INVITED PAPER

Infinite Images: Creating and Exploring a Large Photorealistic Virtual Space

This proposed system uses 3-D-based navigation to browse large photo collections, and arrange them into themes, such as city streets or skylines, in a large image database.

By Biliana Kaneva, *Student Member IEEE*, Josef Sivic, *Member IEEE*, Antonio Torralba, *Member IEEE*, Shai Avidan, *Member IEEE*, and William T. Freeman, *Fellow IEEE*

ABSTRACT | We present a system for generating "infinite" images from large collections of photos by means of transformed image retrieval. Given a query image, we first transform it to simulate how it would look if the camera moved sideways and then perform image retrieval based on the transformed image. We then blend the query and retrieved images to create a larger panorama. Repeating this process will produce an "infinite" image. The transformed image retrieval model is not limited to simple 2-D left/right image translation, however, and we show how to approximate other camera motions like rotation and forward motion/zoom-in using simple 2-D image transforms. We represent images in the database as a graph where each node is an image and different types of edges correspond to different types of geometric transformations simulating different camera motions. Generating infinite images is thus reduced to following paths in the image graph. Given this data structure we can also generate a panorama that connects two query images, simply by finding the shortest path between the two in the image graph. We call this option the "image taxi." Our approach does not assume photographs are of a single real 3-D location, nor that they were taken at the same time. Instead, we organize the photos in themes, such as city streets or skylines and synthesize new virtual scenes by combining images from distinct but visually similar locations. There are a number of potential applications to this technology. It can be used to generate long panoramas as well as content aware transitions between reference images or video shots. Finally, the image graph allows users to interactively explore large photo collections for ideation, games, social interaction, and artistic purposes.

KEYWORDS | Image mosaicing; Internet images; large datasets; scene recognition; transformed image retrieval; virtual scene synthesis

I. INTRODUCTION

The number of digital images captured and shared online is growing at a phenomenal rate. It was recently reported[1] that Facebook currently stores 15 billion images (not including replications) and 850 million new photographs are added every month. Facebook is not alone; ImageShack, the largest photo-sharing service to date, hosts 20 billion images while News Corp's PhotoBucket and Yahoo's Flickr store 7.2 and 3.4 billion photos, respectively. Internet Vision is an emerging field at the intersection of computer

[1]TechCrunch: Who Has The Most Photos Of Them All? Hint: It Is Not Facebook, April 7, 2009.

Manuscript received April 11, 2009; revised July 15, 2009; accepted August 6, 2009. Date of publication May 27, 2010; date of current version July 21, 2010. This work was supported by ARDA VACE, by NGA NEGI-1582-04-0004, by MURI Grant N00014-06-1-0734, and by ANR project HFIBMR (ANR-07-BLAN-0331-01).

B. Kaneva and **A. Torralba** are with MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139 USA (e-mail: biliana@csail.mit.edu; torralba@csail.mit.edu).
J. Sivic is with INRIA, WILLOW Project Team, Laboratoire d'Informatique de l'Ecole Normale Superieure, CNRS/ENS/INRIA UMR 8548, Paris, France (e-mail: josef.sivic@ens.fr).
S. Avidan is with Adobe Research, Newton, MA 02466 USA (e-mail: avidan@adobe.com).
W. T. Freeman is with MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA 02139 USA, and is also with Adobe Research, Newton, MA 02466 USA (e-mail: billf@csail.mit.edu).

Digital Object Identifier: 10.1109/JPROC.2009.2031133

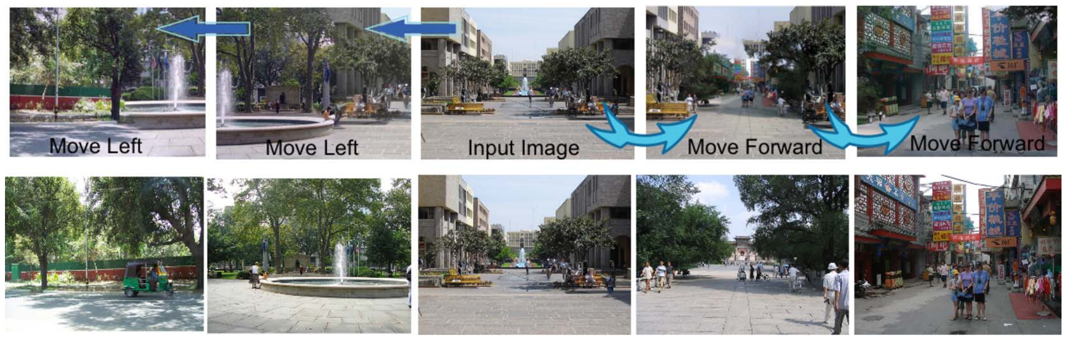


Fig. 1. ***Given the user supplied starting image (middle), our system lets the user navigate through a collection of images as if in a 3-D world. Top row: Snapshots of the synthesized virtual scene. Bottom row: Original images of different real 3-D locations automatically found in the image collection which were blended together to synthesize the virtual scene.***

vision and the Internet. It uses computer vision techniques to exploit this endless stream of images and, on the other hand, leverages these images to improve computer vision algorithms.

In this paper, we focus on ways to help users explore large collections of images using intuitive 3-D controls for ideation, games, social interaction, and artistic purposes. As a motivating example, consider Fig. 1. The user starts with a query image and then wants to naturally navigate the photo collection. Choosing to move left, for example, will cause the system to retrieve an image from the database that can be seamlessly stitched to the query image and produce the desired motion. The key observation is that the query and retrieved images do *not* have to come from the same 3-D location.

We construct an *image graph* where each node is an image and different types of edges correspond to different types of motions between nodes. Exploring a photo collection thus reduces to traversing the graph. The image graph can be used for other applications such as creating infinitely long panoramas by constantly panning the camera, or creating a photorealistic transition between two images by finding the shortest path connecting them in the graph. We describe how to construct the image graph, what are its properties and show a number of applications that are based on this representation.

On a technical level, this work can be viewed as an extension of image retrieval techniques. Traditionally, such systems retrieve images that match the query image. Here, we are interested in retrieving images that can be stitched to the query image and create the illusion of a natural photorealistic 3-D motion. We call our approach *transformed image retrieval*. The key challenge is to establish an edge between two nodes (images) if there is a smooth, photorealistic 3-D motion between them.

We do this by first applying some geometric transformation to the query image before performing the query operation. This transformation lets us predict how the retrieved image should look, at least in some parts of the image. We then proceed to retrieving the matching image from the database using this partial information. We approximate a number of 3-D camera motions with simple 2-D transformations and compute, for each image in the graph, its top matched candidates under various motions. The result is a sparse graph where every node is connected by a small number of edges (corresponding to different motion types) to other nodes in the graph.

The structure of the image graph impacts our ability to create photorealistic tours of the virtual space. If, for example, the image graph consists of a center node that all nodes are directly connected to, then all paths from one image to another will be extremely short and boring. If, on the other hand, the graph consists of many isolated connected components, then it will not be possible to reach many nodes in the graph. We analyze the graph and suggest ways to improve its structure to make paths more appealing.

Several applications can be reduced to finding paths in the image graph. We have made a Web-based interface that lets users explore the image collection interactively using an intuitive 3-D control. In another application, we create infinite panoramas starting with a query image and panning from image to image for as long as needed. We can also create a video clip that is made of the images along the path, which can be used as a transition effect between images. For example, we can create infinite zoom effects that resemble the "Droste effect"[2] which has been used by various artistic groups to generate infinite zooms.[3] The image graph, and transformed image retrieval, may be used to create large photorealistic environments for applications such as Second Life or computer games.

II. BACKGROUND

We represent the virtual photorealistic environment with the image graph and *not* with a 3-D model. As in other

[2]http://en.wikipedia.org/wiki/Droste_effect.
[3]http://zoomquilt2.madmindworx.com/zoomquilt2.swf.

examples of image-based rendering [1], [2], we generate the output images directly from input images, bypassing completely the 3-D model as an intermediate step. This is because capturing images is easy, but constructing 3-D models is time consuming and prone to errors.

We construct the image graph using image search methods. These methods can be roughly divided in two groups: those that use metadata cues such as timestamp, tags or geo-tagging, and those that rely purely on visual data. For example, time can be used to cluster images around events ("Mike's birthday," "Summer vacation of 1997") and, with the recent introduction of geo-tagging, images can be organized based on location ("Trip to Italy"). Google Street View and Microsoft Virtual Earth rely on geo-tagging to let users take a virtual trip around the streets of major cities around the world. Some image search engines organize images by tags that are either entered manually by users or collected automatically from the image caption or the Web page containing the image. The intersection of metadata cues lets users query images on a semantic level ("Find all images taken in New-York in 2004"). Recently, there has been great progress in the field of object [3] and, in particular, face detection [4]. Finding faces and automatically adding the corresponding names as tags to each image extends the range of queries ("Find pictures of Jonathan taken in Spain on November 2004").

In a purely visual search, on the other hand, the user supplies a query image that is then matched to images in the database using low level image features such as color, shape and texture [5], [6]. Image features can be also extracted with a controlled degree of invariance to viewpoint and illumination thus enabling viewpoint invariant object and scene retrieval [7].

In typical image search systems, the retrieved images are often displayed as thumbnails, which is a practical but not very engaging way to browse images. If all images are collected from the same point of view then a big Gigapixel image can be constructed and viewed interactively using familiar 2-D image controls [8].

Another option for browsing large collections of images is to treat images as points in a high dimensional space, compute the distances between them and use multidimensional scaling to display them in the image plane for the user to navigate through [9]. However, there is no effort to create a virtual 3-D world and as a result there is no sense of "being there."

If the image dataset consists of a collection of images of different scenes, taken at different times, one can construct an AutoCollage [10]. This gives visually pleasing collages, but fails to scale to large collections where thousands or millions of images are involved.

Alternatively, one can rely on 3-D context to organize images. This was underscored by the success of the PhotoTourism system [11]. First, images are retrieved using tags ("Find all images of Old City in Prague") and then calibrated to a common 3-D space. Users can then browse the image collection by moving in 3-D space. This was later extended to detect paths in the image collection which gives a better control in navigating the 3-D scene [12].

Our system is decidedly appearance-based as we only rely on image content to retrieve images. But unlike other image retrieval systems, we are not interested in retrieving similar images. Instead, we are looking for transformed image retrieval, where the retrieved image should match the query image after some transformation. Retrieved images are then linked to each query image, forming a large image graph.

This paper is an extended version of [13].

III. IMAGE GRAPH

In this section, we describe how we collect and arrange a large collection of images in an image graph. Each image in the collection corresponds to a node in the graph and an edge in the graph represents a transition between a pair of images. There are different types of edges corresponding to different camera motions. The image graph is illustrated in Fig. 2. The images do not need to correspond to a real unique 3-D space as in Phototourism [11]. Instead, our images are expected to correspond to unrelated, but visually similar, places.

To build the image graph we need to find for each image in the collection a set of candidate images suitable for a transition using a particular camera motion. These transitions will then form the edges in the graph. For instance, we want to find images similar to a photo taken if

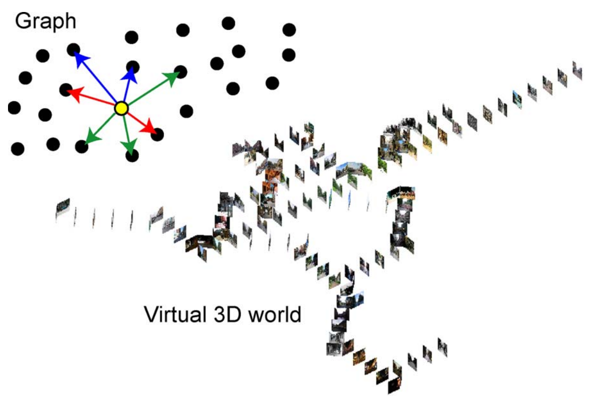


Fig. 2. ***Top: In the image graph, each image is a node and there are different types of edges (color coded in the illustration) that correspond to different camera motions. For each motion, there are several possible transitions to different images and we typically keep just the top ten transitions for each edge/motion type. Bottom: Portion of an image graph laid out in a virtual 3-D world. Relative locations of images are given by the camera motions. For example, for the forward motion, the next image is displayed in front of the query image. Here, only some edges of the graph are shown.***

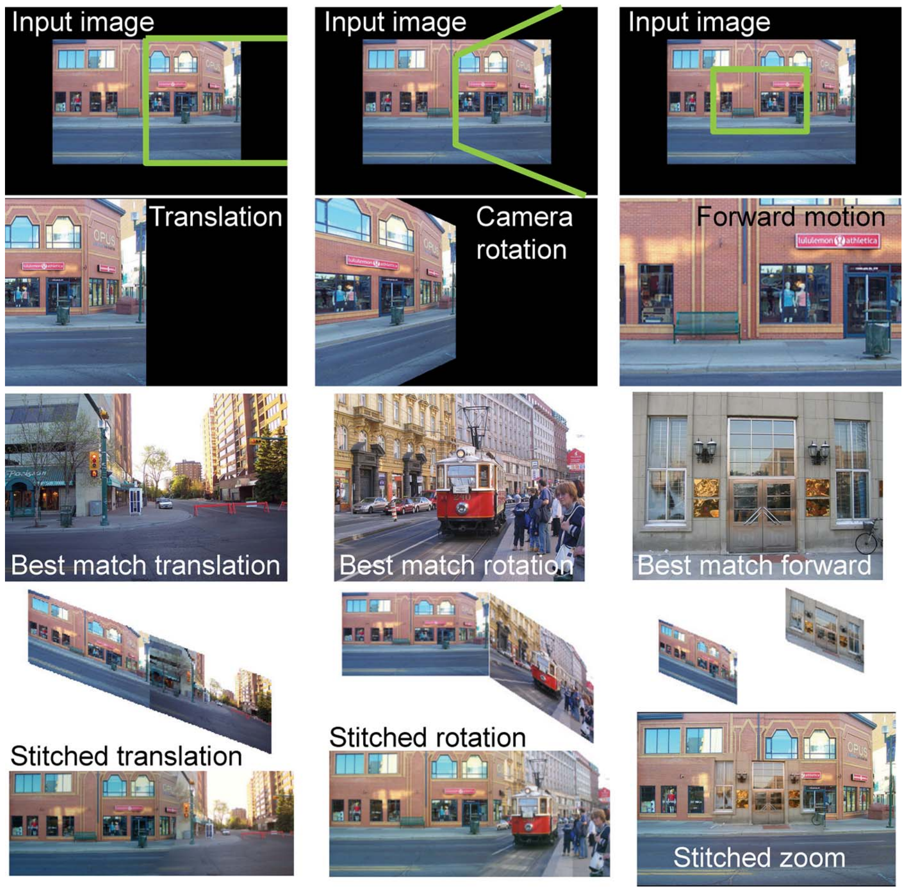


Fig. 3. ***Scene matching with camera view transformations. First row: The input image with the desired camera view overlaid in green. Second row: The synthesized view from the new camera. The goal is to find images which can fill-in the unseen portion of the image (shown in black) while matching the visible portion. The third row shows the top matching image found in the dataset of street scenes for each motion. The fourth row illustrates the induced camera motion between the two pictures. The final row shows the composite after Poisson blending.***

we rotate the camera by 45° with respect to the query image. We hope that the candidate set contains images that can be seamlessly blended with the query image after applying the appropriate camera transformation. We call this process *transformed image retrieval* and show how a simple planar image model can be used to retrieve images that are related to the query image by 3-D camera transformations.

A. Image Collection

We collected a dataset of more than 6 million images from Flickr by querying for suitable image tags and relevant groups. We queried for particular locations in conjunction with tags such as "New York street" or "California beach" and also downloaded images from Flickr groups returned by search on terms relevant to particular themes, such as "Landscape" or "Forest." The typical resolution of the downloaded images is 500 × 375 pixels. One million jpeg compressed images takes about 120 GB of hard-disk space.

B. Transformed Image Retrieval

Our goal is to extend a given image using images from the image database to give an impression of a particular camera motion. We consider three camera motions: i) translation left/right; ii) horizontal rotation (pan); and iii) zoom (forward motion). The camera motions are illustrated in Fig. 3. First, we synthesize a new view of the current image as seen from the new desired camera location. Camera translation is approximated by a translation in the image plane, ignoring parallax effects, horizontal camera rotation is achieved by applying appropriate homography transformation to the image, and zoom is approximated by scaling the image. For computing the horizontal camera rotation homography [14, p. 11], we assume no vertical or in-plane rotation, and set the unknown camera focal length to be half the image width, which corresponds to rather wide-angle lens, in order to exaggerate the image distortion.

We seek to find semantically similar images in the database coarsely matching the geometry of the observed portion of the synthesized view. For example, when the camera rotation changes a fronto-parallel view of a building to a view with a strong perspective (as shown in the middle column of Fig. 3), we find most retrieved images depict scenes looking down a street.

C. Image Representation and Matching

A key component of our approach is finding a set of semantically similar images to a given query image. For example, if the query image contains a cityscape in a sunset with a park in the foreground, we would like to find a candidate set of images with similar objects, scene layout, and lighting.

Semantic scene matching is a difficult task but some success has been recently shown using large databases of millions of images [15], [16]. We show that we can also induce camera transformations without an explicit model of the 3-D geometry of the scene. Matching results are sometimes noisy; for example, a river is sometimes mismatched for a road. In a recent work on image completion [15], this issue was addressed by relying on user interaction, essentially allowing the user to select a visually pleasing result among a set of candidate completions. We wish to find matching images automatically or with minimal user interaction. To reduce the difficulty of scene matching, we train classifiers to preselect images of particular scene types or "themes" from the entire image collection. Details are given in Section III-F. Image graphs are then built only from images of a particular theme or a combination of themes. Next, we describe the image features used for semantic scene matching and their application to transformed image retrieval.

Images are represented using the GIST descriptor, which was found to perform well for scene classification [17]. The GIST descriptor measures the oriented edge energy at multiple scales aggregated into coarse spatial bins. In this work we use three scales with (from coarse to fine) 4, 8, and 8 filter orientations aggregated into 6×4 spatial bins, resulting in a 480-dimensional image descriptor. While images of similar scenes, for example a street in New York and a street in London, can have very different colors and image intensities, we expect the coarse layout and orientation of image edges to be similar. The GIST descriptor can match similar scenes imaged under vastly different illuminations; for example, a street scene in a bright daylight can be matched reasonably well with a street scene in night. However, as we want to blend the matched images into seamless transitions, we would like to find matching scenes with similar illuminations. Hence, we also represent a rough spatial layout of colors by downsampling each of the RGB channels to 8×8 pixels. We normalize both GIST and the color layout vectors to have unit norm to make both measures comparable. Similar image descriptors were used for scene matching in [15].

As illustrated in Fig. 3, not all pixels are observed in the transformed image and hence only descriptor values extracted from the observed descriptor cells in the query image are considered for matching. In this case, the GIST and the color layout vectors are renormalized to have unit norm over the visible region.

The distance between two images is evaluated as the sum of square differences between the GIST descriptors and the low-resolution color layouts of the two images. We set the weight between GIST and color to 0.5, which we found is a good compromise between matching on the geometric structure of the scene captured by GIST and the layout of colors. Currently we consider only images in landscape format with an aspect ratio close to 4 : 3. This represents about 75% of all collected images.

Fig. 4 shows an example of a query image, the bins used to compute the image descriptor and the closest matching images from a dataset of 10 000 street images. The images returned belong to similar scene categories and have similar camera properties (same perspective pattern and similar depth).

D. Creating Seamless Transitions: Alignment and Compositing

We want to simulate transitions between pairs of images depicting different places, such as a street in New York and a street in London. This is a challenging task as image structures (such as windows on buildings) would not necessarily be aligned and can have different shapes and colors. Moreover, images returned by the scene matcher can still be misaligned as the GIST descriptors matches only the rough spatial structure given by the 6×4 grid of cells. For example, in the case of city skylines, the horizon line can be at a slightly different height.

To address these issues, we first align the matched image with the query using a global image transformation (translation and scale change) to compensate for gross misalignments. Next we find a good seam between the aligned images. The goal is to transition between the two images at places where their image intensities are similar. Finally, to reduce the remaining misalignment artifacts and differences in color, the two images are blended together in the gradient domain.

To perform the alignment, we apply a standard gradient descent alignment [14], [18] minimizing the mean of the sum of squared pixel color differences in the overlap region between the two images. We search over three parameters: translation offset in both the x and y direction and scale. The alignment is performed on images downsampled to 1/6 of their original resolution. In the case of translations and zoom, the alignment search is initialized by the synthesized view transformation. In the case of camera rotation, we initialize the alignment with a translation in the x direction matching the image overlap induced by the rotation, e.g., half the image width for the example in middle column of Fig. 3. As a result, the camera rotation is approximated by a translation and scale in the image domain. The camera rotation is only used in the

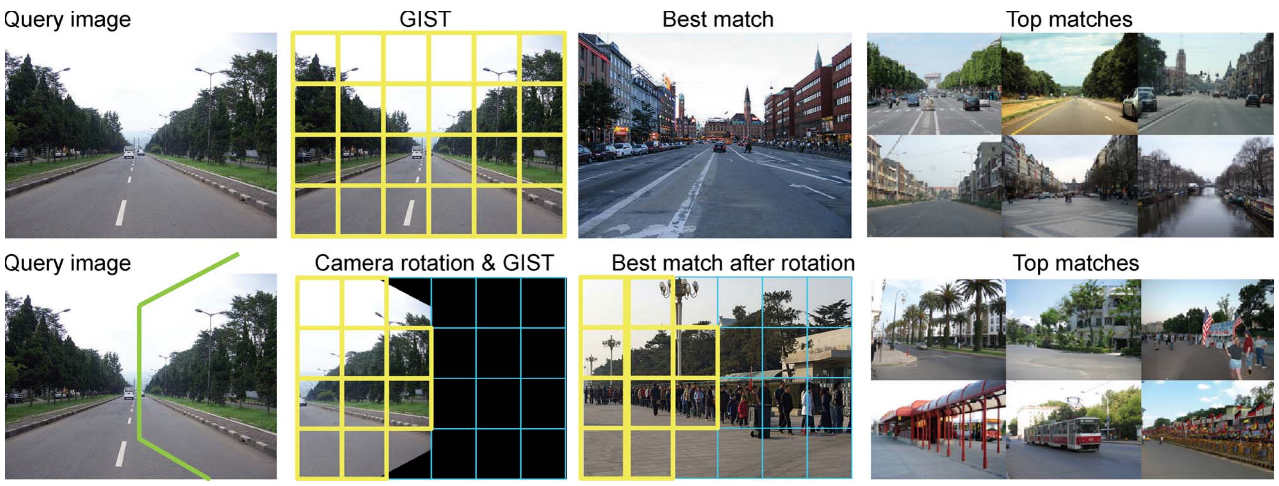


Fig. 4. *Each row shows the input image, the 6 $\times$ 4 spatial bins used to compute the GIST descriptor, the best match on a dataset of 10 000 images, and the next six best matches. Top row: We look for images that match the full GIST descriptor. Bottom row: Result of a query after simulating a camera rotation. The returned images contain a new perspective, similar to the one that we will have obtained by rotating the camera 45° to the right.*

scene matching to induce a change in the geometry of the scene.

The aligned images are blended along a seam in their region of overlap using Poisson blending [19]. In the case of camera translation and rotation, we find a seam running from the top to the bottom of the overlap region minimizing the sum of absolute pixel color differences by solving a dynamic program [20]. In the case of zoom, where images are within each other, we find four seams, one for each side of the overlap region. In addition, to preserve a significant portion of the zoomed-in image, we constrain each seam to lie close to the boundary of the overlap region. Finally, images are blended in the gradient domain using the Poisson solver of [21]. Examples of composited images are shown in the bottom row of Fig. 3.

E. Horizon Line Estimation

Although we do not perform a full estimation of the 3-D geometry of the scene structure, estimation of the horizon line can improve the quality of image transitions, as illustrated in Fig. 5. In this example, we want forward motion to represent a person walking on the ground plane. As shown in the top row, if we zoom into the picture by using the image center as the focus of expansion, we move into the sky region. However, if we zoom in on the horizon line we simulate a person moving on the ground plane. It is important to keep the horizon line within the query region.

We show how to estimate the horizon line from a single image by learning a regression function on the GIST descriptor [22], [23]. In contrast to [24] to [25] in which camera orientation is estimated by an explicit model of the scene geometry, we use machine learning to train a regressor. Our method works even when the scene lacks clear 3-D features such as lines converging towards a vanishing point. We collected 3000 training images for which we entered the location of the horizon line manually (for pictures taken by a person standing on the ground, the horizon line can be approximated by the average vertical location of all the faces present in the image). We use a weighted mixture of linear regressors [26] to estimate the location of the horizon line from the GIST features as described in [23].

F. Organizing Images Into Themes

The issue of semantic mismatches in retrieved images is especially significant in the case of transformed image retrieval where the information available for matching is

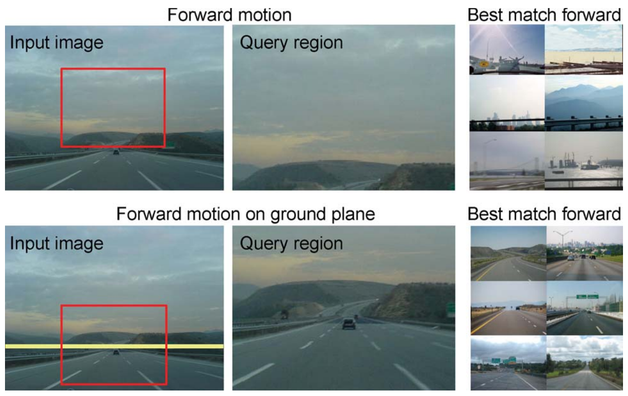


Fig. 5. *The top row shows the query region when we take the central image portion. The bottom row shows the results obtained when centering the query region on the horizon line. The retrieved images contain roads taken from similar viewpoints to the input image.*

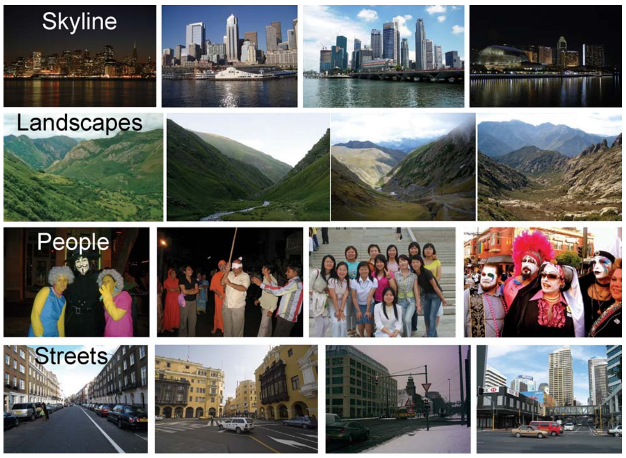


Fig. 6. ***Example of images belonging to different scene themes. Partitioning a large collection of images improves the quality of the results. The classification is done automatically. When navigating through the image collection, it is important to keep the themes constant when moving from one picture to another to avoid undesired transitions. The user can also allow transitions across themes.***

weaker than the original GIST descriptor (due to the smaller image overlap after the transformation). This can result in semantically incoherent transitions between images.

To augment the GIST descriptor, we train a classifier to identify images within some semantic theme. The theme of an image is generally invariant to camera motions. We can dramatically improve retrieval results by matching only pictures that belong to the same theme. This also lowers the memory and CPU requirements as only part of the database needs to be searched. Examples of themes we consider in this work are shown in Fig. 6.

To obtain a visually and semantically coherent set of images for each theme we train theme classifiers from manually labeled training data in a manner similar to [27] and [28]. For each theme, we train 1-versus-all nonlinear Support Vector Machine classifier [29] from about 1000 positive and 2000 negative training images. We have developed a suitable labeling interface so that the classifier can be trained interactively. At each iteration, the most uncertain images are shown to the user for labeling. The interface also allows the user to visually assess the classifier performance and label additional images if needed.

G. Building Image Graphs

We process the image database and create a graph for each type of camera motion: i) rotate left; ii) rotate right; iii) move left; iv) move right; and v) move forward. We call graphs for one type of camera motion a motion graph. Motion graphs for different motions are combined into a single image graph, which we refer to as the combined graph. For each transition between a pair of images (corresponding to an edge in the graph) we store the transformation aligning the images, the GIST matching cost of the two images, and the cost of the seam. In Section IV we show how to convert these costs into probabilities by manually labeling several examples of good and poor quality transitions. We typically retain only the top ten matches measured by the GIST matching cost for each motion. This gives us a sparse graph that can be easily stored and allows for further efficient processing. Currently, we create a separate graph for each theme.

Note that edges in the image graph for each motion are directed, describing transitions from the query image to the (top 10) best matches in the database. However, the direction of each edge could be reversed by reversing its image transformation. For example, a "rotate left" edge from image *A* to image *B*, could be reversed to a "rotate right" edge from *B* to *A*. Note, however, that image matching is not symmetric, as the fact that *B* is among the top ten nearest neighbors of *A* does not imply that *A* is among the top ten nearest neighbors of *B*.

In terms of computation cost, computing GIST descriptors takes about 0.2 s per image, querying a database of 100 K images takes about a second in our Matlab implementation, and it takes a couple of seconds to align and blend the retrieved images with the query image. These timings are for a 2 GHz machine.

In our current implementation, we do not enforce the image graph to respect constraints of a real physical space. Each transition depends only on the particular query image and there are no additional dependencies on other edges in the graph. As a result, it is perfectly legal in our image graph to keep rotating without coming back to the query image. Similarly, it is possible to start with a query image, move left, then move right, and not get back to the original query image.

IV. IMAGE GRAPH PROPERTIES

The motion graphs allow us to create photorealistic tours of the virtual image space using a single 3-D motion. To add versatility to our system and make the tours more interesting, we merge the separate motion graphs into a single graph where the directed edges correspond to a particular type of motion and their weights represent the probability that the edge produces a good quality transition (Fig. 2). The properties of the resulting image graph impact the quality of the tours through the image space. For example, if the graph has nodes with a large number of incoming edges (hubs), it is likely that most tours would pass through the same subset of images and the paths between any pair of nodes would contain only a small number of transitions. In that case, exploring the image space can quickly become boring. In this section, we discuss the presence of such high degree nodes in the graph and contrast the structure of the image graph to that of a random graph. Edges with low transition probabilities can also affect the quality of the tours because they can result in paths connecting semantically incoherent images. We propose several ways to augment the image graph in order to produce better tours and discuss how they affect the connectivity. If the graph has multiple isolated connected components, some portions of the graph would never be reached. A path leading to a small connected component could force the user to visit the same nodes over and over again when trying to generate longer tours. In discussing graph properties in this section, we analyze a graph constructed based on the "streets" theme that combines all five types of motion—rotate right, rotate left, move right, move left, and move forward.

A. Estimating the Probability of a Good Quality Transition

Each motion graph provides the user with the option to explore the image space using one particular camera motion, but to create interesting tours we would like to simulate different camera motions. As described in Section III-G, the edges in the graph can be assigned weights based on the GIST matching cost or the seam cost. Since these are computed differently for each motion graph, they are not necessarily comparable. Instead of using the previously described edge costs for merging the graphs, we fit a model using logistic regression to convert the edge cost to a probability of a good transition for each motion based on 300 hand-labeled training examples. As training data we randomly picked 300 pairs of nodes from each motion graph. After stitching and blending the images using the appropriate motion, we labeled the results as good or poor quality. Fig. 7 shows examples of good and poor quality transitions for rotate right, move right, and move forward camera motions. The labeling is subjective and was based on the labels of a single user. In our training data, we have about 50% positive and 50% negative examples for all motions except move forward where there are only 35% positive examples. The distribution of the GIST matching costs of the positive and negative examples show more overlap than that of the seam costs (Fig. 8) making it harder to learn a good model. For the rest of our experiments, we use models mapping the seam cost (rather than the GIST cost) to the probability of a good transition as shown in Fig. 9(a). The difference in models for different motions shows that using the seam cost directly to determine transition probabilities in a combined graph would give unequal weights for different camera motions.

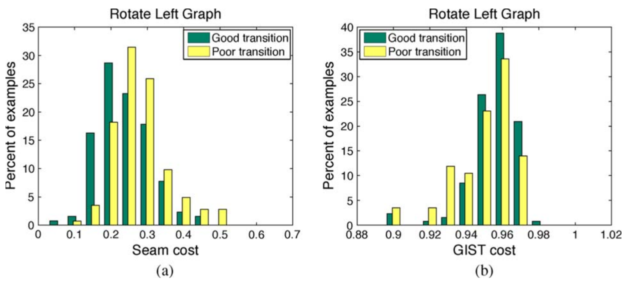


Fig. 8. ***(a) Fitted models mapping seam cost to probability of a good transition based on 300 labeled examples for each motion graph. (b) Histograms of edge probabilities in 1 K, 10 K, and 100 K graphs.***

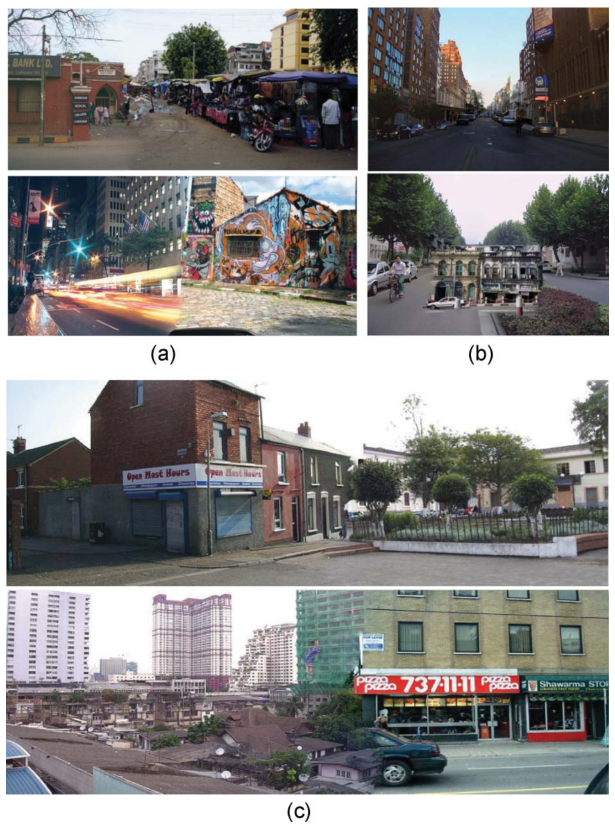


Fig. 7. ***Examples of good (top row) and poor (bottom row) quality transitions for different motion graphs. (a) Rotate right graph. (b) Move forward graph. (c) Move right graph.***

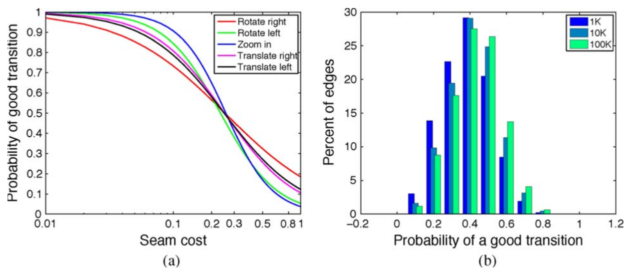


Fig. 9. ***Edge cost distribution for the 300 manually labelled good and poor quality transitions in the "rotate left" graph. (a) Seam cost. (b) GIST matching cost. Lower cost should correspond to better quality transitions.***

B. Combining the Motion Graphs

We use the learned models for mapping the seam cost to the probability of a good quality transition to assign new weights to the edges of the different motion graphs. With comparable edge weights, we then can merge the motion graphs into a single graph. The directed edges in the new image graph correspond to the different types of motion and the weights refer to the probability that the given motion produces a good quality transition. If there is an edge between a given pair of images in multiple motion graphs, we retain only the edge for the motion that results in the best quality transition. We consider the distribution of the edge weights in graphs with 1 K, 10 K, and 100 K nodes. The images of the 1 K and 10 K graphs were a random subset of the images of the 100 K graph. Intuitively, we expect that adding more images to the graph will increase the probability of finding good quality transitions between images. The distribution of the edge weights, indeed, shifts with the increase in graph size [Fig. 9(b)]. The 100 K graph contains a larger proportion of edges with higher probability of producing a good transition than do the smaller image graphs.

C. Popular Images

An interesting aspect of the graph is the node degree for the incoming edges. The number of outgoing edges in the combined graph is almost constant because for each motion graph we only keep a subset of the edges corresponding to the top 10 matches measured by the GIST matching cost. The number of incoming edges, however, varies significantly [Fig. 10(a)]. The majority of the nodes have 50 or fewer incoming edges with 5–8% of nodes having one or none. There are also a number of very well connected nodes with thousands of incoming edges in a graph with 100 K nodes. A sample of the popular images in different motion graphs is shown in Fig. 11(a). Most of the images are very textured, especially those from the forward motion graph, or have a lot of structure. They also have very similar color layout. It is likely that if there is no good match in the database for a given query image, due to their regularity and uniform lighting these images can give good matching costs even though they may not be semantically correct. Even if the transitions to a popular image are semantically correct, too many paths going through one image would make the image space exploration uninteresting. The popular images from the forward motion graph are clearly incorrect semantically. Interestingly, in the combined graph, none of them are amongst the top most high-degree nodes. Most of the popular nodes seem to come from the rotate and translate motion graphs. This can be explained by the fact that we have two graphs for both rotate and translate motions and all four of them favor images with similar structure thereby preserving the connectivity of their popular nodes. A sample of images with no incoming edges are juxtaposed with the popular ones in Fig. 11. The images with no connections show wide color and lighting variation in comparison to the dull colors and uniform lighting of their popular counterparts.

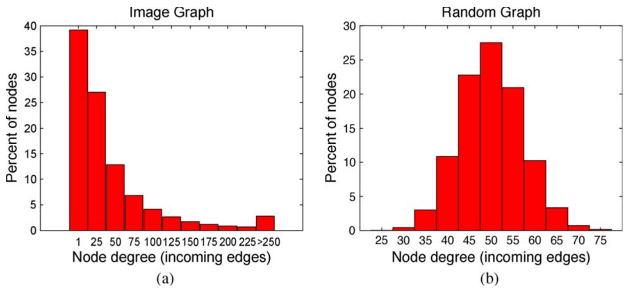


Fig. 10. ***Histograms of incoming edges. (a) 100 K image graph. (b) 100 K random graph.***

To further understand the distribution of edges in the image graph, we constructed a random graph. First we created a graph for each motion by adding 10 randomly chosen outgoing edges to each node and assigning the edge weights to values between 0 and 1 by randomly drawing from the uniform distribution. The graphs were then merged together to create the random combined graph. The distribution of the incoming edges in the random graph is quite different from that in the image graph. Here, we have no nodes with fewer than 20 and with more than 80 incoming edges [Fig. 10(b)]. The properties of similar types of random graphs have been studied in e.g., [30]. The emergence of highly connected nodes in random and real-world graphs was recently studied in [31].

D. Graph Connectivity

To ensure versatility of the tours of the virtual space, we would like to be able to reach a node from any other node in the image graph. In graph theory terms, the image graph should be strongly connected [32]. We already saw that there is a small percentage of nodes in the image graph that have no incoming edges. Unless we start the exploration of the image space from one of those nodes, we will never be able to visit them. It is possible that the image graph is not well connected, i.e., it could have several isolated strongly connected subgraphs. Our tours will then be limited to one portion of the graph which would be undesirable. We consider the size of the largest connected component in the graph. Ideally, most nodes, with the exception of those with no incoming edges, will be part of it. Fig. 12 shows that over 90% of the nodes lie in the largest strongly connected subgraph.

We wish to improve the quality of the transitions but at the same time retain the ability to travel through a large portion of the image graph. One option is to remove edges with low probability of producing a good quality transition. It appears that we can safely discard about 30% of those edges without significantly reducing the size of the largest connected component [Fig. 12(a)]. In contrast, we can

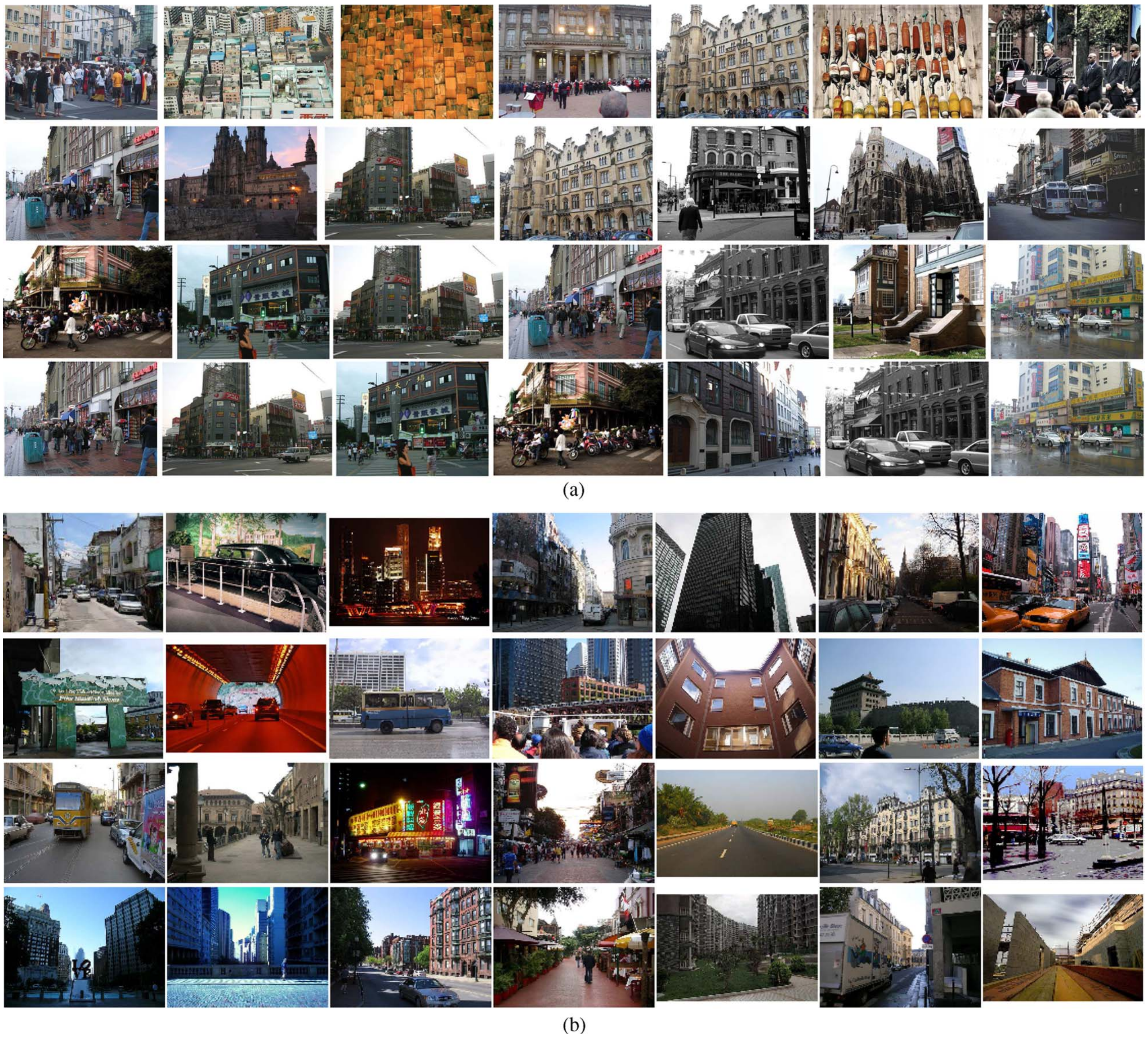


Fig. 11. ***Sample images with (a) many or (b) no incoming edges in different motion graphs. Top row: Forward motion. Second row: Rotate right motion. Third row: Move left motion. Bottom row: Combined motions.***

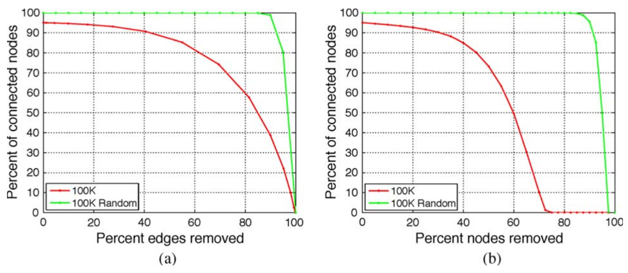


Fig. 12. ***Percentage of nodes in the largest connected component of the 100 K image and random graphs. (a) After removing edges with low probability of a good transition. (b) After removing nodes with the highest number of incoming edges.***

remove almost 90% of the poor quality edges in the random graph, before the graph becomes disconnected at all [Fig. 12(a)]. This behavior can be attributed to the uniform distribution of edges and edge weights in the random graph. In contrast, edge occurrences (and their weights) in the image graph are likely to be correlated and dependent on the type and content of the query image.

Another option for improving the tours through the image space is to reduce the number of times we pass through the same node. To achieve this, we can remove nodes with many incoming edges. At the same time, we would like to preserve the proportion of nodes that belong to the largest connected component in the resulting subgraph. Fig. 12(b) shows the percentage of nodes left in the largest connected component after removing a given number of high-degree nodes from the graph. We can safely discard about 20% of the most popular nodes without significantly affecting the connectedness of the image graph. This suggests that the high-degree nodes are not central to the graph connectivity. As we observed earlier, the random graph is much better connected than the image graph. Due to the uniform distribution of edges in the random graph, even removing 80% of the high-degree nodes still leaves the remaining graph well connected.

E. Path Properties

Our system can be used for finding tours between a given pair of images ("image taxi" described in Section V-B) or providing transitions between images of a personal collection. In both scenarios, we would like to find a path with edges that have high probability of providing a good transition. We use the Dijkstra's shortest path algorithm [32] to find the best path between a pair of nodes selecting the best quality edge at each step. In the image taxi scenario in particular, we prefer longer but still good quality paths. We already know that our graph is well connected as discussed in Section IV-D. The question we ask now is: How far away each node is from all other nodes?

We computed the average length of the best path to 1000 randomly sampled nodes in image graphs with 1 K, 10 K, and 100 K nodes [Fig. 13(a)]. It takes on average 4–6 transitions to reach a node from any given node. This suggests that no two nodes lie very far from each other, which can be explained by the portion of very high-degree nodes in the image graph. In the random graph, on the other hand, the average path has about twice as many transitions as those in the image graph. Despite the fact that the random graph is better connected, its nodes lie farther apart from each other because there are no hubs in it. The number of transitions also increases as we add more nodes to the graph since the number of outgoing edges for each node does not vary. This effect is more pronounced in the random graph but it can be also seen in the image graph.

V. APPLICATIONS OF THE IMAGE GRAPH

Once the image graph is constructed, we can use it to create different applications using the same basic machinery—finding paths in the graph. We present an interactive Web interface that allows the user to navigate the image space using intuitive 3-D controls. We also use the graph to create "infinite" panoramas starting from a given query image and traversing the graph using a particular camera motion. The image taxi tool allows the user to generate a tour between two images by finding a good quality path between the corresponding nodes in the graph. Finally, the graph could be augmented to include a personal photo collection that can later be viewed by using images from the graph as transition "filler" images.

We use three different camera motions in the proposed applications: translation, rotation, and forward motion. When creating smooth transitions (see the accompanying video [33]), the camera translation is simulated by translating with constant per pixel speed between the pair of input images (also applying small scale changes if the consecutive images are of different size). The case of camera rotation is similar to translation but, in addition, we display the resulting images on a cylinder [14]. In the case of forward motion, we apply constant scale change between the two images to create an impression of a person walking at a constant speed in the 3-D world. The focus of expansion for the forward motion is at the center of the next aligned image, which can result in (typically small) changes in forward motion direction between consecutive images.

As mentioned in Section II, in the case of photographs of the same 3-D scene, smooth paths between images can also be defined based on actual (reconstructed) camera positions in the 3-D space [12].

A. Infinite Panoramas

One application of the image graph is to create an infinite panorama by starting with a query image and

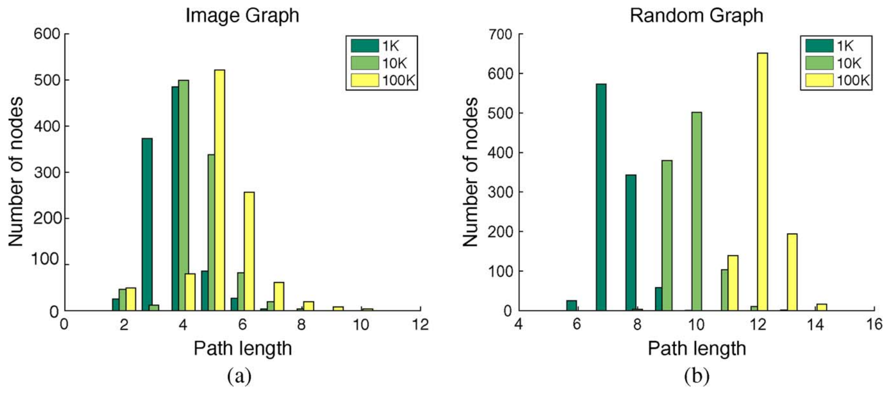


Fig. 13. *Average path length to a given node from all other nodes in the graph (sample of 1000 nodes). (a) Image graph. (b) Random graph.*

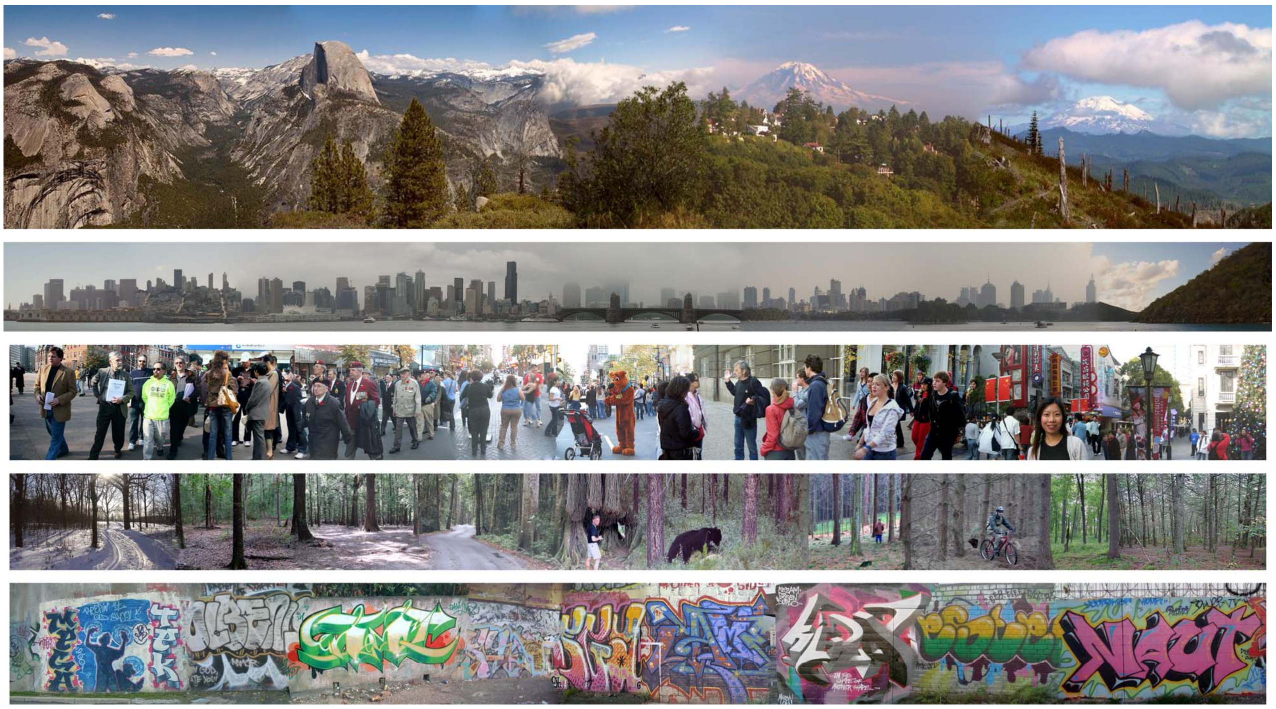


Fig. 14. *Various panoramas created by our system. The top two panoramas were created automatically from the "landscape" and "skyline" themes, respectively. The bottom three panoramas were created interactively from the "people," "forest," and "graffiti" themes, respectively.*

following the "camera translate" edge from image node to image node. Each retrieved image is then blended with the query image to create an ever larger panorama. Fig. 14 shows examples of long panoramas created for various semantic themes.

We can create panoramas for each of the three camera motions: translation, rotation, and forward motion as is demonstrated in Figs. 15(a), 15(b), and 16, respectively. These panoramas were created from the "streets" theme by starting with a query image and following the same camera motion edge from node to node. Note that the left/right translation motion tends to preserve the camera orientation with respect to the scene (the scene remains roughly fronto-parallel), while the rotation motion induces a change in perspective between consecutive images (Fig. 17). The translation and rotation sequences were produced fully automatically. The forward motion sequence was produced interactively by letting the user specify the direction of the motion—toward the horizon of the image. Because it is difficult to display an infinite zoom on paper, please refer to the accompanying video [33]. As mentioned earlier, there is no restriction in our representation for a rotation panorama to go beyond 360° *without* returning to the first query image.

B. Image Taxi: Finding a Path Between Two Images

The image taxi tool lets the user specify the first and last images of a tour and then computes the shortest path in the image graph that connects the two end images. This option can be used to generate smooth transitions between images from a private photo collection, using a large, generic image database, or to generate a smooth transition between two video shots. Given two input images, we first connect them to the graph and then find the shortest path between them using the Dijkstra algorithm [32]. We then follow the path creating a tour based on the different edges along the way. We demonstrate the image taxi in the accompanying video [33].

The video also contains the different tours generated by our system. All the sequences presented in the video were generated automatically, except for the "Hollywood" and "Windows" sequences where the user had to choose the best match out of the top five candidates.

C. Interactive System

We have designed an interactive viewer that allows exploring the image graph using intuitive 3-D commands (move left/right, zoom, rotate left/right). The viewer is implemented in Flex and runs using Flash Player 10 in a standard Web browser. The user can select a particular image graph (theme) to explore. After choosing an image that serves as a gateway to the virtual 3-D space [Fig. 18(a)], s/he can start navigating interactively. Five regions of the screen correspond to the different motions the user can take. Moving the mouse displays the top matching image for each motion overlaid on the current image [Fig. 18(b)]. The

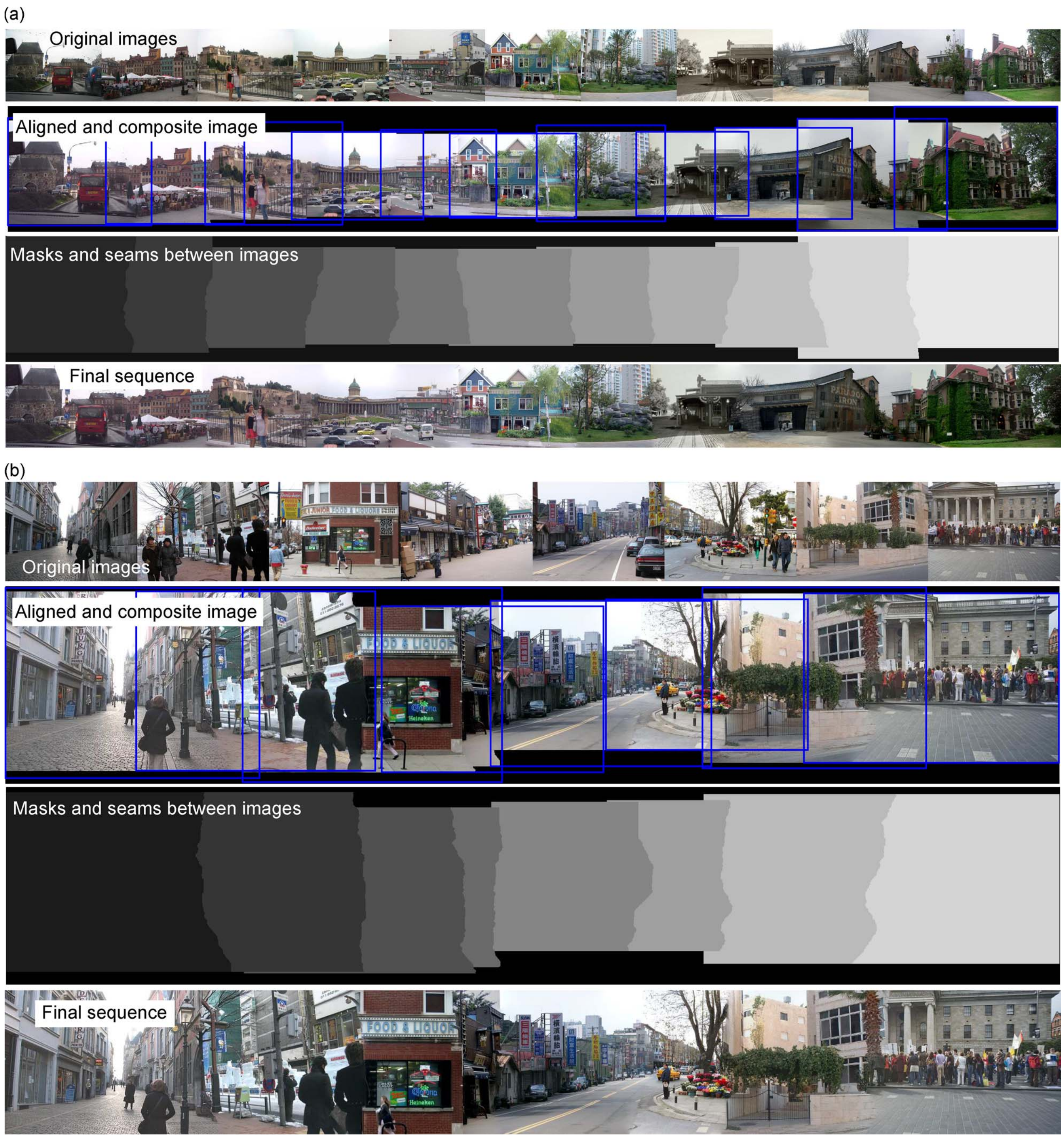

(a)
Original images
Aligned and composite image
Masks and seams between images
Final sequence
(b)
Original images
Aligned and composite image
Masks and seams between images
Final sequence

Fig. 15. *Two sequences generated from a large database of street images by inducing two camera motions. (a) Camera translation. Each consecutive image was obtained by inducing camera translation by half the image size as illustrated in Fig. 3. (b) Camera rotation. Each consecutive image was obtained by inducing camera rotation of 45 degrees as illustrated in Fig. 3. This produces changes in the perspective between consecutive images.*

system automatically moves to the image selected by the user [Fig. 18(f)–(h)]. If the user is not satisfied with the top matching image he can choose a different image from the top ten matching images [Fig. 18(e)]. The system also allows to undo a transition providing the user with an opportunity to explore alternate routes [Fig. 18(d)].

The transition between every pair of images must be smooth to give the impression of "being there." Thus, it is important to be able to synthesize each intermediate image frame efficiently. The images in the graph have already been aligned offline (Section III-D), therefore, to stitch two images we only need to compute on demand the best

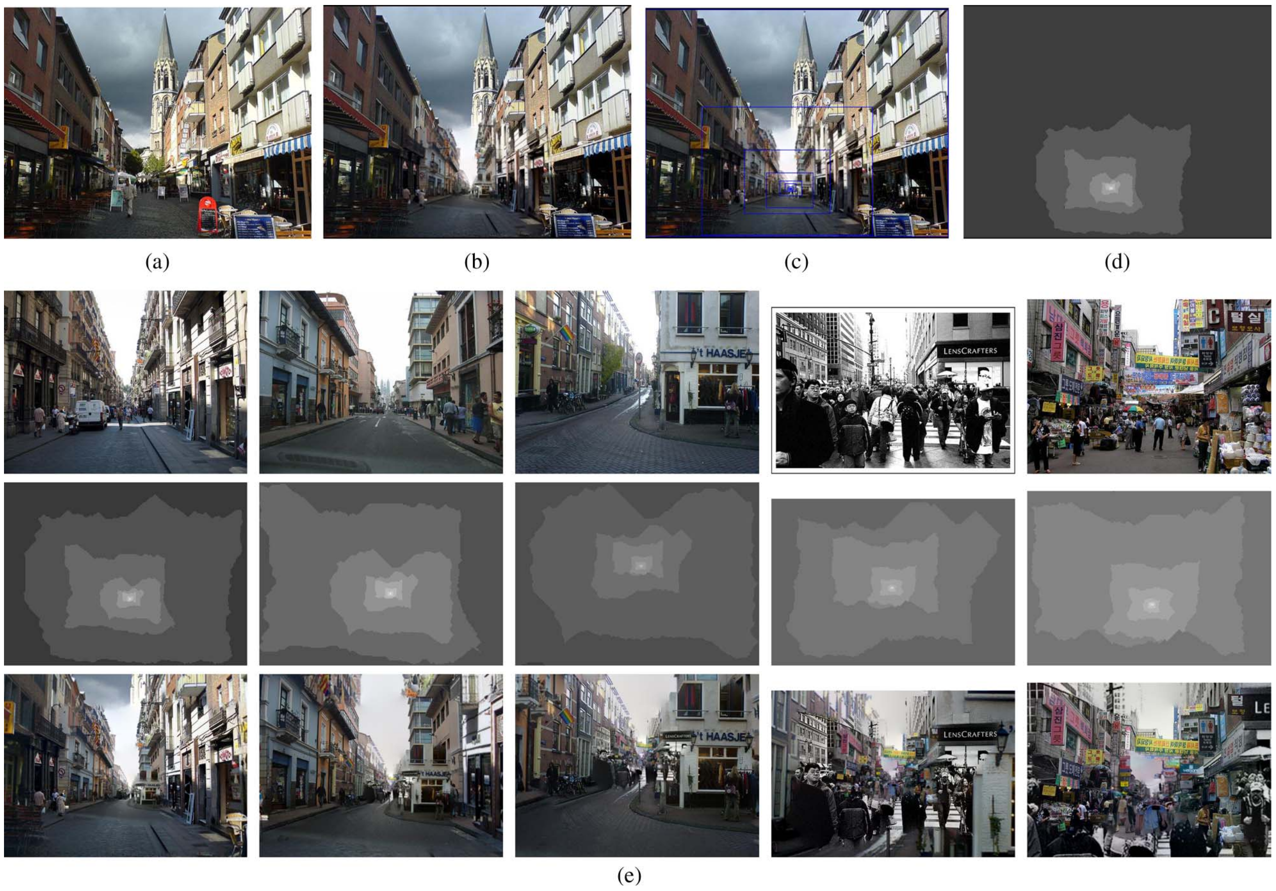


Fig. 16. ***Forward motion sequence of 16 images. (a) The query image. (b) The query image composited with the consecutive images in the sequence. (c) Image boundaries of the following images in the sequence overlaid over the composited image (b). (d) Masks indicating areas in (b) coming from different images. (e) Images 2–6 of the forward motion sequence. Top row: Original images. Middle row: Masks. Bottom row: Composited images. Note that the entire sequence contains perspective views of streets. Each consecutive image was obtained by inducing camera forward motion as illustrated in Fig. 3.***

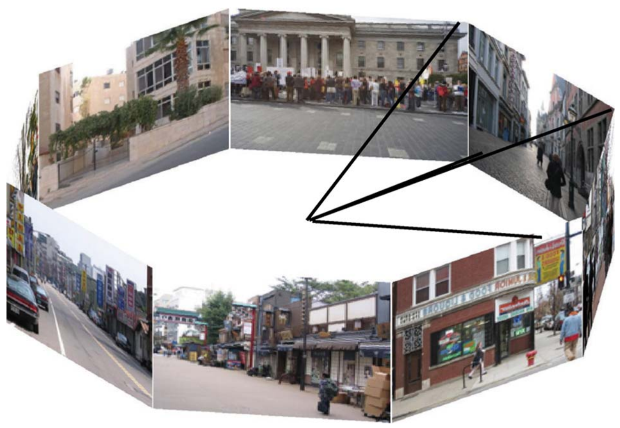


Fig. 17. ***A camera rotation can be used to generate from a single picture a good guess of the surrounding environment not covered by the camera. Here, the system is hallucinating what could be behind the camera (original image marked with a frustum). Note that the back of the camera is also a perspective street, aligned with the camera view.***

seam in their region of overlap and blend them. To ensure real-time transitions, we use the fast pyramid blending instead of the more expensive Poisson blending to construct the blended composite [34] and we apply the blending only on the region of overlap between the two images. The intermediate frames of the motion are synthesized from the composite image. To achieve smooth transitions, we use kernels written in Adobe's Pixel Bender language for hardware-independent description of image processing algorithms [35] to perform the pyramid blending and the synthesis of the intermediate frames. The Pixel Bender kernels can be compiled efficiently for the CPU and/or the GPU and allow significant speed-up of computationally intensive image processing tasks. Currently, Flash Player supports only limited set of features and does not support kernels running on the GPU but if the extra features are added, it will lead to even faster transitions. The computations are memory intensive and are performed on the client machine, therefore they require about 500 MB–1 GB available memory to run smoothly.

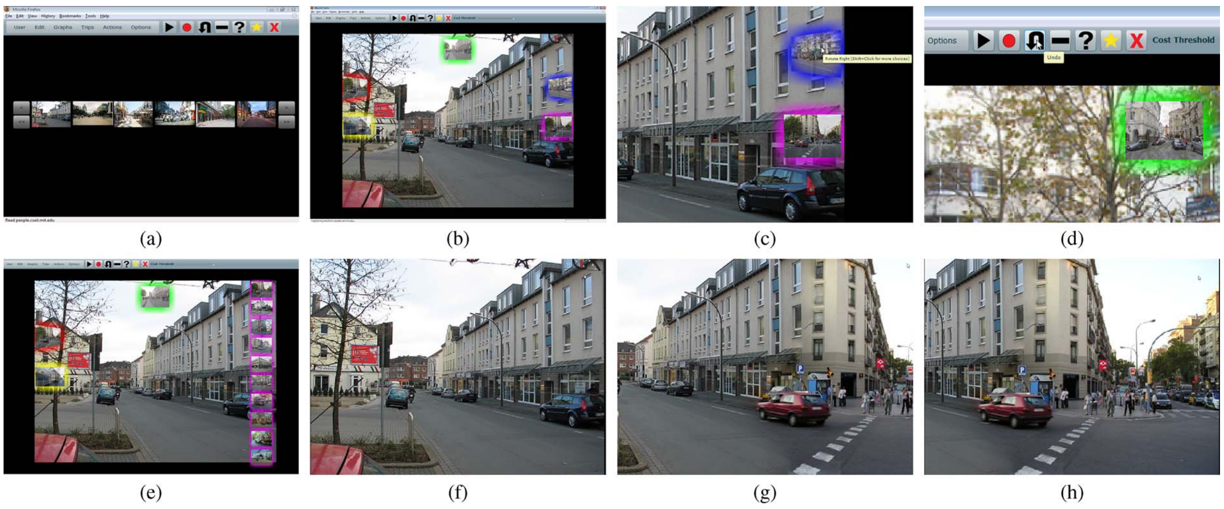


Fig. 18. *A screen shot of the interactive system. (a) Selecting an image to start browsing. (b) The current image is shown in the center and the next image for each motion (forward, rotate left/right, translate left/right) is shown as an inset. (c) Selecting rotate right motion. (d) The user can undo the current transition. (e) The next image for a particular motion (here move right) can be also interactively chosen from the top 10 matching images displayed on the side. (f)–(h) Show the first, middle, and last frame of a rotate right motion sequence.*

After a transition is complete, the image displayed to the user is a blend of two images from the database—the original query and the retrieved image. To produce the next transition, we use the actual retrieved image rather then the displayed blend as the next query. This allows us to take advantage of the pre-computed image graph. Despite the fact that the query image is not identical to the one displayed to the user, we found that it serves as a good enough proxy.

The system also has a noninteractive mode where the next image is chosen automatically by selecting the edge and camera motion type with the highest probability of a good transition. If the user likes a particular sequence, it can be saved as a "trip." The trips can be chained together to create a much larger tour that can also be played automatically. This way, the user can build a library of trips and then mix and match them to achieve a particular tour.

D. Touring Personal Photo Collections

We can also augment the image graph with images from a personal photo collection. The user could then view their photo collection in the interactive Web interface in a manner similar to the image taxi tool. Photos in the personal photo collection can be arranged in any order. Then a new tour is created by finding the shortest path between each pair of images in the collection. The tour can be played automatically in the interactive Web interface. Images from the personal photo collection are highlighted while those from the original image graph serve as transition "filler" images. Augmenting the image graph needs to be done offline before the photo collection can be viewed. Fig. 19 provides an example of a five image personal photo collection from street scenes of Barcelona. The image in the first column shows the interface for placing the images in the personal collection in the desired viewing order. The other columns in Fig. 19 show a path

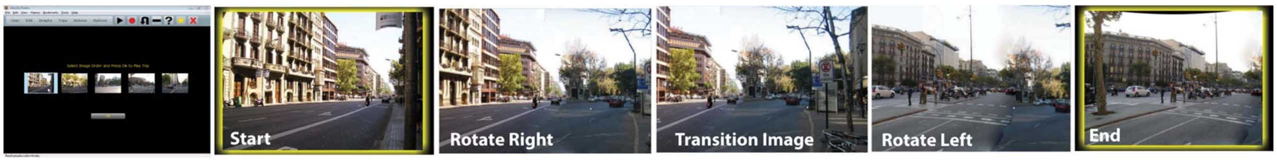


Fig. 19. *Viewing a personal photo collection in the interactive tool. Column 1 shows the interface for placing the images from the personal collection in the desired viewing order. Columns 2–5 show a path between two of the images in the personal collection containing one transition "filler" image. Note the second transition (rotate left) does not bring us back to the starting image since we do not enforce spatial consistency in the graph.*

between two of the images in the personal collection that only contains one transition "filler" image. Notice that the motions used to travel from the start to the end image are rotate right and rotate left. As mentioned in Section III-G, the image graph does not preserve spatial consistency. The rotate left motion does not bring us back to the start image but instead we arrive at a new location.

VI. LIMITATIONS

The system is only as good as the underlying database is. The larger the database, the better the results. We found that there are three typical failure modes. The first is when a semantically wrong image is retrieved. This is mitigated by the use of themes but still sometimes occurs. Second, compositing two distinct images is always a challenge and at times, the seam is quite visible. Finally, there are cases in which the seam runs through important objects in the image which produces noticeable artifacts.

VII. CONCLUSION

The proposed system offers an intuitive 3-D-based navigation approach to browsing large photo collections. Our system arranges the images into semantic themes and performs transformed image retrieval to simulate various camera motions within the large image database. Matches are precomputed offline and the image collection is organized into an image graph, where images correspond to nodes and edges in the graph correspond to transitions between images using different camera motions. We examined properties of the image graph and their effect on tours of the image database. We used the image graph to create synthetic panoramas, interactively tour the virtual 3-D space, or create an image taxi tour from one image to another through the dataset. These tools enable intuitive interaction with large image datasets and may enhance interactive games, movies, photo-sharing Web sites and personal photo collections. ■

REFERENCES

[1] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *SIGGRAPH '96: Proc. 23rd Annu. Conf. Computer Graphics and Interactive Techniques*, New York, 1996, pp. 43–54.

[2] M. Levoy and P. Hanrahan, "Light field rendering," in *SIGGRAPH '96: Proc. 23rd Annu. Conf. Computer Graphics and Interactive Techniques*, New York, 1996, pp. 31–42.

[3] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, *Towards Category-Level Object Recognition*. New York: Springer, 2006, vol. 4170.

[4] P. A. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[5] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[6] N. Vasconcelos, "From pixels to semantic spaces: Advances in content-based image retrieval," *IEEE Computer*, vol. 40, no. 7, pp. 20–26, 2007.

[7] J. Sivic and A. Zisserman, "Efficient visual search for objects in videos," *Proc. IEEE*, vol. 96, no. 4, pp. 548–566, Apr. 2008.

[8] J. Kopf, M. Uyttendaele, O. Deussen, and M. Cohen, "Capturing and viewing gigapixel images," *ACM Trans. Graphics*, vol. 26, no. 3, 2007.

[9] Y. Rubner, C. Tomasi, and L. J. Guibas, "Adaptive color-image embeddings for database navigation," in *Proc. Asian Conf. Computer Vision*, 1998, pp. 104–111.

[10] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake, "Autocollage," in *Proc. ACM SIGGRAPH*, 2006.

[11] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3D," in *Proc. ACM SIGGRAPH*, 2006.

[12] N. Snavely, R. Garg, S. Seitz, and R. Szeliski, "Finding paths through the world's photos," *ACM Trans. Graphics*, vol. 27, no. 3, pp. 11–21, 2008.

[13] J. Sivic, B. Kaneva, A. Torralba, S. Avidan, and W. T. Freeman, "Creating and exploring a large photorealistic virtual space," in *Proc. 1st IEEE Workshop on Internet Vision*, Ancorage, AK, Jun. 2008.

[14] R. Szeliski, "Image alignment and stitching: A tutorial," *Found. Trends Comput. Graph. Comput. Vis.*, 2006.

[15] J. Hays and A. A. Efros, "Scene completion using millions of photographs," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 4:1–4:7, 2007.

[16] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large dataset for non-parametric object and scene recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[17] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[18] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, 1981, pp. 674–679.

[19] P. Perez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

[20] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. ACM SIGGRAPH*, 2001.

[21] A. Agrawal, R. Raskar, and R. Chellappa, "What is the range of surface reconstructions from a gradient field?" in *Proc. Eur. Conf. Computer Vision*, 2006.

[22] J. F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, "Photo clip art," *ACM Trans. Graphics*, vol. 26, no. 3, pp. 3:1–3:10, 2007.

[23] A. Torralba and P. Sinha, "Statistical context priming for object detection," in *Proc. IEEE Int. Conf. Computer Vision*, 2001, pp. 763–770.

[24] J. Coughlan and A. L. Yuille, "Manhattan world: Orientation and outlier detection by bayesian inference," *Neural Comput.*, vol. 15, pp. 1063–1088, 2003.

[25] J. Deutscher, M. Isard, and J. MacCormick, "Automatic camera calibration from a single manhattan image," in *Proc. Eur. Conf. Computer Vision*, 2002, pp. 175–188.

[26] N. Gershenfeld, *The Nature of Mathematical Modeling*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[27] A. Bosch, A. Zisserman, and X. Munoz, "Scene classification using a hybrid generative/discriminative approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 30, no. 4, pp. 712–727, Apr. 2008.

[28] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[29] B. Scholkopf and A. Smola, *Learning With Kernels*. Cambridge, MA: MIT Press, 2002.

[30] P. Erdos and A. Renyi, "Random graphs," *Publ. Math. Inst. Hungarian Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[31] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Nearest neighbors in high-dimensional data: The emergence and influence of hubs," in *Proc. Int. Conf. Machine Learning*, 2009.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[33] [Online]. Available: http://people.csail.mit.edu/biliana/papers/pieee2009/pieee_video.avi

[34] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. COM-31, no. 4, pp. 532–540, 1983.

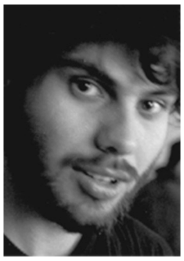
[35] *Adobe Pixel Bender*, 2009. [Online]. Available: http://labs.adobe.com/technologies/pixelbender/

ABOUT THE AUTHORS

Biliana Kaneva (Student Member, IEEE) received the B.A. degree in computer science and mathematics from Smith College in 2000 and the M.S. degree in computer science from the University of Washington in 2005. Currently, she is working towards the Ph.D. degree at the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge.

Prior to joining MIT, she worked as a Software Design Engineer at Microsoft from 2000 to 2006. Her research interests include computer vision, computer graphics, computational photography, and machine learning.

Ms. Kaneva received a Xerox Fellowship in 2009.



Josef Sivic (Member, IEEE) received a degree from the Czech Technical University, Prague, Czech Republic, in 2002 and the Ph.D. degree from the University of Oxford, Oxford, U.K., in 2006. His thesis dealing with efficient visual search of images and videos was awarded the British Machine Vision Association 2007 Sullivan Thesis Prize and was short listed for the British Computer Society 2007 Distinguished Dissertation Award.

After spending six months as a Postdoctoral Researcher at the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, he is currently an INRIA Researcher at the Departement d'Informatique, Ecole Normale Superieure, Paris, France. His research interests include visual search and object recognition applied to large image and video collections.

Antonio Torralba (Member, IEEE) received the degree in telecommunications engineering from Telecom BCN, Spain and the Ph.D. degree in signal, image, and speech processing from the Institut National Polytechnique de Grenoble, France.

He is an Associate Professor of Electrical Engineering and Computer Science at the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology (MIT), Cambridge. He spent postdoctoral training at the Brain and Cognitive Science Department and the Computer Science and Artificial Intelligence Laboratory at MIT.

Shai Avidan (Member, IEEE) received the Ph.D. degree from the Hebrew University, Jerusalem, Israel, in 1999.

He joined Adobe in 2007 as a Senior Researcher and is currently an Assistant Professor at the Department of Electrical Engineering—Systems, Tel-Aviv University, Tel Aviv, Israel. In the past, he was a Researcher at Mitsubishi Electric Research Labs (MERL), a Project Leader at MobilEye, and a Postdoctoral Fellow at Microsoft Research. Between 2002 and 2004, he was a faculty member at the School of Computer Science, Interdisciplinary Center, Herzliya, Israel. His research interests include image editing, tracking, and multicamera systems.

William T. Freeman (Fellow, IEEE) is a Professor of Electrical Engineering and Computer Science at the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, joining the faculty in 2001. His current research interests include machine learning applied to computer vision and graphics, and computational photography. From 1981 to 1987, he worked at the Polaroid Corporation, developing image processing algorithms for electronic cameras and printers. In 1987–1988, he was a Foreign Expert at the Taiyuan University of Technology, China. From 1992 to 2001, he worked at Mitsubishi Electric Research Labs (MERL), Cambridge, MA, most recently as Senior Research Scientist and Associate Director. He holds 25 patents.

Dr. Freeman was an Associate Editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and a member of the IEEE PAMI TCAwards Committee. He is active in the program and organizing committees for Computer Vision and Pattern Recognition (CVPR), the International Conference on Computer Vision (ICCV), Neural Information Processing Systems (NIPS), and SIGGRAPH. He was the Program Co-Chair for ICCV 2005.