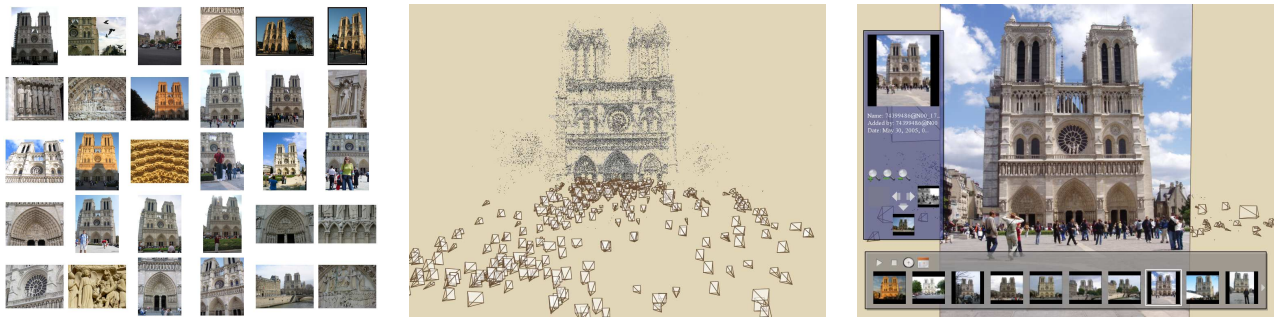
Photo Tourism: Exploring Photo Collections in 3D

Noah Snavely
University of Washington

Steven M. Seitz
University of Washington

Richard Szeliski
Microsoft Research



(a) (b) (c)

Figure 1: Our system takes unstructured collections of photographs such as those from online image searches (a) and reconstructs 3D points and viewpoints (b) to enable novel ways of browsing the photos (c).

Abstract

We present a system for interactively browsing and exploring large unstructured collections of photographs of a scene using a novel 3D interface. Our system consists of an image-based modeling front end that automatically computes the viewpoint of each photograph as well as a sparse 3D model of the scene and image to model correspondences. Our *photo explorer* uses image-based rendering techniques to smoothly transition between photographs, while also enabling full 3D navigation and exploration of the set of images and world geometry, along with auxiliary information such as overhead maps. Our system also makes it easy to construct photo tours of scenic or historic locations, and to annotate image details, which are automatically transferred to other relevant images. We demonstrate our system on several large personal photo collections as well as images gathered from Internet photo sharing sites.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Modeling and recovery of physical attributes

Keywords: image-based rendering, image-based modeling, photo browsing, structure from motion

1 Introduction

A central goal of image-based rendering is to evoke a visceral sense of *presence* based on a collection of photographs of a scene. The last several years have seen significant progress towards this goal through view synthesis methods in the research community and in commercial products such as panorama tools. One of the dreams is that these approaches will one day allow virtual tourism of the world's interesting and important sites.

During this same time, digital photography, together with the Internet, have combined to enable sharing of photographs on a truly massive scale. For example, a Google image search on "Notre Dame Cathedral" returns over *15,000* photos, capturing the scene from myriad viewpoints, levels of detail, lighting conditions, seasons, decades, and so forth. Unfortunately, the proliferation of shared photographs has outpaced the technology for browsing such collections, as tools like Google (www.google.com) and Flickr (www.flickr.com) return pages and pages of thumbnails that the user must comb through.

In this paper, we present a system for browsing and organizing large photo collections of popular sites which exploits the common 3D geometry of the underlying scene. Our approach is based on computing, from the images themselves, the photographers' locations and orientations, along with a sparse 3D geometric representation of the scene, using a state-of-the-art image-based modeling system. Our system handles large collections of unorganized photographs taken by different cameras in widely different conditions. We show how the inferred camera and scene information enables the following capabilities:

- **Scene visualization.** Fly around popular world sites in 3D by morphing between photos.
- **Object-based photo browsing.** Show me more images that contain this object or part of the scene.
- **Where was I?** Tell me where I was when I took this picture.
- **What am I looking at?** Tell me about objects visible in this image by transferring annotations from similar images.

Our paper presents new image-based modeling, image-based rendering, and user-interface techniques for accomplishing these goals, and their composition into an end-to-end 3D photo browsing system. The resulting system is remarkably robust in practice; we include results on numerous sites, ranging from Notre Dame (Figure 1) to the Great Wall of China and Yosemite National Park, as evidence of its broad applicability.

The remainder of this paper is structured as follows. Section 2 gives an overview of the approach. Section 3 surveys related work

in vision, graphics, and image browsing. Section 4 presents our approach to obtain geo-registered camera and scene information. Our photo exploration interface and rendering techniques are described in Section 5, our navigation tools in Section 6, and annotation transfer capabilities in Section 7. Section 8 presents results. We conclude with a discussion of limitations and future work in Section 9.

2 System Overview

In this section, we provide an overview and motivation for the specific features of our system. For visual demonstrations of these features, we refer the reader to the companion video.

Our work is based on the idea of using camera pose (location, orientation, and field of view) and sparse 3D scene information to create new interfaces for browsing large collections of photographs. Knowing the camera pose enables placing the images into a common 3D coordinate system (Figure 1 (b)) and allows the user to virtually explore the scene by moving in 3D space from one image to another using our *photo explorer*. We use morphing techniques to provide smooth transitions between photos. In addition, the reconstructed features provide a sparse but effective 3D visualization of the scene that serves as a backdrop for the photographs themselves. Beyond displaying these features as a simple point cloud, we present a novel non-photorealistic rendering technique that provides a better sense of scene appearance.

While standard 3D navigation controls are useful for some interactions, they are not ideal for other tasks. For example, suppose you want to see images of a particular object or region in the scene. To support this kind of *object-based* browsing, we allow the user to draw a box around an object of interest in one image; the system then moves smoothly to the "best" view of that object. We also provide the ability to geometrically *stabilize* a set of views of the same object, to more effectively compare scene appearance over different times of day, seasons, or weather conditions.

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 (www.wikipedia.org) 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 linked to all other images that contain the same object.

The backbone of our system is a robust structure from motion approach for reconstructing the required 3D information. Our approach first computes feature correspondences between images, using descriptors that are robust with respect to variations in pose, scale, and lighting, and runs an optimization to recover the camera parameters and 3D positions of those features. The resulting correspondences and 3D data enable all of the aforementioned features of our system.

3 Related work

There are three main categories of work related to ours: image-based modeling; image-based rendering; and image browsing, retrieval, and annotation.

3.1 Image-based modeling

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]. Our image-based modeling system is based on recent work in *structure from motion* (SfM), which aims to recover camera parameters, pose estimates, and sparse 3D scene geometry from image sequences [Hartley and Zisserman 2004]. In particular, our SfM approach 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 calibrating unordered image sets, concentrating on efficiently matching interest points between images. While both 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 being 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.

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 UC 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 compute 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 (www.cc.gatech.edu/4d-cities), 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 Pollefeys and Nistér (www.cs.unc.edu/Research/urbanscape/).

3.2 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 overlayed 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. 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. 2003a; Zitnick et al. 2004; Buehler et al. 2001]. In terms of applications, Aliaga *et al.*'s [2003a] *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. 2003b]. 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 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.

3.3 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 been growing interest in using geo-location information to facilitate photo browsing. In particular, the World-Wide Media Exchange [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 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 was 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 geo-referenced 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.).

4 Reconstructing Cameras and Sparse Geometry

Our system requires accurate information about the relative location, orientation, and intrinsic parameters such as focal length for each photograph in a collection, as well as sparse 3D scene geometry. Some 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 invariance to image transformations. A typical image contains several thousand SIFT keypoints. 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. Next, for each pair of images, we match keypoint descriptors between the pair, using the approximate nearest neighbors package of Arya, *et al.*[1998], then robustly estimate a fundamental matrix for the pair using RANSAC [Fischler and Bolles 1987]. During each RANSAC iteration, we compute a candidate fundamental matrix using the eight-point algorithm [Hartley and Zisserman 2004], followed by non-linear refinement. Finally, we remove matches that are outliers to the recovered fundamental matrix. 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 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.

4.2 Structure from motion

Next, we recover a set of camera parameters 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 is formulated as a non-linear least squares problem (see Appendix A) and solved with algorithms such as Levenberg-Marquardt [Nocedal and Wright 1999]. Such algorithms are only guaranteed to find local minima, and large-scale

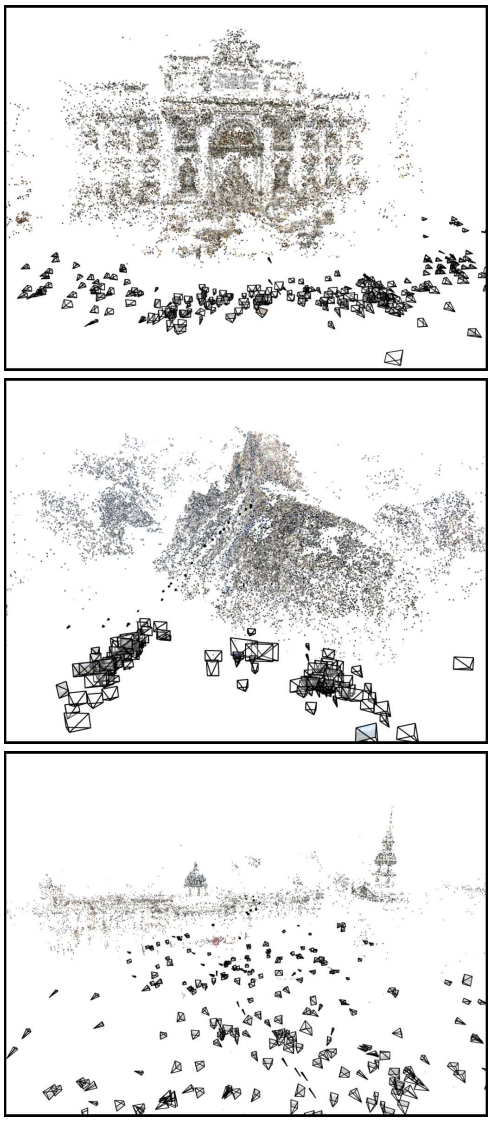


Figure 2: *Camera and 3D point reconstructions from photos on the Internet.* From top to bottom: the Trevi Fountain, Half Dome, and Trafalgar Square.

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 3D locations of the observed points are well-conditioned. 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.

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. The DLT also gives an estimate of the intrinsic parameter matrix

Figure 3: *Estimated camera locations for the Great Wall data set.*

$\mathbf{K}$ as a general upper-triangular matrix. We use $\mathbf{K}$ and the focal length estimated from the EXIF tags of the image to initialize the focal length of the new camera (see Appendix A for more details).

Finally, we add tracks observed by the new camera into the optimization. A track is added if it is observed by at least one other recovered camera, and if triangulating the track gives a well-conditioned estimate of its location. This procedure is repeated, one camera at a time, until no remaining camera observes any reconstructed 3D point. To minimize the objective function at every iteration, we use the sparse bundle adjustment library of Lourakis and Argyros [2004]. After reconstructing a scene, we optionally run a post-processing step to detect 3D line segments in the scene using a line segment reconstruction technique, as in the work of Schmid and Zisserman [1997].

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. We then 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. We first find the camera with the greatest number of matches, M, to existing 3D points, then add any camera with at least $0.75M$ matches to existing 3D points.

Figures 2 and 3 show reconstructed cameras (rendered as frusta) and 3D feature points for several famous world sites reconstructed with this method.

The total running time of the SfM procedure for the datasets we experimented with ranged from a few hours (for Great Wall, 120 photos processed and matched, and 82 ultimately registered) to about two weeks (for Notre Dame, 2,635 photos processed and matched, and 597 photos registered). The running time is dominated by the iterative bundle adjustment, which gets slower as more photos are added, and as the amount of coupling between cameras increases (e.g., when many cameras observe the same set of points).

4.3 Geo-registration

The SfM procedure estimates *relative* camera locations. The final step of the location estimation process is to align the model with a geo-referenced image or map (such as a satellite image, floor plan, or digital elevation map) to enable the determination of absolute geocentric coordinates of each camera. This step is unnecessary for many features of our explorer system to work, but is required for others (such as displaying an overhead map).

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

Figure 4: *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 Figure 5 courtesy of Gefos, a.s. (www.gefos.cz) and Atlas.cz.)

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 [2005]. 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 4 shows a screenshot of such an alignment.

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 (earth.google.com) and with coverage of most of the United States available through the U.S. Geological Survey (www.usgs.com). 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), this manual step will no longer be necessary.

4.4 Scene representation

The representation of the reconstructed scene model that we use for our photo exploration tool is as follows:

- 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, $\mathrm{Points}(C)$ 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.

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 Section 6.

5.1 User interface layout

Figure 5 shows a screen shot 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 view 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 Section 6.2.

The thumbnail pane shows the results of search operations as a filmstrip of thumbnails. When the user visits a camera C_{curr} and mouses over a thumbnail, the corresponding image I_j is projected onto a plane in the main view to show the content of that image and how it is situated in space (we precompute projection planes, $\mathrm{CommonPlane}(C_j, C_k)$, for each pair C_j, C_k of cameras, by robustly fitting a plane to $\mathrm{Points}(C_j) \cup \mathrm{Points}(C_k)$). 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

Figure 5: *Screenshots from the explorer interface.* Left: a view looking down on the Prague dataset, rendered in a non-photorealistic style. Right: when the user visits a photo, that photo appears at full-resolution, and information about it appears in a pane on the left.

the user can see it in detail. The back faces of the other cameras frusta are either texture-mapped with a low-resolution, semi-transparent thumbnail of the photo (if the frustum is near the user's current position), or rendered with a translucent white color.

The recovered points and line segments are used to depict the scene itself. The points are drawn with their acquired color and the lines are drawn in black . The user can control whether or not the points and lines are drawn, the point size, and the line thickness.

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. For each camera C_j, we first robustly fit a plane to $\mathrm{Points}(C_j)$ using RANSAC. If the number of inliers to the recovered plane is at least 20% of the size of $\mathrm{Points}(C_j)$, we then fit a robust bounding box, $\mathrm{Rectangle}(C_j)$, to the inliers in the plane. To render the scene, we project a blurred, semi-transparent version of each image I_j onto $\mathrm{Rectangle}(C_j)$ and use alpha blending to combine the results. In parts of the scene represented with a sparse number of points, our system falls back to the point and line rendering. An example rendering using projected images overlaid with line segments is shown in Figure 5.

5.3 Transitions between photographs

An important element of our user interface is the method used to generate transitions when the user moves between 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. We fade out all other camera frusta before starting the motion, to avoid flickering caused by many frusta rapidly moving through the view.

If the camera moves as the result of an object selection, 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 $\mathrm{Points}(C_j)$ into I_j. The projections of $\mathrm{Lines}(C_j)$ into I_j are imposed as edge constraints on the triangulation [Chew 1987]. The resulting Delaunay triangulation may not cover the entire image, so we overlay a grid onto the image and add each grid point not contained in the original triangulation. Each added grid point is associated with a 3D point on a plane approximating the geometry of the points seen by both C_j and C_k. The connectivity of the triangulation is then used to create a 3D mesh from $\mathrm{Points}(C_j)$ and the endpoints of $\mathrm{Lines}(C_j)$. We texture map the mesh by projecting I_j onto each triangle. We compute a mesh for C_k and texture map it in the same way.

To render an in-between view, we render each mesh from the new viewpoint and blend the two rendered images in proportion to the distance from the in-between camera to the two endpoints. 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. However, missing geometry and outlying points can sometimes cause distracting artifacts, as demonstrated in the accompanying video.

Planar morphs 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 $\mathrm{CommonPlane}(C_j, C_k)$ 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 accompanying video.

There are a few special cases when view interpolation is not used during a transition from image C_j to C_k. First, if the 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 $\mathrm{CommonPlane}(C_j, C_k)$ 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 $\mathrm{CommonPlane}(C_j, C_k)$ is visible in images I_j or I_k, 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 to find 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, $\mathrm{Points}(C_{\mathrm{curr}})$, 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 $\mathrm{Points}(C_{\mathrm{curr}})$ appear to have moved right.

For geometric searches, new images are selected from the set of *neighbors* of the current camera. We define the neighbors of C to be the set of cameras that see at least one point in $\mathrm{Points}(C)$, i.e.,

$$\mathrm{Neighbors}(C) = \{C_j | \mathrm{Points}(C_j) \cap \mathrm{Points}(C) \neq \emptyset\}$$

The tools in the first row find related images at different scales. These tools select images by estimating visibility and "how large" a set of points appears in different views. Because we do not have complete knowledge of the geometry of the scene, to check the visibility of a point in a camera we simply check whether the point projects inside the camera's field of view. To estimate the *apparent size* of a set of points P in an image C, we project the points of P into C, compute the axis-aligned 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 $\mathrm{Size}(P, C)$.

There are three tools in this set: (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 computing $\mathrm{Size}(\mathrm{Points}(C_j), C_{\mathrm{curr}})$ or, conversely, $\mathrm{Size}(\mathrm{Points}(C_{\mathrm{curr}}), C_j)$. We deem a photo C_j to be:

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

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

  and the angle between the viewing directions of C_{curr} and C_j is less than a threshold
- 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 "taking a step back" to see more of the scene.

The tools in the second row 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 precompute a "left" and "right" image, and display them as thumbnails. To find a left and right image for camera C_j, we compute the average 2D motion m_{jk} of the projections of $\mathrm{Points}(C_j)$ from image I_j to each neighboring image I_k. If the angle between m_{jk} and the desired direction is small (and the apparent sizes of $\mathrm{Points}(C_j)$ in both images are similar), then C_k is a candidate left or right image. Out of all the candidates, we select the image I_k whose motion magnitude $||m_{jk}||$ is closest to 10% of the width of image I_j.

Along with the left and right images, we provide a "step back" tool, which is a shortcut to the first zoom-out chosen by the procedure described above.

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 are considered selected. Our system then searches for the "best" picture of the selected points by scoring each image in the database based on how good a representation it is of the selection. The top scoring photo is

Figure 6: *Object-based browsing.* The user drags a rectangle around Neptune in one photo, and the system finds a new, high-resolution photograph.

selected as the representative view, and the virtual camera is moved to that image. The remaining images with scores above a threshold are displayed in the thumbnail pane, sorted by descending score. An example interaction is shown in Figure 6.

Any function that rates the "goodness" of an image with respect to a selection can be used as the scoring function. We choose a function based on three criteria: 1) the selected points are visible, 2) the object is viewed from a good angle, and 3) the object appears in sufficient detail. 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 C.

A point selection can sometimes contain points that the user did not intend to select. In particular, it may include occluded points that happen to project inside the selection rectangle. Because the complete scene geometry is unknown, it is difficult to test for visibility. To avoid problems due to larger-than-intended selections, we first prune the point set to remove likely occluded pixels. In particular, if the selection was made while visiting an image I_j (and is contained in the image boundary), we use the points that are known to be visible from that viewpoint ($\text{Points}(C_j)$) to refine the selection. We compute the 3×3 covariance matrix for the selected points that are also in $\text{Points}(C_j)$, and remove all selected points with a Mahalanobis distance greater than 1.2. If the selection was made directly on the projected point cloud (i.e., not on an image) we instead compute a weighted mean and covariance matrix using the entire set of selected points. The weight for each point is the inverse of the distance from the virtual camera center to the point.

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 is shown in the companion video.

In addition to being able to view search results as a slideshow, the user can load a saved image sequence. This feature can be used to interactively view tours authored by other users.

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 Section 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 3.40GHz 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).

The user can select a region of an image and enter a text annotation. The annotation is then stored, along with the selected points, 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 determine if an annotation is appropriate for a given camera C_j, we check for visibility and scale. To determine visibility, we simply test that at least one of the annotated points is in $\text{Points}(C_j)$. To check that the annotation is at an appropriate scale for the image, we determine the apparent size, $\text{Size}(P_{\text{ann}}, C_j)$, of the annotation in image I_j, where P_{ann} is the set of annotated points. If the annotation is visible and

Figure 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.

$0.05 < \text{Size}(P_{\text{ann}}, C_j) < 0.8$ (to avoid barely visible annotations and annotations that take up the entire image), 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 Figure 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 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 evaluated our system using several data sets. The first two sets were taken in more controlled settings (i.e., a single person with a single camera and lens): **Prague**, a set of 197 photographs of the Old Town Square in Prague, Czech Republic, taken over the course two days, and **Great Wall**, a set of 120 photographs taken along the Great Wall of China (82 of which were ultimately registered).

We have also experimented with "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 four sets are as follows. **Notre Dame** is a set of 597 photos of the Notre Dame Cathedral in Paris. These photos were registered starting from 2,635 photos matching the search term "notredame AND paris." **Yosemite** is a set of 325

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

photos of Half Dome in Yosemite National Park, registered from 1,882 photos matching "halfdome AND yosemite." **Trevi Fountain** is a set of 360 photos of the Trevi Fountain in Rome, registered from 466 photos matching "trevi AND rome." **Trafalgar Square** is a set of 278 photos from Trafalgar Square, from 1893 photos matching "trafalgarsquare."

The average reprojection error over each of the datasets is 1.516 pixels (the average long and short dimensions of the images we registered are 1,611 by 1,128 pixels). Visualizations of these data sets are shown in Figures 1-8. Please see the accompanying video for a demonstration of features of our photo explorer, 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 Section 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 Section 7.1. Figure 8 shows a synthetic rendering of the scene from the estimated position where Ansel Adams took the photo.

9 Discussion and future work

We have presented a novel end-to-end system for taking an unordered set of photos, registering them, and presenting them to a

Figure 9: *Unregistered photographs.* A few representative photos from the Trevi Fountain that our system was unable to register. These include images that are too noisy, dark, cluttered, or whose scale or viewpoint are too different from other images, to be reliably matched.

user in a novel, interactive 3D browser. We evaluated our reconstruction algorithm and exploration interface on several large sets of photos of popular sites gathered from the Internet and from personal photo collections.

For the data sets taken from the Internet, our reconstruction algorithm successfully registered a significant subset of the photos. Most of the photos that were not registered belong to parts of the scene disconnected from the reconstructed portion, but others could not be matched due to other factors, such as excessive blur or noise, underexposure, or too little overlap with other photos. Figure 9 shows a few representative images from the Trevi Fountain data set that our system was unable to register.

Our reconstruction algorithm has several limitations that we would like to address in the future. Our current SfM implementation becomes slow as the number of registered cameras grows. We would like to speed up the process, for instance, by choosing a better order in which to register the photographs. A more efficient ordering might reconstruct the large-scale scene structure with a small number of views, then register the remaining views using local optimization. Additionally, we can improve efficiency using partitioning methods [Steedly et al. 2003]. Our SfM procedure is also not guaranteed to produce a metric scene reconstruction without ground control points, which makes obtaining very accurate models more difficult. This problem will be alleviated as more georeferenced data becomes available. Our camera model does not handle lens distortion, resulting in greater reconstruction error; we plan to use a more sophisticated model in the future. Some scenes are challenging to reconstruct automatically, particularly those with repeating structures or without strong features. Finally, our algorithm only reconstructs one connected component of the input images. We plan to extend it to reconstruct all structures present in the input.

Our explorer interface can also be improved in several ways. For large numbers of photos, the scene can become cluttered with frusta. To alleviate such problems, we wish to explore clustering the photos and allowing the user to display only photos with certain attributes. Better morphing and view interpolation techniques would provide an even more compelling sense of presence.

Ultimately, we wish to scale up our reconstruction algorithm to handle millions of photographs and maintain a database cataloging different scenes and annotations. We envision a system that automatically scours the web for photographs and determines if and how new photos fit into the database. Such a system would also geo-register photographs leveraging existing geo-referenced data.

Acknowledgments This work was supported by Microsoft, the University of Washington Animation Research Labs, an Achievement Rewards for College Scientists (ARCS) fellowship, National Science Foundation grants IIS-0413198 and DGE-0203031, and an endowment by Emer Dooley and Rob Short. We also thank the many Flickr users who agreed to let us use their images for this project. A complete list of acknowledgments is included in Appendix C.

References

ALIAGA, D., FUNKHOUSER, T., YANOVSKY, D., AND CARLBOM, I. 2003. Sea of images. *IEEE Computer Graphics and Applications 23*, 6, 22–30.

ALIAGA, D., YANOVSKY, D., FUNKHOUSER, T., AND CARLBOM, I. 2003. Interactive image-based rendering using feature globalization. In *Proc. SIGGRAPH Symposium on Interactive 3D Graphics*, 163–170.

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. of the ACM 45*, 6, 891–923.

BROWN, M., AND LOWE, D. G. 2005. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Proc. Int. Conf. on 3D Digital Imaging and Modelling*, 56–63.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *SIGGRAPH Conf. Proc.*, 425–432.

CHEN, S., AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *SIGGRAPH Conf. Proc.*, 279–288.

CHEW, L. P. 1987. Constrained delaunay triangulations. In *Proc. Sym. on Computational geometry*, 215–222.

COOPER, M., FOOTE, J., GIRGENSOHN, A., AND WILCOX, L. 2003. Temporal event clustering for digital photo collections. In *Proc. ACM Int. Conf. on Multimedia*, 364–373.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH Conf. Proc.*, 11–20.

DICK, A. R., TORR, P. H. S., AND CIPOLLA, R. 2004. Modelling and interpretation of architecture from several images. *Int. J. of Computer Vision 60*, 2, 111–134.

FEINER, S., MACINTYRE, B., HOLLERER, T., AND WEBSTER, A. 1997. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. In *Proc. IEEE Int. Sym. on Wearable Computers*, 74–81.

FISCHLER, M., AND BOLLES, R. 1987. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in computer vision: issues, problems, principles, and paradigms*, 726–740.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The Lumigraph. In *SIGGRAPH Conf. Proc.*, 43–54.

GRZESZCZUK, R. 2002. Course 44: Image-based modeling. In *SIGGRAPH 2002*.

HARTLEY, R. I., AND ZISSERMAN, A. 2004. *Multiple View Geometry*. Cambridge University Press, Cambridge, UK.

IRANI, M., AND ANANDAN, P. 1998. Video indexing based on mosaic representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence 86*, 5, 905–921.

JOHANSSON, B., AND CIPOLLA, R. 2002. A system for automatic pose-estimation from a single image in a city scene. In *Proc. IASTED Int. Conf. Signal Processing, Pattern Recognition and Applications*.

KADOBAYASHI, R., AND TANAKA, K. 2005. 3d viewpoint-based photo search and information browsing. In *Proc. ACM Int. Conf. on Research and development in information retrieval*, 621–622.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *SIGGRAPH Conf. Proc.*, 31–42.

LIPPMAN, A. 1980. Movie maps: An application of the optical videodisc to computer graphics. In *SIGGRAPH Conf. Proc.*, 32–43.

LOURAKIS, M., AND ARGYROS, A. 2004. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Tech. Rep. 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. *Int. J. of Computer Vision 60*, 2, 91–110.

MCCURDY, N., AND GRISWOLD, W. 2005. A systems architecture for ubiquitous video. In *Proc. Int. Conf. on mobile systems, applications, and services*, 1–14.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH Conf. Proc.*, 39–46.

MIKOLAJCZYK, K., TUYTELAARS, T., SCHMID, C., ZISSERMAN, A., MATAS, J., SCHAFFALITZKY, F., KADIR, T., AND VAN GOOL, L. 2005. A comparison of affine region detectors. *Int. J. of Computer Vision 65*, 1/2, 43–72.

NAAMAN, M., PAEPCKE, A., AND GARCIA-MOLINA, H. 2003. From where to what: Metadata sharing for digital photographs with geographic coordinates. In *Proc. Int. Conf. on Cooperative Information Systems*, 196–217.

NAAMAN, M., SONG, Y. J., PAEPCKE, A., AND GARCIA-MOLINA, H. 2004. Automatic organization for digital photographs with geographic coordinates. In *Proc. ACM/IEEE-CS Joint Conf. on Digital libraries*, 53–62.

NOCEDAL, J., AND WRIGHT, S. J. 1999. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, NY.

POLLEFEYS, M., VAN GOOL, L., VERGAUWEN, M., VERBIEST, F., CORNELIS, K., TOPS, J., AND KOCH, R. 2004. Visual modeling with a hand-held camera. *Int. J. of Computer Vision 59*, 3, 207–232.

ROBERTSON, D. P., AND CIPOLLA, R. 2002. Building architectural models from many views using map constraints. In *Proc. European Conf. on Computer Vision*, vol. II, 155–169.

RODDEN, K., AND WOOD, K. R. 2003. How do people manage their digital photographs? In *Proc. Conf. on Human Factors in Computing Systems*, 409–416.

ROMÁN, A., GARG, G., AND LEVOY, M. 2004. Interactive design of multi-perspective images for visualizing urban landscapes. In *Proc. IEEE Visualization*, 537–544.

RUSSELL, B. C., TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2005. Labelme: A database and web-based tool for image annotation. Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology.

SCHAFFALITZKY, F., AND ZISSERMAN, A. 2002. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *Proc. European Conf. on Computer Vision*, vol. 1, 414–431.

SCHMID, C., AND ZISSERMAN, A. 1997. Automatic line matching across views. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 666–671.

SEITZ, S. M., AND DYER, C. M. 1996. View morphing. In *SIGGRAPH Conf. Proc.*, 21–30.

SIVIC, J., AND ZISSERMAN, A. 2003. Video Google: A text retrieval approach to object matching in videos. In *Proc. Int. Conf. on Computer Vision*, 1470–1477.

STEEDLY, D., ESSA, I., AND DELLEART, F. 2003. Spectral partitioning for structure from motion. In *Proc. Int. Conf. on Computer Vision*, 996–103.

SZELISKI, R. 2005. Image alignment and stitching: A tutorial. Tech. Rep. MSR-TR-2004-92, Microsoft Research.

TELLER, S., ET AL. 2003. Calibrated, registered images of an extended urban area. *Int. J. of Computer Vision 53*, 1, 93–107.

TOYAMA, K., LOGAN, R., AND ROSEWAY, A. 2003. Geographic location tags on digital images. In *Proc. Int. Conf. on Multimedia*, 156–166.

VON AHN, L., AND DABBISH, L. 2004. Labeling images with a computer game. In *Proc. Conf. on Human Factors in Computing Systems*, 319–326.

ZITNICK, L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. In *SIGGRAPH Conf. Proc.*, 600–608.

A 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). We use an incremental rotation, ω to parameterize the 3D rotation, where

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

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

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

We group the seven parameters into a vector, $\mathbf{\Theta} = [\omega, c, f]$. Each point is parameterized by a 3D position, p.

Recovering the parameters can be formulated as an optimization problem. In particular, we have a set of n cameras, parameterized by $\mathbf{\Theta}_i$. We also have a set of m tracks, parameterized by p_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 $\mathbf{P}(\mathbf{\Theta}, p)$ be the equation mapping a 3D point p to its 2D projection in a camera with parameters $\mathbf{\Theta}$. $\mathbf{P}$ transforms p to homogeneous image coordinates and performs the perspective division:

$$p'(\mathbf{\Theta}, p) = \mathbf{KR}(p - c)$$

$$\mathbf{P}(\mathbf{\Theta}, p) = \begin{bmatrix} -p'_x/p'_z & -p'_y/p'_z \end{bmatrix}^T$$

where $\mathbf{K} = \mathrm{diag}(f, f, 1)$.

We wish to minimize the sum of the reprojection errors:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} ||q_{ij} - \mathbf{P}(\mathbf{\Theta}_i, p_j)||$$

(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 is a reasonable estimate. If this test fails, we use f_1.

B Image selection criteria

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

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

Next, to compute E_{angle}, we first attempt to find a dominant plane in P_{inliers} by fitting a plane to the points using orthogonal regression inside a RANSAC loop. If the plane fits most of the points fairly tightly, i.e., if the percentage of points in P_{inliers} is above a threshold of 30%, we favor cameras that view the object head-on (i.e., with the camera pointing parallel to the normal, $\hat{n}$, to the plane), by setting $E_{\mathrm{angle}} = V(C_j) \cdot \hat{n}$, where V indicates viewing direction. If enough points do not fit a plane, we set $E_{\mathrm{angle}} = 0$.

Finally, we compute E_{detail} to be the area, in pixels, of the bounding box of the projections of P_{inliers} 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.

C 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[1]
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[2]
Ted Wang	http://www.flickr.com/photos/mtwang[3]
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[4]
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 Creative Photography, University of Arizona, © Trustees of The Ansel Adams Publishing Rights Trust.

[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/