Int J Comput Vis
DOI 10.1007/s11263-007-0107-3

Modeling the World from Internet Photo Collections

Noah Snavely · Steven M. Seitz · Richard Szeliski

Received: 30 January 2007 / Accepted: 31 October 2007
© Springer Science+Business Media, LLC 2007

Abstract There are billions of photographs on the Internet, comprising the largest and most diverse photo collection ever assembled. How can computer vision researchers exploit this imagery? This paper explores this question from the standpoint of 3D scene modeling and visualization. We present structure-from-motion and image-based rendering algorithms that operate on hundreds of images downloaded as a result of keyword-based image search queries like "Notre Dame" or "Trevi Fountain." This approach, which we call *Photo Tourism*, has enabled reconstructions of numerous well-known world sites. This paper presents these algorithms and results as a first step towards 3D modeling of the world's well-photographed sites, cities, and landscapes from Internet imagery, and discusses key open problems and challenges for the research community.

Keywords Structure from motion · 3D scene analysis · Internet imagery · Photo browsers · 3D navigation

1 Introduction

Most of the world's significant sites have been photographed under many different conditions, both from the ground and from the air. For example, a Google image search for "Notre Dame" returns over one million hits (as of September, 2007), showing the cathedral from almost every conceivable viewing position and angle, different times of day and night, and changes in season, weather, and decade. Furthermore, entire cities are now being captured at street level and from a birds-eye perspective (e.g., Windows Live Local,[1,2] and Google Streetview[3]), and from satellite or aerial views (e.g., Google[4]).

The availability of such rich imagery of large parts of the earth's surface under many different viewing conditions presents enormous opportunities, both in computer vision research and for practical applications. From the standpoint of shape modeling research, Internet imagery presents the ultimate data set, which should enable modeling a significant portion of the world's surface geometry at high resolution. As the largest, most diverse set of images ever assembled, Internet imagery provides deep insights into the space of natural images and a rich source of statistics and priors for modeling scene appearance. Furthermore, Internet imagery provides an ideal test bed for developing robust and general computer vision algorithms that can work effectively "in the wild." In turn, algorithms that operate effectively on such imagery will enable a host of important applications, ranging from 3D visualization, localization, communication (media sharing), and recognition, that go well beyond traditional computer vision problems and can have broad impacts for the population at large.

To date, this imagery is almost completely untapped and unexploited by computer vision researchers. A major reason is that the imagery is not in a form that is amenable to processing, at least by traditional methods: the images are

N. Snavely (✉) · S.M. Seitz
University of Washington, Seattle, WA, USA
e-mail: snavely@cs.washington.edu

R. Szeliski
Microsoft Research, Redmond, WA, USA

[1] Windows Live Local, http://local.live.com.

[2] Windows Live Local—Virtual Earth Technology Preview, http://preview.local.live.com.

[3] Google Maps, http://maps.google.com.

[4] Google Maps, http://maps.google.com.

unorganized, uncalibrated, with widely variable and uncontrolled illumination, resolution, and image quality. Developing computer vision techniques that can operate effectively with such imagery has been a major challenge for the research community. Within this scope, one key challenge is *registration*, i.e., figuring out correspondences between images, and how they relate to one another in a common 3D coordinate system (structure from motion). While a lot of progress has been made in these areas in the last two decades (Sect. 2), many challenging open problems remain.

In this paper we focus on the problem of geometrically registering Internet imagery and a number of applications that this enables. As such, we first review the state of the art and then present some first steps towards solving this problem along with a visualization front-end that we call *Photo Tourism* (Snavely et al. 2006). We then present a set of open research problems for the field, including the creation of more efficient correspondence and reconstruction techniques for extremely large image data sets. This paper expands on the work originally presented in (Snavely et al. 2006) with many new reconstructions and visualizations of algorithm behavior across datasets, as well as a brief discussion of *Photosynth*, a Technology Preview by Microsoft Live Labs, based largely on (Snavely et al. 2006). We also present a more complete related work section and add a broad discussion of open research challenges for the field. Videos of our system, along with additional supplementary material, can be found on our Photo Tourism project Web site, http://phototour.cs.washington.edu.

2 Previous Work

The last two decades have seen a dramatic increase in the capabilities of 3D computer vision algorithms. These include advances in feature correspondence, structure from motion, and image-based modeling. Concurrently, image-based rendering techniques have been developed in the computer graphics community, and image browsing techniques have been developed for multimedia applications.

2.1 Feature Correspondence

Twenty years ago, the foundations of modern feature detection and matching techniques were being laid. Lucas and Kanade (1981) had developed a patch tracker based on two-dimensional image statistics, while Moravec (1983) introduced the concept of "corner-like" feature points. Förstner (1986) and then Harris and Stephens (1988) both proposed finding keypoints using measures based on eigenvalues of smoothed outer products of gradients, which are still widely used today. While these early techniques detected keypoints at a single scale, modern techniques use a quasi-continuous sampling of scale space to detect points invariant to changes in scale and orientation (Lowe 2004; Mikolajczyk and Schmid 2004) and somewhat invariant to affine transformations (Baumberg 2000; Kadir and Brady 2001; Schaffalitzky and Zisserman 2002; Mikolajczyk et al. 2005).

Unfortunately, early techniques relied on matching patches around the detected keypoints, which limited their range of applicability to scenes seen from similar viewpoints, e.g., for aerial photogrammetry applications (Hannah 1988). If features are being tracked from frame to frame, an affine extension of the basic Lucas-Kanade tracker has been shown to perform well (Shi and Tomasi 1994). However, for true *wide baseline matching*, i.e., the automatic matching of images taken from widely different views (Baumberg 2000; Schaffalitzky and Zisserman 2002; Strecha et al. 2003; Tuytelaars and Van Gool 2004; Matas et al. 2004), (weakly) affine-invariant feature descriptors must be used.

Mikolajczyk et al. (2005) review some recently developed view-invariant local image descriptors and experimentally compare their performance. In our own Photo Tourism research, we have been using Lowe's Scale Invariant Feature Transform (SIFT) (Lowe 2004), which is widely used by others and is known to perform well over a reasonable range of viewpoint variation.

2.2 Structure from Motion

The late 1980s also saw the development of effective structure from motion techniques, which aim to simultaneously reconstruct the unknown 3D scene structure and camera positions and orientations from a set of feature correspondences. While Longuet-Higgins (1981) introduced a still widely used two-frame *relative orientation* technique in 1981, the development of multi-frame structure from motion techniques, including *factorization methods* (Tomasi and Kanade 1992) and global optimization techniques (Spetsakis and Aloimonos 1991; Szeliski and Kang 1994; Oliensis 1999) occurred quite a bit later.

More recently, related techniques from photogrammetry such as *bundle adjustment* (Triggs et al. 1999) (with related sparse matrix techniques, Szeliski and Kang 1994) have made their way into computer vision and are now regarded as the *gold standard* for performing optimal 3D reconstruction from correspondences (Hartley and Zisserman 2004).

For situations where the camera calibration parameters are unknown, *self-calibration* techniques, which first estimate a *projective* reconstruction of the 3D world and then perform a *metric upgrade* have proven to be successful (Pollefeys et al. 1999; Pollefeys and Van Gool 2002). In our own work (Sect. 4.2), we have found that the simpler approach of simply estimating each camera's focal length as part of the bundle adjustment process seems to produce good results.

The SfM approach used in this paper is similar to that of Brown and Lowe (2005), with several modifications to improve robustness over a variety of data sets. These include initializing new cameras using pose estimation, to help avoid local minima; a different heuristic for selecting the initial two images for SfM; checking that reconstructed points are well-conditioned before adding them to the scene; and using focal length information from image EXIF tags. Schaffalitzky and Zisserman (2002) present another related technique for reconstructing unordered image sets, concentrating on efficiently matching interest points between images. Vergauwen and Van Gool have developed a similar approach (Vergauwen and Van Gool 2006) and are hosting a web-based reconstruction service for use in cultural heritage applications[5]. Fitzgibbon and Zisserman (1998) and Nistér (2000) prefer a bottom-up approach, where small subsets of images are matched to each other and then merged in an agglomerative fashion into a complete 3D reconstruction. While all of these approaches address the same SfM problem that we do, they were tested on much simpler datasets with more limited variation in imaging conditions. Our paper marks the first successful demonstration of SfM techniques applied to the kinds of real-world image sets found on Google and Flickr. For instance, our typical image set has photos from hundreds of different cameras, zoom levels, resolutions, different times of day or seasons, illumination, weather, and differing amounts of occlusion.

2.3 Image-Based Modeling

In recent years, computer vision techniques such as structure from motion and model-based reconstruction have gained traction in the computer graphics field under the name of *image-based modeling*. IBM is the process of creating three-dimensional models from a collection of input images (Debevec et al. 1996; Grzeszczuk 2002; Pollefeys et al. 2004).

One particular application of IBM has been the creation of large scale architectural models. Notable examples include the semi-automatic Façade system (Debevec et al. 1996), which was used to reconstruct compelling fly-throughs of the University of California Berkeley campus; automatic architecture reconstruction systems such as that of Dick et al. (2004); and the MIT City Scanning Project (Teller et al. 2003), which captured thousands of calibrated images from an instrumented rig to construct a 3D model of the MIT campus. There are also several ongoing academic and commercial projects focused on large-scale urban scene reconstruction. These efforts include the 4D Cities project (Schindler et al. 2007), which aims to create a spatial-temporal model of Atlanta from historical photographs; the Stanford CityBlock Project (Román et al. 2004), which uses video of city blocks to create multi-perspective strip images; and the UrbanScape project of Akbarzadeh et al. (2006). Our work differs from these previous approaches in that we only reconstruct a *sparse* 3D model of the world, since our emphasis is more on creating smooth 3D transitions between photographs rather than interactively visualizing a 3D world.

2.4 Image-Based Rendering

The field of image-based rendering (IBR) is devoted to the problem of synthesizing new views of a scene from a set of input photographs. A forerunner to this field was the groundbreaking Aspen MovieMap project (Lippman 1980), in which thousands of images of Aspen Colorado were captured from a moving car, registered to a street map of the city, and stored on laserdisc. A user interface enabled interactively moving through the images as a function of the desired path of the user. Additional features included a navigation map of the city overlaid on the image display, and the ability to touch any building in the current field of view and jump to a facade of that building. The system also allowed attaching metadata such as restaurant menus and historical images with individual buildings. Recently, several companies, such as Google[6] and EveryScape[7] have begun creating similar "surrogate travel" applications that can be viewed in a web browser. Our work can be seen as a way to automatically create MovieMaps from unorganized collections of images. (In contrast, the Aspen MovieMap involved a team of over a dozen people working over a few years.) A number of our visualization, navigation, and annotation capabilities are similar to those in the original MovieMap work, but in an improved and generalized form.

More recent work in IBR has focused on techniques for new view synthesis, e.g., (Chen and Williams 1993; McMillan and Bishop 1995; Gortler et al. 1996; Levoy and Hanrahan 1996; Seitz and Dyer 1996; Aliaga et al. 2003; Zitnick et al. 2004; Buehler et al. 2001). In terms of applications, Aliaga et al.'s (2003) *Sea of Images* work is perhaps closest to ours in its use of a large collection of images taken throughout an architectural space; the same authors address the problem of computing consistent feature matches across multiple images for the purposes of IBR (Aliaga et al. 2003). However, our images are casually acquired by different photographers, rather than being taken on a fixed grid with a guided robot.

In contrast to most prior work in IBR, our objective is *not* to synthesize a photo-realistic view of the world from all viewpoints *per se*, but to browse a specific collection of

[5]Epoch 3D Webservice, http://homes.esat.kuleuven.be/~visit3d/webservice/html/.

[6]Google Maps, http://maps.google.com.

[7]Everyscape, http://www.everyscape.com.

photographs in a 3D spatial context that gives a *sense* of the geometry of the underlying scene. Our approach therefore uses an approximate plane-based view interpolation method and a non-photorealistic rendering of background scene structures. As such, we side-step the more challenging problems of reconstructing full surface models (Debevec et al. 1996; Teller et al. 2003), light fields (Gortler et al. 1996; Levoy and Hanrahan 1996), or pixel-accurate view interpolations (Chen and Williams 1993; McMillan and Bishop 1995; Seitz and Dyer 1996; Zitnick et al. 2004). The benefit of doing this is that we are able to operate robustly with input imagery that is beyond the scope of previous IBM and IBR techniques.

2.5 Image Browsing, Retrieval, and Annotation

There are many techniques and commercial products for browsing sets of photos and much research on the subject of how people tend to organize photos, e.g., (Rodden and Wood 2003). Many of these techniques use metadata, such as keywords, photographer, or time, as a basis of photo organization (Cooper et al. 2003).

There has recently been growing interest in using geolocation information to facilitate photo browsing. In particular, the World-Wide Media Exchange (WWMX) (Toyama et al. 2003) arranges images on an interactive 2D map. PhotoCompas (Naaman et al. 2004) clusters images based on time and location. Realityflythrough (McCurdy and Griswold 2005) uses interface ideas similar to ours for exploring video from camcorders instrumented with GPS and tilt sensors, and Kadobayashi and Tanaka (2005) present an interface for retrieving images using proximity to a virtual camera. In Photowalker (Tanaka et al. 2002), a user can manually author a walkthrough of a scene by specifying transitions between pairs of images in a collection. In these systems, location is obtained from GPS or is manually specified. Because our approach does not require GPS or other instrumentation, it has the advantage of being applicable to existing image databases and photographs from the Internet. Furthermore, many of the navigation features of our approach exploit the computation of image feature correspondences and sparse 3D geometry, and therefore go beyond what has been possible in these previous location-based systems.

Many techniques also exist for the related task of retrieving images from a database. One particular system related to our work is Video Google (Sivic and Zisserman 2003) (not to be confused with Google's own video search), which allows a user to select a query object in one frame of video and efficiently find that object in other frames. Our object-based navigation mode uses a similar idea, but extended to the 3D domain.

A number of researchers have studied techniques for automatic and semi-automatic image annotation, and annotation transfer in particular. The LOCALE system (Naaman et al. 2003) uses proximity to transfer labels between georeferenced photographs. An advantage of the annotation capabilities of our system is that our feature correspondences enable transfer at much finer granularity; we can transfer annotations of specific *objects and regions* between images, taking into account occlusions and the motions of these objects under changes in viewpoint. This goal is similar to that of augmented reality (AR) approaches (e.g., Feiner et al. 1997), which also seek to annotate images. While most AR methods register a 3D computer-generated model to an image, we instead transfer 2D image annotations to other images. Generating annotation content is therefore much easier. (We can, in fact, import existing annotations from popular services like Flickr.) Annotation transfer has been also explored for video sequences (Irani and Anandan 1998).

Finally, Johansson and Cipolla (2002) have developed a system where a user can take a photograph, upload it to a server where it is compared to an image database, and receive location information. Our system also supports this application in addition to many other capabilities (visualization, navigation, annotation, etc.).

3 Overview

Our objective is to geometrically register large photo collections from the Internet and other sources, and to use the resulting 3D camera and scene information to facilitate a number of applications in visualization, localization, image browsing, and other areas. This section provides an overview of our approach and summarizes the rest of the paper.

The primary technical challenge is to robustly match and reconstruct 3D information from hundreds or thousands of images that exhibit large variations in viewpoint, illumination, weather conditions, resolution, etc., and may contain significant clutter and outliers. This kind of variation is what makes Internet imagery (i.e., images returned by Internet image search queries from sites such as Flickr and Google) so challenging to work with.

In tackling this problem, we take advantage of two recent breakthroughs in computer vision, namely feature-matching and structure from motion, as reviewed in Sect. 2. The backbone of our work is a robust SfM approach that reconstructs 3D camera positions and sparse point geometry for large datasets and has yielded reconstructions for dozens of famous sites ranging from Notre Dame Cathedral to the Great Wall of China. Section 4 describes this approach in detail, as well as methods for aligning reconstructions to satellite and map data to obtain geo-referenced camera positions and geometry.

One of the most exciting applications for these reconstructions is 3D scene visualization. However, the sparse

points produced by SfM methods are by themselves very limited and do not directly produce compelling scene renderings. Nevertheless, we demonstrate that this sparse SfM-derived geometry and camera information, along with morphing and non-photorealistic rendering techniques, is sufficient to provide compelling view interpolations as described in 5. Leveraging this capability, Section 6 describes a novel *photo explorer* interface for browsing large collections of photographs in which the user can virtually explore the 3D space by moving from one image to another.

Often, we are interested in learning more about the content of an image, e.g., "which statue is this?" or "when was this building constructed?" A great deal of annotated image content of this form already exists in guidebooks, maps, and Internet resources such as Wikipedia[8] and Flickr. However, the image you may be viewing at any particular time (e.g., from your cell phone camera) may not have such annotations. A key feature of our system is the ability to transfer annotations automatically between images, so that information about an object in one image is propagated to all other images that contain the same object (Sect. 7).

Section 8 presents extensive results on 11 scenes, with visualizations and an analysis of the matching and reconstruction results for these scenes. We also briefly describe *Photosynth*, a related 3D image browsing tool developed by Microsoft Live Labs that is based on techniques from this paper, but also adds a number of interesting new elements. Finally, we conclude with a set of research challenges for the community in Sect. 9.

4 Reconstructing Cameras and Sparse Geometry

The visualization and browsing components of our system require accurate information about the relative location, orientation, and intrinsic parameters such as focal lengths for each photograph in a collection, as well as sparse 3D scene geometry. A few features of our system require the *absolute* locations of the cameras, in a geo-referenced coordinate frame. Some of this information can be provided with GPS devices and electronic compasses, but the vast majority of existing photographs lack such information. Many digital cameras embed focal length and other information in the EXIF tags of image files. These values are useful for initialization, but are sometimes inaccurate.

In our system, we do not rely on the camera or any other piece of equipment to provide us with location, orientation, or geometry. Instead, we compute this information from the images themselves using computer vision techniques. We first detect feature points in each image, then match feature points between pairs of images, and finally run an iterative, robust SfM procedure to recover the camera parameters. Because SfM only estimates the *relative* position of each camera, and we are also interested in absolute coordinates (e.g., latitude and longitude), we use an interactive technique to register the recovered cameras to an overhead map. Each of these steps is described in the following subsections.

4.1 Keypoint Detection and Matching

The first step is to find feature points in each image. We use the SIFT keypoint detector (Lowe 2004), because of its good invariance to image transformations. Other feature detectors could also potentially be used; several detectors are compared in the work of Mikolajczyk et al. (2005). In addition to the keypoint locations themselves, SIFT provides a local descriptor for each keypoint. A typical image contains several thousand SIFT keypoints.

Next, for each pair of images, we match keypoint descriptors between the pair, using the approximate nearest neighbors (ANN) kd-tree package of Arya et al. (1998). To match keypoints between two images I and J, we create a kd-tree from the feature descriptors in J, then, for each feature in I we find the nearest neighbor in J using the kd-tree. For efficiency, we use ANN's priority search algorithm, limiting each query to visit a maximum of 200 bins in the tree. Rather than classifying false matches by thresholding the distance to the nearest neighbor, we use the ratio test described by Lowe (2004): for a feature descriptor in I, we find the two nearest neighbors in J, with distances d_1 and d_2, then accept the match if $\frac{d_1}{d_2} < 0.6$. If more than one feature in I matches the same feature in J, we remove all of these matches, as some of them must be spurious.

After matching features for an image pair (I, J), we robustly estimate a fundamental matrix for the pair using RANSAC (Fischler and Bolles 1981). During each RANSAC iteration, we compute a candidate fundamental matrix using the eight-point algorithm (Hartley and Zisserman 2004), normalizing the problem to improve robustness to noise (Hartley 1997). We set the RANSAC outlier threshold to be 0.6% of the maximum image dimension, i.e., 0.006 max(image width, image height) (about six pixels for a 1024×768 image). The F-matrix returned by RANSAC is refined by running the Levenberg-Marquardt algorithm (Nocedal and Wright 1999) on the eight parameters of the F-matrix, minimizing errors for all the inliers to the F-matrix. Finally, we remove matches that are outliers to the recovered F-matrix using the above threshold. If the number of remaining matches is less than twenty, we remove all of the matches from consideration.

After finding a set of geometrically consistent matches between each image pair, we organize the matches into *tracks*, where a track is a connected set of matching keypoints across multiple images. If a track contains more than

[8] Wikipedia, http://www.wikipedia.org.

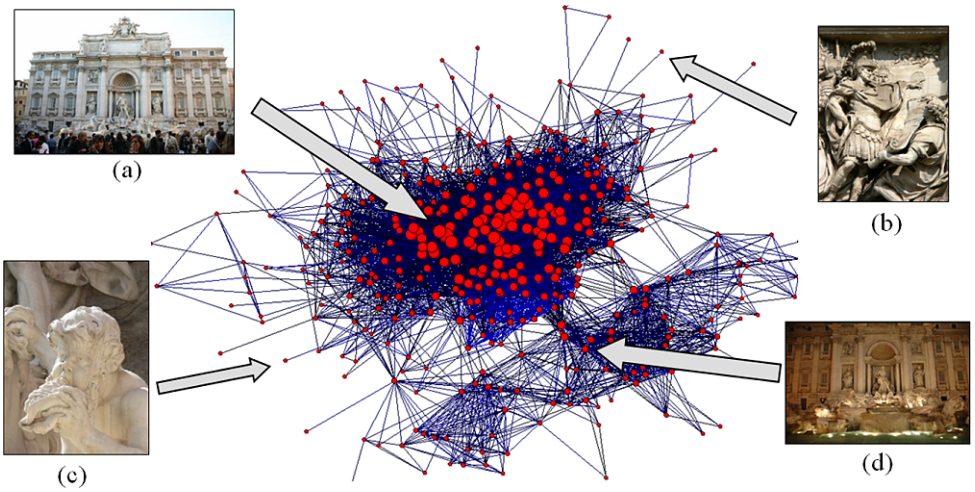

Fig. 1 Photo connectivity graph. This graph contains a node for each image in a set of photos of the Trevi Fountain, with an edge between each pair of photos with matching features. The size of a node is proportional to its degree. There are two dominant clusters corresponding to day (**a**) and night time (**d**) photos. Similar views of the facade cluster together in the center, while nodes in the periphery, e.g., (**b**) and (**c**), are more unusual (often close-up) views

one keypoint in the same image, it is deemed inconsistent. We keep consistent tracks containing at least two keypoints for the next phase of the reconstruction procedure.

Once correspondences are found, we can construct an *image connectivity graph*, in which each image is a node and an edge exists between any pair of images with matching features. A visualization of an example connectivity graph for the Trevi Fountain is Fig. 1. This graph embedding was created with the *neato* tool in the Graphviz toolkit.[9] *Neato* represents the graph as a mass-spring system and solves for an embedding whose energy is a local minimum.

The image connectivity graph of this photo set has several distinct features. The large, dense cluster in the center of the graph consists of photos that are all fairly wide-angle, frontal, well-lit shots of the fountain (e.g., image (a)). Other images, including the "leaf" nodes (e.g., images (b) and (c)) and night time images (e.g., image (d)), are more loosely connected to this core set. Other connectivity graphs are shown in Figs. 9 and 10.

4.2 Structure from Motion

Next, we recover a set of camera parameters (e.g., rotation, translation, and focal length) for each image and a 3D location for each track. The recovered parameters should be consistent, in that the reprojection error, i.e., the sum of distances between the projections of each track and its corresponding image features, is minimized. This minimization problem can formulated as a non-linear least squares problem (see Appendix 1) and solved using bundle adjustment. Algorithms for solving this non-linear problem, such as Nocedal and Wright (1999), are only guaranteed to find local minima, and large-scale SfM problems are particularly prone to getting stuck in bad local minima, so it is important to provide good initial estimates of the parameters. Rather than estimating the parameters for all cameras and tracks at once, we take an incremental approach, adding in one camera at a time.

We begin by estimating the parameters of a single pair of cameras. This initial pair should have a large number of matches, but also have a large baseline, so that the initial two-frame reconstruction can be robustly estimated. We therefore choose the pair of images that has the largest number of matches, subject to the condition that those matches cannot be well-modeled by a single homography, to avoid degenerate cases such as coincident cameras. In particular, we find a homography between each pair of matching images using RANSAC with an outlier threshold of 0.4% of max(image width, image height), and store the percentage of feature matches that are inliers to the estimated homography. We select the initial image pair as that with the lowest percentage of inliers to the recovered homography, but with at least 100 matches. The camera parameters for this pair are estimated using Nistér's implementation of the five point algorithm (Nistér 2004),[10] then the tracks visible in the two images are triangulated. Finally, we do a two frame bundle adjustment starting from this initialization.

Next, we add another camera to the optimization. We select the camera that observes the largest number of tracks whose 3D locations have already been estimated, and initialize the new camera's extrinsic parameters using the direct linear transform (DLT) technique (Hartley and Zisserman 2004) inside a RANSAC procedure. For this RANSAC step, we use an outlier threshold of 0.4% of max(image width, image height). In addition to providing an estimate of the camera rotation and translation, the DLT technique returns an upper-triangular matrix **K** which can

[9] Graphviz—graph visualization software, http://www.graphviz.org/.

[10] We only choose the initial pair among pairs for which a focal length estimate is available for both cameras, and therefore a calibrated relative pose algorithm can be used.

be used as an estimate of the camera intrinsics. We use **K** and the focal length estimated from the EXIF tags of the image to initialize the focal length of the new camera (see Appendix 1 for more details). Starting from this initial set of parameters, we run a bundle adjustment step, allowing only the new camera, and the points it observes, to change; the rest of the model is held fixed.

Finally, we add points observed by the new camera into the optimization. A point is added if it is observed by at least one other recovered camera, and if triangulating the point gives a well-conditioned estimate of its location. We estimate the conditioning by considering all pairs of rays that could be used to triangulate that point, and finding the pair of rays with the maximum angle of separation. If this maximum angle is larger than a threshold (we use 2.0 degrees in our experiments), then the point is triangulated. Note that this check will tend to reject points at infinity. While points at infinity can be very useful for estimating accurate camera rotations, we have observed that they can sometimes cause problems, as using noisy camera parameters to triangulate points at infinity can result in points at erroneous, finite 3D locations. Once the new points have been added, we run a global bundle adjustment to refine the entire model. We find the minimum error solution using the sparse bundle adjustment library of Lourakis and Argyros (2004).

This procedure is repeated, one camera at a time, until no remaining camera observes enough reconstructed 3D points to be reliably reconstructed (we use a cut-off of twenty points to stop the reconstruction process). Therefore, in general, only a subset of the images will be reconstructed. This subset is not selected beforehand, but is determined by the algorithm while it is running in the form of a termination criterion.

For increased robustness and speed, we make a few modifications to the basic procedure outlined above. First, after every run of the optimization, we detect outlier tracks that contain at least one keypoint with a high reprojection error, and remove these tracks from the optimization. The outlier threshold for a given image adapts to the current distribution of reprojection errors for that image. In particular, for a given image I, we compute d_{80}, the 80th percentile of the reprojection errors for that image, and use $\text{clamp}(2.4d_{80}, 4.0, 16.0)$ as the outlier threshold (where $\text{clamp}(x, a, b) = \min(\max(x, a), b)$). The effect of this clamping function is that all points with a reprojection error above 16.0 pixels will be rejected as outliers, and all points with a reprojection error less than 4.0 will be kept as inliers, with the exact threshold lying between these two values. After rejecting outliers, we rerun the optimization, rejecting outliers after each run, until no more outliers are detected.

Second, rather than adding a single camera at a time into the optimization, we add multiple cameras. To select which cameras to add, we first find the camera with the greatest number of matches, M, to the existing 3D points, then add any camera with at least $0.75M$ matches to the existing 3D points.

Fig. 2 Estimated camera locations for the Great Wall data set

We have also found that estimating radial distortion parameters for each camera can have a significant effect on the accuracy of the reconstruction, because many consumer cameras and lenses produce images with noticeable distortion. We therefore estimate two radial distortion parameters κ_1 and κ_2, for each camera. To map a projected 2D point $p = (p_x, p_y)$ to a distorted point $p' = (x', y')$, we use the formula:

$$\rho^2 = \left(\frac{p_x}{f}\right)^2 + \left(\frac{p_y}{f}\right)^2,$$

$$\alpha = \kappa_1\rho^2 + \kappa_2\rho^4,$$

$$p' = \alpha p$$

where f is the current estimate of the focal length (note that we assume that the center of distortion is the center of the image, and that we define the center of the image to be the origin of the image coordinate system). When initializing new cameras, we set $\kappa_1 = \kappa_2 = 0$, but these parameters are freed during bundle adjustment. To avoid undesirably large values of these parameters, we add a term $\lambda(\kappa_1^2 + \kappa_2^2)$ to the objective function for each camera (we use a value of 10.0 for λ in our experiments). This term discourages values of κ_1 and κ_2 which have a large magnitude.

Figure 2 shows an example of reconstructed points and cameras (rendered as frusta), for the **Great Wall** data set, superimposed on one of the input images, computed with this method. Many more results are presented in Sect. 8. For the various parameter settings and thresholds described in this section, we used the same values for each of the image sets in Sect. 8, and the reconstruction algorithm ran completely automatically for most of the sets. Occasionally, we found

that the technique for selecting the initial image pair would choose a pair with insufficient baseline to generate a good initial reconstruction. This usually occurs when the selected pair has a large number of mismatched features, which can appear to be outliers to a dominant homography. When this happens, we specify the initial pair manually. Also, in some of our Internet data sets there are a small number of "bad" images, such as fisheye images, montages of several different images, and so on, which our camera model cannot handle well. These images tend to be have very poor location estimates, and in our current system these must be identified manually if the user wishes to remove them.

The total running time of the SfM procedure for the data sets we experimented with ranged from about three hours (for the **Great Wall** collection, 120 photos processed and matched, and 82 ultimately reconstructed) to more than 12 days (for **Notre Dame**, 2,635 photos processed and matched, and 598 photos reconstructed). Sect. 8 lists the running time for the complete pipeline (feature detection, matching, and SfM) for each data set. The running time is dominated by two steps: the pairwise matching, and the incremental bundle adjustment. The complexity of the matching stage is quadratic in the number of input photos, but each pair of images can be matched independently, so the running time can be improved by a constant factor through parallelization. The speed of the bundle adjustment phase depends on several factors, including the number of photos and points, and the degree of coupling between cameras (e.g., when many cameras observe the same set of points, the bundle adjustment tends to become slower). In the future, we plan to work on speeding up the reconstruction process, as described in Sect. 9.

4.3 Geo-Registration

The SfM procedure estimates *relative* camera locations. The final step of the location estimation process is to optionally align the model with a geo-referenced image or map (such as a satellite image, floor plan, or digital elevation map) so as to determine the absolute geocentric coordinates of each camera. Most of the features of the photo explorer can work with relative coordinates, but others, such as displaying an overhead map require absolute coordinates.

The estimated camera locations are, in theory, related to the absolute locations by a similarity transform (global translation, rotation, and uniform scale). To determine the correct transformation the user interactively rotates, translates, and scales the model until it is in agreement with a provided image or map. To assist the user, we estimate the "up" or gravity vector using the method of Szeliski (2006). The 3D points, lines, and camera locations are then rendered superimposed on the alignment image, using an orthographic projection with the camera positioned above the scene, pointed downward. If the up vector was estimated correctly, the user needs only to rotate the model in 2D, rather than 3D. Our experience is that it is fairly easy, especially in urban scenes, to perform this alignment by matching the recovered points to features, such as building façades, visible in the image. Figure 3 shows a screenshot of such an alignment.

Fig. 3 Example registration of cameras to an overhead map. Here, the cameras and recovered line segments from the Prague data set are shown superimposed on an aerial image. (Aerial image shown here and in Fig. 4 courtesy of Gefos, a.s.[11] and Atlas.cz)

In some cases the recovered scene cannot be aligned to a geo-referenced coordinate system using a similarity transform. This can happen if the SfM procedure fails to obtain a fully metric reconstruction of the scene, or because of low-frequency drift in the recovered point and camera locations. These sources of error do not have a significant effect on many of the navigation controls used in our explorer interface, as the error is not usually locally noticeable, but are problematic when an accurate model is desired.

One way to "straighten out" the recovered scene is to pin down a sparse set of ground control points or cameras to known 3D locations (acquired, for instance, from GPS tags attached to a few images) by adding constraints to the SfM optimization. Alternatively, a user can manually specify correspondences between points or cameras and locations in an image or map, as in the work of Robertson and Cipolla (2002).

4.3.1 Aligning to Digital Elevation Maps

For landscapes and other very large scale scenes, we can take advantage of Digital Elevation Maps (DEMs), used for example in Google Earth[12] and with coverage of most of

[12]Google Earth, http://earth.google.com.

the United States available through the U.S. Geological Survey.[13] To align point cloud reconstructions to DEMs, we manually specify a few correspondences between the point cloud and the DEM, and estimate a 3D similarity transform to determine an initial alignment. We then re-run the SfM optimization with an additional objective term to fit the specified DEM points. In the future, as more geo-referenced ground-based imagery becomes available (e.g., through systems like WWMX (Toyama et al. 2003) or Windows Live Local[14]), this manual step will no longer be necessary.

4.4 Scene Representation

After reconstructing a scene, we optionally detect 3D line segments in the scene using a technique similar to that of Schmid and Zisserman (1997). Once this is done, the scene can be saved for viewing in our interactive photo explorer. In the viewer, the reconstructed scene model is represented with the following data structures:

- A set of points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Each point consists of a 3D location and a color obtained from one of the image locations where that point is observed.
- A set of cameras, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Each camera C_j consists of an image I_j, a rotation matrix $\mathbf{R}_j$, a translation t_j, and a focal length f_j.
- A mapping, Points, between cameras and the points they observe. That is, Points(C) is the subset of $\mathcal{P}$ containing the points observed by camera C.
- A set of 3D line segments $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ and a mapping, Lines, between cameras and the set of lines they observe.

We also pre-process the scene to compute a set of 3D planes for each camera or pair of cameras:

- For each camera C_i, we compute a 3D plane, Plane(C_i), by using RANSAC to robustly fit a plane to Points(C_i).
- For each pair of neighboring cameras (i.e., cameras which view at least three points in common), C_i, C_j, we compute a 3D plane, CommonPlane(C_i, C_j) by using RANSAC to fit a plane to Points(C_i) $\cup$ Points(C_j).

5 Photo Explorer Rendering

Once a set of photographs of a scene has been registered, the user can browse the photographs with our photo explorer interface. Two important aspects of this interface are how we render the explorer display, described in this section, and the navigation controls, described in Sect. 6.

[13] U.S. Geological Survey, http://www.usgs.com.

[14] Windows Live Local—Virtual Earth Technology Preview, http://preview.local.live.com.

5.1 User Interface Layout

Figure 4 (left-hand image) shows a screenshot from the main window of our photo exploration interface. The components of this window are the main view, which fills the window, and three overlay panes: an information and search pane on the left, a thumbnail pane along the bottom, and a map pane in the upper-right corner.

The main view shows the world as seen from a virtual camera controlled by the user. This view is not meant to show a photo-realistic rendering of the scene, but rather to display photographs in spatial context and give a sense of the geometry of the true scene.

The information pane appears when the user visits a photograph. This pane displays information about that photo, including its name, the name of the photographer, and the date and time when it was taken. In addition, this pane contains controls for searching for other photographs with certain geometric relations to the current photo, as described in Sect. 6.2.

The thumbnail pane shows the results of search operations as a filmstrip of thumbnails. When the user mouses over a thumbnail, the corresponding image I_j is projected onto Plane(C_j) to show the content of that image and how it is situated in space. The thumbnail panel also has controls for sorting the current thumbnails by date and time and viewing them as a slideshow.

Finally, the map pane displays an overhead view of scene that tracks the user's position and heading.

5.2 Rendering the Scene

The main view displays a rendering of the scene from the current viewpoint. The cameras are rendered as frusta. If the user is visiting a camera, the back face of that camera frustum is texture-mapped with an opaque, full-resolution version of the photograph, so that the user can see it in detail. The back faces of the other cameras frusta are texture-mapped with a low-resolution, semi-transparent thumbnail of the photo. The scene itself is rendered with the recovered points and lines.

We also provide a non-photorealistic rendering mode that provides more attractive visualizations. This mode uses a washed-out coloring to give an impression of scene appearance and geometry, but is abstract enough to be forgiving of the lack of detailed geometry. To generate the rendering, we project a blurred, semi-transparent version of each image I_j onto Plane(C_j) and use alpha blending to combine the projections. An example rendering using projected images overlaid with line segments is shown in Fig. 4.

5.3 Transitions between Photographs

An important element of our user interface is the method used to generate transitions when the user moves between

Fig. 4 Screenshots from the explorer interface. *Left*: when the user visits a photo, that photo appears at full-resolution, and information about it appears in a pane on the left. *Right*: a view looking down on the Prague dataset, rendered in a non-photorealistic style

photos in the explorer. Most existing photo browsing tools cut from one photograph to the next, sometimes smoothing the transition by cross-fading. In our case, the geometric information we infer about the photographs allows us to use camera motion and view interpolation to make transitions more visually compelling and to emphasize the spatial relationships between the photographs.

5.3.1 Camera Motion

When the virtual camera moves from one photograph to another, the system linearly interpolates the camera position between the initial and final camera locations, and the camera orientation between unit quaternions representing the initial and final orientations. The field of view of the virtual camera is also interpolated so that when the camera reaches its destination, the destination image will fill as much of the screen as possible. The camera path timing is non-uniform, easing in and out of the transition.

If the camera moves as the result of an object selection (Sect. 6.3), the transition is slightly different. Before the camera starts moving, it orients itself to point at the mean of the selected points. The camera remains pointed at the mean as it moves, so that the selected object stays fixed in the view. This helps keep the object from undergoing large, distracting motions during the transition. The final orientation and focal length are computed so that the selected object is centered and fills the screen.

5.3.2 View Interpolation

During camera transitions, we also display in-between images. We have experimented with two simple techniques for morphing between the start and destination photographs: triangulating the point cloud and using planar impostors.

Triangulated Morphs To create a triangulated morph between two cameras C_j and C_k, we first compute a 2D Delaunay triangulation for image I_j using the projections of Points(C_j) into I_j. The projections of Lines(C_j) into I_j are imposed as edge constraints on the triangulation (Chew 1987). The resulting constrained Delaunay triangulation may not cover the entire image, so we overlay a grid onto the image and add to the triangulation each grid point not contained inside the original triangulation. Each added grid point is associated with a 3D point on Plane(C_j). The connectivity of the triangulation is then used to create a 3D mesh; we project I_j onto the mesh in order to texture map it. We compute a mesh for C_k and texture map it in the same way.

Then, to render the transition between C_j and C_k, we move the virtual camera from C_j and C_k while cross-fading between the two meshes (i.e., the texture-mapped mesh for C_j is faded out while the texture-mapped mesh for C_k is faded in, with the depth buffer turned off to avoid popping). While this technique does not use completely accurate geometry, the meshes are often sufficient to give a sense of the 3D geometry of the scene. For instance, this approach works well for many transitions in the Great Wall data set (shown as a still in Fig. 2, and as an animation in the video on the project website). However, missing geometry and outlying points can sometimes cause distracting artifacts.

Planar Morphs We have also experimented with using planes, rather than 3D meshes, as our projection surfaces. To create a morph between cameras C_j and C_k using a planar impostor, we simply project the two images I_j and I_k onto CommonPlane(C_j, C_k) and cross-fade between the projected images as the camera moves from C_j to C_k. The resulting in-betweens are not as faithful to the underlying geometry as the triangulated morphs, tending to stabilize only a dominant plane in the scene, but the resulting artifacts are usually less objectionable, perhaps because we are used to seeing distortions caused by viewing planes from different angles. Because of the robustness of this method, we prefer to use it rather than triangulation as the default for transitions. Example morphs using both techniques are shown in the video on our project website.[15]

[15] Photo tourism website, http://phototour.cs.washington.edu/.

There are a few special cases which must be handled differently during transitions. First, if the two cameras observe no common points, our system currently has no basis for interpolating the images. Instead, we fade out the start image, move the camera to the destination as usual, then fade in the destination image. Second, if the normal to CommonPlane(C_j, C_k) is nearly perpendicular to the average of the viewing directions of C_j and C_k, the projected images would undergo significant distortion during the morph. In this case, we revert to using a plane passing through the mean of the points common to both views, whose normal is the average of the viewing directions. Finally, if the vanishing line of CommonPlane(C_j, C_k) is visible in images I_j or I_k (as would be the case if this plane were the ground plane, and the horizon were visible in either image), it is impossible to project the entirety of I_j or I_k onto the plane. In this case, we project as much as possible of I_j and I_k onto the plane, and project the rest onto the plane at infinity.

6 Photo Explorer Navigation

Our image exploration tool supports several modes for navigating through the scene and finding interesting photographs. These modes include free-flight navigation, finding related views, object-based navigation, and viewing slideshows.

6.1 Free-Flight Navigation

The free-flight navigation controls include some of the standard 3D motion controls found in many games and 3D viewers. The user can move the virtual camera forward, back, left, right, up, and down, and can control pan, tilt, and zoom. This allows the user to freely move around the scene and provides a simple way to find interesting viewpoints and nearby photographs.

At any time, the user can click on a frustum in the main view, and the virtual camera will smoothly move until it is coincident with the selected camera. The virtual camera pans and zooms so that the selected image fills as much of the main view as possible.

6.2 Moving Between Related Views

When visiting a photograph C_{curr}, the user has a snapshot of the world from a single point of view and an instant in time. The user can pan and zoom to explore the photo, but might also want to see aspects of the scene beyond those captured in a single picture. He or she might wonder, for instance, what lies just outside the field of view, or to the left of the objects in the photo, or what the scene looks like at a different time of day.

To make it easier to find related views such as these, we provide the user with a set of "geometric" browsing tools. Icons associated with these tools appear in two rows in the information pane, which appears when the user is visiting a photograph. These tools find photos that depict parts of the scene with certain spatial relations to what is currently in view. The mechanism for implementing these search tools is to project the points observed by the current camera, Points(C_{curr}), into other photos (or vice versa), and select views based on the projected motion of the points. For instance, to answer the query *"show me what's to the left of this photo,"* we search for a photo in which Points(C_{curr}) appear to have moved right.

The geometric browsing tools fall into two categories: tools for selecting the *scale* at which to view the scene, and directional tools for looking in a particular direction (e.g., left or right).

There are three scaling tools: (1) find *details*, or higher-resolution close-ups, of the current photo, (2) find *similar* photos, and (3) find *zoom-outs*, or photos that show more surrounding context. If the current photo is C_{curr}, these tools search for appropriate neighboring photos C_j by estimating the relative "apparent size" of set of points in each image, and comparing these apparent sizes. Specifically, to estimate the apparent size of a set of points P in a image I, we project the points into I, compute the bounding box of the projections that are inside the image, and calculate the ratio of the area of the bounding box (in pixels) to the area of the image. We refer to this quantity as Size(P, C).

When one of these tools is activated, we classify each neighbor C_j as:

- a *detail* of C_{curr} if Size(Points(C_j), C_{curr}) < 0.75 and most points visible in C_{curr} are visible in C_j
- *similar* to C_{curr} if

$$0.75 < \frac{\text{Size}(\text{Points}(C_{\text{curr}}), C_j)}{\text{Size}(\text{Points}(C_{\text{curr}}), C_{\text{curr}})} < 1.3$$

  and the angle between the viewing directions of C_{curr} and C_j is less than a threshold of 10 degrees
- a *zoom-out* of C_{curr} if C_{curr} is a detail of C_j.

The results of any of these searches are displayed in the thumbnail pane (sorted by increasing apparent size, in the case of details and zoom-outs). These tools are useful for viewing the scene in more detail, comparing similar views of an object which differ in other respects, such as time of day, season, and year, and for "stepping back" to see more of the scene.

The directional tools give the user a simple way to "step" left or right, i.e., to see more of the scene in a particular direction. For each camera, we compute a left and right neighbor, and link them to arrows displayed in the information pane. To find a left and right image for camera C_j,

Fig. 5 *Object-based navigation.* The user drags a rectangle around Neptune in one photo, and the system finds a new, high-resolution photograph

we compute the average 2D motion m_{jk} of the projections of Points(C_j) from image I_j to each neighboring image I_k. If the angle between m_{jk} and the desired direction (i.e., left or right), is small, and the apparent sizes of Points(C_j) in both images are similar, C_k is a candidate left or right image to C_j. Out of all the candidates, we select the left of right image to be the image I_k whose motion magnitude $\|m_{jk}\|$ is closest to 20% of the width of image I_j.

6.3 Object-Based Navigation

Another search query our system supports is *"show me photos of this object,"* where the object in question can be directly selected in a photograph or in the point cloud. This type of search, applied to video in (Sivic and Zisserman 2003), is complementary to, and has certain advantages over, keyword search. Being able to select an object is especially useful when exploring a scene—when the user comes across an interesting object, direct selection is an intuitive way to find a better picture of that object.

In our photo exploration system, the user selects an object by dragging a 2D box around a region of the current photo or the point cloud. All points whose projections are inside the box form the set of selected points, S. Our system then searches for the "best" photo of S by scoring each image in the database based on how well it represents the selection. The top scoring photo is chosen as the representative view, and the virtual camera is moved to that image. Other images with scores above a threshold are displayed in the thumbnail pane, sorted in descending order by score. An example object selection interaction is shown in Fig. 5.

Our view scoring function is based on three criteria: (1) the visibility of the points in S, (2) the angle from which the points in S are viewed, and (3) the image resolution. For each image I_j, we compute the score as a weighted sum of three terms, E_{visible}, E_{angle}, and E_{detail}. Details of the computation of these terms can be found in Appendix 2.

The set S can sometimes contain points that the user did not intend to select, especially occluded points that happen to project inside the selection rectangle. If we had complete knowledge of visibility, we could cull such hidden points. Because we only have a sparse model, however, we use a set of heuristics to prune the selection. If the selection was made while visiting an image I_j, we can use the points that are known to be visible from that viewpoint (Points(C_j)) to refine the selection. In particular, we compute the 3×3 covariance matrix for the points in $S \cap \text{Points}(C_j)$, and remove all from S all points with a Mahalanobis distance greater than 1.2 from the mean. If the selection was made while not visiting an image, we instead compute a weighted mean and covariance matrix for the entire set S. The weighting favors points which are closer to the virtual camera, the idea being that those are more likely to be unoccluded than points which are far away. Thus, the weight for each point is computed as the inverse of its distance from the virtual camera.

6.4 Creating Stabilized Slideshows

Whenever the thumbnail pane contains more than one image, its contents can be viewed as a slideshow by pressing the "play" button in the pane. By default, the virtual camera will move through space from camera to camera, pausing at each image for a few seconds before proceeding to the next. The user can also "lock" the camera, fixing it to the its current position, orientation, and field of view. When the images in the thumbnail pane are all taken from approximately the same location, this mode stabilizes the images, making it easier to compare one image to the next. This mode is useful for studying changes in scene appearance as a function of time of day, season, year, weather patterns, etc. An example stabilized slideshow from the Yosemite data set is shown in the companion video.[16]

6.5 Photosynth

Our work on visualization of unordered photo collections is being used in the Photosynth Technology Preview[17] released by Microsoft Live Labs. Photosynth is a photo visualization tool that uses the same underlying data (camera

[16] Photo tourism website, http://phototour.cs.washington.edu/.

[17] Microsoft Live Labs, Photosynth technology preview, http://labs.live.com/photosynth.

positions and points) as in our work and has a user interface with many similarities to Photo Tourism, but also has several important differences. In Photosynth, rather than the user dragging a box around an object to see a detailed photo of it, the system suggests different photos as the user moves the mouse cursor around the screen. If a close-up of the object that the cursor is hovering over is available, a semi-transparent quadrilateral appears, highlighting that region of the screen. When the user clicks on a quadrilateral the virtual view moves to the selected photo. Photosynth also supports a "splatter" mode, in which the photos are viewed in a 2D arrangement. The currently selected photo is placed in the center of the arrangement, and the other photos are ordered by similarity in a spiral around the center photo. In the 2D mode, when a new photo is selected it becomes the new center image, and the photos are rearranged smoothly. Transitions between 2D and 3D modes involve similarly fluid rearrangement of the photos.

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 is then sent over the network, and higher-resolution data is smoothly blended onto the screen as it is received.

7 Enhancing Scenes

Our system allows users to add content to a scene in several ways. First, the user can register their own photographs to the scene at run-time, after the initial set of photos has been registered. Second, users can annotate regions of images, and these annotations can be propagated to other images.

7.1 Registering New Photographs

New photographs can be registered on the fly, as follows. First, the user switches to a mode where an overhead map fills the view, opens a set of images, which are displayed in the thumbnail panel, and drags and drops each image onto its approximate location on the map. After each image has been dropped, the system estimates the location, orientation, and focal length of each new photo by running an abbreviated version of the SfM pipeline described in Sect. 4 at a local level. First, SIFT keypoints are extracted and matched to the keypoints of the twenty cameras closest to the initial location; the matches to each other camera are pruned to contain geometrically consistent matches; the existing 3D points corresponding to the matches are identified; and finally, these matches are used to refine the pose of the new camera. After a set of photos has been dragged onto the map, it generally takes around ten seconds to optimize the parameters for each new camera on our test machine, a 3.80 GHz Intel Pentium 4.

7.2 Annotating Objects

Annotations are supported in other photo organizing tools, but a unique feature of our system is that annotations can be automatically *transferred* from one image to all other images that contain the same scene region(s).

In the photo explorer, the user can select a region of an image and enter a textual annotation. The annotation is then stored, along with the 3D points S_{ann} which lie in the selected area, and appears as a semi-transparent box around the selected points. Once annotated, an object can be linked to other sources of information, such as web sites, guidebooks, and video and audio clips.

When an annotation is created, it is automatically transferred to all other relevant photographs. To transfer an annotation to another image I_j, we first check whether the annotation is visible in I_j, and whether it is at an appropriate scale for the image—that it neither fills the image entirely nor labels a very small region of the image. To determine visibility, we simply test that at least one of the annotated points S_{ann} is in Points(C_j). To check whether the annotation is at an appropriate scale, we compute the apparent size, Size(S_{ann}, C_j), of the annotation in image I_j. If the annotation is visible and $0.05 < \text{Size}(S_{\text{ann}}, C_j) < 0.8$, we transfer the annotation to C_j. When the user visits C_j, the annotation is displayed as a box around the annotated points, as shown in Fig. 7.

Besides quickly enhancing a scene with semantic information, the ability to transfer annotations has several applications. First, it enables a system in which a tourist can take a photo (e.g., from a camera phone that runs our software) and instantly see information about objects in the scene super-imposed on the image. In combination with a head-mounted display, such a capability could offer a highly portable, computer-vision-based augmented reality system (Feiner et al. 1997). Second, it makes labeling photographs in preparation for keyword search more efficient. If an object is annotated with a set of keywords in one photo, transferring the annotation to other photos enables multiple images to be added to a keyword search database based on a single annotation.

We can also leverage the many existing images that have already been annotated. There are several sources of existing annotations. On Flickr, for instance, users can attach notes to rectangular regions of photos. Tools such as the ESP Game (von Ahn and Dabbish 2004) and LabelMe (Russell et al. 2005) encourage users to label images on the web, and have accumulated a database of annotations. By registering

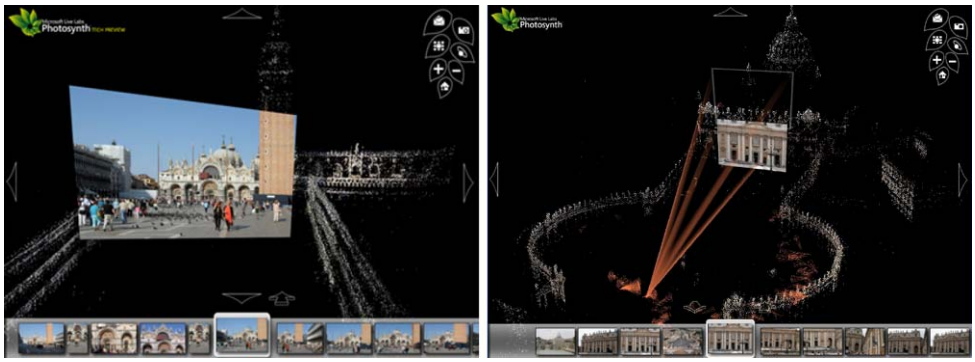

Fig. 6 Photosynth technology preview. This screenshot shows a user exploring photos of Piazza San Marco in Venice (*left*) and St. Peter's Basilica in the Vatican (*right*)

Fig. 7 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

such labeled images with an existing collection of photos using our system, we could transfer the existing labels to every other relevant photo in the system. Other images on the web are implicitly annotated: for instance, an image on a Wikipedia page is "annotated" with the URL of that page. By registering such images, we could link other photos to the same page.

8 Results

We have applied our system to several input photo collections, including "uncontrolled" sets consisting of images downloaded from Flickr. In each case, our system detected and matched features on the entire set of photos and automatically identified and registered a subset corresponding to one connected component of the scene. The uncontrolled sets we have tested are as follows:

1. **Notre Dame**, a set of photos of the Notre Dame Cathedral in Paris.
2. **Mount Rushmore**, a set of photos of Mount Rushmore National Monument, South Dakota.
3. **Trafalgar Square**, a set of photos from Trafalgar Square, London.
4. **Yosemite**, a set of photos of Half Dome in Yosemite National Park.
5. **Trevi Fountain**, a set of photos of the Trevi Fountain in Rome.
6. **Sphinx**, a set of photos of the Great Sphinx of Giza, Egypt.
7. **St. Basil's**, a set of photos of Saint Basil's Cathedral in Moscow.
8. **Colosseum**, a set of photos of the Colosseum in Rome.

Three other sets were taken in more controlled settings (i.e., a single person with a single camera and lens):

1. **Prague**, a set of photos of the Old Town Square in Prague.
2. **Annecy**, a set of photos of a street in Annecy, France.
3. **Great Wall**, a set of photos taken along the Great Wall of China.

More information about these data sets (including the number of input photos, number of registered photos, running time, and average reprojection error), is shown in Table 1. The running times reported in this table were generated by running the complete pipeline on one or more 3.80 GHz Intel Pentium 4 processors. The keypoint detection and matching phases were run in parallel on ten processors, and the structure from motion algorithm was run on a single processor.

Visualizations of these data sets are shown in Figs. 9 and 10. Please see the video and live demo on the project website[18] for a demonstration of features of our photo ex-

[18] Photo tourism website, http://phototour.cs.washington.edu/.

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 comparison (DEM and color texture courtesy of the U.S. Geological Survey)

Table 1 Data sets. Each row lists information about each data set used

Collection	Search term	# photos	# registered	# points	runtime	error
Notre Dame	notredame AND paris	2635	598	305535	12.7 days	0.616
Mt. Rushmore	mount rushmore	1000	452	133994	2.6 days	0.444
Trafalgar Sq.	trafalgar square	1893	278	27224	3.5 days	1.192
Yosemite	halfdome AND yosemite	1882	678	264743	10.4 days	0.757
Trevi Fountain	trevi AND rome	466	370	114742	20.5 hrs	0.698
Sphinx	sphinx AND egypt	1000	511	206783	3.4 days	0.418
St. Basil's	basil AND red square	627	220	25782	23.0 hrs	0.816
Colosseum	colosseum AND (rome OR roma)	1994	390	188306	5.0 days	1.360
Prague	N/A	197	171	38921	3.1 hrs	0.731
Annecy	N/A	462	424	196443	2.5 days	0.810
Great Wall	N/A	120	81	24225	2.8 hrs	0.707

Collection, the name of the set; *search term* the search term used to gather the images; *# photos*, the number of photos in the input set; *# registered* the number of photos registered; *# points*, the number of points in the final reconstruction; *runtime*, the approximate total time for reconstruction; *error*, the mean reprojection error, in pixels, after optimization. The first eight data sets were gathered from the Internet, and the last three were each captured by a single person

plorer, including object selection, related image selection, morphing, and annotation transfer, on several data sets.

For the Half Dome data set, after initially constructing the model, we aligned it to a digital elevation map using the approach described in Sect. 4.3.1. We then registered a historical photo, Ansel Adam's "Moon and Half Dome," to the data set, by dragging and dropping it onto the model using the method described in Sect. 7.1. Figure 8 shows a synthetic rendering of the scene from the estimated position where Ansel Adams took the photo.

8.1 Discussion

The graphs shown in the third column of Figs. 9 and 10 contain edges between pairs of photos with matching features, as in Fig. 1. These connectivity graphs suggest that many of the uncontrolled datasets we tested our reconstruction algorithm on consist of several large clusters of photos with a small number of connections spanning clusters, and a sparse set of photos hanging off the main clusters. The large clusters usually correspond to sets of photos from similar viewpoints. For instance, the large cluster that dominates the **Mount Rushmore** connectivity graph are all frontal-view photos taken from the observation terrace or the trails around it, and the two large clusters on the right side of the **Colosseum** connectivity graph correspond to the inside and the outside of the Colosseum. Sometimes clusters correspond not to viewpoint but to different lighting conditions, as in the case of the **Trevi Fountain** collection (see Fig. 1), where there is a "daytime" cluster and a "nighttime" cluster.

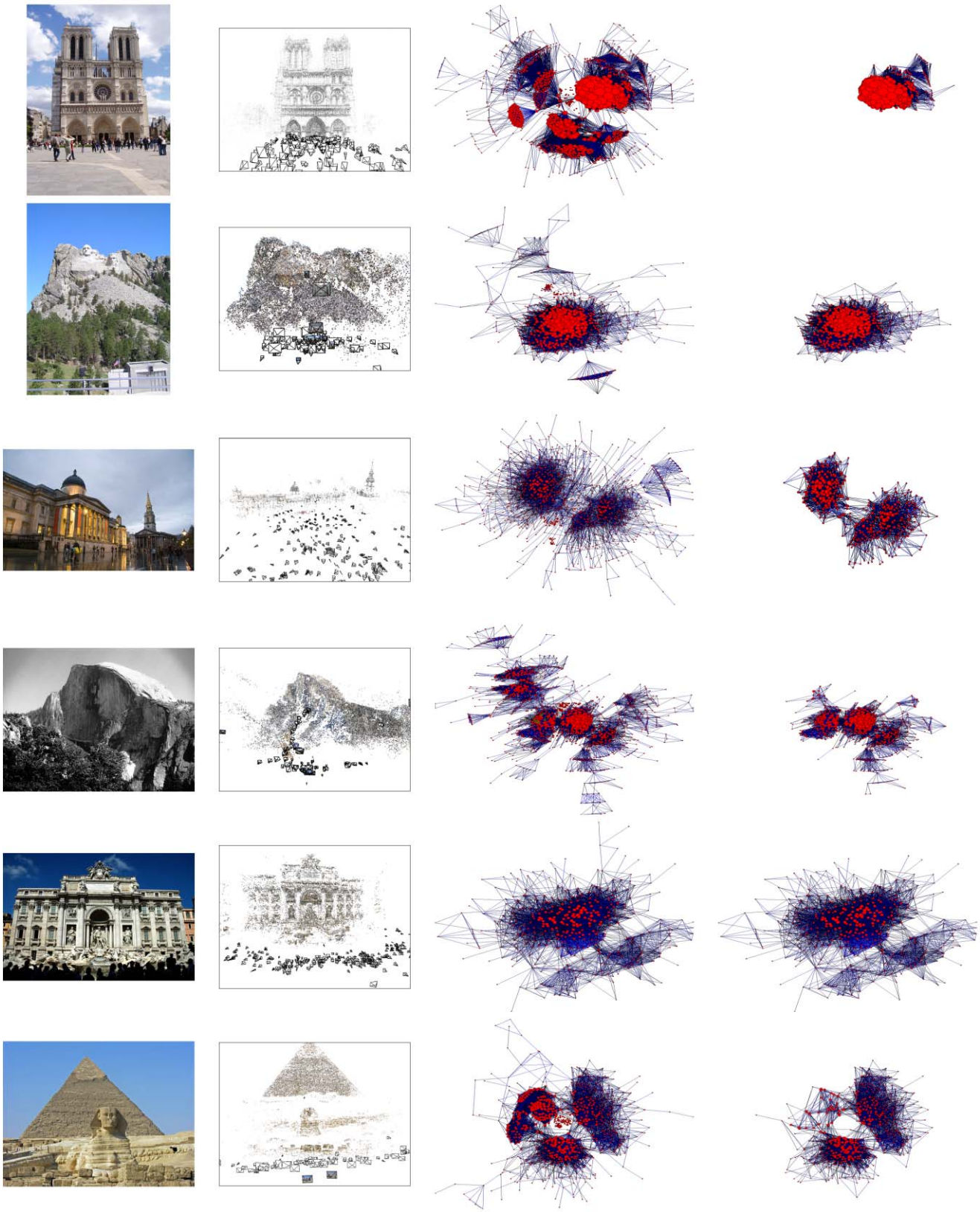

Fig. 9 Sample reconstructed scenes. *From top to bottom*: **Notre Dame**, **Mount Rushmore**, **Trafalgar Square**, **Yosemite**, **Trevi Fountain**, and **Sphinx**. The *first column* shows a sample image, and the *second column* shows a view of the reconstruction. The *third and fourth columns* show photo connectivity graphs, in which each image in the set is a node and an edge links each pair of images with feature matches. The *third column* shows the photo connectivity graph for the full image set, and the fourth for the subset of photos that were ultimately reconstructed

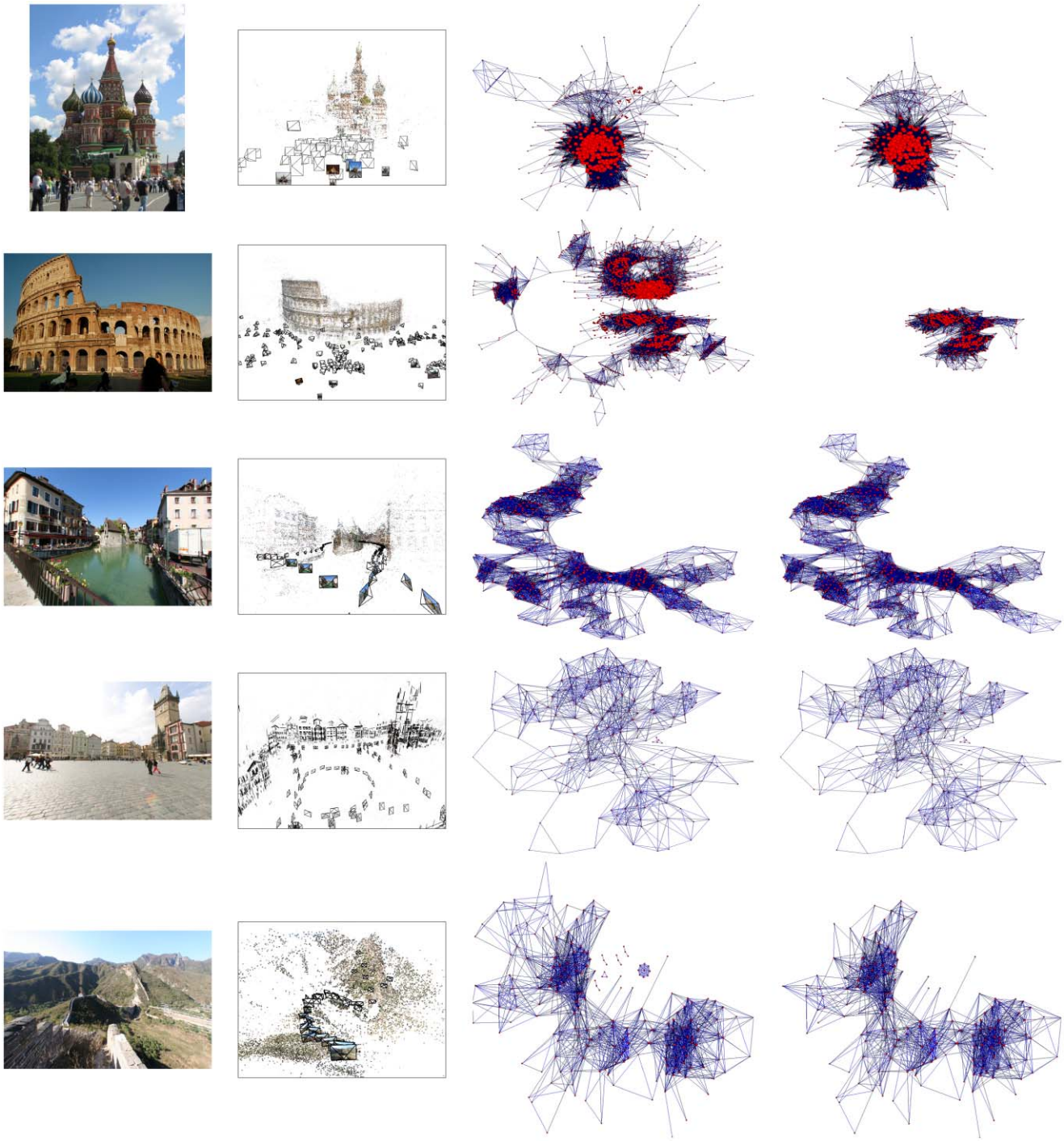

Fig. 10 More sample reconstructed scenes. *From top to bottom*: **St. Basil's**, **Colosseum**, **Annecy**, **Prague** and **Great Wall**. The *last three photo collections* were each taken by a single person

Inside of a cluster, the *neato*[19] mass-spring system used to embed the graph into the plane tends to pull together the most similar photos and push apart the least similar ones. This behavior can result in photos being laid out along intuitive dimensions. For instance, in the large cluster at the top of the connectivity graph for the **Trevi Fountain** dataset, as well as the sparser cluster on the bottom, the x-axis roughly corresponds to the angle from which the fountain is viewed. The photos that *span* clusters tend to be those that happen to see two parts of a scene (for instance, both the inside and

[19] Graphviz—graph visualization software, http://www.graphviz.org/.

outside of a building), or those that are sharp, in focus, and well-lit, in the case of clusters taken at different times of day. The "leaf" nodes in the graph generally correspond to images that are at extremes along some dimension, such as photos that are very zoomed in, photos taken from a significantly different viewpoint, or photos taken under very different lighting.

While the graphs in the third column of Figs. 9 and 10 represent the connectivity of entire photo sets, the fourth column shows the part of the graph our algorithm was able to reconstruct (the *reconstruction graph*). As described in Sect. 4, in general, our algorithm does not reconstruct all input photos, because the input set may form separate connected components, or clusters that are too weakly connected to be reliable reconstructed (during reconstruction, photos are added until no remaining photo observes enough 3D points to reliably add it to the scene). These reconstruction graphs suggest that for unstructured datasets, our reconstruction algorithm tends to reconstruct most of one of the main clusters, and can sometimes bridge gaps between clusters with enough connections between them. For instance, in the **Sphinx** collection, our algorithm reconstructed two prominent clusters, one on the right side of the graph, and one on the bottom. These clusters correspond to two sides of the Sphinx (the front and the right side) which are commonly photographed; a few photos were taken from intermediate angles, allowing the two clusters to be connected. In the **Colosseum** collection, only the outside of the structure was successfully reconstructed, and therefore only a single cluster in the connectivity graph is represented in the reconstruction graph. More images would be needed to bridge the other clusters in the graph. In general, the more "connected" the image graph is, the more images can successfully be registered.

For the controlled datasets (**Annecy**, **Prague**, and **Great Wall**), the photos were captured with the intention of generating a reconstruction from them, and the connectivity graphs are less clustered, as they were taken, for the most part, while walking along a path. In the *Prague* photo set, for instance, most of the photos were taken all around the Old Town Square, looking outward at the buildings. A few were taken looking across the square, so a few longer range connections between parts of the graph are evident. Our reconstruction algorithm was able to register most of the photos in these datasets.

9 Research Challenges

A primary objective of this paper is to motivate more research in the computer vision community on analyzing the diverse and massive photo collections available on the Internet. While this paper presented some initial steps towards processing such imagery for the purpose of reconstruction and visualization, huge challenges remain. Some of the open research problems for our community include:

- **Scale.** As more and more of the world's sites and cities are captured photographically, we can imagine using SfM methods to reconstruct a significant portion of the world's urban areas. Achieving such a goal will require *Internet-scale* matching and reconstruction algorithms that operate on millions of images. Some recent matching algorithms (Grauman and Darrell 2005; Nistér and Stewénius 2006) have demonstrated the ability to operate effectively on datasets that approach this scale, although more work is needed to improve the accuracy of these methods, especially on Internet 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. These and other factors lead us to believe that Internet-scale SfM is feasible.
- **Variability.** While SIFT and other feature matching techniques are surprisingly robust with respect to appearance variation, more significant appearance changes still pose a problem. More robust matching algorithms could enable a number of exciting capabilities, such as matching ground-based to aerial/satellite views, aligning images of natural sites through changes in seasons and weather conditions, registering historical photos and artistic renderings with modern-day views (*rephotography*[20]), and robust matching using low-quality (e.g., cell phone camera) devices and imagery.
- **Accuracy.** Accuracy is an important concern for applications such as localization (e.g., figuring out where you are by taking a photograph), navigation, and surveillance. Most SfM methods operate by minimizing reprojection error and do not provide guarantees on metric accuracy. However, satellite images, maps, DEMs, surveys, and similar data provide a rich source of metric data for a large percentage of world sites; such data could be used to obtain more accurate metric SfM results. There is also a need for evaluation studies, in the spirit of (Scharstein and Szeliski 2002; Seitz et al. 2006; Szeliski et al. 2006), that benchmark the best-of-breed SfM algorithms against ground truth datasets and encourage the development of more accurate techniques.
- **Shape.** While SfM techniques provide only sparse geometry, the ability to compute accurate camera parameters opens the door for techniques such as multi-view stereo that compute dense surface shape models. In turn, shape enables computing scene reflectance properties (e.g., BRDFs) and illumination. We therefore envision a new breed of shape and reflectance modeling techniques

[20] Rephotography, http://en.wikipedia.org/wiki/Rephotography.

that operate robustly on the free-form imagery typical in Internet photo collections (see Goesele et al. 2007 for exciting initial steps along these lines).

- **Online algorithms.** Imagine pointing your cell phone camera at a scene of interest, and have it instantly recognize where you are and annotate the image in real time with information about objects visible in the image. This and a number of other applications become feasible only with online algorithms that produce results in real time. Similarly, one can imagine highlighting areas of the world that are not sufficiently covered with photographs, and provide a way for users to take photos of these areas, upload the results, and see the resulting refinement to the model. (The *Geograph British Isles* project[21] uses manual geotagging to get at least one representative photograph for each square kilometer of the British Isles.) This is, in effect, a form of active vision (Aloimonos 1993; Blake and Yuille 1993) but on a massive scale, where people in the world population are the active agents filling in the missing data.

In conclusion, we believe Internet imagery provides very fertile ground for computer vision research. The massive amount of data available on the Internet is starting to be used to address several problems in computer graphics and computer vision, including object category recognition, (Fergus et al. 2005), scene completion (Hays and Efros 2007), and object insertion (Lalonde et al. 2007). We expect to see major advances in this area over the next few years.

Appendix 1: Structure from Motion Optimization

A perspective camera can be parameterized by an eleven-parameter projection matrix. Making the common additional assumptions that the pixels are square and that the center of projection is coincident with the image center, the number of parameters is reduced to seven: the 3D orientation (three parameters), the camera center c (three parameters), and the focal length f (one parameter). In our system, we also solve for two radial distortion parameters, κ_1 and κ_2, so the total number of parameters per camera is nine.

To parameterize the rotations, we use an incremental rotation, ω, where

$$\mathbf{R}(\theta, \hat{n}) = \mathbf{I} + \sin\theta[\hat{n}]_\times + (1 - \cos\theta)[\hat{n}]_\times^2, \qquad \omega = \theta\hat{n}$$

is the incremental rotation matrix applied to an initial rotation, and

$$[\hat{n}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}.$$

[21] Geograph British Isles project website, http://www.geograph.org.uk/.

We group the nine parameters into a vector, $\Theta = [\omega, c, f, \kappa_1, \kappa_2]$. Each point is parameterized by a 3D position, X.

Recovering the parameters can be formulated as an optimization problem. In particular, we have a set of n cameras, parameterized by Θ_i. We also have a set of m tracks, parameterized by X_j, and a set of 2D projections, q_{ij}, where q_{ij} is the observed projection of the j-th track in the i-th camera.

Let $P(\Theta, X)$ be the equation mapping a 3D point X to its 2D projection in a camera with parameters Θ. P transforms X to homogeneous image coordinates and performs the perspective division, that applies radial distortion:

$$X' = \mathbf{R}(X - c),$$
$$P_0 = \left[-fX'_x/X'_z \; -fX'_y/X'_z \right]^T,$$
$$P = g_{rd}(P_0)$$

where $g_{rd}(p)$ is the distortion equation given in Sect. 4.2. We wish to minimize the sum of the reprojection errors:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}\|q_{ij} - P(\Theta_i, X_j)\|$$

(w_{ij} is used as an indicator variable where $w_{ij} = 1$ if camera i observes point j, and $w_{ij} = 0$ otherwise).

When initializing a new camera, we have one or two independent estimates of the focal length. One estimate, f_1, is $\frac{1}{2}(\mathbf{K}_{11} + \mathbf{K}_{22})$, where $\mathbf{K}$ is the intrinsic matrix estimated using DLT. The other, f_2, is calculated from the EXIF tags of an image, and is undefined if the necessary EXIF tags are absent. This estimate is usually good, but can occasionally be off by more than a factor of two. We prefer to use f_2 as initialization when it exists, but first we check that $0.7f_1 < f_2 < 1.4f_1$ to make sure f_2 does not disagree to much with the DLT estimate. If this test fails, we use f_1. Otherwise, we use f_2, and add the term $\gamma(f - f_2)^2$ to the objective function, in order to keep the focal length of the camera close to the initial estimate. We used $\gamma = 0.001$ for each of our tests.

Appendix 2: Image Selection Criteria

When determining how well an image I_j represents a point set S during an object selection, the score is computed as a weighted sum of three terms, $E_{\text{visible}} + \alpha E_{\text{angle}} + \beta E_{\text{detail}}$ (we use $\alpha = \frac{1}{3}$ and $\beta = \frac{2}{3}$).

To compute the visibility term E_{visible}, we first check whether $S \cap \text{Points}(C_j)$ is empty. If so, the object is deemed not to be visible to C_j at all, and $E_{\text{visible}} = -\infty$. Otherwise, $E_{\text{visible}} = \frac{n_{\text{inside}}}{|S|}$, where n_{inside} denotes the number of points in S that project inside the boundary of image I_j.

The term E_{angle} is used to favor head-on views of a set of points over oblique views. To compute E_{angle}, we first fit a plane to the points in S using a RANSAC procedure. If the percentage of points in S which are inliers to the recovered plane is above a threshold of 50% (i.e., there appears to be a dominant plane in the selection), we favor cameras that view the object head-on by setting $E_{\text{angle}} = V(C_j) \cdot \hat{n}$, where V indicates viewing direction, and $\hat{n}$ the normal to the recovered plane. If fewer than 50% of the points fit the plane, we set $E_{\text{angle}} = 0$.

Finally, E_{detail} favors high-resolution views of the object. E_{detail} is defined to be the area, in pixels, of the bounding box of the projections of S into image I_j (considering only points that project inside the boundary of I_j). E_{detail} is normalized by the area of the largest such bounding box, so the highest resolution available view will have a score of 1.0.

Appendix 3: Photo Credits

We would like to thank the following people for allowing us to reproduce their photographs:

Holly Ables, of Nashville, TN
Rakesh Agrawal
Pedro Alcocer
Julien Avarre (http://www.flickr.com/photos/eole/)
Rael Bennett (http://www.flickr.com/photos/spooky05/)
Loïc Bernard
Nicole Bratt
Nicholas Brown
Domenico Calojero (mikuzz@gmail.com)
DeGanta Choudhury (http://www.flickr.com/photos/deganta/)
dan clegg
Claude Covo-Farchi
Alper Çuğun
W. Garth Davis
Stamatia Eliakis
Dawn Endico (endico@gmail.com)
Silvana M. Felix
Jeroen Hamers
Caroline Härdter
Mary Harrsch
Molly Hazelton
Bill Jennings (http://www.flickr.com/photos/mrjennings), supported by grants from the National Endowment for the Humanities and the Fund for Teachers
Michelle Joo
Tommy Keswick
Kirsten Gilbert Krenicky
Giampaolo Macorig
Erin K Malone (photographs copyright 2005)
Daryoush Mansouri
Paul Meidinger
Laurete de Albuquerque Mouazan
Callie Neylan
Robert Norman
Dirk Olbertz
Dave Ortman
George Owens
Claire Elizabeth Poulin
David R. Preston
Jim Sellers and **Laura Kluver**
Peter Snowling
Rom Srinivasan
Jeff Allen Wallen Photographer/Photography
Daniel West
Todd A. Van Zandt
Dario Zappalà
Susan Elnadi

We also acknowledge the following people whose photographs we reproduced under Creative Commons licenses:

Shoshanah
http://www.flickr.com/photos/shoshanah[22]
Dan Kamminga
http://www.flickr.com/photos/dankamminga[1]
Tjeerd Wiersma http://www.flickr.com/photos/tjeerd[1]
Manogamo
http://www.flickr.com/photos/se-a-vida-e[23]
Ted Wang http://www.flickr.com/photos/mtwang[24]
Arnet http://www.flickr.com/photos/gurvan[3]
Rebekah Martin http://www.flickr.com/photos/rebekah[3]
Jean Ruaud http://www.flickr.com/photos/jrparis[3]
Imran Ali http://www.flickr.com/photos/imran[3]
Scott Goldblatt http://www.flickr.com/photos/goldblatt[3]
Todd Martin http://www.flickr.com/photos/tmartin[25]
Steven http://www.flickr.com/photos/graye[4]
ceriess http://www.flickr.com/photos/ceriess[1]
Cory Piña http://www.flickr.com/photos/corypina[3]
mark gallagher
http://www.flickr.com/photos/markgallagher[1]
Celia http://www.flickr.com/photos/100n30th[3]
Carlo B. http://www.flickr.com/photos/brodo[3]
Kurt Naks http://www.flickr.com/photos/kurtnaks[4]
Anthony M. http://www.flickr.com/photos/antmoose[1]
Virginia G http://www.flickr.com/photos/vgasull[3]

Collection credit and copyright notice for *Moon and Half Dome*, 1960, by Ansel Adams: Collection Center for Cre-

[1] http://creativecommons.org/licenses/by/2.0/.

[2] http://creativecommons.org/licenses/by-nd/2.0/.

[3] http://creativecommons.org/licenses/by-nc-nd/2.0/.

[4] http://creativecommons.org/licenses/by-nc/2.0/.

ative Photography, University of Arizona, © Trustees of The Ansel Adams Publishing Rights Trust.

References

Akbarzadeh, A., Frahm, J.-M., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Merrell, P., Phelps, M., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., Towles, H., Nistér, D., & Pollefeys, M. (2006). Towards urban 3D reconstruction from video. In *Proceedings of the international symposium on 3D data processing, visualization, and transmission*.

Aliaga, D. G. et al. (2003). Sea of images. *IEEE Computer Graphics and Applications*, *23*(6), 22–30.

Aliaga, D., Yanovsky, D., Funkhouser, T., & Carlbom, I. (2003). Interactive image-based rendering using feature globalization. In *Proceedings of the SIGGRAPH symposium on interactive 3D graphics* (pp. 163–170).

Aloimonos, Y. (Ed.). (1993). *Active perception*. Mahwah: Lawrence Erlbaum Associates.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, *45*(6), 891–923.

Baumberg, A. (2000). Reliable feature matching across widely separated views. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (Vol. 1, pp. 774–781), June 2000.

Blake, A., & Yuille, A. (Eds.). (1993). *Active vision*. Cambridge: MIT Press.

Brown, M., & Lowe, D. G. (2005). Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Proceedings of the international conference on 3D digital imaging and modelling* (pp. 56–63).

Buehler, C., Bosse, M., McMillan, L., Gortler, S., & Cohen, M. (2001). Unstructured lumigraph rendering. In *SIGGRAPH conference proceedings* (pp. 425–432).

Chen, S., & Williams, L. (1993). View interpolation for image synthesis. In *SIGGRAPH conference proceedings* (pp. 279–288).

Chew, L. P. (1987). Constrained Delaunay triangulations. In *Proceedings of the symposium on computational geometry* (pp. 215–222).

Cooper, M., Foote, J., Girgensohn, A., & Wilcox, L. (2003). Temporal event clustering for digital photo collections. In *Proceedings of the ACM international conference on multimedia* (pp. 364–373).

Debevec, P. E., Taylor, C. J., & Malik, J. (1996). Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH conference proceedings* (pp. 11–20).

Dick, A. R., Torr, P. H. S., & Cipolla, R. (2004). Modelling and interpretation of architecture from several images. *International Journal of Computer Vision*, *60*(2), 111–134.

Feiner, S., MacIntyre, B., Hollerer, T., & Webster, A. (1997). A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In *Proceedings of the IEEE international symposium on wearable computers* (pp. 74–81).

Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2005). Learning object categories from Google's image search. In *Proceedings of the international conference on computer vision* (Vol. 2, pp. 816–823), October 2005.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fitzgibbon, A. W., & Zisserman, A. Automatic camera recovery for closed and open image sequences. In *Proceedings of the European conference on computer vision* (pp. 311–326), June 1998.

Förstner, W. (1986). A feature-based correspondence algorithm for image matching. *International Archives Photogrammetry & Remote Sensing*, *26*(3), 150–166.

Goesele, M., Snavely, N., Seitz, S. M., Curless, B., & Hoppe, H. (2007, to appear). Multi-view stereo for community photo collections. In *Proceedings of the international conference on computer vision*.

Gortler, S. J., Grzeszczuk, R., Szeliski, R., & Cohen, M. F. (1996). The lumigraph. In *SIGGRAPH conference proceedings* (pp. 43–54), August 1996.

Grauman, K., & Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. In *Proceedings of the international conference on computer vision* (pp. 1458–1465).

Grzeszczuk, R. (2002). Course 44: image-based modeling. In *SIGGRAPH 2002*

Hannah, M. J. (1988). Test results from SRI's stereo system. In *Image understanding workshop* (pp. 740–744), Cambridge, MA, April 1988. Los Altos: Morgan Kaufmann.

Harris, C., & Stephens, M. J. (1988). A combined corner and edge detector. In *Alvey vision conference* (pp. 147–152).

Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(6), 580–593.

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry*. Cambridge: Cambridge University Press.

Hays, J., & Efros, A. A. (2007). Scene completion using millions of photographs. In *SIGGRAPH conference proceedings*.

Irani, M., & Anandan, P. (1998). Video indexing based on mosaic representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *86*(5), 905–921.

Johansson, B., & Cipolla, R. (2002). A system for automatic pose-estimation from a single image in a city scene. In *Proceedings of the IASTED international conference on signal processing, pattern recognition and applications*.

Kadir, T., & Brady, M. (2001). Saliency, scale and image description. *International Journal of Computer Vision*, *45*(2), 83–105.

Kadobayashi, R., & Tanaka, K. (2005). 3D viewpoint-based photo search and information browsing. In *Proceedings of the ACM international conference on research and development in information retrieval* (pp. 621–622).

Lalonde, J.-F., Hoiem, D., Efros, A. A., Rother, C., Winn, J., & Criminisi, A. (2007). Photo clip art. In *SIGGRAPH conference proceedings*.

Levoy, M., & Hanrahan, P. (1996). Light field rendering. In *SIGGRAPH conference proceedings* (pp. 31–42).

Lippman, A. (1980). Movie maps: an application of the optical videodisc to computer graphics. In *SIGGRAPH conference proceedings* (pp. 32–43).

Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, *293*, 133–135.

Lourakis, M., & Argyros, A. (2004). *The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg–Marquardt algorithm* (Technical Report 340). Inst. of Computer Science-FORTH, Heraklion, Crete, Greece. Available from www.ics.forth.gr/~lourakis/sba.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application in stereo vision. In *International joint conference on artificial Intelligence* (pp. 674–679).

Matas, J. et al. (2004). Robust wide baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, *22*(10), 761–767.

McCurdy, N., & Griswold, W. (2005). A systems architecture for ubiquitous video. In *Proceedings of the international conference on mobile systems, applications, and services* (pp. 1–14).

McMillan, L., & Bishop, G. (1995) Plenoptic modeling: An image-based rendering system. In *SIGGRAPH conference proceedings* (pp. 39–46).

Mikolajczyk, K., & Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, *60*(1), 63–86.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & van Gool, L. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, *65*(1/2), 43–72.

Moravec, H. (1983). The Stanford cart and the CMU rover. *Proceedings of the IEEE*, *71*(7), 872–884.

Naaman, M., Paepcke, A., & Garcia-Molina, H. (2003). From where to what: Metadata sharing for digital photographs with geographic coordinates. In *Proceedings of the international conference on cooperative information systems* (pp. 196–217).

Naaman, M., Song, Y. J., Paepcke, A., & Garcia-Molina, H. (2004). Automatic organization for digital photographs with geographic coordinates. In *Proceedings of the ACM/IEEE-CS joint conference on digital libraries* (pp. 53–62).

Nistér, D. (2000). Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *Proceedings of the European conference on computer vision* (pp. 649–663).

Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(6), 756–777.

Nistér, D., & Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2118–2125).

Nocedal, J., & Wright, S. J. (1999). *Springer series in operations research. Numerical optimization*. New York: Springer.

Oliensis, J. (1999). A multi-frame structure-from-motion algorithm under perspective projection. *International Journal of Computer Vision*, *34*(2–3), 163–192.

Pollefeys, M., Koch, R., & Van Gool, L. (1999). Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *International Journal of Computer Vision*, *32*(1), 7–25.

Pollefeys, M., & Van Gool, L. (2002). From images to 3D models. *Communications of the ACM*, *45*(7), 50–55.

Pollefeys, M., van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., & Koch, R. (2004). Visual modeling with a hand-held camera. *International Journal of Computer Vision*, *59*(3), 207–232.

Robertson, D. P., & Cipolla, R. (2002). Building architectural models from many views using map constraints. In *Proceedings of the European conference on computer vision* (Vol. II, pp. 155–169).

Rodden, K., & Wood, K. R. (2003). How do people manage their digital photographs? In *Proceedings of the conference on human factors in computing systems* (pp. 409–416).

Román, A., et al. (2004). Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE visualization 2004* (pp. 537–544).

Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2005). *Labelme: a database and web-based tool for image annotation* (Technical Report MIT-CSAIL-TR-2005-056). Massachusetts Institute of Technology.

Schaffalitzky, F., & Zisserman, A. (2002). Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?" In *Proceedings of the European conference on computer vision* (Vol. 1, pp. 414–431).

Scharstein, D., & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, *47*(1), 7–42.

Schindler, G., Dellaert, F., & Kang, S. B. (2007). Inferring temporal order of images from 3D structure. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Schmid, C., & Zisserman, A. (1997). Automatic line matching across views. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 666–671).

Seitz, S. M., & Dyer, C. M. (1996). View morphing. In *SIGGRAPH conference proceedings* (pp. 21–30).

Seitz, S., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (Vol. 1, pp. 519–526), June 2006.

Shi, J., & Tomasi, C. Good features to track. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 593–600), June 1994.

Sivic, J., & Zisserman, A. (2003). Video Google: a text retrieval approach to object matching in videos. In *Proceedings of the international conference on computer vision* (pp. 1470–1477), October 2003.

Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, *25*(3), 835–846.

Spetsakis, M. E., & Aloimonos, J. Y. (1991). A multiframe approach to visual motion perception. *International Journal of Computer Vision*, *6*(3), 245–255.

Strecha, C., Tuytelaars, T., & Van Gool, L. (2003). Dense matching of multiple wide-baseline views. In *Proceedings of the international conference on computer vision* (pp. 1194–1201), October 2003.

Szeliski, R. (2006). Image alignment and stitching: a tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, *2*(1).

Szeliski, R., & Kang, S. B. (1994). Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, *5*(1), 10–28.

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., & Rother, C. (2006). A comparative study of energy minimization methods for Markov random fields. In *Proceedings of the European conference on computer vision* (Vol. 2, pp. 16–29), May 2006.

Tanaka, H., Arikawa, M., & Shibasaki, R. (2002). A 3-d photo collage system for spatial navigations. In *Revised papers from the second Kyoto workshop on digital cities II, computational and sociological approaches* (pp. 305–316).

Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M., & Master, N. (2003). Calibrated, registered images of an extended urban area. *International Journal of Computer Vision*, *53*(1), 93–107.

Tomasi, C., & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, *9*(2), 137–154.

Toyama, K., Logan, R., & Roseway, A. (2003). Geographic location tags on digital images. In *Proceedings of the international conference on multimedia* (pp. 156–166).

Triggs, B., et al. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms* (pp. 298–372), September 1999.

Tuytelaars, T., & Van Gool, L. (2004). Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, *59*(1), 61–85.

Vergauwen, M., & Van Gool, L. (2006). Web-based 3D reconstruction service. *Machine Vision and Applications*, *17*(2), 321–329.

von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the conference on human factors in computing systems* (pp. 319–326).

Zitnick, L., Kang, S. B., Uyttendaele, M., Winder, S., & Szeliski, R. (2004). High-quality video view interpolation using a layered representation. In *SIGGRAPH conference proceedings* (pp. 600–608).