RECONSTRUCTING ROME

Sameer Agarwal and Yasutaka Furukawa, *Google*
Noah Snavely, *Cornell University*
Brian Curless, *University of Washington*
Steven M. Seitz, *Google and University of Washington*
Richard Szeliski, *Microsoft Research*

Community photo collections like Flickr offer a rich, ever-growing record of the world around us. New computer vision techniques can use photographs from these collections to rapidly build detailed 3D models.

Amateur photography was once largely a personal endeavor. Traditionally, a person with a camera would capture a moment on film and share it with a small number of friends and family members, perhaps inserting the best images in photo albums and storing a few hundred extra ones in a shoebox. However, the advent of digital photography and the recent growth in photo-sharing websites such as Flickr have brought about a seismic change in photography and the use of photo collections. Today, a photo that we take and share online can potentially be seen by millions of people.

Consequently, 3D city modelers now have access to a vast, ever-growing collection of photographs spanning the entire globe, capturing all of its cities and landmarks innumerable times. For instance, a search for the term "Rome" on Flickr returns more than 2 million photos. This collection represents an increasingly comprehensive photographic record of the city, capturing every popular site, facade, interior, fountain, sculpture, painting, and café. Virtually anything that people find interesting in Rome has been captured from thousands of viewpoints and under myriad illumination and weather conditions. The Trevi Fountain, for example, appears in more than 50,000 photographs on Flickr.

How much of Rome can we reconstruct in 3D from this photo collection? In principle, the images of Rome on Flickr represent an ideal data set for 3D modeling research, as they capture the city's highlights in exquisite detail and from a broad range of viewpoints. However, extracting high-quality 3D models from such a collection is extremely challenging for several reasons. First, the photos are *unstructured*—they're taken in no particular order, and we have no control over the distribution of camera viewpoints. Second, they're *uncalibrated*—thousands of people take the photos, and we know little, if anything, about the camera settings. Third, the *scale* of the problem is enormous—whereas prior 3D modeling methods operated on hundreds or at most a few thousand photos, we need to process collections two to three orders of magnitude larger. Finally, the algorithms must be *fast*—we seek to reconstruct an entire city in a single day, making it possible to repeat the process many times to reconstruct all of the world's significant cultural centers.

Published by the IEEE Computer Society 0018-9162/10/$26.00 © 2010 IEEE

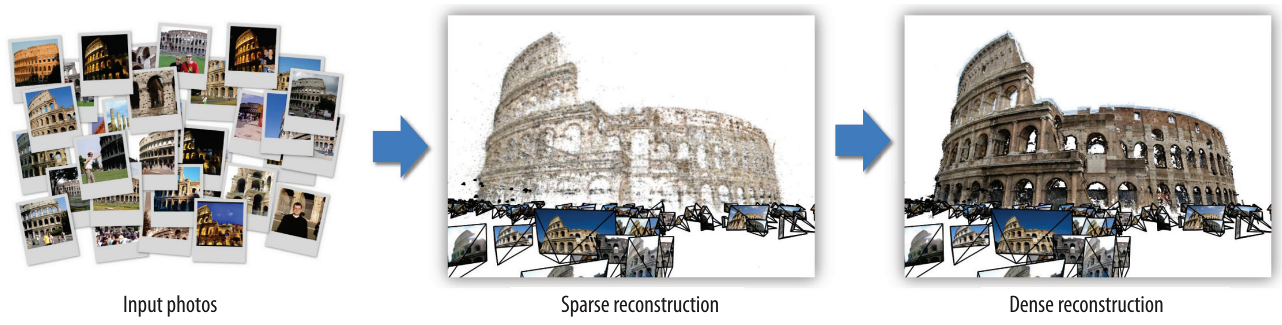


Figure 1. New 3D computer-vision techniques make it possible to create dense 3D city reconstructions from extremely diverse, large, and unconstrained photo collections.

Creating accurate 3D city models is a problem with broad applications. In the government sector, city models are vital for urban planning and visualization. They're equally important in a wide range of academic disciplines, including history, archaeology, geography, and computer graphics research. And digital city models are central to popular consumer mapping and visualization applications such as Google Earth and Bing Maps as well as GPS-enabled navigation systems. In the near future, these models will enable augmented-reality capabilities that recognize and annotate objects in camera phone or other displays. Such capabilities will allow tourists to locate points of interest, obtain driving directions, and situate themselves in the environment.

Computer-vision researchers have explored many approaches to city-scale 3D reconstruction. However, existing large-scale systems operate on data from a structured source—for example, aerial photographs taken by a survey aircraft or street-level imagery captured by a moving vehicle. These systems rely on photos captured using the same calibrated cameras at a regular sampling rate and typically leverage other sensors such as GPS and inertial navigation units, vastly simplifying computation.

Images harvested from the Web have none of these simplifying characteristics. They're taken from many different kinds of cameras under varying lighting conditions, have little to no geographic information associated with them, and often include no camera-calibration data. Thus, a key focus of our research is developing new 3D computer-vision techniques that work "in the wild" on extremely diverse, large, and unconstrained image collections to create dense 3D city reconstructions, as Figure 1 shows.

Our approach to this challenging problem builds on progress made in computer vision in recent years, including our own work on Photo Tourism[1] and Photosynth (http://photosynth.net), and draws from many other areas of computer science including distributed systems, algorithms, information retrieval, and scientific computing. Here we discuss progress in two areas in particular: *structure from motion* (SfM), the problem of inferring camera viewpoints and sparse 3D scene structure from a set of 2D photos, and *multi-view stereo* (MVS), the problem of producing dense 3D geometry from multiple calibrated photos.

CITY-SCALE STRUCTURE FROM MOTION

Suppose we have a 1-Tbyte hard drive full of JPEG files taken by tourists in Rome, with no accompanying information, and must build a 3D model of the city—in a day, if possible. How would we recover 3D geometry from this large collection of images?

The key problem we face is that a photograph is a projection of the world, and in the process of projection we lose a dimension. Inverting this projection is difficult because we've lost the depth of each point in the image. As humans, we can experience this problem by closing one eye and noting our diminished depth perception. Fortunately, most of us have two eyes, and our brains can estimate depth by correlating points between the two images we perceive. This suggests that from multiple photos of a scene, we can recover its shape. If we knew the viewpoints of the cameras used—their relative positions and orientations—then we could recover the depth of corresponding image points through simple triangulation.

Unfortunately, in this case, we don't know a priori the viewpoints of the photographs we've been given, and thus can't directly triangulate points. In other words, both the camera viewpoints and the 3D positions of points in the scene are unknown, and we must solve for both—simultaneously—using image data alone. This classic computer-vision problem, known as structure from motion, is a challenging one, but thanks to key advances during the past 20 years, SfM technology is now mature and widely used, most notably in the film and special-effects industries. Our own work indicates that SfM can also be applied to large, uncontrolled photo collections downloaded from the Internet.

Structure from motion

At first, SfM seems like a chicken-and-egg problem: if we knew the camera viewpoints, we could then triangulate the points to find their 3D positions; conversely, if we

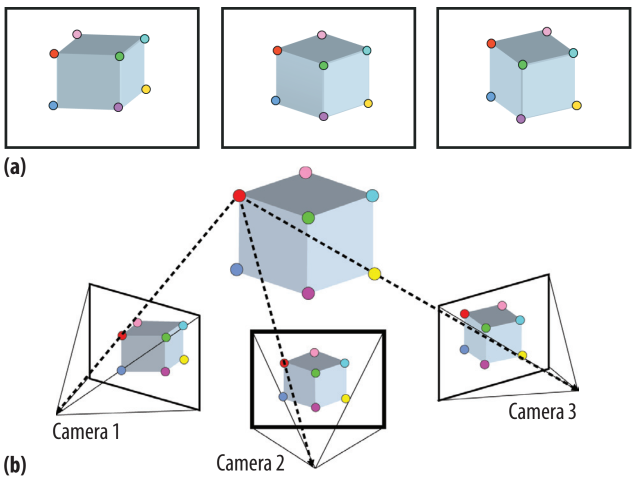


Figure 2. Structure-from-motion (SfM) problem. (a) Three images of a cube from unknown viewpoints. The color-coded dots on the corners show the known correspondence between certain 2D points in these images. Each set of dots of the same color are projections of the same 3D point. (b) A candidate reconstruction of the 3D points (larger-colored points) and cameras for the same image collection. Each image has an associated position and orientation. The reconstruction largely agrees with the observed 2D projections: when the red 3D point is projected into each image (dotted lines), the predicted projection is close to the observed one. In the case of camera 3, the projection is slightly off, resulting in a reprojection error. The middle image in Figure 1 shows a similar visualization but with real data.

knew the 3D positions of a set of points in the scene—for example, a set of fiducial markers carefully placed in the scene in advance—we could then estimate the camera viewpoints from the projections of these 3D points into the photos using surveying techniques. But how can we solve for both at once?

Consider an example. Suppose we have three images of a cube as shown in Figure 2a. We don't know where these images were taken or that they depict a cube. However, we do know that the corners of the cube as seen in the images—the 2D projections of the 3D corners—are in *correspondence*, that is, the 2D dots with the same color correspond to the same 3D points.

This correspondence is a powerful constraint on the 3D geometry of the cameras and points. Whatever the geometry is, we know that it had to have produced this exact set of 2D projections. Thus, we could imagine trying all possible configurations of 3D points and cameras, keeping the one that gives rise to the observed 2D projections through the equations of perspective geometry.

We can formulate this idea as an optimization problem. Figure 2b shows a candidate reconstruction of the 3D points and cameras for the images in Figure 2a. Each image has an associated position and orientation. This reconstruction largely agrees with the observed 2D projections: when the red 3D point is projected into each image, the predicted projection is close to the observed one. In the case of camera 3, the projection is slightly off; the resulting residual is called the *reprojection error*. We seek to minimize the sum of the squares of the reprojection errors. This nonlinear least-squares problem is difficult to solve, as the objective function has many local minima and large scenes have many millions of parameters. We use an incremental approach whereby we initially solve for a small number of cameras and points and then grow the scene a few cameras at a time, pausing after each round to shift around the cameras and points so as to reoptimize the objective function in a process known as *bundle adjustment*.

Correspondence

In the cube example above, we assumed that we were given as input a set of 2D correspondences between the input images. In reality, these correspondences aren't given and also must be estimated from the images. How can we do so automatically?

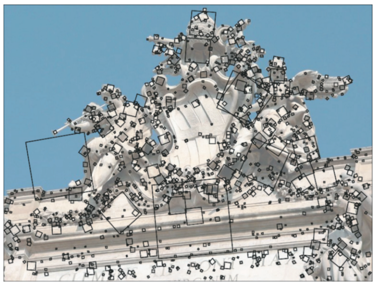
Even for two images, solving this correspondence problem isn't trivial; we must do so for hundreds of thousands of images. In addition, many images may not have any correspondences because they were taken in different parts of the city or even at the same place but looking in different directions. Thus, a subproblem is finding pairs of images that overlap.

A major computer-vision breakthrough in the past decade is the development of algorithms that detect the most distinctive, repeatable features in an image. Such feature detectors not only reduce an image representation to a more manageable size but also produce much more robust features for matching. One of the most successful of these detectors is *scale-invariant feature transform* (SIFT).[2] Figure 3a shows SIFT features for an image of the Trevi Fountain.

Once we detect features in an image, we can match them across image pairs by finding similar-looking features. We then link pairs of matching feature points together to form *tracks* corresponding to the same 3D point in the scene. Figure 3b shows the track for a point on the face of the central statue of Oceanus in Rome. Given these tracks of corresponding points, we can apply SfM techniques.

Large-scale image matching

For image collections with only a few thousand images, comparing all pairs of images is reasonable.[1] However, given an unordered collection of millions of images, and thus trillions of image pairs, solving the matching problem is akin to looking for needles in a very large haystack, as most pairs of images from a given city won't overlap. Thus, instead of comparing every pair of images, we must



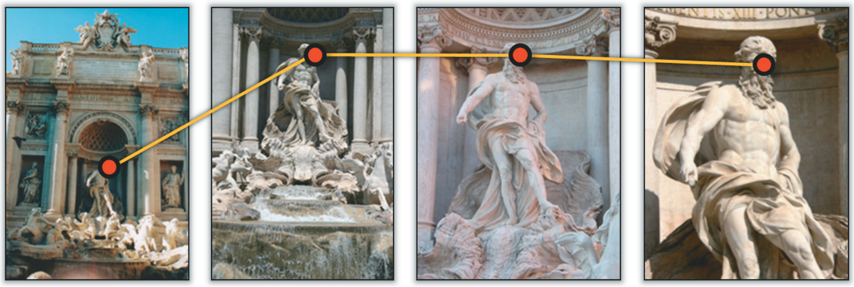


Figure 3. Feature detection and matching. (a) The position and orientation of scale-invariant feature transform (SIFT) features on an image of the Trevi Fountain. (b) A track corresponding to a point on the face of the central statue of Oceanus at the Trevi Fountain, the embodiment of a river encircling the world in Greek mythology.

carefully choose those pairs that the system spends time matching.

The solution to this problem is to devise a method for quickly determining pairs of images that look "similar" and then find corresponding points in these image pairs. How do we find such similar images? If, instead of images we were dealing with textual documents, we could use the text-retrieval methods that lie at the heart of modern Web search engines. These methods represent each document as a vector of weighted word frequencies; it turns out that the distance between two such vectors is a good predictor of the similarity between corresponding documents.

Inspired by this progress in document analysis, computer-vision researchers have recently begun to apply comparable techniques to visual object recognition, with great success. The basic idea is to cluster the SIFT features in a photo collection into "visual words." By treating the images as documents composed of these visual words, we can apply document-retrieval technology to efficiently match large data sets of photos. In particular, we exploit well-known methods like term-frequency analysis and query expansion in our distributed image-matching system, which can process hundreds of thousands of images in less than a day.[3]

Large-scale model reconstruction

While traditional SfM methods work well for collections of images numbering in the hundreds, they become prohibitively expensive in time and space when applied to large, unstructured image collections. Internet photo collections are inherently redundant. Many photographs are taken from nearby viewpoints—for example, the front of the Colosseum—and processing all of them doesn't necessarily add to the reconstruction. It's thus preferable to find and reconstruct a minimal subset of photos that capture the scene's essential geometry.[4] Once we've reconstructed this subset, we can consider the remaining images at a later processing phase, improving performance by an order of magnitude or more.

We've experimented with three city-scale data sets downloaded from Flickr: Dubrovnik, Rome, and Venice consist of 58,000, 150,000, and 250,000 photos, respectively. Figure 4 shows reconstructions of the largest connected components in these data sets.

DENSE CITY-SCALE 3D RECONSTRUCTION

SfM recovers camera poses and 3D points by leveraging distinctive image features that match well across photos. This conservative feature selection leads to a sparse point-set reconstruction. The next stage in 3D reconstruction is to recover dense and accurate models from registered photos using a multi-view stereo algorithm. MVS algorithms have advanced dramatically in the past decade. In a recent evaluation within a controlled, laboratory setting, many methods successfully recovered complete models of objects with accuracy rivaling that of laser scanners.[5]

Multi-view stereo

MVS algorithms recover 3D geometric information in much the same way our visual system perceives depth by fusing two views. When our eyes see a point on a surface, we match in our minds the appearance of that point across the two views and then approximately intersect the lines of sight to that point—triangulation given our known eye viewpoints—and perceive its depth. With MVS, we may have many views that see the same point and can use them simultaneously for depth estimation.

Suppose we wish to estimate the depth of a pixel in one reference image, as Figure 5 shows. We take a small image window (for example, 5×5 pixels) around the pixel, and hypothesize possible depths along its line of sight. At each hypothesized depth, we position the reference window along the line of sight, project it into all the other images, and measure overall consistency between the colors in the reference window and those in the projected windows. The chosen pixel depth is the one with the highest consistency score.

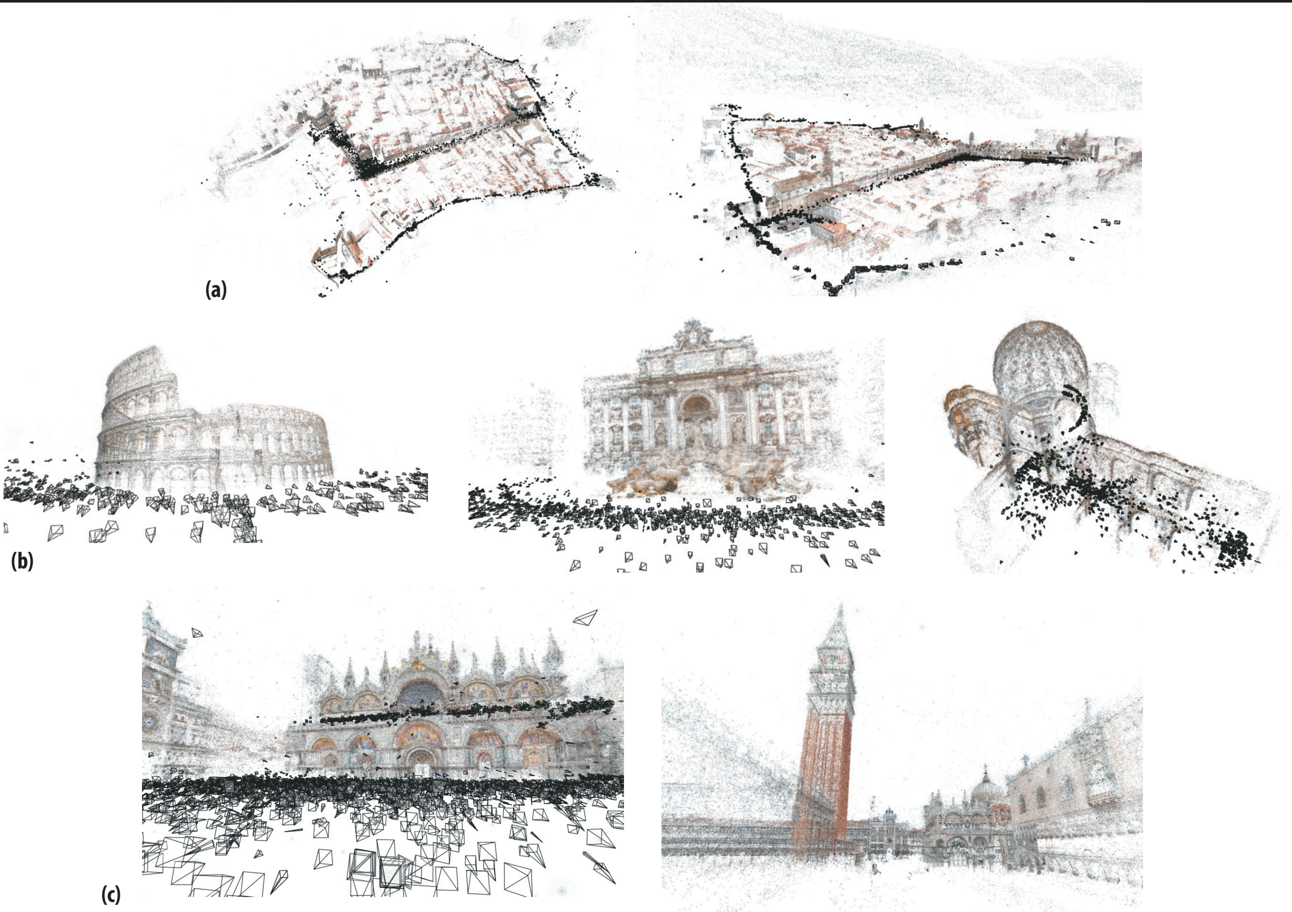


Figure 4. Large-scale model reconstruction. (a) Two views of the city of Dubrovnik, Croatia, containing 4,585 images. Starting with 58,000 images from Flickr, it took us 5 hours to do the matching and 18 hours to do the reconstructions using 352 processors. (b) The three largest reconstructions obtained from the Rome data set. Starting with 150,000 Flickr images, it took us 18 hours to do the matching and 8 hours to do the reconstructions using 496 processors. The matching process decomposed the images into numerous famous, as well as little-known, Roman landmarks. The Colosseum (left) with 2,106 images was the largest, followed by the Trevi Fountain (middle) with 1,936 images, and the interior of St. Peter's Basilica (right) with 1,294 images. (c) Two views of the San Marco Plaza reconstructed from the Venice data set of 250,000 images. The reconstruction contains 13,699 images.

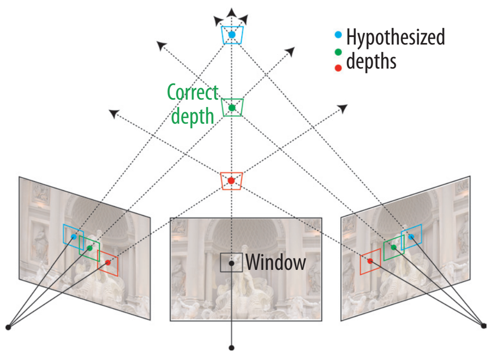


Figure 5. Standard window-based multi-view stereo (MVS) algorithm. Given a pixel and an image window around it, we hypothesize a finite number of depths along its viewing ray. At each depth, we project the window into the other images and evaluate consistency among textures at these image projections. At the true depth (highlighted in green), the consistency score is at its maximum.

This kind of window matching is susceptible to several sources of error: differences in lighting and cameras will lead to color differences in windows that should otherwise match, a view may not see the desired point if another scene object is obstructing its line of sight, and uniformly colored regions may match at many different depths. These problems can be largely addressed by robust matching that normalizes for brightness differences, eliminates contributions from views that disagree with the others for a given scene point, and avoids estimating depths when the window contains little or no color variation.

To recover a dense model, we estimate depths for every pixel in every image and then combine the resulting 3D points into a single model. There are many alternative approaches, including algorithms that solve for dense object shape directly from the images without intermediate per-view depths.

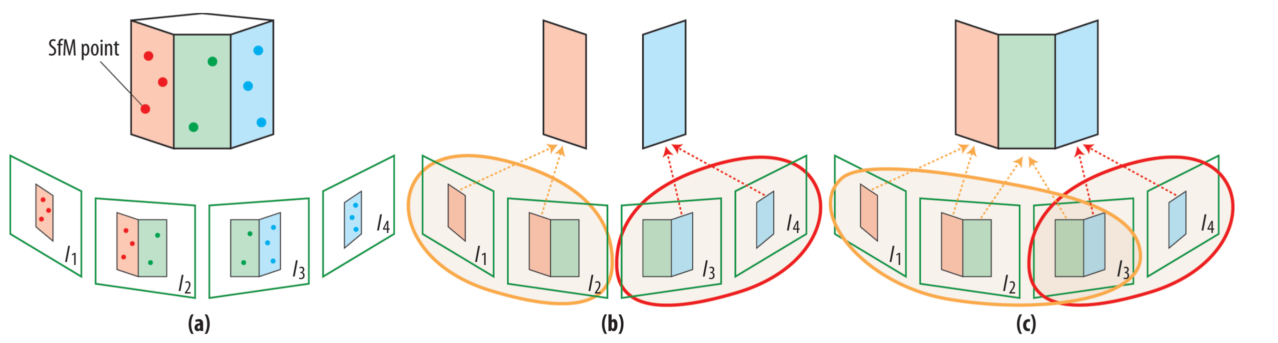


Figure 6. Image selection and clustering. (a) Our algorithm takes images and SfM outputs and extracts image clusters suitable for distributed MVS reconstruction. (b) Nonoverlapping clusters can result in failure to reconstruct scene geometry observed at the boundaries between the clusters. (c) We avoid this problem by allowing clusters to overlap.

Large-scale MVS

For city-scale MVS, the volume of photos is well beyond what any standard MVS algorithm can operate on at once; the amount of memory required is itself prohibitive. Our approach to solving this problem is to group photos into manageable-sized clusters, calculate scene geometry within each cluster independently using an MVS algorithm, and combine the results. This strategy makes it straightforward to perform the reconstruction in parallel on many processors.

Which photographs should be clustered together? The natural choice is to select images that are looking at the same part of the scene. Indeed, the results of structure from motion are already rich with this kind of information; photos with many features in common are surely looking at the same part of the scene. Our overall strategy then is to form small clusters that cover the SfM points well.

However, this simple approach—clustering photographs that look at the same scene component—may not work well for Internet image collections, because hundreds or even thousands of photos may be taken from nearly the same viewpoint, often at famous sightseeing spots. The reconstruction from a cluster consisting of very similar viewpoints would be poor, as small errors in pixel matching would translate into considerable triangulation errors. To avoid this problem, we simply remove redundant images that are very close, or even identical, to other viewpoints in the image set. Another advantage of eliminating such images is reducing the computation time for the MVS algorithm eventually to be applied.

With the remaining images, it's tempting to run a standard partitioning algorithm—for example, form a graph over the photo set and perform normalized cuts—to arrive at nonoverlapping clusters. However, such a partitioning can result in portions of the scene not being reconstructed, especially at the boundaries between clusters. Consider Figure 6, in which three facets of an object are visible in four images I_1, I_2, I_3, and I_4. If the images are partitioned into two clusters containing images (I_1, I_2) and (I_3, I_4), respectively, the middle facet will not be reconstructed because neither cluster contains the images needed to triangulate its shape. Thus, it's important to allow overlaps among the clusters.

We formulate image clustering as an optimization problem, in which we constrain the clustering to ensure that each cluster is small enough for MVS reconstruction on a single processor and the clusters as a whole cover the scene completely.[6] As we don't yet have a reconstruction of the scene, we use the SfM points as a proxy for the scene. However, cluster size and completeness don't guarantee a good clustering—for example, creating a cluster for every pair of images in the data set satisfies both constraints but is extremely redundant and inefficient. To avoid this oversegmentation, we seek clusterings that satisfy the size and completeness constraints while minimizing the sum total of the cluster sizes. This ensures that no cluster is extraneous and each contains only images needed to satisfy the constraints.

Having extracted image clusters, we use an MVS algorithm[7] to reconstruct dense 3D points for each cluster independently (one at a time or in parallel). Finally, we merge these points into a single model. The MVS algorithm produces a point cloud, and thus the final model is also a point cloud.

Figure 7 shows MVS reconstructions, rendered as colored points, of St. Peter's Basilica and the Colosseum in Rome, the city of Dubrovnik, and San Marco Plaza in Venice. The largest of these point models—San Marco Plaza—used 14,000 input images, was processed as 67 clusters, and yielded 28 million surface points in less than 3 hours when we performed MVS reconstruction in parallel on each cluster. While our system successfully reconstructed dense, high-quality 3D points for these very large scenes, our models contain holes in certain places—for example, rooftops where image coverage is poor and ground planes where surfaces usually aren't clearly visible. On the other hand, as the close-ups in Figure 7 illustrate, for places with many images the reconstruction quality is very high.

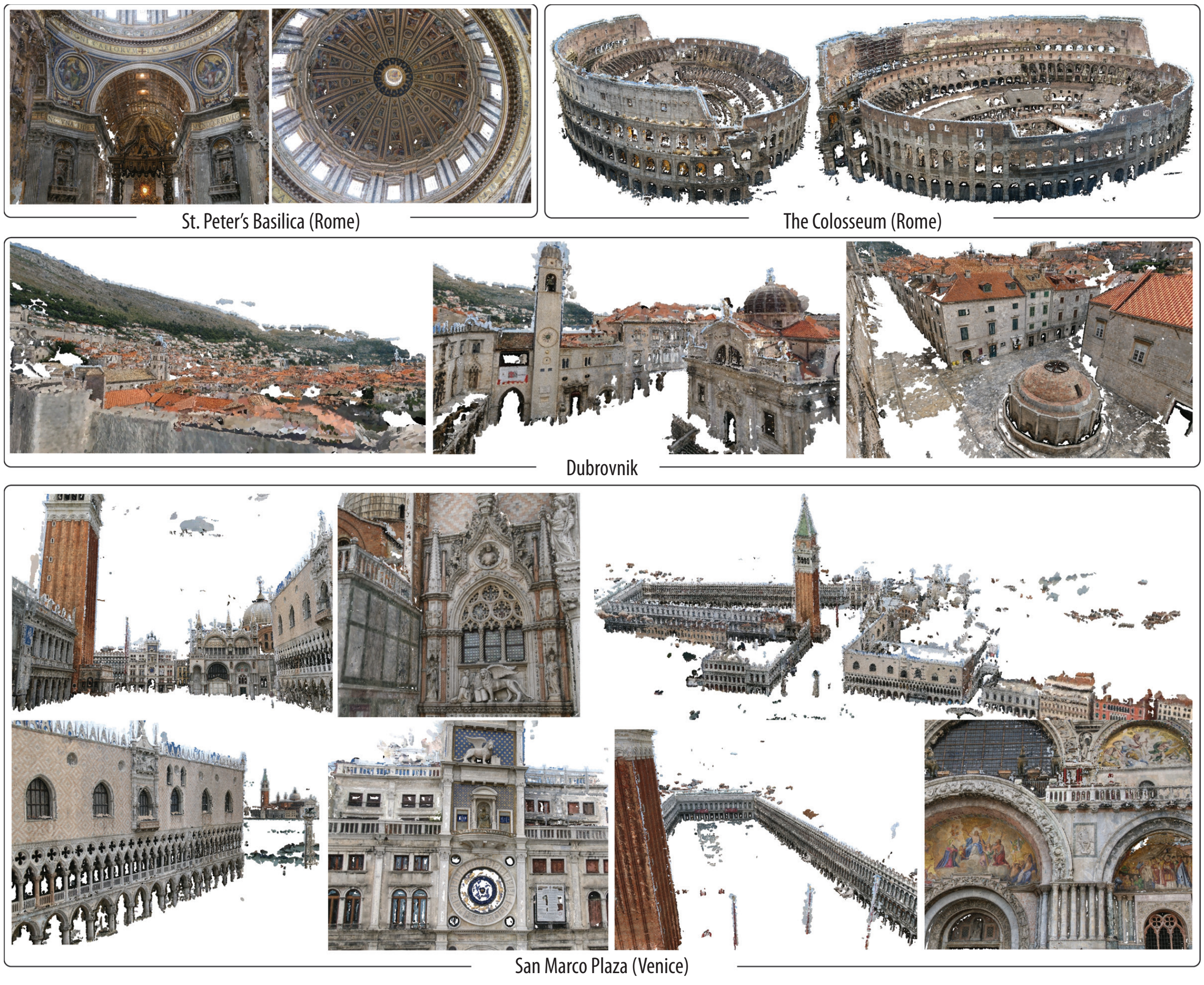


Figure 7. Reconstructed dense 3D point models. For places with many available images, reconstruction quality is very high.

Community photo collections offer a rich, ever-growing record of the world around us. Our ultimate goal is to use such collections to reconstruct every building, landscape, and (static) object, an ambitious goal toward which we've only taken the first step. Thus far, we've restricted our attention to tourist photographs, but future efforts will include structured photo surveys of street-side (for example, Google Street View) and aerial imagery. And while we've focused on the question of how far it's possible to go using computer-vision algorithms on raw image data, we plan to use other sensor information like GPS and compass data as well as laser scans to further improve the processing pipeline.

Readers can visit http://grail.cs.washington.edu/rome for more results, papers, data, and source code. While our work represents state-of-the-art 3D reconstruction from images, many challenges remain. Key among them is coverage—it's hard to get a complete model of a building or object. One reason is that reconstruction algorithms tend to fail in areas that have specular reflection or limited texture such as flat, painted walls. A more fundamental reason is the difficulty of acquiring a complete set of images of an object, sampling every nook and cranny. A promising way to address this issue is to devise algorithms that consider the structure of architectural scenes and thus can intelligently fill in gaps and unseen regions. Other research challenges include recovering surface reflectance properties, incorporating time variation in the models, and rendering high-quality CAD models rather than unstructured point clouds. C

References

1. N. Snavely, S.M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *ACM Trans. Graphics*, July 2006, pp. 835-846.

2. D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision*, Nov. 2004, pp. 91-110.
3. S. Agarwal et al., "Building Rome in a Day," *Proc. 12th IEEE Int'l Conf. Computer Vision* (ICCV 09), 2009; http://grail.cs.washington.edu/rome/rome_paper.pdf.
4. N. Snavely, S.M. Seitz, and R. Szeliski, "Skeletal Graphs for Efficient Structure from Motion," *Proc. 2008 IEEE CS Conf. Computer Vision and Pattern Recognition* (CVPR 08), IEEE CS Press, 2008; doi 101109/CVPR.2008.4587678.
5. S.M. Seitz et al., "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms," *Proc. 2006 IEEE CS Conf. Computer Vision and Pattern Recognition* (CVPR 06), IEEE CS Press, 2006, pp. 519-528.
6. Y. Furukawa et al., "Towards Internet-Scale Multi-View Stereo," *Proc. 2010 IEEE CS Conf. Computer Vision and Pattern Recognition* (CVPR 10), IEEE CS Press, 2010; www.cs.washington.edu/homes/furukawa/papers/cvpr10.pdf.
7. Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multi-View Stereopsis," to appear in *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2010.

Sameer Agarwal *is a software engineer at Google. His research interests include computer vision, machine learning, and numerical optimization. Agarwal received a PhD in computer science and engineering from the University of California, San Diego. He is a member of IEEE. Contact him at sameeragarwal@google.com.*

Yasutaka Furukawa *is a software engineer at Google. His research interests include computer vision and computer graphics. Furukawa received a PhD in computer science from the University of Illinois at Urbana-Champaign. Contact him at furukawa@google.com.*

Noah Snavely *is an assistant professor in the Department of Computer Science at Cornell University. His research interests are in computer vision and computer graphics. Snavely received a PhD in computer science and engineering from the University of Washington. He is a member of IEEE. Contact him at snavely@cs.cornell.edu.*

Brian Curless *is an associate professor in the Department of Computer Science and Engineering at the University of Washington. His research interests include computer graphics and computer vision. Curless received a PhD in electrical engineering from Stanford University. He is a member of IEEE and the ACM. Contact him at curless@cs.washington.edu.*

Steven M. Seitz *is on the technical staff at Google and a professor in the Department of Computer Science and Engineering at the University of Washington. His research interests are in computer vision and computer graphics. Seitz received a PhD in computer sciences from the University of Wisconsin—Madison. He is a senior member of IEEE and a member of the ACM. Contact him at seitz@cs.washington.edu.*

Richard Szeliski *leads the Interactive Visual Media Group at Microsoft Research and is also an affiliate professor at the University of Washington. His research interests are in computer vision and computer graphics. Szeliski received a PhD in computer science from Carnegie Mellon University. He is a Fellow of IEEE and the ACM. Contact him at szeliski@microsoft.com.*