# Photo Tourism: Exploring Photo Collections in 3D

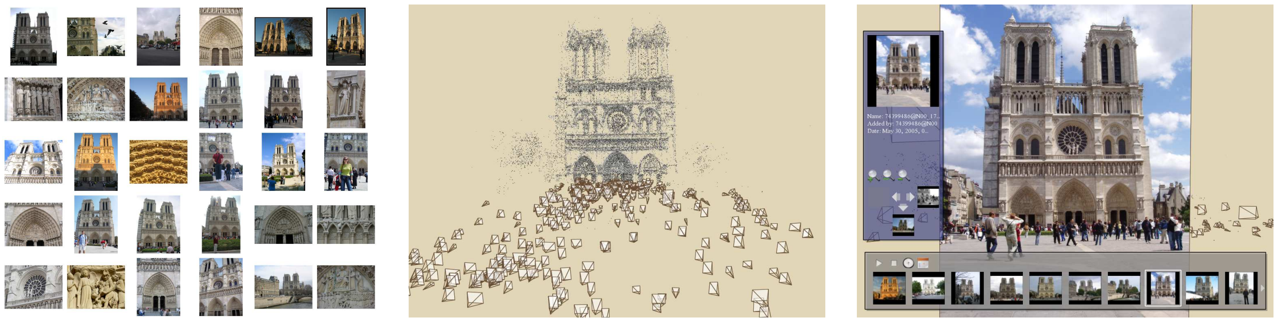

Figure 1: Our system takes unstructured collections of photographs such as those from online image searches (a) and reconstructs 3D points and viewpoints (b) to enable novel ways of browsing the photos (c).

## Abstract

We present a system for interactively browsing and exploring a large unstructured collection of photographs of a scene using a novel 3D interface. Our system consists of an image-based modeling front end, which automatically computes the viewpoint of each photograph as well as a sparse 3D model of the scene and image to model correspondences. Our photo navigation tool 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, as well as to annotate image details, which are automatically transferred to other relevant images in the collection. We demonstrate our system on several large personal photo collections as well as images gathered from photo sharing Web sites on the Internet.

## 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, that 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 better 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 enable 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 contents 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 [Lowe 2004], 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 2002]. 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 scalability and robustness. 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 conditions, and differing amounts of occlusion.

One particular application of image-based modeling has been the creation of large scale architectural models. Notable examples include the semi-automatic Façade system by Debevec, *et al.* [1996], which was used to reconstruct compelling fly-throughs of the UC Berkeley campus 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, and the Stanford CityBlock Project [Román et al. 2004], which uses video of city blocks to create multi-perspective strip images.

### 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 image-based rendering 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]. 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. 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 current IBM and IBR techniques.

### 3.3 Image browsing, retrieval, and annotation

There are many techniques and commercial products for browsing sets of photographs and much research on the subject of how people tend to organize their own photographs, 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.)

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 capabilities (visualization, navigation, annotation, etc.).

## 4 Reconstructing Cameras and Sparse Geometry

In order to use our browsing tools, we need accurate information about the relative location and orientation of the camera used to take each photograph in the set, and sparse 3D scene geometry. Some features of our browser also require the absolute locations of the cameras, in a geo-referenced coordinate frame. In addition to these *extrinsic* parameters, it is useful to know the intrinsic parameters, such as the focal length, of each camera. How can this information be derived? GPS devices can determine position, and electronic compasses can give orientation, but it is not yet common for people to carry around this kind of equipment. Many digital cameras embed the focal length of the lens (along with other information) in the EXIF tags of the image files. These values are useful for initialization, but are not always accurate.

In our system, we do not rely on the camera or any other piece of equipment to provide us with accurate location, orientation, or geometry information. Instead, we compute this information from the images themselves using computer vision techniques. Our approach is similar to that of Brown and Lowe [2005], with several motifications to improve scalability and robustness, plus additional geo-registration capabilities. We first detect feature points in each image, then match feature points between pairs of image, keeping only geometrically consistent matches, and finally run an iterative, robust SfM procedure to recover the intrinsic and extrinsic 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. In our system, we use the SIFT keypoint detector [Lowe 2004], because of its invariance to image transformations. A typical image contains several thousand such keypoints. In addition to the keypoint locations themselves, SIFT provides a local descriptor for each keypoint. We match keypoint descriptors between each pair of images, using the approximate nearest neighbors package of [Arya et al. 1998]. We remove pairs with too few matches, using a relatively low threshold of twenty matches, since false matches are pruned in several later stages of our algorithm. For each remaining pair, we use RANSAC [Hartley and Zisserman 2004] to robustly estimate a fundamental matrix. Inside each iteration of RANSAC, the eight-point algorithm [2004], followed by a non-linear refinement step, is used to compute a candidate fundamental matrix. We then remove matches that are outliers to the recovered fundamental matrix.

After finding a set of geometrically consistent matches, we organize the matches into sets of *tracks*, where a track is a connected set of matching keypoints. Some tracks may be inconsistent, in that one image observes multiple keypoints in the track. We keep only consistent tracks containing at least two keypoints for the next phase of the location estimation 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 distance between the projections of each track and its observations, is minimized.

This error minimization problem can be 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, so it is important to provide good initial estimates of the parameters. Large-scale SfM problems are particularly prone to bad local minima if the parameters are not carefully initialized. Rather than try to estimate the parameters for all cameras and tracks at once, we take an incremental approach in which we add cameras one at a time using pose estimation techniques for initialization.

We begin by estimating the parameters of a single pair of cameras. This initial pair should have a large number of matches, but at the same time 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 $\mathbf{K}$, as a general upper-triangular matrix. We use this matrix, as well as the focal length estimate given by the the EXIF tags of the image,

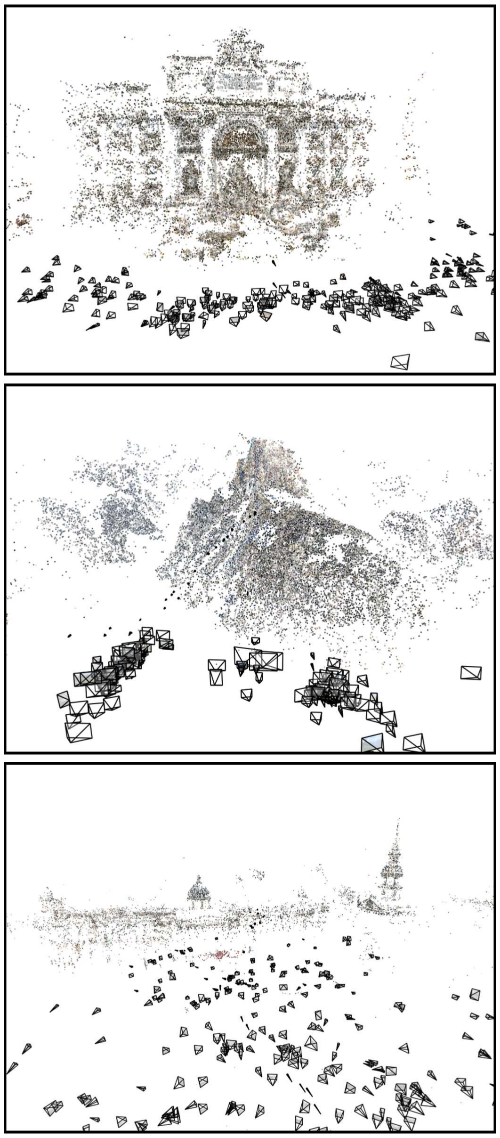

Figure 2: *Camera and 3D point reconstructions from photos on the Internet.* From top to bottom: the Trevi Fountain, Half Dome, and Trafalgar Square.

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 existing 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-process step in which we detect 3D line segments in the scene, for use in rendering, using a line segment reconstruction technique as in [Schmid and Zisserman 1997].

For increased robustness and speed, we make a few modifications to the 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.

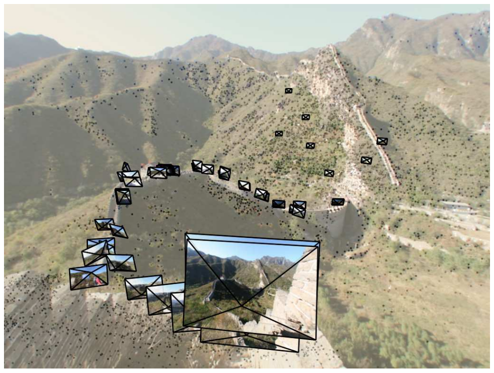

Figure 3: *Estimated camera locations for the Great Wall data set.*

In order to speed up the procedure, rather than adding a single camera at a time into the optimization, we add multiple cameras. To decide which cameras to add during a given iteration, we first find the camera with the greatest number of matches, $M$ to existing 3D points. We then add any camera with at least $0.75M$ matches to existing 3D points. Each new camera is initialized by first applying the DLT technique, then refining the parameters of that camera alone by running bundle adjustment, keeping all other parameters fixed.

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

The total running time of the SfM procedure for the datasets we experimented with ranged from a few hours (for Great Wall, 82 photos registered) to about two weeks (for Notre Dame, 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, a floor plan, or a 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 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 by either 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 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 re-

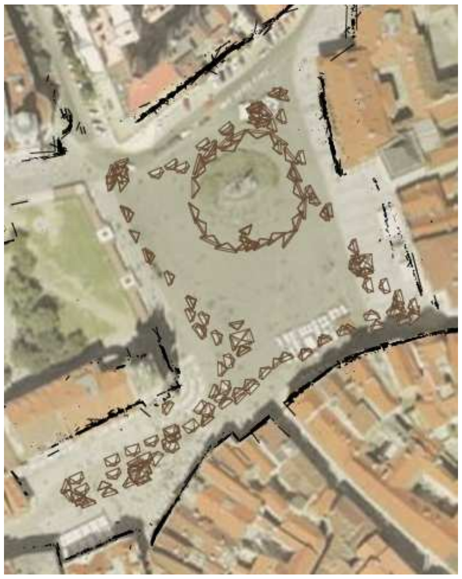

Figure 4: *Example registration of cameras to an overhead map.* Here, the cameras (and recovered points) from the Prague data set are shown superimposed on a satellite image.

construction 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 [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, pinning down points from computed from SfM to points on 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, which is obtained by averaging the pixel colors at all 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, $\mathrm{Points}(c)$ between cameras and the points they observe. That is, $\mathrm{Points}(c)$ is the subset of $P$ containing the points observed by camera $c$.
- A set of 3D line segments $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$.
- A mapping, $\mathrm{Lines}(c)$ 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 photorealistic view of the scene, but rather to display photographs in spatial context and give a sense of the geometry of the true scene. We have explored several ways of rendering the scene, described later in this section.

The information pane appears when the user is visiting 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 is visiting a camera $C_{\mathrm{curr}}$ and mouses over a thumbnail, the corresponding image $I_j$ is projected onto a plane the main view to give the user an idea of 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 movement.

### 5.2 Rendering the scene

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 either texture-mapped with a low-resolution, semi-transparent thumbnail of the photo (if the frustum is visible and near the user's current position), or rendered with a translucent white color.

The recovered points and lines are used to depict the scene itself. The points are rendered with their acquired color and the lines are drawn with thick black borders, to achieve a "line-drawn" look. The

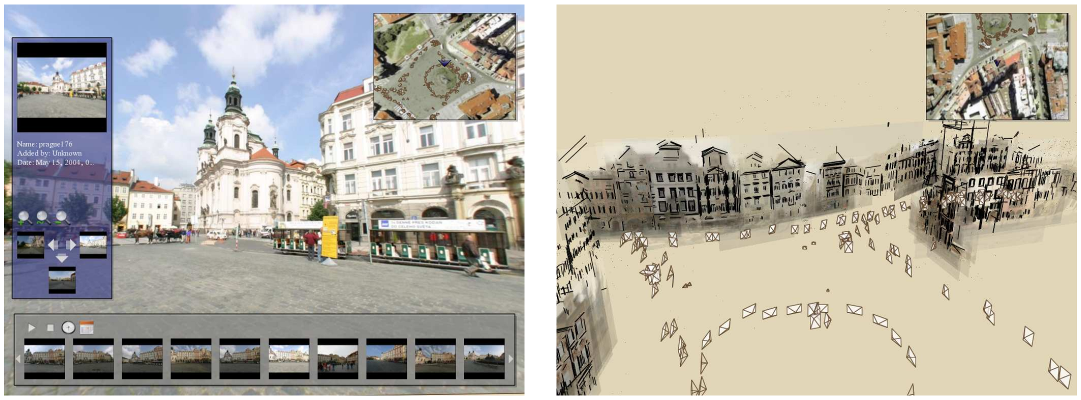

Figure 5: *Screenshots from the explorer interface.*

user can control whether or not the points and lines are drawn, the size of the points, and the thickness of the lines.

We also provide an non-photorealistic rendering mode that provides more attractive visualizations; it 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 between two photographs 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 information we infer about the photographs (location, orientation, and sparse correspondence) 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

Many 3D games and exploration tools (such as Google Earth) use smooth camera motion, rather than sudden cuts, to make transitions more appealing and intuitive. We also use smooth camera paths in our system, but tailor them to our application in several ways.

When the virtual camera is moved from one photograph to another, by default the camera center is linearly interpolated between the start and end camera locations and the camera orientation is linearly interpolated between unit quaternions representing the start and end orientations. The field of view of the virtual camera is also linearly interpolated so that when it reaches its destination, the destination image will fill as much of the screen as possible. The camera path timing is nonuniform, to ease in and out of the transition. We also fade out all the camera frusta before starting the motion, to avoid flickering caused by the 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 end 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 [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 "virtual" 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 $\mathrm{Lines}(C_j)$. Finally, we texture map the mesh by projecting image $I_j$ onto each triangle. A Delaunay triangulation is also computed for $C_k$ and texture mapped 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 good 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 cameras are neighbors, but the normal to the plane $\text{CommonPlane}(C_j, C_k)$ is nearly perpendicular to the average of $\text{Direction}(C_j)$ and $\text{Direction}(C_k)$, the projected images would undergo a large amount of 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 $\text{Direction}(C_j)$ and $\text{Direction}(C_k)$. Finally, if the vanishing line of $\text{CommonPlane}(C_j, C_k)$ 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 optionally zooms in so that the selected image fills as much as the main view as possible. When the virtual camera reaches its destination, the selected photo will be centered in the screen.

### 6.2 Moving between related views

When visiting a photograph $C_{\text{curr}}$, 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" tools. Icons associated with these tools appear in two rows in the information pane, which pops up when the user is visiting a photograph. To implement these tools, we take a *scene*-centered (as opposed to a *camera*-centered) approach. In other words, these tools find photos that depict parts of the scene with certain spatial relations to what is currently in view. The basic mechanism for implementing these search tools is to project the points observed by the current camera, $\text{Points}(C_{\text{curr}})$ 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 would attempt to find a photo in which $\text{Points}(C_{\text{curr}})$ appear to have moved right.

In these "geometric" searches, new images are selected from the set of *neighbors* of the current photo. We define the neighbors of $C$ to be the set of cameras that observe at least one point in common with $C$, i.e.,

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

The first row of tools 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 point set in an image, we project the points into that view, find the axis-aligned bounding box of the projections, and calculate the ratio of the area of the bounding box (in pixels) to the area of the image.

The first tool finds *details*, or higher-resolution close-ups, of the current photo, and is useful for quickly finding out which objects in the current view can be seen in more detail. We deem a neighbor $C_j$ of $C_{\text{curr}}$ a detail if most points in $\text{Points}(C_j)$ are visible in $C_j$ and the apparent size of $\text{Points}(C_j)$ in image $I_{\text{curr}}$ is less than $0.75$. The details are sorted by increasing apparent size and displayed in the thumbnail pane.

The second tool finds *similar* photos, and is useful for comparing similar views of an object which differ in other respects, such as time of day, season, year, and so on. For this operation, we deem a neighbor $C_j$ to be similar to $C_{\text{curr}}$ if most points in $\text{Points}(C_{\text{curr}})$ are visible in $C_j$, the apparent size of $\text{Points}(C_{\text{curr}})$ in $C_j$ is within $30\%$ of the apparent size of $\text{Points}(C_{\text{curr}})$ in $C_{\text{curr}}$ itself, and the angle between the two camera's viewing directions is less than a threshold.

The third tool finds photographs that are "zoom-outs" of the current image, i.e., photos that show more surrounding context. We deem a neighbor $C_j$ of $C_{\text{curr}}$ to be a zoom-out of $C_{\text{curr}}$ if the apparent size of $\text{Points}(C_{\text{curr}})$ in $I_j$ is smaller than the apparent size of $\text{Points}(C_{\text{curr}})$ in $I_{\text{curr}}$ itself. The zoom-outs are sorted by increasing bounding box area.

The second set of tools gives 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 flow $\mathbf{m}_{jk}$ of the projections of $\text{Points}(C_j)$ from image $I_j$ to each neighboring image $I_k$. If the angle between $\mathbf{m}_{jk}$ and the desired direction is small (and the apparent size of $\text{Points}(C_j)$ in both images is comparable), then $C_k$ is a candidate left or right image. Out of all the candidates, we select the image $I_k$ whose motion magnitude $||\mathbf{m}_{jk}||$ is closest to $10\%$ of the width of image $I_j$.

Along with the left and right cameras, we show a "step back" camera, 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 rectangular box around a region of the current photo, or around a region of the point cloud. All points whose projections are inside the box are considered selected. Our system then takes over, searching for the "best" picture of the selected points. Each

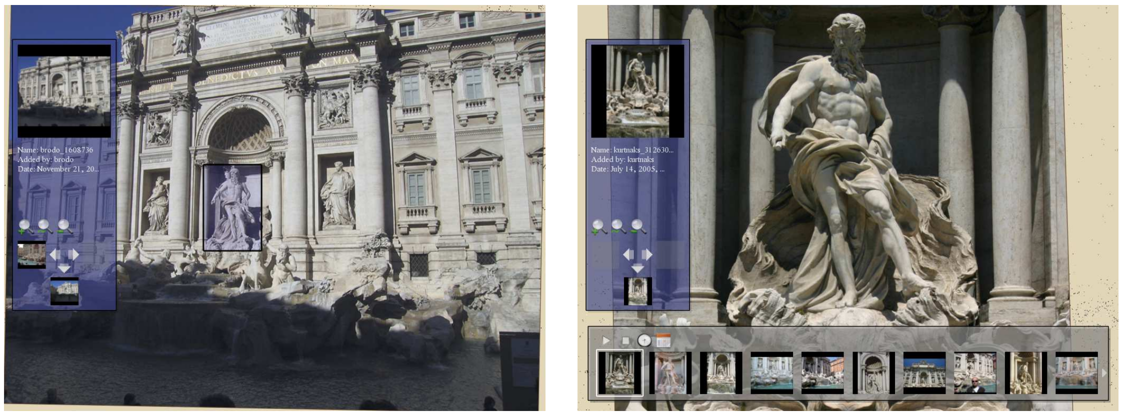

Figure 6: *Object-based browsing.* The user drags a rectangle around Neptune in one photo, and the camera moves to a new, high-resolution photograph of the statue.

image in the database is scored based on how good a representation it is of the selection. The top scoring photo is selected as the representative view, and the virtual camera is moved to that image. The rest of the 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, $E_{\mathrm{visible}}$, $E_{\mathrm{angle}}$, and $E_{\mathrm{detail}}$. Details of the computation of these terms can be found in Appendix B.

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 full 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 ($\mathrm{Points}(C_j)$) to refine the selection. We then compute the $3 \times 3$ covariance matrix for the selected points that are also in $\mathrm{Points}(C_j)$, and remove 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) the covariance matrix is defined with respect to the entire selected point set.

### 6.4 Creating stabilized slideshows

Whenever the thumbnails pane contains more than one image, its contents can be viewed as a slideshow by pressing the "play" button on 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 locked 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 in and view a previously saved sequence of images. 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 photographs 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 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 structure from motion 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 have 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. That annotation is then stored, along with the selected points, and appears as a semi-transparent rectangular box around the selected points. Once annotated, an object can also 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 $\mathrm{Points}(C_j)$. To check that the annotation is at an appropriate scale for the image, we determine the apparent size of the annotation in image $I_j$. If the annotation is visible and the

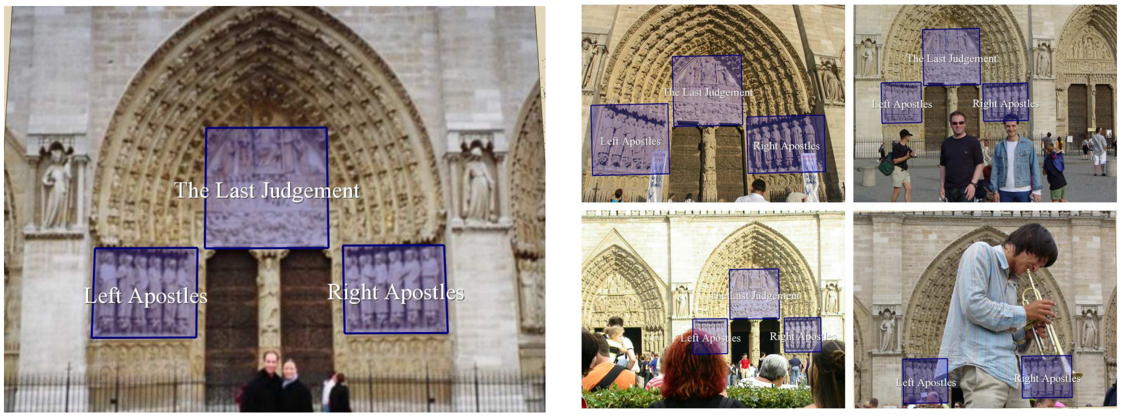

Figure 7: *Example of annotation transfer.* Three regions were annotated in the photograph on the left; the annotations were automatically transferred to the other photographs, a few of which are shown on the right. Our system can handle partial and full occlusions.

apparent size is greater than 0.05 (to avoid barely visible annotations), and less than 0.8 (to avoid 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 projections of the annotated points, with the label in the center of the box.

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 photographs 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 photographs. Tools such as 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 photographs using our system, we can potentially transfer the existing labels to every other photograph in the system. Other images on the web are implicitly annotated—for instance, an image appearing on a Wikipedia page is "annotated" with the URL of that page. By registering such images, we can link other photographs 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 of two days.

**Great Wall**: A set of 82 photographs taken during a walk along the Great Wall of China.

We have also experimented with "uncontrolled" sets consisting of images downloaded from the web. In each case, we ran our system on all of the downloaded photos; the system registered a subset of them.

**Notre Dame**: A set of 597 photos of the Notre Dame Cathedral in Paris. These photos were registered starting from 2635 photos from Flickr matching the search term "notredame AND paris."

**Yosemite**: A set of 325 photos of Half Dome in Yosemite National Park. These were registered from 1882 Flickr photos matching the search term "halfdome AND yosemite."

**Trevi Fountain**: A set of 350 photos of the Trevi Fountain in Rome, registered from 466 Flickr photos matching the search term "trevi AND rome."

**Trafalfar Square**: A set of 278 photos from Trafalgar Square, from 1893 Flickr photos matching the search term "trafalgarsquare."

Visualizations of these data sets are shown in Figures 1-8. Please see the accompanying video for a demonstration of features of our explorer interface, including object selection, related image selection, morphing, and annotation transfer, on several of these 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

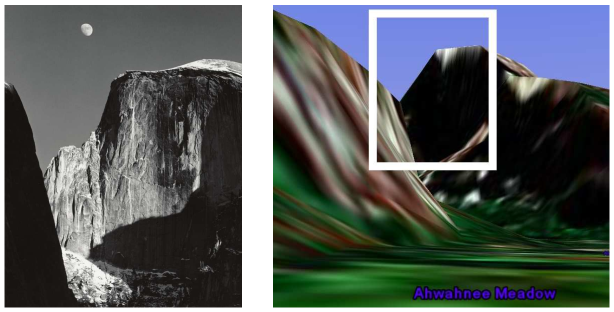

Figure 8: *A registered historical photo.* Left: Ansel Adams, "Moon and Half Dome," 1960. We registered this historical photo to our Half Dome model. Right: Google Earth's rendering of Half Dome from where Ansel Adams was standing, as estimated by our system (note that the Google Earth DEM is fairly low-resolution).

in Section 4.3.1. We then tried registering 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 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 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 web and from personal photo collections.

Our reconstruction algorithm has several limitations that we would like to address in the future. Our current structure from motion implementation becomes slow as the number of registered cameras grows larger. We would like to make the process more efficient, for instance, by choosing a better order in which to register the photographs. A more efficient ordering might reconstruct the large-scale structure of the scene with a relatively small number of cameras, then register the remaining cameras using more local optimizations. Also, our SfM procedure is not guaranteed produce a metric reconstruction from a set of photographs without ground control points, which makes reconstructing very accurate scene models more difficult. This problem will be alleviated as more georeferenced data becomes available. Scenes with many repeating structures can generate many erroneous key matches, and can be challenging to reconstruct with our technique. Finally, our algorithm currently only reconstructs one connected component of matching input images; we would like to be able 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 potential 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.

## References

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

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. *Journal of the ACM 45*, 6, 891–923.

BROWN, M., AND LOWE, D. G. 2005. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *International Conference on 3D Imaging and Modelling*.

CHEN, S., AND WILLIAMS, L. 1993. View interpolation for image synthesis. *Computer Graphics (SIGGRAPH'93)* (August), 279–288.

CHEW, L. P. 1987. Constrained delaunay triangulations. In *SCG '87: Proceedings of the third annual symposium on Computational geometry*, ACM Press, New York, NY, USA, 215–222.

COOPER, M., FOOTE, J., GIRGENSOHN, A., AND WILCOX, L. 2003. Temporal event clustering for digital photo collections. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, ACM Press, New York, NY, USA, 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 '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 11–20.

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 *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, IEEE Computer Society, Washington, DC, USA, 74.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The Lumigraph. In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, Proc. SIGGRAPH'96 (New Orleans), 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, March.

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

KADOBAYASHI, R., AND TANAKA, K. 2005. 3d viewpoint-based photo search and information browsing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM Press, New York, NY, USA, 621–622.

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

LIPPMAN, A. 1980. Movie maps: An application of the optical videodisc to computer graphics. *Computer Graphics (SIGGRAPH'80) 14*, 3 (July), 32–43.

LOURAKIS, M. I., AND ARGYROS, A. A. 2004. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. Available from `http://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.

MCCURDY, N. J., AND GRISWOLD, W. G. 2005. A systems architecture for ubiquitous video. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, ACM Press, New York, NY, USA, 1–14.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH'95)* (August), 39–46.

NAAMAN, M., PAEPCKE, A., AND GARCIA-MOLINA, H. 2003. From where to what: Metadata sharing for digital photographs with geographic coordinates. In *Proceedings of the 10th Interational Conference on Cooperative Information Systems*.

NAAMAN, M., SONG, Y. J., PAEPCKE, A., AND GARCIA-MOLINA, H. 2004. Automatic organization for digital photographs with geographic coordinates. In *JCDL '04: Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, ACM Press, New York, NY, USA, 53–62.

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

POLLEFEYS, M. 2002. Course 54: Obtaining 3d models with a hand-held camera. In *SIGGRAPH 2002*.

ROBERTSON, D. P., AND CIPOLLA, R. 2002. Building architectural models from many views using map constraints. In *Seventh European Conference on Computer Vision (ECCV 2002)*, Springer-Verlag, Copenhagen, vol. II, 155–169.

RODDEN, K., AND WOOD, K. R. 2003. How do people manage their digital photographs? In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press, New York, NY, USA, 409–416.

ROMÁN, A., GARG, G., AND LEVOY, M. 2004. Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE Visualization 2004*, 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 *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, vol. 1, 414–431.

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

SEITZ, S. M., AND DYER, C. M. 1996. View morphing. In *Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, Proc. SIGGRAPH'96 (New Orleans), 21–30.

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

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. *International Journal of Computer Vision 53*, 1 (June), 93–107.

TOYAMA, K., LOGAN, R., AND ROSEWAY, A. 2003. Geographic location tags on digital images. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, ACM Press, New York, NY, USA, 156–166.

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

## A Structure from motion optimization

Assuming a pinhole model, each 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 for each camera is reduced to seven: the 3D rotation expressing the orientation (three parameters), the camera center $\mathbf{c}$ (three parameters), and the focal length $f$ (one parameter). We use an incremental rotation, $\omega$ to parameterize the 3D rotation, where

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

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

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 $\mathbf{p}_j$. We also have a set of 2D projections, $\mathbf{q}_{ij}$, where $\mathbf{q}_{ij}$ is the observed projection of the $j$-th track in the $i$-th camera.

Let $\mathbf{P}(\mathbf{\Theta}, \mathbf{p})$ be the projection equation that maps a 3D point $\mathbf{p}$ to its 2D projection in a camera with parameters $\mathbf{\Theta}$. $\mathbf{P}$ first transforms $\mathbf{p}$ to homogeneous image coordinates, then performs the perspective division:

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

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

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

We thus wish to minimize the sum of the reprojection errors:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}||\mathbf{q}_{ij} - \mathbf{P}(\mathbf{\Theta}_i, \mathbf{p}_j)||$$

($w_{ij}$ 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; we have observed that this is often a good estimate. The other, $f_2$, is calculated using the EXIF tags of an image, and is undefined if the necessary EXIF tags are absent. This estimate is also usually good, but can sometimes 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 reasonable. If this test fails, we use $f_1$.

## B Image selection criteria

When evaluating the scoring function to determine how well an image represents a point select, the score for each image $I_j$ is 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 $E_{\text{visible}}$, we first check whether $P_{\text{inliers}} \cap \text{Points}(C_j)$ is empty. If it is empty, then the object is deemed not to be visible to $C_j$ at all, and $E_{\text{visible}} = -\infty$. Otherwise,

$$E_{\text{visible}} = \frac{n_{\text{inside}}}{|P_{\text{inliers}}|}$$

where $n_{\text{inside}}$ denotes the number of points in $P_{\text{inliers}}$ that project inside the boundary of image $I_j$.

Next, to compute $E_{\text{angle}}$, we first attempt to find a dominant plane in $P_{\text{inliers}}$ 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 $P_{\text{inliers}}$ 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 $E_{\text{angle}} = \text{Direction}(C_j) \cdot \hat{n}$. If enough points do not fit a plane, we set $E_{\text{angle}} = 0$.

Finally, we compute $E_{\text{detail}}$ to be the area, in pixels, of the bounding box of the projections of $P_{\text{inliers}}$ into image $I_j$ (considering only points that project inside the boundary of $I_j$). $E_{\text{detail}}$ is normalized by the area of the largest such bounding box of all images.