the lookup table. The quality of the synthesized images depends on the number of reference images. The closer the virtual view is to the reference images, the better the quality of the synthesized image. Note that constructing the light slab model of the light field does not require establishing correspondences between images. It should be noted that, unlike most methods for image-based rendering that rely on texture mapping and thus assume (implicitly) that the observed surfaces are Lambertian, light-field techniques can be used to render (under a fixed illumination) pictures of objects with *arbitrary* reflectance functions.

In practice, a sample of the light field is acquired by taking a large number of images and mapping pixel coordinates onto slab coordinates. Figure 19.22 illustrates the general case: the mapping between any pixel in the (x, y) image plane and the corresponding areas of the (u, v) and (s, t) plane defining a light slab is a planar projective transformation. Hardware- or software-based texture mapping can thus be used to populate the light field on a four-dimensional rectangular grid. In the experiments described in Levoy and Hanrahan (1996), light slabs are acquired in the simple setting of a camera mounted on a planar gantry and equipped with a pan-tilt head so it can rotate about its optical center and always point toward the center of the object of interest. In this context, all calculations can be simplified by taking the (u, v) plane to be the plane in which the camera's optical center is

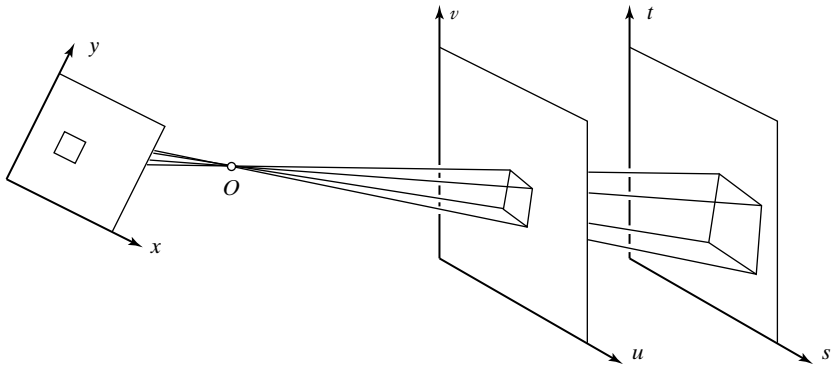

FIGURE 19.22: The acquisition of a light slab from images and the synthesis of new images from a light slab can be modeled via projective transformations between the (x, y) image plane and the (u, v) and (s, t) planes defining the slab.

constrained to remain.

At rendering time, the projective mapping between the (virtual) image plane and the two planes defining the light slab can once again be used to efficiently synthesize new images. Figure 19.23 shows sample pictures generated using the light-field approach. The top three image pairs were generated using synthetic pictures of various objects to populate the light field. The last pair of images was constructed by using the planar gantry mentioned earlier to acquire 2,048 256×256 images of a toy lion, grouped into four slabs consisting of 32×16 images each.

An important issue is the size of the light slab representation; for example, the raw input images of the lion take 402MB of disk space. There is, of course, much redundancy in these pictures, as in the case of successive frames in a motion sequence. A simple but effective two-level approach to image (de)compression is proposed in Levoy and Hanrahan (1996): the light slab is first decomposed into four-dimensional tiles of color values. These tiles are encoded using *vector quantization* (Gersho and Gray 1992), a lossy compression technique where the 48-dimensional vectors representing the RGB values at the 16 corners of the original tiles are replaced by a relatively small set of reproduction vectors, called *codewords*, that best approximate in the mean-squared-error sense the input vectors. The light slab is thus represented by a set of indexes in the *codebook* formed by all codewords. In the case of the lion, the codebook is relatively small (0.8MB) and the size of the set of indexes is 16.8MB. The second compression stage consists of applying the *gzip* implementation of *entropy coding* (Ziv and Lempel 1977) to the codebook and the indexes. The final size of the representation is only 3.4MB, corresponding to a compression rate of 118:1. At rendering time, entropy decoding is performed as the file is loaded in main memory. Dequantization is performed on demand during display, and it allows interactive refresh rates.

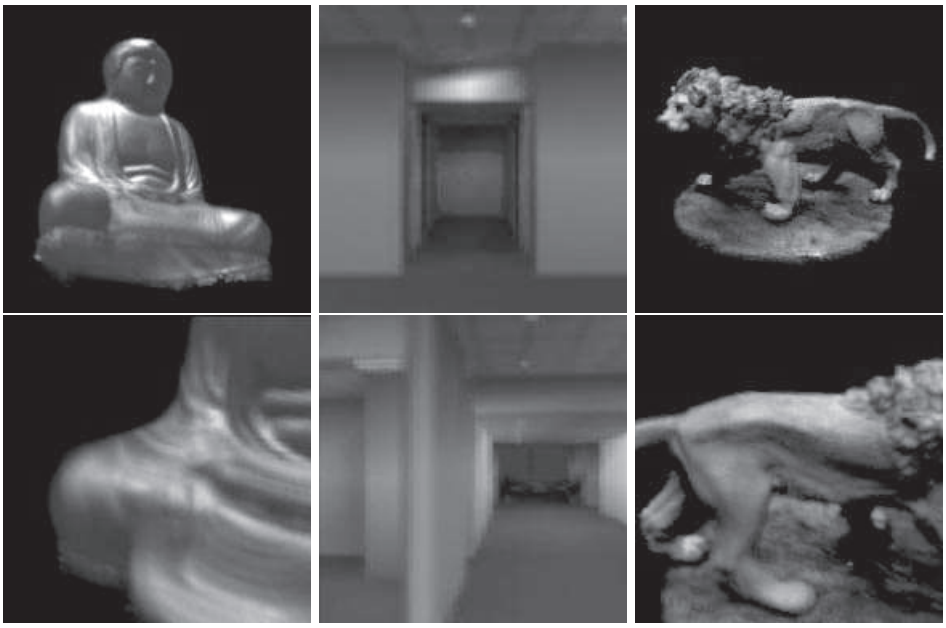

FIGURE 19.23: Images of three scenes synthesized with the light field approach. *Reprinted from "Light Field Rendering," by M. Levoy and P. Hanrahan, Proc. SIGGRAPH, (1996). © 1996 ACM, Inc.* `http://doi.acm.org/10.1145/10.1145/237170.237199` *Reprinted by permission.*

19.4 NOTES

Visual hulls date back to Baumgart's PhD thesis (1974), and their geometric properties have been studied in Laurentini (1995) and Petitjean (1998). Voxel- and octree-based volumetric methods for computing the visual hull include Martin and Aggarwal (1983) and Srivastava and Ahuja (1990); see also Kutulakos and Seitz (1999) for a related approach, called *space carving*, where empty voxels are iteratively removed using photoconsistency constraints. Visual hull algorithms based on polyhedral models include Baumgart (1974), Connolly and Stenstrom (1989), Niem and Buschmann (1994), Matusik, Buehler, Raskar, Gortler, and McMillan (2001), and, more recently, Franco and Boyer (2009). The problem of computing the visual hull of a solid bounded by a smooth surface is addressed in Lazebnik, Boyer, and Ponce (2001) and Lazebnik, Furukawa, and Ponce (2007). The algorithm described in Section 19.1 in the context of cameras with known intrinsic and extrinsic parameters actually also applies to weakly calibrated images. Not suprisingly, its output is defined only up to a projective transformation. Combining photometric information with the geometric constraints associated with visual hulls was first proposed in Sullivan and Ponce (1998). Variants of the carved visual hulls proposed in Furukawa and Ponce (2009*a*) and described in Section 19.1 include, for example, Hernandez Esteban and Schmitt (2004) and Sinha and Pollefeys (2005). The fact that a viewing cone is tangent to the surface observed by the corresponding camera, used in Furukawa and Ponce (2009*a*) to carve a visual hull, is also the