INVITED PAPER

Scene Reconstruction and Visualization From Community Photo Collections

Recent progress is described in digitizing and visualizing the world from data captured by people taking photos and uploading them to the web.

By Noah Snavely, *Member IEEE*, Ian Simon, Michael Goesele, *Member IEEE*, Richard Szeliski, *Fellow IEEE*, and Steven M. Seitz, *Senior Member IEEE*

ABSTRACT | There are billions of photographs on the Internet, representing an extremely large, rich, and nearly comprehensive visual record of virtually every famous place on Earth. Unfortunately, these massive community photo collections are almost completely unstructured, making it very difficult to use them for applications such as the virtual exploration of our world. Over the past several years, advances in computer vision have made it possible to automatically reconstruct 3-D geometry—including camera positions and scene models—from these large, diverse photo collections. Once the geometry is known, we can recover higher level information from the spatial distribution of photos, such as the most common viewpoints and paths through the scene. This paper reviews recent progress on these challenging computer vision problems, and describes how we can use the recovered structure to turn community photo collections into immersive, interactive 3-D experiences.

KEYWORDS | Internet photo collections; multiview stereo; scene summarization; structure from motion; 3-D navigation and visualization; 3-D reconstruction

I. INTRODUCTION

The Internet has become a vast, ever-growing repository of visual information about our world. Virtually all of the world's famous landmarks and cities (and many not-so-famous ones) have been photographed many different times, both from the ground and from the air. Billions of these photos can be found on mapping sites such as Google Maps [23] and Microsoft's Bing Maps [76], and on public photo-sharing websites such as Flickr [19] and Photobucket [53]. For instance, a Flickr search for "Trafalgar Square" results in nearly 150 000 photos (as of June 2010), showing the square from almost every conceivable viewing position and angle, different times of day and night, changes in season, weather, and decade, and during different events. Moreover, these photos do not exist in isolation, but are surrounded by rich context, such as textual tags describing the contents of photos, metadata including who took the photo and when it was taken, and Wikipedia articles [75]. There is also a wealth of information represented in the *behavior* of large numbers of photographers. For example, given all the world's photos of Trafalgar Square, we could potentially identify the most-photographed objects in the scene and find the most common paths taken by tourists.

In short, these *community photo collections* and their context represent a very rich set of information about our world, and open up enormous opportunities, both for research and for practical applications. Imagine mining these collections to create the ultimate virtual experience of a famous site.

Manuscript received April 14, 2009; revised August 13, 2009; accepted March 10, 2010. Date of publication June 10, 2010; date of current version July 21, 2010. This work was supported in part by Microsoft, Adobe, Google, the University of Washington Animation Research Labs, an Achievement Rewards for College Scientists (ARCS) fellowship, the National Science Foundation under Grants IIS-0413198, IIS-0811878, and DGE-0203031, the DFG Emmy Noether Fellowship GO 1752/3-1, the Office of Naval Research, and an endowment by Emer Dooley and Rob Short. Collection credit and copyright notice for *Moon and Half Dome*, 1960, by Ansel Adams: Collection Center for Creative Photography, University of Arizona, © Trustees of The Ansel Adams Publishing Rights Trust.

N. Snavely is with the Computer Science Department, Cornell University, Ithaca, NY 14853-7501 USA (e-mail: snavely@cs.cornell.edu).
I. Simon is with the Computer Science and Engineering Department, University of Washington, Seattle, WA 98195-2350 USA (e-mail: iansimon@cs.washington.edu).
M. Goesele is with the Technische Universität Darmstadt, Darmstadt 64289, Germany (e-mail: michael.goesele@gris.informatik.tu-darmstadt.de).
R. Szeliski is with Microsoft Research, Redmond, WA 98052-6399 USA (e-mail: szeliski@microsoft.com).
S. M. Seitz is with the Computer Science and Engineering Department, University of Washington, Seattle, WA 98195-2350 USA, and also with Google, Inc., Seattle, WA 98102 USA (e-mail: seitz@cs.washington.edu).

Digital Object Identifier: 10.1109/JPROC.2010.2049330

Such an experience could be extremely visually compelling, giving us the ability to walk around in a photorealistic 3-D version of the scene, to let us dial in any time of day or year, and to let us revisit different events. In addition, such an experience could be informed by additional contextual and behavioral information. For instance, we could be guided to the most interesting parts of the scene and provided information about different buildings, statues, and paintings, all through automatic analysis of existing data on the Web.

Unfortunately, a large gap exists between unstructured community photo collections on sites such as Flickr and this vision of a photorealistic digital replica of the world augmented with contextual knowledge. How can we automatically recover useful geometric and semantic structure from the unorganized, uncalibrated clouds of photos on the Internet? Researchers in the field of computer vision have for decades studied aspects of these problems, such as 3-D reconstruction from images. Historically, vision algorithms have only been applied to very controlled data sets—such as video frames or images captured in the lab—far removed from the kinds of photos found on the Web. Recent breakthroughs in computer vision, however, have closed this gap considerably, opening up the possibility of extracting structure from photos found "in the wild." It is now possible, for example, to automatically create 3-D models from images downloaded from Flickr, as illustrated with Trafalgar Square in Fig. 1.

This paper describes our work on computer vision algorithms and visualization tools that are taking us closer to this vision of an immersive, photorealistic, and intelligent virtual experience of our world. We begin by discussing the problem of accurate 3-D reconstruction, as well as recovery of appearance information, such as color and glossiness, from community photo collections (Sections II and III). We then describe our work on using the behavior of photographers to infer information about a scene. This includes computing *scene summaries* that identify interesting views, segmenting, and labeling individual objects, and finding common paths through the scene (Sections IV and V). These algorithms give us both geometric and semantic scene structure. We then show how we can use this information to create an interactive scene visualization system. This system, described in Section VI, uses the derived geometry to display the scene as a user moves around, and uses the inferred semantic structure to suggest interesting views and paths. Finally, we conclude by discussing some of the many remaining open problems in recovering structure from community photo collections (Section VII).

II. RECONSTRUCTING CAMERAS AND SPARSE GEOMETRY

Each of the applications described in this paper rely on a few essential computer vision algorithms. These algorithms take the raw, unorganized images, and produce a midlevel, geometric representation of the underlying scene, which supports higher level tasks such as finding the most common views and paths (Sections IV and V). This midlevel representation consists of three basic elements:

- a position and orientation for each input photo describing where it was taken (the *camera pose*);
- a sparse set of 3-D points in the scene;
- a description of which 3-D points are visible in each image.

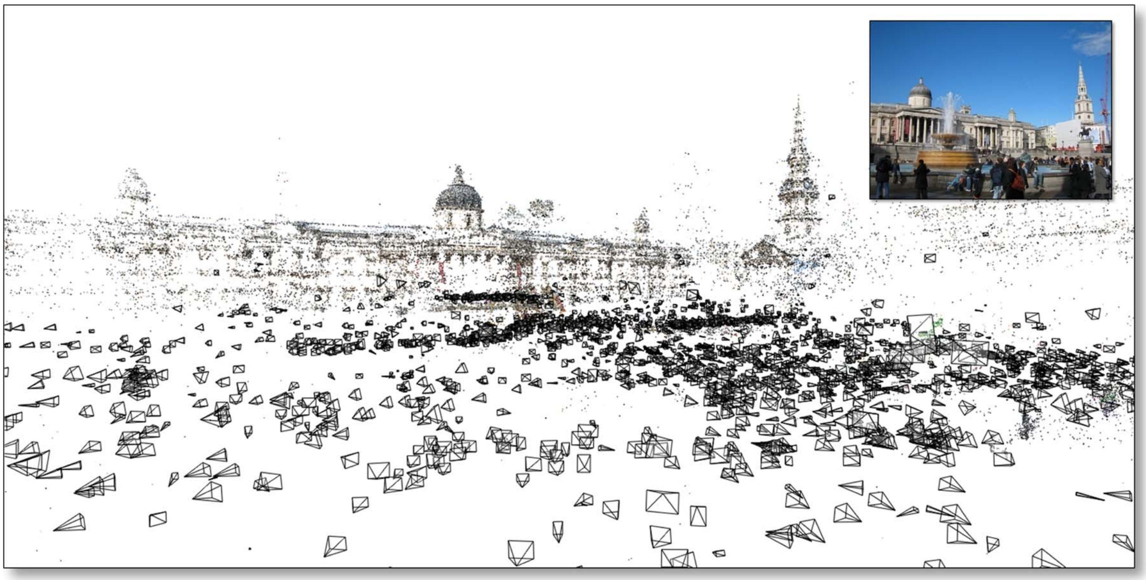

Fig. 1. ***Point cloud reconstruction of Trafalgar Square from several thousand Internet photos. The reconstructed cameras are shown as black wire-frame pyramids. Inset: one of the input photos taken from approximately the same viewpoint.***

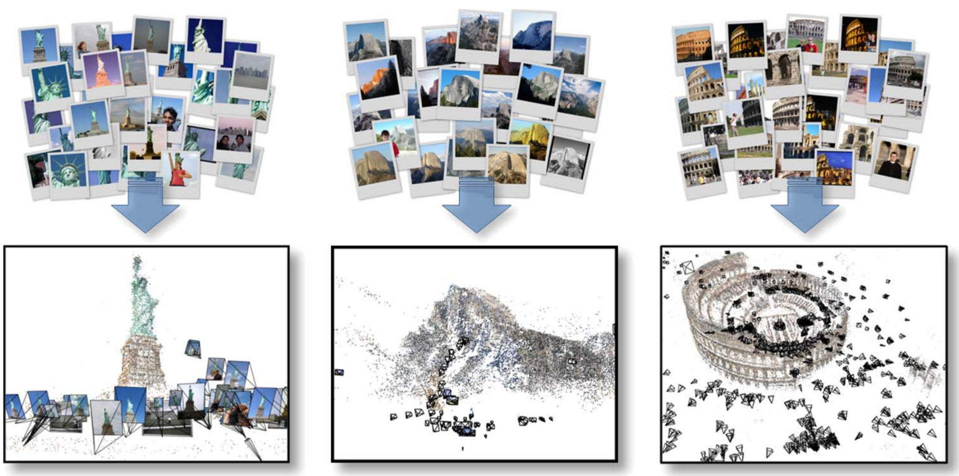

Fig. 2. ***3-D reconstructions from Internet photo collections. We take image results from keyword search (top), and automatically recover camera positions and scene geometry (bottom).***

We refer to this representation as a *reconstruction*. Example reconstructions of three different scenes—the Statue of Liberty, Half Dome in Yosemite, and the Colosseum—are shown in Fig. 2. Once a collection of photos has been reconstructed, we can answer several questions about that collection, including the following.

- Where was a given photo taken?
- What parts of the scene was it looking at?
- Do two photos see any common parts of the scene? That is, do they overlap?

The problem of reconstructing geometry from images has a long history in computer vision and photogrammetry. Until recently, however, there were no computer vision techniques for recovering this kind of structure from the large, diverse sets of photos found on the Internet. However, the last decade has seen remarkable advances in computer vision, and this kind of information can now be reliably extracted completely automatically, without requiring human input or GPS information. This technology has started making its way into commercial applications such as Microsoft's Photosynth [54], where users can create 3-D visualizations from their own photo collections.

Recovering 3-D geometry from a set of 2-D images is a challenging problem, because the process of photographing a scene results in the loss of a dimension (the depth of the scene). However, the basic principles behind recovering geometry from multiple images are fairly straightforward. Humans and many animals implicitly use multiview geometry to sense depth with binocular vision. If we see the same point in the world (the corner of a window, say) in both eyes, we can implicitly "triangulate" that point to determine its rough distance.[1] This form of depth perception depends on two key components: 1) identifying which parts of the two images we perceive correspond to the same point in the world, and 2) knowing where the eyes are roughly located relative to each other (to enable triangulation of corresponding points). We as humans use these faculties without even thinking about them. A basic problem in computer vision is to generalize and automate these abilities, given many views taken from unknown viewpoints.

The first problem, that of finding 2-D point matches between images, is known as the *correspondence problem*. There are many automated techniques for finding correspondences between two images, but most work on the principle that the same 3-D point in the world (the window corner, for instance) will have a similar appearance in different images, particularly if those images are taken close together. Once we have solved the 2-D correspondence problem, we then face the second problem: How can we determine where each photo was taken and the 3-D location of each scene point, given just a set of corresponding 2-D points among the photos? If we knew the camera poses but not the point positions, we could find the points through triangulation; conversely, if we knew the 3-D points, we could find the camera poses through a process similar to triangulation called resectioning. Unfortunately, we know neither the camera poses nor the points.

As it turns out, however, the correspondences place constraints on the physical configuration of the cameras and points.[2] For instance, the two cameras must be situated so that rays through corresponding pixels actually intersect (or nearly intersect, given noise in the system). This is a powerful constraint, as two 3-D rays chosen at random are

[1] Binocular vision is only one of a number of cues we can use to estimate depth. Others include focus, parallax induced by the motion of the head, and the apparent size of objects.

[2] We refer to each image as being taken by a different "camera," meaning a specific 3-D pose, even if the same physical device is used from photo to photo.

very unlikely to pass close to one another. Thus, given enough point matches between two images, the geometry of the system becomes sufficiently constrained that we can determine the two camera poses and the 3-D point locations (up to a similarity transformation) [36], after which we can estimate the 3-D point positions through standard triangulation. The problem of using pixel correspondences to determine camera and point geometry in this manner is known as structure from motion (SfM).

The correspondence and SfM problems are challenging tasks in computer vision, with a long history of research. These problems are especially challenging for community photo collections, which are captured by many different cameras and under many different conditions (including various times of day, seasons, and weather conditions). However, research in computer vision over the past few decades has made great strides towards solving these fundamental problems.

The rest of this section briefly describes our approach to solving these problems. First, a set of pixel correspondences between all images is determined through feature detection and matching. Second, an incremental SfM algorithm is used to estimate the camera and scene geometry. This approach is illustrated in Fig. 3.

A. Finding Correspondences

The goal of correspondence estimation is to take a raw set of images and to find sets of matching 2-D pixels across all the images. Each set of matching pixels ideally represents a single point in 3-D. Given a completely unorganized set of images, how can we go about finding matching 2-D points? One common approach, and the one we use, is to first find a set of distinctive local features in each image—the image's fingerprints, so to speak—and to then identify similar-looking features in different images. This *feature matching* problem brings up many interesting questions. What is a "feature" and what makes it distinctive? How do we represent the appearance of a feature? How can we quickly find similar features in different images? These questions have been the basis of a great deal of research in computer vision for nearly three decades [20], [29], [44].

For a long time these techniques were largely limited to working with sets of images that were very similar, such as consecutive frames of a video sequence. However, in the past ten years, more powerful feature extractors have been developed that achieve invariance to a wide class of image transformations, including rotations, scales, changes in brightness or contrast, and, to some extent, changes in viewpoint [43], [48]. These techniques allow us to match features between images taken with different cameras, with different zoom and exposure settings, from different angles, and—in some cases—at completely different times of day. Thus, recent feature extractors open up the possibility of matching features in Internet collections, which vary along all of these dimensions (and more). In our system, we use the scale-invariant feature transform (SIFT) [43], which has become a popular tool for many computer vision applications. Fig. 4 shows an example of the output of SIFT given an example image from the Trevi Fountain, and Fig. 5 shows the power of modern feature matching techniques in matching features between very different images.

Feature extractors such as SIFT take an image and return a set of pixel locations that are highly distinctive. For each of these features, the detector also computes a "signature" for the neighborhood of that feature, also known as a *feature descriptor*: a vector describing the local image appearance around the location of that feature.

Once features have been extracted from each image, we match features by finding similar features in other

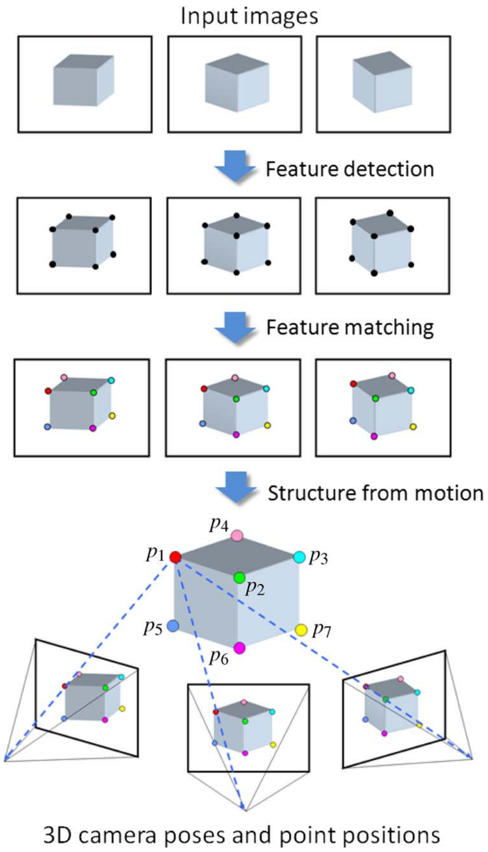

Fig. 3. ***Scene reconstruction pipeline. Reconstructing 3-D structure from photo collections typically has three basic steps. We start with a set of input images (in this case, three images of a cube) without any knowledge of the 3-D shape of the scene or the camera positions. First, we detect distinctive 2-D features in the input images, shown as black disks over the detected features. We then look for matching features between input images, shown by color-coding corresponding features. Finally, we use the correspondences to estimate the 3-D geometry of the cameras and points in a procedure known as structure from motion. Structure from motion optimizes the recovered 3-D structure for self-consistency—3-D points should project close to their detected 2-D locations. The cameras here are depicted as wireframe pyramids; the apex of each pyramid is the center of projection of the corresponding camera.***

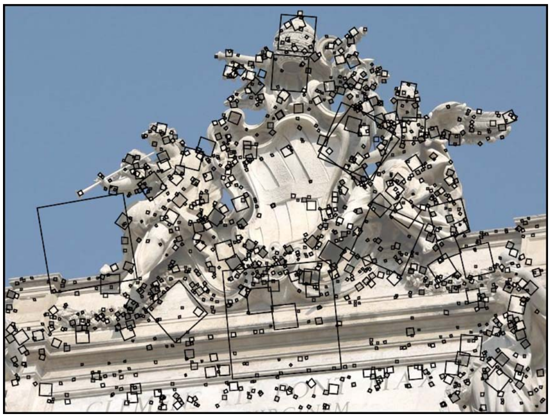

Fig. 4. *Example set of detected SIFT features. Each detected SIFT feature is displayed as a black box centered on the detected feature location. SIFT detects a canonical scale and orientation for each feature, depicted by scaling and rotating each box.*

images. In a brute force approach, we would compare every feature in an image to every other feature in every other image, looking for matches. However, using data structures such as *kd*-trees, we can find nearby features without doing a completely exhaustive search [3]. In our system, we still consider every pair of images, using a *kd*-tree to find matches between each image pair, resulting in a matching algorithm that takes time quadratic in the number of input images. Though this is a high computational cost, the *kd*-trees make matching each individual pair of images very efficient, and in addition the matching can be very easily parallelized across a cluster of machines. However, better algorithms are needed to scale image matching to entire cities. Recent work has used ideas from the text-retrieval community to create much more efficient image matching algorithms [9], [50], [64]. These approaches create *vocabularies* of visual features, and then index images using the type of inverted file data structures commonly used in text-retrieval systems such as Google.

Once all pairs of images have been matched, we can construct an *image connectivity graph* to represent the connections between the images in the collection. An image connectivity graph contains a node for each image, and an edge between any pair of images that have matching features. A visualization of the connectivity graph for a collection of Internet photos of the Trevi Fountain is shown in Fig. 6. To create this visualization, the graph was embedded in the plane using the *neato* tool in the Graphviz graph visualization toolkit [25]. *Neato* works by modeling the graph as a mass-spring system and solving for an embedding whose energy is a local minimum.

The image connectivity graph for this collection has several notable properties. There is a large, dense cluster in the center of the graph that consists of photos that are mostly wide-angle, frontal, well-lit shots of the fountain [such as Fig. 6(a)]. Other images, including the "leaf" nodes [such as Fig. 6(b) and (c)] corresponding to tightly cropped details, and nighttime images [such as Fig. 6(d)], are more loosely connected to this core set.

B. Structure From Motion

Once we have a set of correspondences between images, we can use them to recover the 3-D camera poses and the 3-D position of each point represented in the set of feature matches. This is known as the SfM problem. SfM is, at heart, an optimization problem: we seek the configuration of cameras and 3-D points which, when related through the equations of perspective projection, best agrees with the correspondences. This objective is illustrated at the bottom of Fig. 3. When we project any reconstructed 3-D point (such as the red corner of the cube) into a reconstructed camera, the projection should lie close to the 2-D image location where we detected that point. Here, the projection operation is visualized with the

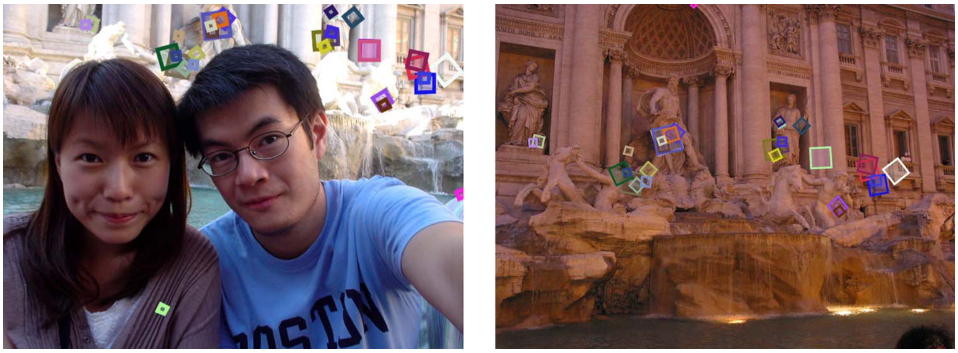

Fig. 5. *SIFT feature matches between two images of the Trevi Fountain. Each match is shown by a small square of the same color in both images. Matches are found despite the fact that the images are taken at different times of day, and in one image the scene is significantly hidden by people in the foreground.*

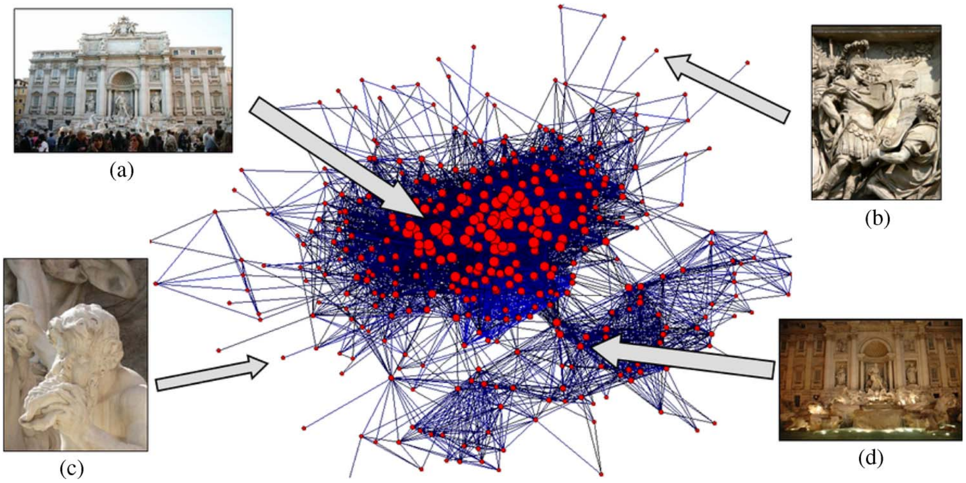

Fig. 6. ***Image connectivity graph for the Trevi Fountain. This graph contains a node (red dot) for each image in a set of photos of the Trevi Fountain, and an edge between each pair of photos with matching points. The size of a node is proportional to its degree. There are two dominant clusters corresponding to daytime photos [e.g., image (a)] and nighttime photos [image (d)]. Similar views of the facade are clustered together in the center of the graph, while nodes in the periphery of the graph, e.g., (b) and (c), are more unusual (often closeup) views. An interactive version of this graph can be found at http://phototour.cs.washington.edu/imagegraphs/Trevi/.***

dotted blue arrows connecting the red point to the center of projection of each camera.

SfM, though still an active research area, is another crowning achievement of computer vision and photogrammetry over the past few decades. While the basics of projective geometry were understood in the Renaissance, our understanding of the mathematical theory underlying multiview geometry has been significantly improved by fundamental research since the late 1980s [18], [30]. Early work focused on reconstructing geometry from two views [42], with more recent work addressing the multiview reconstruction problem [69]–[71]. Unfortunately, the optimization underlying SfM involves a complex, nonlinear objective function with no closed-form solution, due to nonlinearities in perspective geometry. Most modern approaches use nonlinear least squares algorithms to minimize this objective function, a process known as *bundle adjustment* [72].

Until recently, most work on the multiview SfM problem focused on controlled image sequences, such as frames of a video. However, the advent of more powerful feature matching techniques opened up the possibility of reconstructing geometry from fundamentally *uncontrolled* sets of photos, such as Internet imagery. Prior to our work, Schaffalizsky and Zisserman [58] and Brown and Lowe [5] developed SfM systems that successfully reconstructed unordered image sets captured by a single photographer.

In our work, we showed that SfM could be applied to photos found "in the wild," reconstructing scenes from several large Internet photo collections [66]. To solve the resulting large-scale nonlinear optimization, we reconstruct the scene incrementally, starting from a single pair of images, then adding new images and points in rounds, running a global nonlinear optimization after each round. This incremental process is visualized for the Trevi Fountain photo collection in Fig. 7.[3]

Our system has been remarkably effective in reconstructing geometry from images from the Internet, working on almost every example collection we have downloaded. Several reconstructions are shown in Figs. 1, 2, and 7. The system is also robust to problem cases such as incorrect matches (images which do not match the desired scene; such images are automatically discarded), crowds occluding much of the scene, and low-quality images (up to a point). We have also been able to register historical images, such as the photograph of Half Dome taken by Ansel Adams shown in Fig. 8. Hence, one application of our system is determining the viewpoint from which famous photographs were captured, then taking a new photo from the same viewpoint, a process often known as *rephotography* [57].

The rest of this paper discusses other types of information we can derive once this basic structure is in place, then describes how we can use this information in an interactive scene visualization tool.

III. DENSE SCENE RECONSTRUCTION

Given a set of input images in a community photo collection, SfM is able to reconstruct a sparse geometric model consisting of the 3-D positions of matched image features. While this is sufficient for some applications such as the image-based visualizations presented in Section VI, we might wonder how much more information can be

[3] More details on this algorithm can be found in [65], and code for our SfM system is freely available at http://phototour.cs.washington.edu/bundler/.

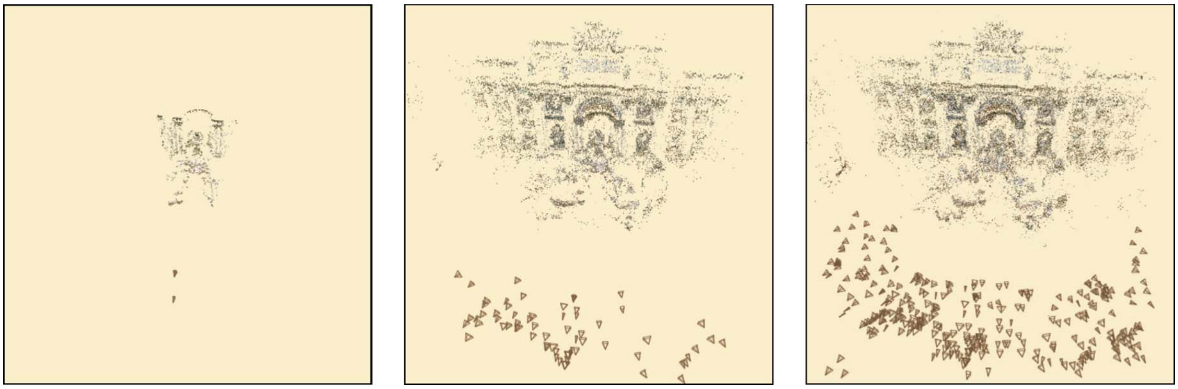

Fig. 7. ***Incremental SfM. Our incremental SfM approach reconstructs the scene a few cameras at a time. This sequence of images shows the Trevi data set at three different stages of incremental reconstruction. Left: the initial two-frame reconstruction. Middle: an intermediate stage, after 60 images have been added. Right: the final reconstruction with 360 photos.***

extracted. In particular, is it possible to reconstruct an accurate and dense geometric model of objects in our world from community photo collections alone? Likewise, can we also reconstruct appearance information such as surface color and the degree of gloss?

For decades, researchers in the vision and graphics community have been developing a wide variety of techniques to capture scene geometry and appearance. Traditionally, the input data for these approaches are images or videos captured specifically for the purpose of reconstruction. In many cases, capture conditions such as camera positions or illumination need to be tightly controlled (e.g., the system of Debevec *et al.*, used to capture the shape and reflectance of the Parthenon [12], [14] required a carefully controlled capture setup). In contrast, images in a community photo collection are typically captured by different photographers under uncontrolled conditions. As illustrated in Figs. 9 and 10, lighting, foreground clutter, and scale can differ substantially from image to image. Thus, one of the major challenges for scene reconstruction from community photo collections is to develop techniques that work even for such general and uncontrolled input data. Note that these techniques enable us not only to make use of the wealth of existing image data on the Web for reconstruction, but can also drastically simplify the requirements for "traditional" capture applications since conditions do not need to be tightly controlled.

In this section, we first discuss an approach to recover *dense geometry* for a scene using multiview stereo techniques. We then describe a technique to capture detailed *appearance* information for these scenes, such as

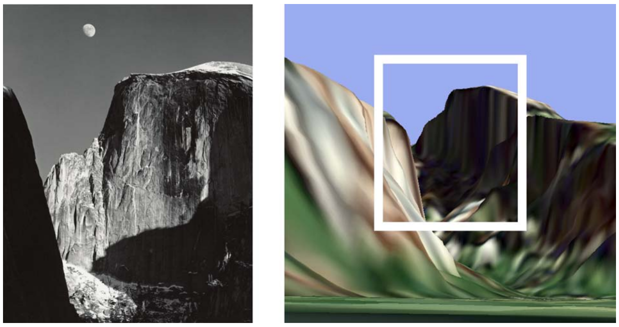

Fig. 8. ***A registered historical photo. Left: Moon and Half Dome, 1960. Photograph by Ansel Adams. We registered this historical photo to our Half Dome model. Right: rendering of DEM data for Half Dome from where Ansel Adams was standing, as estimated by our system. The white border was drawn manually for clarity. (DEM and color texture courtesy of the U.S. Geological Survey.)***

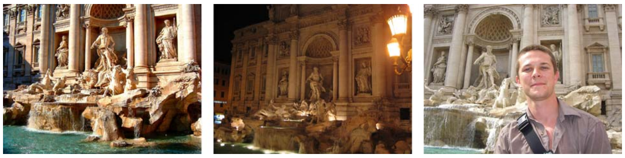

Fig. 9. ***Community photo collection consisting of images of the Trevi Fountain collected from the Internet. Varying illumination and camera response yield strong appearance variations. In addition, images often contain clutter, such as the tourist in the rightmost image, that varies significantly from image to image.***

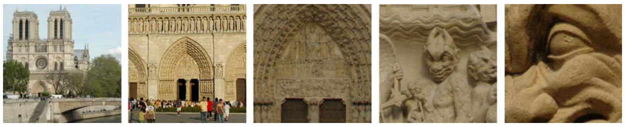

Fig. 10. ***Images of Notre Dame with drastically different sampling rates. All images are shown at native resolution, cropped to a size of 200 × 200 pixels to demonstrate a variation in sampling rate of more than three orders of magnitude.***

color and degree of gloss, again using images from community photo collections.

A. Multiview Stereo Reconstruction

Multiview stereo (MVS) techniques take as input a set of images with known camera parameters (position and orientation of the camera, focal length, image distortion parameters). These parameters can be determined in a separate calibration step or reconstructed using the SfM system described in Section II. Given the camera parameters for an image, we can compute a viewing ray per pixel, i.e., a ray in space containing all 3-D scene points that project to this pixel. The distance of the visible scene point to the camera along the viewing ray (and therefore also its 3-D position) is however not yet known. The goal of MVS is to find these distances, or *depths* for each pixel, and to use this information to construct a dense 3-D object model. See [60] for a classification and evaluation of recent MVS techniques.

As illustrated in Fig. 11, each depth along a viewing ray in one image yields a different projected location in the other images. Thus, finding the correct depth can be posed as a matching problem (similar to the correspondence problem described in Section II), where we want to find the depth for which the projected locations in all images look as similar to each other as possible. We therefore search for the depth for which the resulting corresponding patches, small regions in the images around the projected locations as shown in Fig. 11 (top), are consistent. For traditional data sets captured under laboratory conditions such as the temple data set [60] shown in Fig. 11, this matching problem can be challenging, e.g., in areas of low texture where matches are not unique or in case of occlusions where corresponding areas may not be visible in all images. Matching becomes even more difficult for community photo collections, which exhibit tremendous variation in appearance and viewing parameters, as they are acquired by an assortment of cameras at different times of day and in various weather.

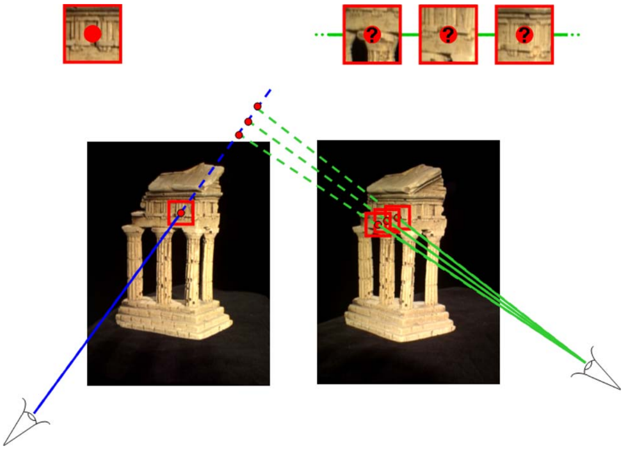

Fig. 11. ***The stereo principle. Depending on the depth along the viewing ray in the left image, a pixel and the surrounding image patch are projected onto different locations in other images. For the correct depth, the patch and its projections in other images are consistent barring occlusions or other unmodeled effects. Multiview stereo approaches follow the same principle using more than two images. Images are taken from a standard benchmark data set [60].***

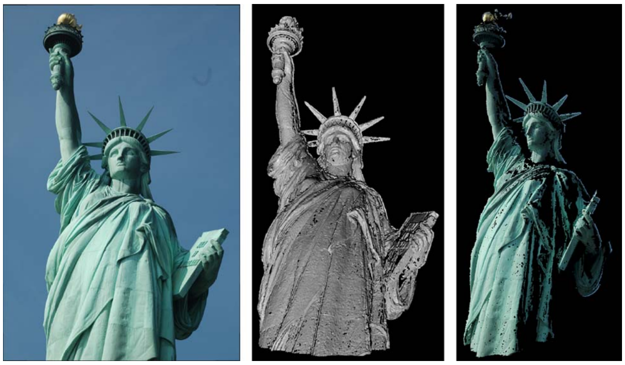

Fig. 12. ***A view of the Statue of Liberty and the corresponding depth map. Left: original input image from a community photo collection. Center: reconstructed depth map rendered from the original camera viewpoint. Right: the same depth map textured with the input image and rendered from a novel viewpoint.***

We introduced a novel MVS approach [22] that is based on the following observation about large community photo collections: given the massive numbers of images available online, there should be large subsets of images of any particular site that are captured under compatible lighting, weather, and exposure conditions, as well as sufficiently similar resolutions and large enough distance between the cameras (just as with our own two eyes, two cameras need to be separated by some distance to yield good information about depth through *parallax*, i.e., differing displacements of objects at different depths). By automatically identifying such subsets, we can dramatically simplify the matching problem, by using images that are similar in appearance and scale while providing enough parallax for accurate reconstruction.

Our system incorporates this *view selection* idea into an existing MVS approach by Otto and Chau [52] and reconstructs a *depth map*, i.e., an image with depth values per pixel, for each image in the community photo collection. Pixels that cannot be matched reliably are left undefined. Fig. 12 shows an example view from a community photo collection of the Statue of Liberty and the reconstructed depth map from two different viewpoints.

The reconstructed depth maps for all the different images can be combined into a single consistent 3-D model using standard geometry processing techniques (see [22] for details). This fusion of depth maps yields two major advantages. First, the depth maps typically contain varying noise and outliers due to the difficulty of the matching

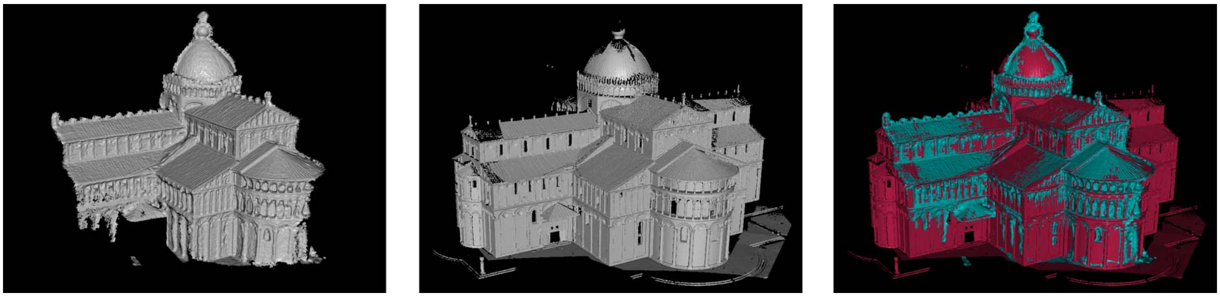

Fig. 13. *Comparison of a reconstructed geometry model with ground truth. Left: merged model of the Duomo in Pisa reconstructed from a community photo collection with 56 images. Middle: laser scanned model of the Duomo serving as ground truth. Right: false color rendering of the two registered models overlaid on top of each other demonstrating that the two models match closely. (Scanned Duomo model courtesy of Visual Computing Group, CNR, Pisa, Italy.)*

problem discussed above. Combining multiple depth maps can reduce these artifacts and therefore improves the *quality* of the reconstructed surface. Second, the input images in a community photo collection depict the scene from different perspectives and under different conditions (e.g., lighting and scene clutter as shown in Fig. 9). Consequently, the reconstructed depth maps typically contain different parts of the scene so that combining them also improves the *completeness* of the reconstructed model. Fig. 13 shows a merged 3-D model of the Duomo in Pisa reconstructed from a community photo collection with 56 images. The model in the center was created from several 3-D scans acquired with a laser scanner and serves as ground truth. As is evident from the rightmost image, the two models match closely. The MVS reconstruction is within about 0.25% of the laser scanned model [22] using the accuracy metric of [60].

B. Relighting Objects From Image Collections

The appearance of an object is characterized not only by its geometry, but also by its *reflectance*, i.e., the way in which it interacts with light. Different materials, such as stone, bronze, wood, and plastic, all reflect light in different ways, giving each material a characteristic look. Some materials, such as chalk, are *diffuse* and reflect incoming light equally in all directions. The appearance of such diffuse materials does not change as we look at the material from different directions. Most opaque objects, however, are not completely diffuse, and reflect light in a more complex way that depends on our viewpoint.[4] For instance, many materials are *glossy*, resulting in highlights that appear to move on the surface as we view the material from different directions. Gloss itself is not simple, and can have different characteristics for different materials (as can be observed when one goes to buy paint for a room, and has to choose between eggshell, satin, semigloss, high gloss, and other types of paint). Fig. 14 shows an example of how different materials interact with light.

In order to fully capture the appearance of an object and enable rendering of that object from any viewpoint, we need to recover reflectance properties for every point on the surface of the object. There are many techniques for acquiring reflectance in the lab, using specialized equipment and controlled lighting including [13], [38], [46], and [80]. However, it is not possible to take objects like the Statue of Liberty or the Notre Dame Cathedral into the lab. Other techniques that work in outdoor settings need additional information about the lighting during capture [12], [14], [78]. Can we recover material properties solely from community photo collections where lighting is neither known nor controlled?

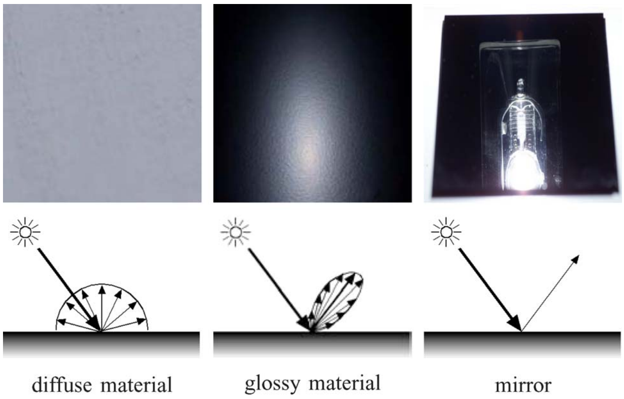

Fig. 14. *Reflectance properties for three different materials. Here, samples of different materials are placed in the scene, and photographed under illumination from a bright light source. The camera is close to the mirror direction of the light source. The left image shows a diffuse surface, which scatters incoming light equally in all directions. The middle image is of a glossy material that reflects most of the light in and close to the mirror direction forming a broad highlight. The right image shows a standard mirror that reflects all light in the mirror direction—note that we get an image of the light source. (Images and drawings courtesy of Hendrik Lensch [37].)*

[4] In fact, transparent or translucent objects interact with light in even more complex ways, as light rays can enter the object at one point, bounce around inside, and exit at a different point. In our work, we only consider opaque objects.

In [27], we approach this question by developing a technique to recover reflectance and illumination from community photo collections alone. The output of this technique is a model where we can change both the lighting and the viewpoint, and get back a realistic image of the object. Our approach relies on a number of assumptions about the scene and about the lighting. For instance, we assume that all light sources are very far away, which is true for the sun and the sky, but not for floodlights or light rays bouncing between objects in the scene. We also assume that each material present in a scene can be described as a linear combination of a small number of basis materials (see [27] for complete details of our assumptions). While there is a great deal of work left to be done on this problem, our work takes a step towards enabling reflectance estimation solely from uncontrolled images gathered "in the wild."

The idea behind the approach is straightforward and uses a technique known in the computer graphics literature as *inverse rendering* [4], [45], [56], [77]. We treat each image in the community photo collection as a *measurement* of what the scene looks like from a specific point of view (the known camera position) and under the (unknown) illumination for that image. Our goal is to determine a complete scene model, including the scene geometry (recovered using MVS as described in the previous section), the reflectance at each surface point, and the illumination present in each image, that closely predicts the measurements. Given an estimate of these unknowns, we can render the scene, i.e., create an image of what the scene model would look like with the estimated material properties and under the estimated illumination. If these estimates are correct, each rendered image should match the corresponding real image. For efficiency reasons, we perform this rendering step in the wavelet domain [49], [74] exploiting the structure of natural images and natural lighting environments. Comparing the real and rendered images, we can update the estimates of the unknown reflectance and illumination conditions and iterate until convergence.

Fig. 15 illustrates an example for the Statue of Liberty using the six input images in Fig. 15(a). After estimating the reflectance and per-image illumination, we can create high-quality renderings using the estimated illumination, as shown in Fig. 15(b). Fig. 15(c) shows the Statue of Liberty rendered under two novel illumination conditions—a single spot light (left two images) and a measured illumination environment [11] (right three images). The latter are renderings of the statue as it would appear if miniaturized and placed in the Uffizi Gallery, Florence, Italy.

IV. SCENE SUMMARIZATION AND SEGMENTATION

A Flickr search for Trafalgar Square results in nearly 150 000 photos, presented as a list of thumbnails in seemingly random order. Can we use a reconstruction of such a set of photos to construct an artifact that is easier to digest? Our goal in scene summarization is to condense a photo collection into a smaller set of *canonical views* that quickly conveys the essence of the full collection. A good summary should contain views that appear frequently in the collection, without much redundancy. (See Fig. 16 for a summary of Rome computed automatically by our algorithm.) A summary can also be interactive, allowing a user to browse the scene at a high level before "drilling down" into specific views.

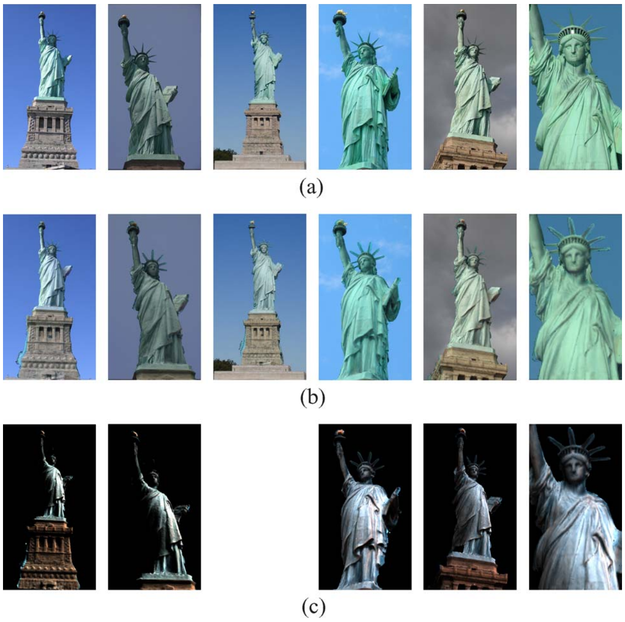

Fig. 15. ***Relighting a model of the Statue of Liberty. (a) Input images for the Liberty data set; (b) object rendered under estimated illumination; (c) object rendered under point light illumination (left two images) and under the Uffizi environment map [11] (right three images).***

Earlier summarization approaches did not use computer vision, and relied solely on metadata associated with the images, such as timestamps, textual tags, and geotags [2], [34]. Unfortunately, user-supplied metadata is often absent or unreliable, leading to the use of image analysis (and the resulting scene reconstructions) as the basis for our summarization technique. (A subsequent user study by Kennedy and Naaman [35] found that using visual feature matching led to summaries that users found 39% more "representative" and 22% more "satisfying" than summaries computed from tags alone.) The success of our vision-based method depends on a simple assumption that the distribution of photos taken by users and submitted to Flickr reflects the relative importance of different views of a scene.

Our approach to scene summarization involves three related subproblems. The first is to partition the image set into groups of images, each corresponding to a different

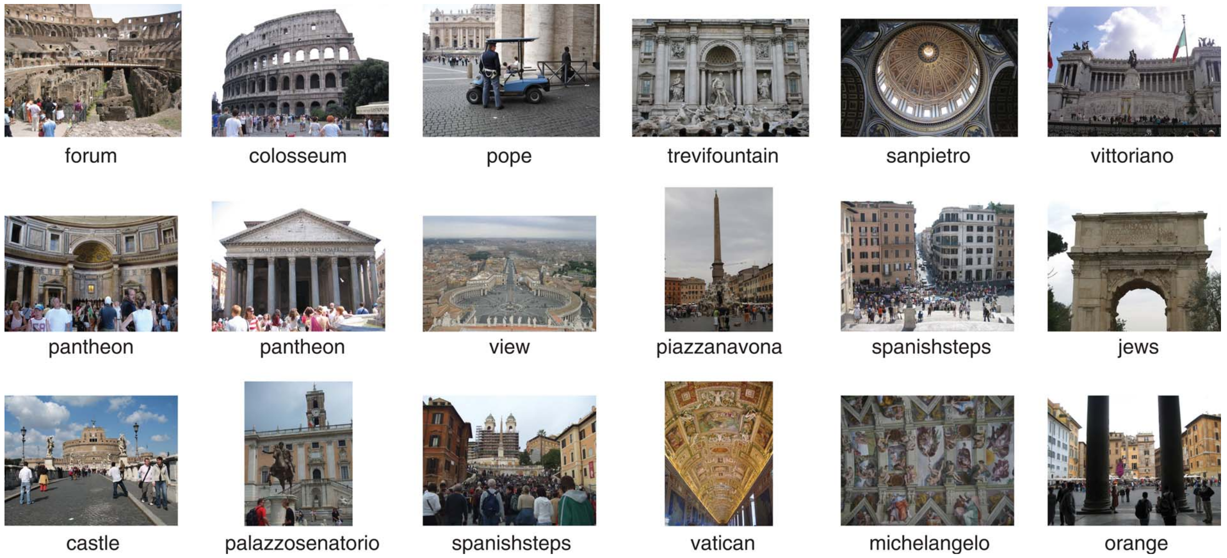

Fig. 16. ***Rome summary. A summary of 20 000 images of Rome, consisting of a representative image from the 18 largest scenes, along with automatically selected tags.***

representative view of that scene. The second is to identify a canonical image to represent each group. The third is to compute textual tag information that best represents each view. Computing a city summary further requires identifying all of the distinct sites of interest in that city.

At a technical level, our approach works by applying clustering techniques to partition the image set into groups of related images, based on the visual structure represented in the image connectivity graph for the collection (such as the one shown in Fig. 6). The clustering and representative images are computed using a greedy approximation to a *k*-means-like objective that is guaranteed to achieve a solution with quality above a constant fraction of the optimal solution. The resulting clustering can also be used to browse the collection of images by content. Descriptive textual tags for each cluster are computed using probabilistic reasoning on histograms of image-tag co-occurrences. A good tag should appear in many images within the cluster (preferably from multiple users) and less frequently outside the cluster. Due to the large amount of noise in user tags, obtaining high-quality tags turns out to be nontrivial. Further details on our scene summarization method can be found in [61].

Can we go further? Instead of clustering the images, is it possible to segment the scene itself into *objects* that people find interesting? It turns out that we can perform such a segmentation using a *field-of-view* cue—photographers are likely to take pictures "of" interesting objects, rather than pictures where multiple objects are partially overlapping with the frame. This means that scene points that appear together in many images are likely to belong to the same object. We can use this cue, along with a spatial proximity cue (nearby scene points are more likely than distant points to belong to the same object), to cast the problem of scene segmentation into a framework similar to latent semantic analysis [15], [32], ordinarily used on word-document co-occurrences in text. With this, we can decompose a sparse set of 3-D scene points and their image co-occurrences into a set of latent objects. This sparse 3-D segmentation can then be used to highlight interesting objects in images (Fig. 17), and in some cases, select good text labels from the pool of noisy image tags from Flickr (Fig. 18). More details are available in [62].

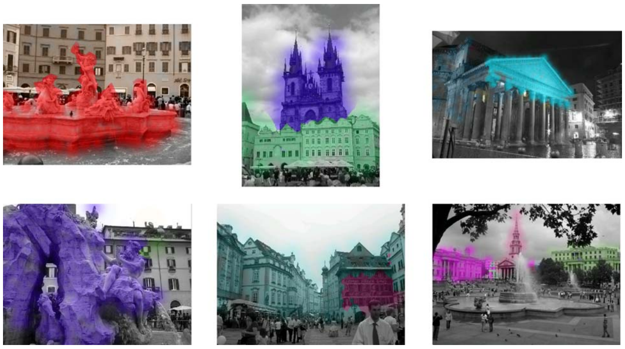

Fig. 17. ***Regions highlighted by importance. These importance images are from several scenes. Cluster importance is a function of the number of images containing scene points from the cluster, as well as the cluster size (to discount large background clusters). Importance is indicated by color saturation, with different hues for different clusters. In other words, the colored regions correspond to parts of the scene that are photographed frequently relative to their size.***

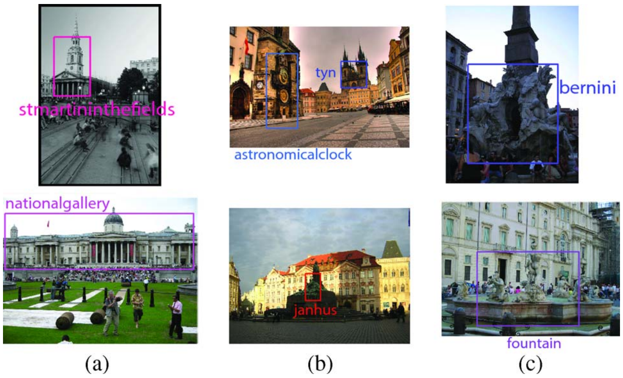

Fig. 18. ***Automatically tagged regions. Region tags automatically computed from our segmentation, and used to label two images of (a) Trafalgar Square, (b) Old Town Square, and (c) Piazza Navona. We compute these labels using noisy user-submitted tags downloaded from Flickr, automatically associating a single tag (or no tags) with each scene point cluster. In these images, we manually moved the labels to make them more readable.***

V. FINDING PATHS THROUGH THE WORLD'S PHOTOS

Section IV describes techniques for identifying interesting, representative viewpoints given a large collection of photos of a scene. In addition to common viewpoints, there tend to be common *paths* through a scene. These paths often correspond to actual paths and walkways, but the way people move through and photograph a scene is also influenced by the content of the scene. For instance, if the scene has a dominant object, people often tend to take photos of that object from a range of surrounding viewpoints. If we look at the spatial distribution of reconstructed photos taken by many different people, we often see "orbits" form around such objects. Fig. 19 shows a set of reconstructed viewpoints from Flickr photos of the Statue of Liberty. Notice how these views form two orbits around the statue, one on the island, and one further out in the water (captured from boats). Even if we knew nothing about the contents of the scene, these viewpoints alone indicate that there is an interesting object in the center (because all the cameras are trained on that single point), and that people tend to photograph that object from two different distances.

These paths are interesting in themselves, as they can be used to infer people's behavior as they move through a scene. In addition, these paths are extremely useful in building a scene-exploration interface for people who have *never* been to that place, as described in Section VI. By identifying consensus views and paths over many photographers, we can potentially learn a set of good navigation controls specific to each scene, i.e., controls that guide new virtual visitors to the interesting objects and paths. For instance, in the Statue of Liberty example, we can detect the two orbits around the statue, and then present these orbits as controls to a user of our 3-D interface. This section briefly describes our work on finding orbits and other paths [67].

In our work, we find three types of paths, in addition to the representative views of the previous section:

- orbits;
- panoramas;
- paths from one representative view, orbit, or panorama to another.

An orbit is an arc of a circle focused on a nodal point (usually the center of an object). A panorama is a set of images all taken near the same location, but looking in different directions. Photos taken at the top of the Leaning Tower of Pisa, for instance, would likely make a good panorama.[5] Finally, it should be easy for a user of a 3-D interface to travel from one good path or viewpoint to another. To enable this, our system computes good paths between any two views. These computed paths attempt to emulate how a person would move through the scene. We now briefly discuss how we find these different types of paths.

A. Orbits and Panoramas

Finding orbits and panoramas in a set of reconstructed camera locations is a form of clustering—we want to identify sets of cameras that lie close to an arc (for an orbit) or to a single point (for a panorama). At the same time, we want the arc (or range of viewing directions, for a panorama) to be large enough to seem meaningful, to avoid detecting many small, spurious paths.

To find good orbits, we first look for evidence of strong orbit centers. A strong orbit center is a point in the scene towards which many different cameras are pointed (in other words, a point which appears in the center of many different photos). We find such candidate orbit centers by clustering the intersection points between rays passing through the centers of pairs of cameras. Regions where many such rays intersect indicate potentially interesting objects.[6] We then look for strong arcs centered on each orbit center, i.e., long arcs that pass close to a large number of camera centers (see [67] for more detail on this process). For the Statue of Liberty, for example (Fig. 19), we detect a single orbit point, and two arcs around this point. Panoramas are detected through a similar process of clustering camera locations to identify good panorama centers.

B. Paths Between Views

We also want to enable transitions from any view to any other view, as a means of automatically moving a user from

[5] Note that we do not refer to a panorama in the traditional sense of a 360° image, often stitched together from several normal images. Instead, by panorama, we simply mean a location where different people have stood and taken photos in different directions. Note that a panorama is a degenerate "path" in terms of changes in camera location, but we define a path to include changes in orientation as well.

[6] A similar procedure was used by Epshtein *et al.* to find salient objects in a scene [17].

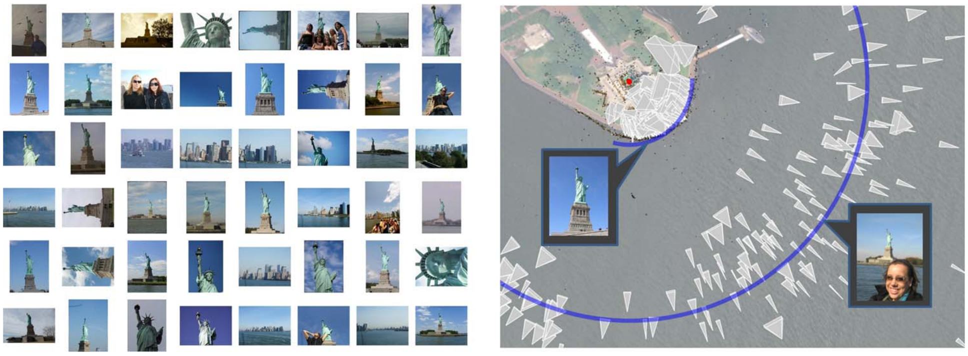

Fig. 19. ***Paths through the Statue of Liberty photo collection. Left: several Flickr images of the Statue of Liberty, from a collection of 388 input photos. Right: reconstructed camera viewpoints for this collection, revealing two clear orbits, shown here superimposed on a satellite view. Our goal is to automatically discover such orbits and other paths through view space to create scene-specific controls for browsing photo collections.***

one part of the scene to another. One approach would be to simply linearly interpolate the starting and ending camera positions and orientations. However, this approach often fails to produce paths that naturally emulate how a person would move through the scene. For instance, Fig. 20 shows an overhead view of a point cloud reconstruction of the Pantheon in Rome. Consider a transition between the green and red views shown in Fig. 20. A linear path (dotted black line) between the two viewpoints passes through the wall of the Pantheon, hence this is an impossible path. How can we generate more natural paths? Our approach is to leverage the fact that we have samples of where people actually stood when they visited the scene. These locations are guaranteed to be free space regions unobstructed by walls and other obstacles. Just as important, they likely lie along common paths through the scene and are taken in interesting viewing directions.

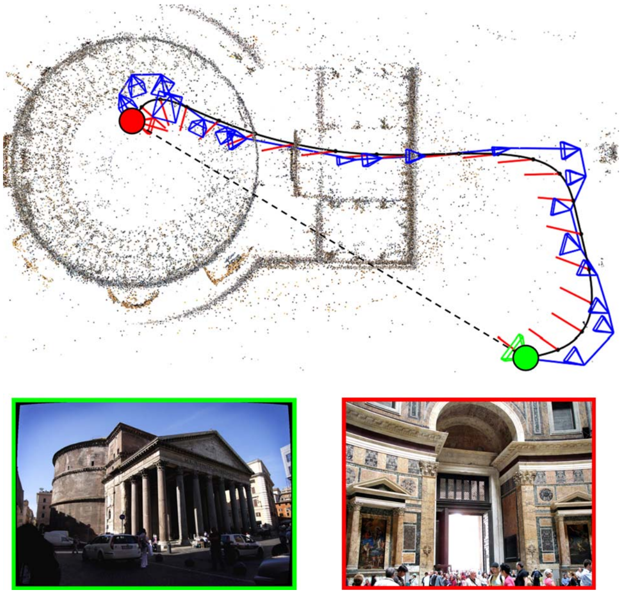

Fig. 20. ***Using path planning to compute a transition between photos. A transition from an image outside the Pantheon (green) to an image inside (red) computed using our path-planning algorithm. The black dotted line shows a linearly interpolated path between these two nodes. The blue cameras are the intermediate nodes visited on the transition graph. The black solid curve shows our smooth path through these cameras. Red line segments indicate the viewing directions at samples along this path.***

Thus, we have developed an algorithm to compute paths through scenes based on two criteria:

- proximity: the path should always be close to the location (and orientation) of an input photo;
- smoothness: the path should be as smooth as possible (to avoid unnatural, zig-zagging paths).

Our algorithm is a variant of Dijkstra's shortest path algorithm [16] on a graph defined on the set of input viewpoints, giving us a sequence of cameras that our path should pass through (the blue cameras shown in Fig. 20). Using a piecewise linear path through these cameras as an initial path, we apply a smoothing filter to generate the final, smooth transition.

VI. SCENE VISUALIZATION

The preceding sections have described several ways of extracting structure from large, unorganized photos of famous places. Now, we discuss how to use this structure to create interactive, immersive 3-D experiences of scenes underlying these photo collections. There are several important components of a 3-D user interface, including:

- **rendering**: how do we display and animate a scene?
- **navigation**: what controls should we give to the user for moving around in a scene?

These two problems have been the focus of a great deal of research in computer graphics and human–computer interaction. The techniques used for these problems depend greatly on the representation of the underlying scene. In computer animation and games, the dominant approach is to manually create detailed 3-D models by hand, then render them using standard 3-D rendering algorithms such as raytracing and *z*-buffering. Unfortunately, these models often require substantial time and expertise to create. As we described in Sections II and III, our work is a step towards creating such geometry completely automatically from existing photos on the Web.

In addition to rendering approaches based on dense geometry, we can also render scenes directly from the photos themselves using an approach known as image-based rendering (IBR). IBR techniques use a database of images to synthesize new views of a scene by combining pixels from multiple input images. One of the first IBR systems was the Aspen Moviemap project of Andrew Lippman [41]. This project sought to recreate an experience of the town of Aspen, CO, by capturing omnidirectional images while driving through the streets of the town, then assembling the images into an interactive city walkthrough. This pioneering work laid the groundwork for many other systems, including Google's Streetview [73]. While the Aspen Moviemap always displayed the closest database image to the user's current viewpoint—a form of nearest-neighbor viewpoint interpolation—more recent work in IBR has developed ways to synthesize a continuum of views [7], [24], [39], [47].

Most work in IBR has focused on rendering scenes from databases of dense, fairly uniformly captured images, quite unlike the photos found in community photo collections (one notable exception is the work of Buehler *et al.* on the unstructured lumigraph [6]). In addition, the problem of navigating these scenes has been less well studied than the rendering problem, despite its importance to the user experience. In our work, we have addressed the challenge of rendering scenes from Internet photo collections, and of giving users understandable, intuitive 3-D navigation controls. This section describes how we use recovered geometric and semantic structure for rendering and navigation.[7]

A. Rendering

Given a set of diverse, nonuniformly sampled photos of a scene, how can we display coherent renderings as a user moves around in 3-D? One approach is to simply display the kinds of point clouds or geometric models shown in Figs. 2 and 12. However, these renderings simply do not have the richness inherent in an actual photo (although the appearance modeling work of Section III-B is moving us closer to photorealism). The input photos contain many elements—sky, clouds, water, crowds—that are difficult to model explicitly with geometry and reflectance information. Thus, in order to preserve the character of the photographs, we have mainly explored image-based rendering techniques.

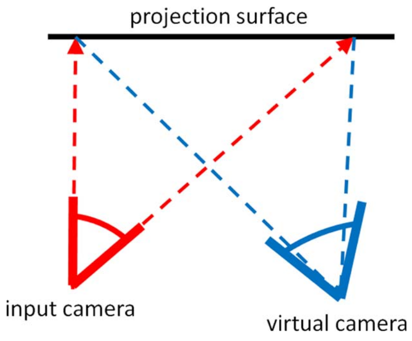

Fig. 21. ***Reprojection. To render the scene from a new viewpoint corresponding to the user's current position and orientation (the "virtual camera" shown in blue), we take an input photo (the red camera on the left) and reproject it into the virtual camera. The reprojection process involves first projecting the photo onto a virtual projection surface in the scene (here, simply a plane indicated by the solid black line), then projecting back into the virtual camera.***

Our basic approach to rendering can be thought of as follows: we treat the input cameras as slide projectors, and treat the scene itself as a projection surface onto which the camera images are projected. To render the scene into a novel view—the user's virtual camera—we project an input view onto the scene, then reproject back into the output viewpoint, as illustrated in Fig. 21. The design of our image-based rendering system requires the answers to two basic questions:

1) **What input image do we project into the output view?** The answer depends on what navigation mode the user is in. In the next section, we describe several different styles of controls for scene navigation. One type of control moves the user from one photo to another based on requests such as *zoom in* and *move left*. In this mode, we project both the starting photo and the destination photo into the user's virtual view during each transition, cross-fading them to produce a *morph*. Other navigation modes allow the user to move freely through the scene (or freely along a constrained path, such as an orbit). For these modes, there is no starting and ending photo, so the system continually selects the best input photo to display based on a photo scoring function. This scoring function takes into account several factors: the proximity of the input photo to the virtual viewpoint, the resolution of the input photo, and how much of the virtual view it covers when reprojected. As the user moves around, different images become the "best" photo to display. For instance, when the user zooms in, the system tends to select a higher resolution photo, and when the user moves, the system selects a photo closer to the new viewpoint. Whenever a new photo is

[7] The interfaces we describe are best shown through video, and examples can be found online at http://phototour.cs.washington.edu/ and http://phototour.cs.washington.edu/findingpaths/.

selected, the system cross fades between the previously selected photo and the new photo.

2) **What geometry do we use for the projection surface?** The natural choice would be the dense reconstructions of Section III. The advantage of using such detailed, accurate models is that the resulting reprojection looks more accurate when seen from a different virtual viewpoint. However, these models also have their disadvantages. Some parts of the scene, such as the sky and the ground, may not have been reconstructed, and thus the projection surface may have large holes. In addition, people and other dynamic elements often appear in Internet photos. Such transient objects will not be represented in the 3-D model, and often look unnatural when projected onto detailed surface geometry.

In contrast, we have found that using an extremely simple projection surface—a single plane per input image—produces good results in many cases, and avoids some of the problems of the more detailed geometry. The use of planar projection surfaces is illustrated in Fig. 21. For each input camera, we select its projection surface by robustly fitting a plane to the 3-D points visible in that camera. This approach results in less accurate reprojections, but looks more natural, perhaps because we are used to seeing planar distortions from everyday life (e.g., whenever we see a billboard from an oblique angle). Moreover, planes are very common in man-made environments, so planar projection surfaces are often good approximations to the true geometry. On the other hand, in scenes that are not well approximated by single planes, this approach does not always work well. Developing better projection surfaces for rendering is still an active research area, with recent work focusing on fitting piecewise planar surfaces to architectural scenes [21], [63].

B. Navigation

In addition to rendering, navigation is a key component of a 3-D user interface. We have developed several new types of photo browsing and scene navigation controls based on the geometric and semantic structure described in the previous sections.

1) Photo-Based Controls: First, the recovered geometric structure enables powerful new *photo-based* controls for exploring scenes and finding new photos. For instance, once we know where each photo was taken, it becomes easy to answer questions such as: Which photos show a wider angle view of the scene than a given photo? Which photos show what's to the left? Which photos are similar? Our system allows for queries based on these types of questions. For instance, Fig. 22 shows, for an example image of the Notre Dame cathedral, the results of using our system to search for all photos which 1) show more detail, 2) show similar parts of the scene, and 3) show more surrounding context.

2) Object Selection: For each photo, we also know which 3-D points are visible, and where they appear in that photo—and we know where these points appear in *other* photos as well. This knowledge enables a new type of *object selection* control. To find a photo of an object, a user can simply select that object in one photo, and the system automatically finds the best photo of the selected object, then moves the user's viewpoint to that photo. When users select a region of a photo, they are implicitly selecting a set of 3-D points as well. Thus, the question of finding a good photo reduces to that of finding a good photo of the selected 3-D points. We define a "good" photo of a set of points to be one that shows most of the points (but is well cropped) and views them from a frontal angle. An example of this search feature is shown in Fig. 23.

3) Suggested Controls: Even more powerful are navigation controls based on the scene summaries and common paths described in Sections IV and V. Building good 3-D scene navigation controls is a surprisingly difficult problem; it is often easy for users to get lost in a scene, especially one they are unfamiliar with [28]. A completely

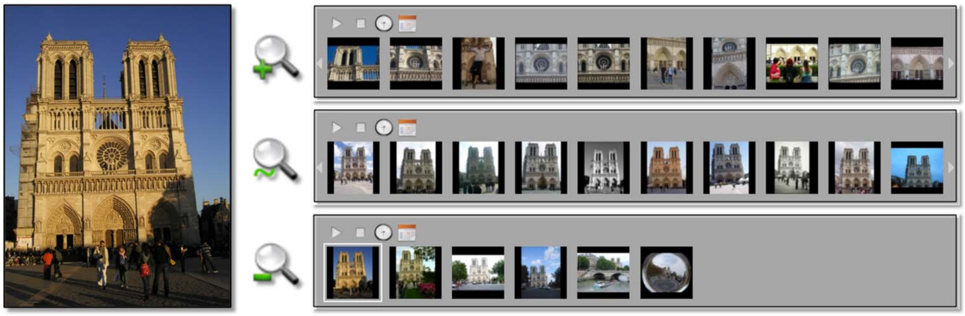

Fig. 22. ***Photo-based controls. Left: an image from a reconstruction of the Notre Dame cathedral. Right: the results of searching for details (top), similar images (middle), and zoom-outs (bottom), starting from this input image.***

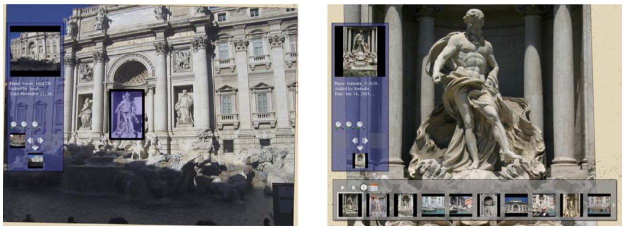

Fig. 23. ***Object selection. Left: the user drags a rectangle around the statue in the current photo. Right: the system finds a new, high-resolution photograph of the selected object, and moves to that view. Additional images of the object are shown in the thumbnail pane at the bottom of the screen.***

free-form navigation interface, such as those common in 3-D games, allows a user to go wherever they choose, but users might not know where they *should* go to see the interesting parts of a scene. For instance, consider Prague's Old Town Square, a well-known tourist site with many notable landmarks. These include the Old Town Hall with its famous astronomical clock, the Tyn Cathedral, the St. Nicholas Church, and the Jan Hus Memorial statue. An ideal interface for exploring this scene would make it easy to find all of the important landmarks, and offer additional controls—such as an orbit control for the statue—when appropriate. These landmarks and controls are precisely the kind that our scene summarization and path-finding algorithms attempt to find based on the consensus of people who have been there before and taken photos.

Our system suggests these detected views and paths to the user as a set of constrained navigation controls for orbiting, panning, and moving to interesting views. These controls are particularly useful for users who are unfamiliar with a scene. The controls instantly inform the user which parts of the scene are worth exploring, and automatically take the user to those places using the path-planning capabilities described in Section V. Fig. 24 shows several suggested controls for the Statue of Liberty and the Pantheon.

C. Additional Applications

The structure we extract from photo collections also enables a number of additional applications. The object selection capability above takes advantage of the fact that we know where each 3-D point appears in each image. This information also allows us to create powerful tools for *image annotation*. In Section IV, we described how we can automatically use Flickr tags to annotate photos and objects. In addition, we can also take an annotated region in one photo and propagate that annotation to all the other photographs through their common 3-D points. Thus, a user can label an entire photo collection by annotating just a single photo. In our system, a user does this by simply dragging a box around a region of a photo and typing in a label. An example of automatic propagation of annotations is shown in Fig. 25.

Our system also allows users to classify photos into different categories, such as day and night. We can then create controls for changing the appearance of a scene by toggling between categories of photos. We created such a control for the reconstructed Trevi Fountain photo collection, giving the user the ability to alternate between daytime and nighttime views of the fountain. An example is shown in Fig. 26.

Another application of our system is to create a 3-D slideshow from a personal photo collection. Suppose that you visit the Pantheon and take your own set of photos, showing friends and family in a few discrete locations (e.g., the front façade, the altar, looking up at the oculus). SfM techniques are not likely to register these photos together due to insufficient overlap, unless they are captured much more frequently than is customary with personal photo collections. However, if these photos are registered with

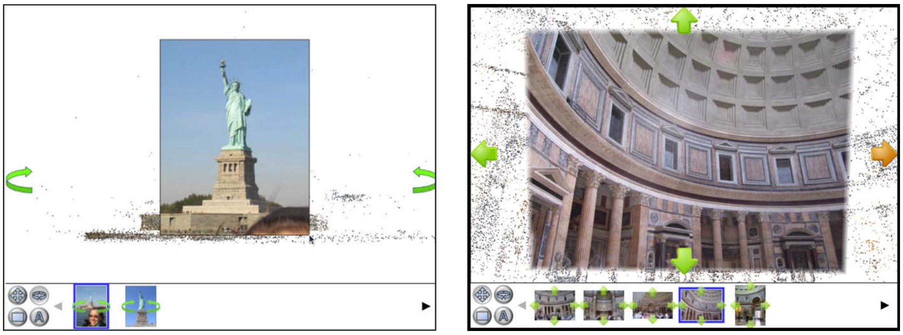

Fig. 24. ***Suggested orbits and panoramas. Two screenshots from our user interface showing suggested controls. The main view of the scene takes up most of the image, and the suggested controls are shown at the bottom. Left: the pane at the bottom shows the two orbits detected for the Statue of Liberty. The user is currently exploring the outer orbit. Green arrows on the sides of the screen indicate that the user can orbit to see additional views. Right: here, the user is at the Pantheon, and the pane at the bottom shows detected panoramas. The user is exploring one such panorama, and again arrows appear to show which directions the user can pan in to see additional views.***

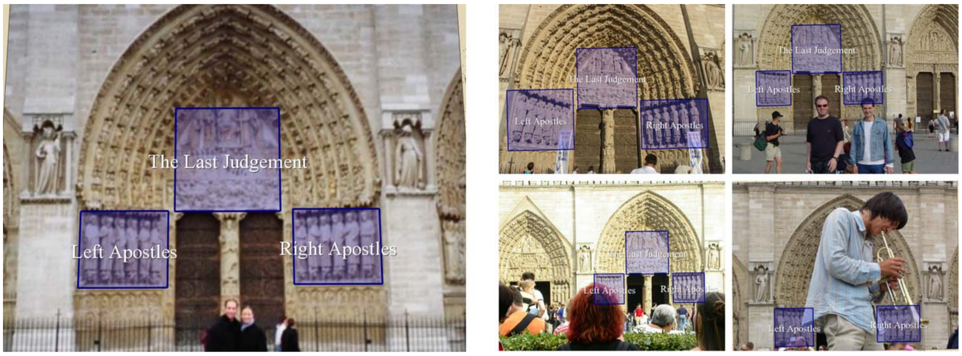

Fig. 25. ***Example of annotation transfer. Three regions were annotated in the photograph on the left; the annotations were automatically transferred to the other photographs, a few of which are shown on the right. Our system can handle partial and full occlusions.***

the much denser set of photos of the Pantheon archived on the Internet, we can plan paths between the personal image set to create a 3-D slideshow—in a sense, retracing our steps through the Pantheon by leveraging other peoples' photos to fill in the gaps. The video on our project website shows such a slideshow created from four personal photos, combined with several hundred community photos of the Pantheon. Fig. 27 shows several images in this personal photo tour in context in our viewer.

D. Photosynth

Our work on reconstruction and visualization of unordered photo collections is being used in Microsoft's Photosynth [54] system. Photosynth, shown in Fig. 28, is a 3-D photo visualization tool that uses the same underlying representation (camera positions and points) as in our work and that supports some similar types of controls, including zooming and orbiting. There are also several interesting differences. In Photosynth, instead of having the user drag a box around an object to see a more detailed photo, the system suggests different photos as the user moves the mouse cursor around the screen. If a closeup of the object that the cursor is hovering over is available, a quadrilateral outline appears, highlighting that region of the screen, as shown in the left image of Fig. 28. Clicking on an outline results in a transition to the selected photo. Photosynth also supports a "splatter" mode, in which the photos are viewed in a 2-D arrangement.

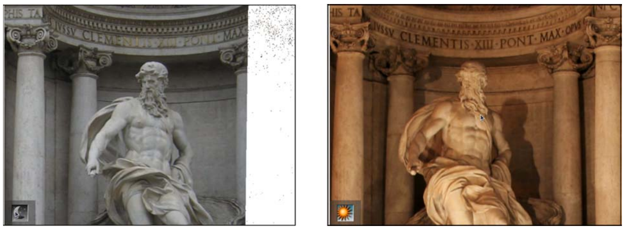

Fig. 26. ***Switching between day and night in the Trevi Fountain. Left: a daytime view of the Trevi Fountain. Right: the result of switching to a nighttime view; the same viewpoint is shown at a different time of day.***

In order to support interactive browsing of large collections of high-resolution photos over a network connection, Photosynth efficiently streams image data using a system called *Seadragon*, which computes which parts of which photos are visible on the screen, and at what resolution each photo (or part of a photo) is viewed. Only the required data are then sent over the network, and higher resolution data are smoothly blended onto the screen as they are received.

To date, enthusiastic Photosynth users have created tens of thousands of Photosynths, some of which, such as those of the Presidential inauguration created by MSNBC photographers, have been viewed over a quarter million times.

VII. CONCLUSION

In this paper, we have described recent progress in digitizing and visualizing our world from data gathered through the ultimate distributed capture process—millions of people simply taking photos and uploading them to the Web. We believe that these tools and models can have significant impact in education, in science, and in everyday life. A virtual, photorealistic model of the world allows anyone with an Internet connection to remotely experience and learn about any location on Earth. Our tools also allow scientists and historians—armed only with a digital camera—to easily create 3-D models of cultural artifacts, archaeological sites, and other places and objects, or to view a progression of registered photographs showing the evolution of a site over time (see, for instance, the 4-D Cities project at Georgia Institute of Technology, which is building a time-varying 3-D model of Atlanta, GA [1], [59]). This ability is especially critical for sensitive sites where it might be difficult to use a laser scanner, or for objects that have been

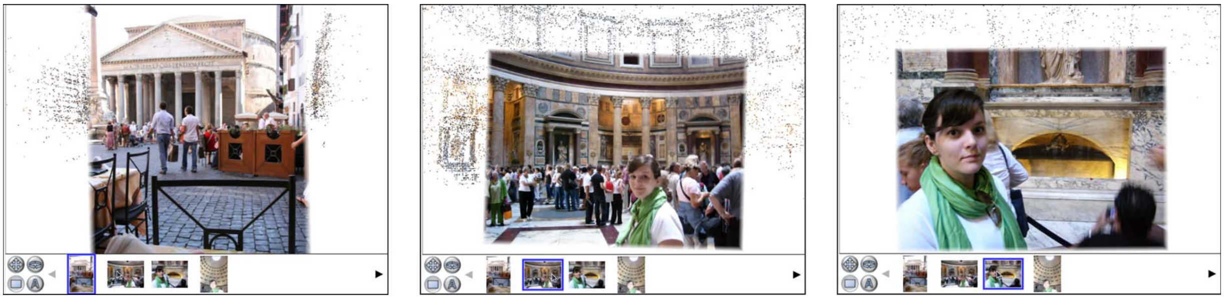

Fig. 27. ***Personal photo tour of the Pantheon. In this sequence, a user has added their own personal collection to the Pantheon data set. This allows the user to create a 3-D slideshow through their own photos, using the community's photos to fill in the gaps (please see the video at http://phototour.cs.washington.edu/findingpaths/ for an animated version of this tour).***

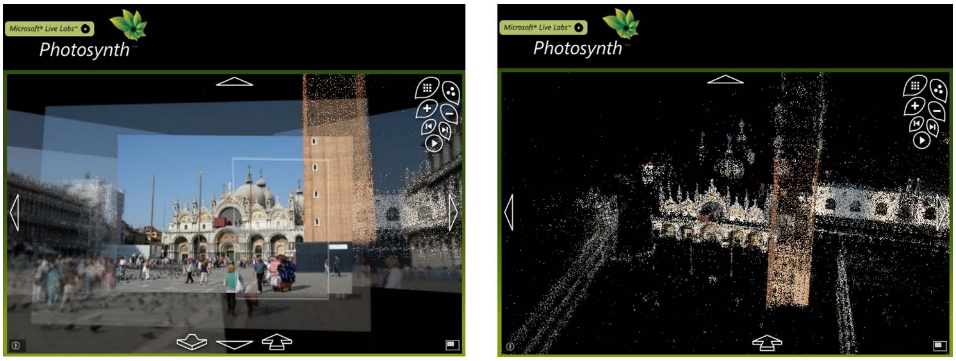

Fig. 28. ***Screenshots from Microsoft's Photosynth. A reconstruction of Venice's Piazza San Marco from 467 photos. Left: the normal display showing one main photo and several other semitransparent photos showing context. Right: a view of the point cloud.***

damaged or destroyed and have only been captured through photography. In the hands of the general public, these tools will let anyone create, explore, and share 3-D models of their home, neighborhood, car, cat, or anything else they might wish to document.

A number of significant challenges still lie ahead. One of the most important is scale. Ultimately, we would like to use our methods to reconstruct as much of the Earth's surface as possible, requiring *Internet-scale* matching and reconstruction algorithms. For many cities, there are *millions* of photos on the Internet. For instance, a search for "Rome" on Flickr results in nearly two million photos. How can we scale to such vast photo collections without requiring many years of running time? Recent matching algorithms [8], [9], [26], [51] have demonstrated the ability to operate effectively on data sets that approach this scale, although more work is needed to improve the accuracy of these methods, especially for community photo collections. Furthermore, the large redundancy in online photo collections means that a small fraction of images may be sufficient to produce high-quality reconstructions, an observation we have begun to explore by extracting image "skeletons" from large collections [68]. Perhaps the most important challenge is to find ways to effectively parallelize all the steps of our reconstruction pipeline to take advantage of multicore architectures and cloud computing [81]–[83].

There has also been interesting recent work in analyzing image collections at Web scale. For instance, Quack *et al.* [55], Crandall *et al.* [10], and Zheng *et al.* [79] used massive collections of georegistered images (35 million in the case of [10]) to find, label, and summarize *all* of the world's significant landmarks, and to trace the most common paths on a city scale [10]. Once these databases have been gathered, they also open up the possibility for *world-scale* matching algorithms—imagine taking a random photo taken anywhere in the world, and localizing it not using GPS, but by matching it to a massive collection of georegistered photographs. Researchers including Hays and Efros [31] and Li *et al.* [40] have begun looking at this problem.

There is also a great deal of work left to be done on reconstructing better geometry and appearance models for scenes. This is especially true for dynamic elements such as clouds, crowds, flowing water in fountains and streams, and weather conditions such as rain and fog. These phenomena are critical for creating a vivid experience of a place, but are traditionally very difficult to model. For this problem, the growing amounts of video data on the Web might help complement the still photos that have been our primary source of data.

While our initial work has focused on what is possible given large Internet collections, rigorous *evaluation* of these methods is also an important direction for future research. How accurate is the geometry we are getting? How good are the scene summaries and object segmentations we compute? How effective are the user interfaces we have developed? These are to some extent open questions, although we have done some initial quantitative evaluation of the accuracy in geometry (e.g., see Fig. 13). However, evaluation is itself a challenging problem for these applications. For 3-D reconstruction, there is usually no accurate ground truth for locations for images found on the Internet, and most places in the world have not been scanned in 3-D; thus, an interesting direction is to gather accurate ground truth for a number of landmarks through laser scanning. For the more subjective applications we explore, such as scene summarization, user studies may be an appropriate means of evaluation.

Finally, there is a large opportunity to connect the visual data on the Web to other types of information, such as Wikipedia pages, landmark databases, weather information, and street maps. The same community of photographers that capture and upload images could also potentially specify contextual information about their photos, especially if they *purposefully* contribute to a growing visual database of the world. Instead of finding photos that are "accidentally" useful for 3-D reconstruction, we envision a site where people intentionally submit photos for the purpose of reconstructing the world. Users could also annotate images and provide links to additional content, resulting in a kind of visual Wiki [33]. Such a site would have the additional benefit of encouraging people to submit *missing* photos that fill in gaps in the model, rather than, say, yet another identical photo of the same famous landmark.

Computer vision as a whole is having increasing success in a number of practical applications. Our hope is that, as the techniques we describe here become more and more robust and scalable, computer vision will have the same impact on the Internet (and the world at large) in the next ten years as text search and retrieval have had in the last ten. ■

Acknowledgment

The authors would like to thank B. Curless and H. Hoppe for their contribution to the multiview stereo work in Section III; T. Haber, C. Fuchs, P. Bekaert, H.-P. Seidel, and H. Lensch for their contribution to the reflectance estimation and relighting work in Section III-B; and R. Garg for his contribution to the path analysis for visualization work in Section V. They would also like to thank K. Chiu and A. Hou for their assistance in the work in Section V. The authors would like to thank all of the generous Flickr users who let them make use of their photos in our research.

REFERENCES

[1] *4D Cities—Spatio-Temporal Reconstruction from Images*. [Online]. Available: http://www-static.cc.gatech.edu/projects/4d-cities/dhtml/index.html

[2] S. Ahern, M. Naaman, R. Nair, and J. H. Yang, "World explorer: Visualizing aggregate data from unstructured text in geo-referenced collections," in *Proc. Conf. Digital Libraries*, 2007, pp. 1–10.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. Assoc. Comput. Mach.*, vol. 45, no. 6, pp. 891–923, 1998.

[4] S. Boivin and A. Gagalowicz, "Image-based rendering of diffuse, specular and glossy surfaces from a single image," in *Proc. SIGGRAPH*, 2001, pp. 107–116.

[5] M. Brown and D. Lowe, "Unsupervised 3D object recognition and reconstruction in unordered datasets," in *Proc. Int. Conf. 3D Imag. Model.*, 2003, pp. 1218–1225.

[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proc. SIGGRAPH*, 2001, pp. 425–432.

[7] S. Chen and L. Williams, "View interpolation for image synthesis," in *Proc. 20th Annu. Conf. Comput. Graph. Interactive Tech.*, 1993, pp. 279–288.

[8] O. Chum, M. Perdoch, and J. Matas, "Geometric min-hashing: Finding a (thick) needle in a haystack," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 17–24.

[9] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *Proc. Int. Conf. Comput. Vis.*, 2007, DOI: 10.1109/ICCV.2007.4408891.

[10] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg, "Mapping the world's photos," in *Proc. Int. World Wide Web Conf.*, 2009, pp. 761–770.

[11] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. SIGGRAPH*, 1998, pp. 189–198.

[12] P. Debevec, "Making 'The Parthenon'," in *Proc. Int. Symp. Virtual Reality Archaeol. Cultural Heritage*, 2005.

[13] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar, "Acquiring the reflectance field of a human face," in *Proc. SIGGRAPH*, 2000, pp. 145–156.

[14] P. Debevec, C. Tchou, A. Gardner, T. Hawkins, C. Poullis, J. Stumpfel, A. Jones, N. Yun, P. Einarsson, T. Lundgren, M. Fajardo, and P. Martinez, "Estimating surface reflectance properties of a complex scene under captured natural illumination," Univ. Southern California Inst. Creative Technol. (USC ICT), Marina del Rey, CA, Tech. Rep. ICT-TR-06.2004, 2004.

[15] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Amer. Soc. Inf. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[16] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[17] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang, "Hierarchical photo organization using geo-relevance," in *Proc. ACM Int. Symp. Adv. Geographic Inf. Syst.*, 2007. [Online]. Available: http://doi.acm.org/10.1145/1341012.1341036

[18] O. Faugeras and Q.-T. Luong, *The Geometry of Multiple Images*. Cambridge, MA: MIT Press, Mar. 2001.

[19] *Flickr*. [Online]. Available: http://www.flickr.com

[20] W. Förstner, "A feature-based correspondence algorithm for image matching," *Int. Arch. Photogrammetry Remote Sens.*, vol. 26, no. 3, pp. 150–166, 1986.

[21] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 80–87.

[22] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *Proc. Int. Conf. Comput. Vis.*, 2007. [Online]. Available: doi.ieeecomputersociety.org/10.1109/ICCV.2007.4408933

[23] *Google Maps*. [Online]. Available: http://maps.google.com

[24] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Proc. SIGGRAPH*, 1996, pp. 43–54.

[25] M*Graphviz—Graph Visualization Software*. [Online]. Available: http://www.graphviz.org/

[26] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Proc. Int. Conf. Comput. Vis.*, 2005, pp. 1458–1465.

[27] T. Haber, C. Fuchs, P. Bekaert, H.-P. Seidel, M. Goesele, and H. P. A. Lensch, "Relighting objects from image collections," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 627–634.

[28] C. Hand, "A survey of 3D interaction techniques," *Comput. Graph. Forum*, vol. 16, no. 5, pp. 269–281, 1997.

[29] C. Harris and M. J. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, 1988, pp. 147–152.

[30] R. I. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge, U.K.: Cambridge Univ. Press, Sep. 2000.

[31] J. Hays and A. A. Efros, "IM2GPS: Estimating geographic information from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, DOI: 10.1109/CVPR.2008.4587784.

[32] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. Uncertainty Artif. Intell.*, Stockholm, Sweden, 1999, pp. 289–296.

[33] A. Irschara, C. Zach, and H. Bischof, "Towards Wiki-based dense city modeling," in *Proc. Workshop Virtual Represent. Model. Large-Scale Environ.*, 2007, DOI: 10.1109/ICCV.2007.4409216.

[34] A. Jaffe, M. Naaman, T. Tassa, and M. Davis, "Generating summaries for large collections of geo-referenced photographs," in *Proc. Int. Conf. World Wide Web*, 2006, pp. 853–854.

[35] L. Kennedy and M. Naaman, "Generating diverse and representative image search results for landmarks," in *Proc. Int. World Wide Web Conf.*, 2008, pp. 297–306.

[36] E. Kruppa, "Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung," *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw. Kl., Abt. Ila.*, vol. 122, pp. 1939–1948, 1913.

[37] H. P. A. Lensch, "Efficient, image-based appearance acquisition of real-world objects," Ph.D. dissertation, Dept. Comput. Sci., Saarland Univ., Saarbrücken, Germany, 2004.

[38] H. P. A. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel, "Image-based reconstruction of spatial appearance and geometric detail," *ACM Trans. Graphics*, vol. 22, no. 2, pp. 234–257, 2003.

[39] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 1996, pp. 31–42.

[40] Y. Li, D. Crandall, and D. P. Huttenlocher, "Landmark classification in large-scale image collections," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2–9.

[41] A. Lippman, "Movie maps: An application of the optical videodisc to computer graphics," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 1980, pp. 32–43.

[42] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, pp. 133–135, 1981.

[43] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[44] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application in stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[45] S. Marschner, "Inverse rendering for computer graphics," Ph.D. dissertation, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 1998.

[46] S. Marschner, S. Westin, E. Lafortune, K. Torrance, and D. Greenberg, "Image-based BRDF measurement including human skin," in *Proc. Eurographics Rendering Workshop*, 2000, pp. 139–152.

[47] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 1995, pp. 39–46.

[48] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. J. Van Gool, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1–2, pp. 43–72, Nov. 2005.

[49] R. Ng, R. Ramamoorthi, and P. Hanrahan, "Triple product wavelet integrals for all-frequency relighting," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 2004, pp. 477–487.

[50] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2118–2125.

[51] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2118–2125.

[52] G. P. Otto and T. K. W. Chau, "'Region-growing' algorithm for matching of terrain images," *Image Vis. Comput.*, vol. 7, no. 2, pp. 83–94, 1989.

[53] *Photobucket*. [Online]. Available: http://photobucket.com/

[54] *Microsoft's Photosynth*. [Online]. Available: http://photosynth.net

[55] T. Quack, B. Leibe, and L. Van Gool, "World-scale mining of objects and events from community photo collections," in *Proc. Int. Conf. Content-Based Image Video Retrieval*, 2008, pp. 47–56.

[56] R. Ramamoorthi and P. Hanrahan, "A signal-processing framework for inverse rendering," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 2001, pp. 117–128.

[57] *Rephotography*. [Online]. Available: http://en.wikipedia.org/wiki/Rephotography

[58] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or 'How do I organize my holiday snaps?'," in *Proc. Eur. Conf. Comput. Vis.*, 2002, vol. I, pp. 414–431.

[59] G. Schindler, F. Dellaert, and S. B. Kang, "Inferring temporal order of images from 3D structure," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, DOI: 10.1109/CVPR.2007.383088.

[60] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, vol. 1, pp. 519–526.

[61] I. Simon, N. Snavely, and S. M. Seitz, "Scene summarization for online image collections," in *Proc. Int. Conf. Comput. Vis.*, 2007, DOI: 10.1109/ICCV.2007.4408863.

[62] I. Simon and S. M. Seitz, "Scene segmentation using the wisdom of crowds," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 541–553.

[63] S. N. Sinha, D. Steedly, and R. Szeliski, "Piecewise planar stereo for image-based rendering," in *Proc. Int. Conf. Comput. Vis.*, 2009.

[64] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 591–606, Apr. 2009.

[65] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from Internet photo collections," *Int. J. Comput. Vis.*, vol. 80, no. 2, pp. 189–210, Nov. 2008.

[66] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 2006, pp. 835–846.

[67] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski, "Finding paths through the world's photos," in *Proc. Annu. Conf. Comput. Graph. Interactive Tech.*, 2008, article 15.

[68] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal graphs for efficient structure from motion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, DOI: 10.1109/CVPR.2008.4587678.

[69] M. E. Spetsakis and J. Y. Aloimonos, "A multiframe approach to visual motion perception," *Int. J. Comput. Vis.*, vol. 6, no. 3, pp. 245–255, Aug. 1991.

[70] R. Szeliski and S. B. Kang, "Recovering 3D shape and motion from image streams using nonlinear least squares," *J. Visual Commun. Image Represent.*, vol. 5, no. 1, pp. 10–28, Mar. 1994.

[71] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: A factorization method," *Int. J. Comput. Vis.*, vol. 9, no. 2, pp. 137–154, Nov. 1992.

[72] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—A modern synthesis," in *Proc. Int. Workshop Vis. Algorithms*, Sep. 1999, pp. 298–372.

[73] L. Vincent, "Taking online maps down to street level," *IEEE Computer*, vol. 40, no. 12, pp. 118–120, Dec. 2007.

[74] R. Wang, R. Ng, D. Luebke, and G. Humphreys, "Efficient wavelet rotation for environment map rendering," in *Proc. Eurograhics Symp. Rendering*, 2006, pp. 173–182.

[75] *Wikipedia*. [Online]. Available: http://www.wikipedia.org

[76] *Windows Bing Maps*. [Online]. Available: http://maps.bing.com

[77] T. Yu, H. Wang, N. Ahuja, and W.-C. Chen, "Sparse lumigraph relighting by illumination and reflectance estimation from multi-view images," in *Proc. Eurographics Symp. Rendering*, 2006, pp. 41–50.

[78] Y. Yu and J. Malik, "Recovering photometric properties of architectural scenes from photographs," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Tech.*, 1998, pp. 207–217.

[79] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven, "Tour the world: Building a web-scale landmark recognition engine," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1085–1092.

[80] T. Zickler, R. Ramamoorthi, S. Enrique, and P. Belhumeur, "Reflectance sharing: Predicting appearance from a sparse set of

images of a known shape," *Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1287–1302, Aug. 2006.

[81] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a day," in *Proc. 12th IEEE Int. Conf. Comput. Vis.*, Kyoto, Japan, Sep. 2009, pp. 72–79. [Online]. Available: http://grail.cs.washington.edu/rome/

[82] S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing Rome," *Computer*, vol. 43, no. 6, pp. 40–47, Jun. 2010. [Online]. Available: http://grail.cs.washington.edu/rome/

[83] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards Internet-scale multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, Jun. 2010.

ABOUT THE AUTHORS

Noah Snavely (Member, IEEE) received the B.S. degree in computer science and mathematics from the University of Arizona, Tucson, in 2003, and the Ph.D. degree in computer science and engineering from the University of Washington, Seattle, in 2008.

While at the University of Washington, his research was funded by a National Science Foundation Graduate Research Fellowship and a Microsoft Live Labs Ph.D. Fellowship. Since 2009, he has worked as an Assistant professor in Computer Science at Cornell University, Ithaca, NY. He works in the areas of computer vision and graphics, in particular, in applying computer vision techniques to the Internet for modeling and visualization of our world.

Ian Simon received the B.Sc. degree in computer science from Washington University in St. Louis, St. Louis, MO, in 2001. Currently, he is working towards the Ph.D. degree in computer science and engineering, advised by Dr. S. Seitz, at the University of Washington, Seattle.

He is currently a member of the Graphics and Imaging Laboratory (GRAIL), University of Washington, working in the area of computer vision. His current research is focused on the extraction of high-level information from Internet image collections. Some of his other research interests include machine learning and music synthesis, and he is one of the creators of the infamous Microsoft Research Songsmith.

Michael Goesele (Member, IEEE) studied computer science at Ulm University and the University of North Carolina at Chapel Hill. He received the Dipl.-Inf. degree from Ulm University, Ulm, Germany, in 1999 and the Dr.-Ing. degree from Saarland University, Saarbrücken, Germany, and Max-Planck-Institut für Informatik, Saarbrücken, Germany, in 2004.

In 2005, he received a Feodor Lynen-Fellowship from the Alexander von Humboldt-Foundation to work as a Postdoctoral Researcher at the University of Washington, Seattle. Since 2007, he has been an Assistant Professor of Computer Graphics at Technische Universität Darmstadt, Darmstadt, Germany. Since 2009, he has been additionally leading an Emmy Noether Research group funded by the German National Science Foundation (DFG). His research interests include capturing and modeling techniques for graphics and vision as well as high-performance computing on modern massively parallel hardware.

Dr. Goesele received several awards including the Dr. Eduard Martin-Preis in 2005 and the Eurographics 2008 Young Researcher Award.

Richard Szeliski (Fellow, IEEE) received the B.Eng. (honors) degree in electrical engineering from McGill University, Montreal, QC, Canada, in 1979 and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in 1988.

Since 1995, he has worked at Microsoft Research, Redmond, WA, where he is currently a Principal Researcher and Head of the Interactive Visual Media Group. He is also an Affiliate Professor at the University of Washington, Seattle. Prior to Microsoft, he worked at Bell-Northern Research, Schlumberger Palo Alto Research, the Artificial Intelligence Center of SRI International, and the Cambridge Research Lab of Digital Equipment Corporation. He pioneered the field of Bayesian methods for computer vision, as well as image-based modeling, image-based rendering, and computational photography, which lie at the intersection of computer vision and computer graphics. He has published over 150 research papers in computer vision, computer graphics, medical imaging, neural nets, and numerical analysis, as well as the book *Bayesian Modeling of Uncertainty in Low-Level Vision* (Norwell, MA: Kluwer, 2007) and the forthcoming book *Computer Vision: Algorithms and Applications* (New York: Springer-Verlag, 2010).

Dr. Szeliski is a Fellow of the Association for Computing Machinery (ACM). He was a Program Committee Chair for the 2001 International Conference on Computer Vision (ICCV) and the 1999 Vision Algorithms Workshop, served as an Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and on the Editorial Board of the *International Journal of Computer Vision*, and is a Founding Editor of *Foundations and Trends in Computer Graphics and Vision*.

Steven M. Seitz (Senior Member, IEEE) received the B.A. degree in computer science and mathematics from the University of California, Berkeley, in 1991 and the Ph.D. degree in computer sciences from the University of Wisconsin, Madison, in 1997.

He is a Professor in the Department of Computer Science and Engineering, University of Washington, Seattle, and also directs an imaging group at Google's Seattle office. Following his doctoral work, he spent one year visiting the Vision Technology Group, Microsoft Research, and subsequently two years as an Assistant Professor in the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. He joined the faculty at the University of Washington in July 2000. He is interested in problems in computer vision and computer graphics. His current research focuses on 3-D modeling and visualization from large photo collections.

Dr. Seitz was twice awarded the David Marr Prize for the best paper at the International Conference of Computer Vision, and has received a National Science Foundation (NSF) Career Award, an Office of Naval Research (ONR) Young Investigator Award, and an Alfred P. Sloan Fellowship. His work on Photo Tourism (joint with N. Snavely and R. Szeliski) formed the basis of Microsoft's Photosynth technology.