Photo Navigator

Chi-Chang Hsieh[1], Wen-Huang Cheng[2], Chia-Hu Chang[2], Yung-Yu Chuang[1], Ja-Ling Wu[2]

[1]Department of Computer Science and Information Engineering
[2]Graduate Institute of Networking and Multimedia
National Taiwan University
Taipei 10617, Taiwan, R.O.C.
{nonrat, wisley, chchang, cyy, wjl}@cmlab.csie.ntu.edu.tw
project website: http://www.cmlab.csie.ntu.edu.tw/navigator

ABSTRACT

Nowadays, travel has become a popular activity for people to relax their body and mind. Taking photos is then often an inevitable and frequent event during one's trip for recording the enjoyable experience. To help people to relive the wonderful travel experience they had recorded in photos, this paper presents a system, *Photo Navigator*, for enhancing the photo browsing experience by creating a new browsing style with a realistic feel to users as being into the scenes and taking a trip back in time to revisit the place. The proposed system is characterized by two main features. First, it better reveals the spatial relations among photos and offers a strong sense of space by taking users to fly into the scenes. Second, it is fully automatic and makes plausible for novice users to utilize the 3D technologies that are traditionally complex to manipulate. The proposed system is compared with two other photo browsing tools, ACDSee's photo slideshow and Microsoft's PhotoStory. User studies show that people would comparatively favor the browsing style we offer and appreciate the ease to create such a style.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]: Animations; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*representations*.

General Terms

Algorithms, Design, Experimentation.

Keywords

Photo browsing, Fly-through, Sense of space, Automation, Image-based modeling.

1. INTRODUCTION

After days of hard work, travel has become a popular activity for people to relax their body and mind. During their trips, people like to take pictures for anything that they find interested, such as a splendid view from the Great Wall, a busy street scene of the New York city, or the quiet scenery in an European village. It is common that hundreds or thousands of photographs are taken for a single trip alone. Although taking a trip is fun and memorable, the process required to organize the large number of travel photos (e.g. tagging metadata) is often tedious and painful. Sometimes, it is cumbersome even for the simple browsing task. Photo slideshows or thumbnails generated using available tools, such as ACDSee [1] and Picasa [3], are popular means for people to review their trips. However, their plain styles of display tend to be dull. It is possible to create more vivid and eye-catching presentations. However, it is less accessible to most users because it requires not only much time and labor, but also skills, creativity and art sense. As a result, the creation of vivid presentations for travel photos is largely privileges of professionals or expert users.

In this paper, we present *Photo Navigator*, a system for enhancing the photo browsing experience by creating a realistic feel to users as being personally into the scenes and taking a trip back in time to revisit the place. By human nature, people would like to relive the wonderful experiences they had during their travels. To go beyond the traditional browsing paradigms, we believe that people would enjoy the experience of virtually walking through their photos in a realistic sense of returning to the scenes. Our system produces such an experience by creating an approximate 3D model for each photo. These models are then connected to generate a sequence of views according to a well-routed browsing path. Without the need for user intervention, the procedure is fully automatic and only requires users to specify a set of photos as the input. A key feature of our system is its capability for better revealing the spatial relations between photos. For example, six pictures in Figure 1 were taken at a memorial. It is hard to tell their spatial relations at first glance by simply browsing the photos. For those who have never been there, it is even doubtful if these photos were taken at the same scene. Our system, by contrast, offers a strong sense of space by taking users to fly into and through photographs taken at the same scene.

For our applications, two neighboring photos (i.e. the two are taken in physical proximity but not necessarily closely in time) are assumed to share certain sight in common. For example, a landmark (partial or whole) is photographed from different positions or angles. It seems a strong assumption but actually quite acceptable in practice. Travelers often walk leisurely on a pre-scheduled traveling route and the

MM'08, October 26–31, 2008, Vancouver, British Columbia, Canada.

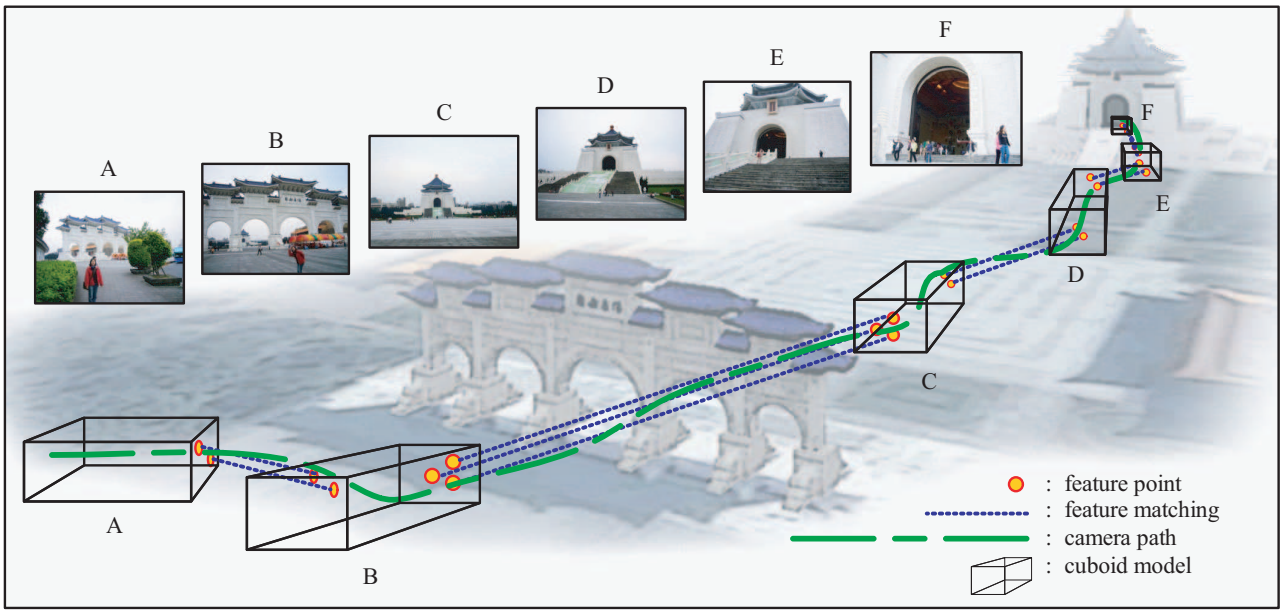

Figure 1: The basic idea of the Photo Navigator system. Given a set of unordered photographs (A–F), the system automatically arranges these photos by their spatial relationship. For each photograph, a cuboid model and a couple of camera poses are estimated automatically. With these, the system generates a "fly-through" path for each photograph and smooth transitions between photographs. Thus, browsing photos with the slideshow generated by the system offers a more enjoyable "being there" experience.

photos they took naturally tend to exhibit high spatial correlations. On the other hand, once users are aware of the potential to generate interesting fly-through slideshows, they may take pictures this way so that they can later relive their wonderful trip experiences. An analogy is taking panoramas. Before panorama stitching tools exist, people won't take pictures in a panoramic way. However, once they are aware of such tools, it becomes popular to take pictures this way if they think it is worth it. In addition to the above motivation, the main ideas and general design considerations of our system are also motivated by the following observations:

- Travel photos are personal data and not necessarily relating to famous tourist spots. The pleasure of travel, for example, sometimes comes from the discovery of special places where few other travelers reached, such as a scenic trail in deep mountains. Thus, it is hard to find relevant photos of such a scene from external sources, e.g. travel reviews on personal blogs or media posted on social networking websites [13]. Therefore, to create a personal photo browsing experience, it is better to enhance the visual presentation with the input photographs themselves rather than relying on other external relevant information or other community-based multimedia services.

- Travel photos are rich in content and there are high variations in the number of photos taken at different physical locations. This may pose difficulty in the effective realization of advanced photo applications. For example, modeling a travel site with a full 3D representation is favorable since it offers the viewer a more realistic sense of space [16]. However, if the available photos for the site are in a small quantity, it is often not affordable to obtain accurate geometry for the site. Furthermore, even if the number of photos is large enough, the recovery is still often ill-posed since the travelers tend to take pictures of the same site from various angles and distances. Thus, a practical system should only attempt to approximate such visual experience instead of the pursuit of reconstructing accurate models.

- People would prefer browsing photos in 3D, even if the constructed 3D models are not precise. An example is the active research on the *Tour into the Picture* [10, 9]. However, previous work is mostly confined to a single image and its extensions to a photo set need more investigation. For example, making visually seamless transitions between individual photos remains a technical issue [17]. In addition, enabling 3D navigating effect is difficult because it lacks friendly user interfaces and could be complex when dealing with multiple photos. Automation of the process would make the creation of such effects more feasible for practical use.

- It may not be less convenient to browse travel photos in an interactive navigation style (e.g. *Photo Tourism* [17]) that allows users to freely choose every next step to move in a 3D scene. For example, people are used to share their pleasure of travel by showing the photos to friends. If the viewers have never been to the same places, they would soon feel bored for not knowing where are worthy of seeing and just aimlessly walking around in the 3D scenes. In such a case, they would rather passively watch a dynamic photo presentation than actively participate in it.

The main contributions of our work are twofold. First, a practical system is proposed and realized for enhancing the enjoyment of browsing travel photos. It provides a new type of personalized applications that make visually seamless 3D presentations for user's own photos, in support of the user's desire to relive wonderful travels by creating photo browsing experiences of really walking around the scenes. Second, our system effectively automates a series of complex interactive tasks (e.g. 3D modeling of a scene) into a simple step (i.e. specifying the input photos). It greatly reduces the user's burden. We believe such automation would facilitate the widespread use of the cool 3D technologies that are not easily accessible to naïve users.

Two systems are related to our system, Photo2Video [11, 12] (and Microsoft Photo Story) and Photo Tourism [17] (and Microsoft's Photosynth). The former generates a set of 2D panning and zooming camera motions within a photo to create slideshows. The later analyzes spatial relations among photos to allow users to interactively switch between spatially related photos. Compared to Photo2Video, Photo Navigator's 3D walk-through style reveals the sense of space for the scene better than plain 2D camera motions. In addition, Photo Navigator exposes spatial relations among photos by navigating through them while Photo2Video does not spatially relate photos at all. Compared to Photo Tourism, Photo Navigator works with much fewer photos. Photo Tourism requires a dense set of photos for a site and thus often needs to seek for them from public photo collections. As pointed out previously, travel photo browsing is a personal application and it is preferred to only use personal photographs. In addition, Photo Tourism does not pay much attention to navigation within a photo. Its 3D illusion is synthesized merely by switching between densely related photos. On the other hand, to expose sense of space with much fewer photos, we build approximate 3D models for navigation within single photos.

The rest of this paper is organized as follows. Section 2 describes an overview to the system, mainly consisting of 3D scene model reconstruction and slideshow generation. Details of the 3D scene model reconstruction are described in Section 3, and the process of generating slideshow is described in Section 4. Section 5 presents the experimental results and Section 6 shows evaluations and comparisons with other tools. Finally, Section 7 concludes the paper with conclusions and future work.

2. SYSTEM OVERVIEW

Figure 1 sketches the basic idea for the proposed Photo Navigator system. In this example, a user took six photographs (A–F) at different locations of Chiang Kai-shek Memorial Hall in Taipei. Note that these photographs are not necessarily taken temporally in that order. Photo Navigator automatically analyzes spatial relationships between photographs by feature matching (orange points and blue dotted liens in Figure 1) and suggests a proper navigation order (from A to F in this example) through this set of photographs so that the navigation exhibits a more natural camera motion. Once the path is determined, a cuboid model is constructed automatically for each photograph. A cuboid is a rectangular prism consisting of five axis-aligned rectangular faces. The photograph is then partitioned into five areas, one for each rectangular face. Each area of photograph is texture-mapped onto the corresponding face. With

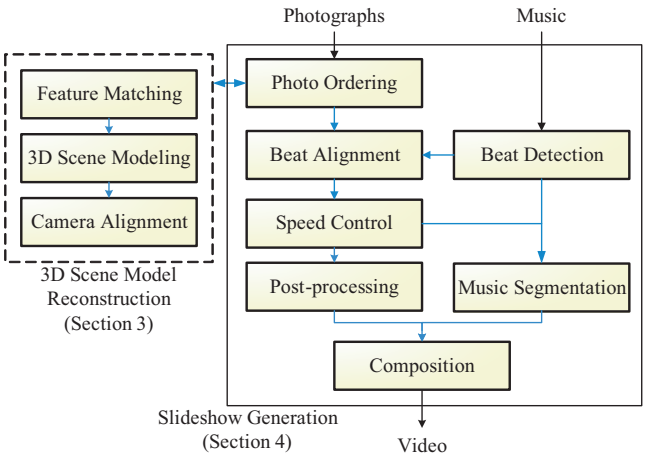

Figure 2: System flowchart of the proposed Photo Navigator system.

the cuboid model, by setting a virtual camera, we can take a picture of this textured mapped cuboid to obtain an image approximating viewing the scene from the virtual view. Here, a virtual camera is specified by its rotation and translation. After constructing the cuboid models, Photo Navigator automatically determines a smooth camera path (the green path in Figure 1) to let us navigate from photograph A to photograph B and so forth until we reach the final photograph F. The system then takes a series of pictures with virtual camera settings along the determined path to synthesize a navigation video for the input photo set. Music is accompanied to complete the slideshow. Note that the system is essentially automatic. However, optionally, users can manually cut out foreground objects, such as the girl in Photo A of Figure 1, to even further improve the slideshow.

Technically, as illustrated in Figure 2, the proposed Photo Navigator system consists of a set of processes for 3D scene model reconstruction and slideshow generation. The core technical module of Photo Navigator is its 3D scene model reconstruction procedure which consists of three main components: feature matching, 3D scene modeling, and camera alignment. Given two photographs, feature matching is first used to identify the spatial relationship between them. A cuboid model is automatically created by finding the four boundaries of the back wall and a vanishing point. Then, camera alignment estimates the initial and final camera poses for the best photograph transition.

Given a collection of unordered photographs, the photo ordering process automatically selects an ordered sequence of photographs and generates a camera path for them. In terms of music content, beat detection is performed to characterize beats. Switches between photographs are aligned with the nearest beat in the beat alignment process. Instead of using a constant touring speed, speed control is performed to enhance the virtual walk-through of the photographs. According to the varying speed, frames of the slideshow are then rendered along the determined path. Furthermore, post-processing is applied to the rendered frames for better visual quality. Finally, a video consisting of these rendered frames and the segmented music is generated as the slideshow for the input photographs.

3. 3D SCENE MODEL CONSTRUCTION

This section describes the core algorithm of Photo Navigator, 3D scene model construction. Given a *source image* I and a *destination image* J, an automatic procedure is proposed to obtain the following:

- a *cuboid model* $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ for the source image I,
- *initial extrinsic parameters* (rotation and translation) $\left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]$ for the virtual camera to match I,
- *final extrinsic camera parameters* $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ so that rendering of $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ matches the destination image J.

More specifically, if the source image I is texture-mapped onto the cuboid model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$, the image of the textured model taken by a virtual camera with its location and orientation specified by $\left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]$ should resemble I and the image of $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ taken with $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ should resemble J, i.e.

$$\begin{aligned} \mathrm{I} &\approx \texttt{Render}(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]) \\ \mathrm{J} &\approx \texttt{Render}(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, [\mathbf{R}_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}_{\mathrm{I}\to\mathrm{J}}]), \end{aligned} \tag{1}$$

where $\texttt{Render}(\mathbf{M}, [\mathbf{R}|\mathbf{T}])$ is a function which returns the rendering of a texture-mapped cuboid $\mathbf{M}$ for a given camera orientation $\mathbf{R}$ and location $\mathbf{T}$. Figure 3 illustrates the flowchart for the proposed procedure which consists of three main components: feature matching (Section 3.1), 3D scene modeling (Section 3.2) and camera alignment (Section 3.3).

3.1 Feature matching

In order to acquire the spatial relationship between two photographs, we first match features extracted from them. The popular SIFT (Scale Invariant Feature Transform) [14] feature is used to detect and describe local features. SIFT is used because it is famous for its invariance to scale and rotation, and its robustness to occlusion, illumination variations, noise and minor variances in viewpoints. However, even with robust SIFT features, for our application, it is still possible to have a fair number of false feature matches. This is because the photos we used were often taken from very different viewpoints. To reduce the adverse effect of falsely matched features, we estimate the fundamental matrix $\mathbf{F}$ between I and J. The fundamental matrix imposes the constraint that correctly matched features $\mathbf{x}$ and $\mathbf{x}'$ of two images should satisfy the relation $\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$. We can use this constraint to remove false feature matches. We use the normalized 8-point algorithm and RANSAC to compute $\mathbf{F}$ [8] and decide inliers as correctly matched features denoted as $(\mathbf{x}^{\mathrm{I}}_i, \mathbf{x}^{\mathrm{J}}_i)$. The whole procedure for feature matching is described in Algorithm 1. On the top of Figure 3, the cyan lines represent all matches without imposing the fundamental matrix constraint. The blue lines shows correct feature matches using Algorithm 1.

3.2 3D scene modeling

The next step is building a 3D scene model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ for the source image I. Our approach is based on the "tour into picture" paper [10, 4] in which a cuboid is used as the scene model. Figure 4 shows the configuration of our cuboid model consisting of five rectangular faces: floor, ceiling, rear wall, and two side walls. Figure 5 shows the partition of the source image corresponding to the cuboid model. In the original tour into picture paper [10], users need to manually specify

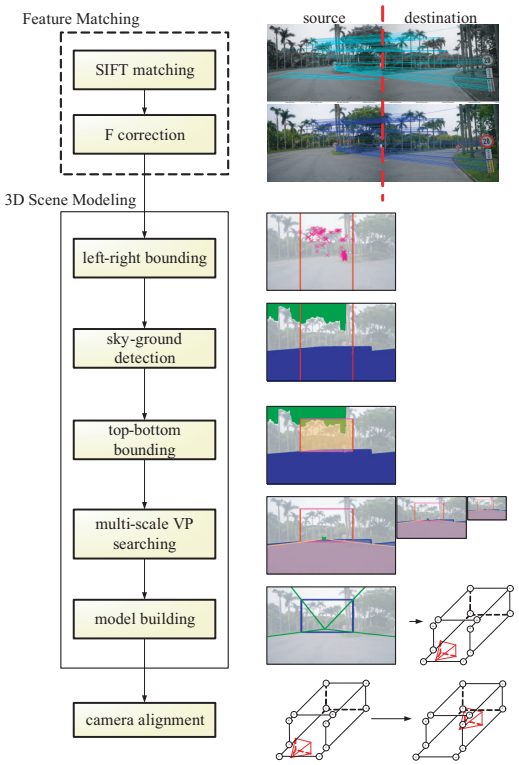

Figure 3: 3D scene model reconstruction.

Algorithm 1 $(\mathbf{x}^{\mathrm{I}}_i, \mathbf{x}^{\mathrm{J}}_i)$=**FeatureMatching**(I, J). Given two images I and J, find a set of correct feature matches $(\mathbf{x}^{\mathrm{I}}_i, \mathbf{x}^{\mathrm{J}}_i)$.

```
detect and match SIFT features of I and J
repeat
    randomly select 8 matched pairs
    estimate F' using the selected pairs
    count the number of inliers
until it has been performed N times
select the configuration with most inliers
compute F using all inliers
return all (x_i^I, x_i^J) satisfying x_i^I^T F x_i^J = 0
```

five points for building the model. These points are the four corners of the rear wall (vertices 1, 2, 7, 8) and the vanishing point $\mathbf{V}$ of the scene, as shown in Figure 5. Assuming the focal length is set to 1, once those 2D points are specified, the corresponding 3D coordinates of the twelve points (vertices $1 \sim 12$ in Figure 4) can be directly determined [4]. Thus, once those five points are specified, one can uniquely determine the cuboid model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$. In addition, since the focal length is assumed to be 1, the projection matrix Π and the extrinsic camera parameters $\left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]$ can also be determined so that the projection of $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ matches I. In order to make this process automatic, we devise methods to determine those five points from image content.

To determine those five points, we need to determine the vanishing point and the positions for left, right, top, and bottom boundaries of the rear wall. We first find the leftmost and the rightmost features among $\mathbf{x}^{\mathrm{I}}_i$ and use them respectively as the left and right boundaries of the rear wall. To

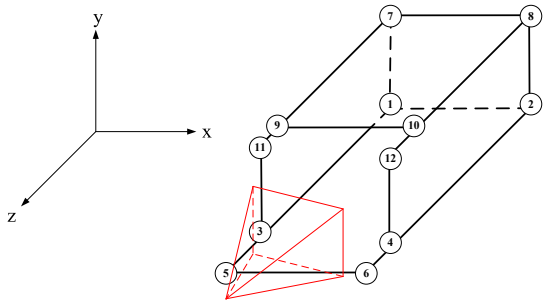

Figure 4: The 3D cuboid model. It is specified by five rectangular faces and twelve vertices. The red pyramid represents a virtual camera.

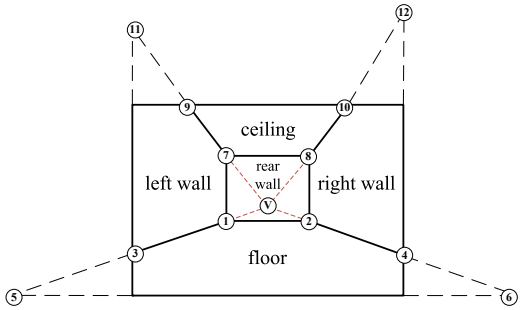

Figure 5: The partition of an image for a cuboid model. An image is partitioned into five areas, corresponding to the five faces of a cuboid model, rear wall, right wall, left wall, ceiling and floor.

prevent constructing a bizarre model, we require that the width of the rear wall must be larger than a quarter of the image width. If the estimated rear wall width is too small, we use the mean position of all matched feature points as the middle of the rear wall and set the left and right boundaries of the rear wall so that its width equals the minimal width. Magenta points in Figure 3 are feature points and the two vertical red lines represent the left and right boundaries.

Next, we determine the top and bottom of the rear wall by detecting sky and ground portion of the image. We use the method described in Hoiem *et al.*'s paper [9] to detect sky area and ground area of the image I. The image is first segmented by colors to obtain superpixels [7]. Superpixels are then classified into "sky", "vertical", and "ground," using pre-trained classifiers based on color, texture, location and geometry features. In Figure 3, the green area is the detected sky area and the blue one is the detected ground area. We set the top of the rear wall as the top of the non-sky area within left and right boundaries of the rear wall. The reason is to prevent distortion caused by the ceiling. Similarly, we assign the bottom of rear wall as the bottom of the non-ground area within left and right boundaries. However, when the road is a trapezoid or inaccurate due to people in the image, the bottom might not be a good boundary. Thus, when the distance between top and bottom of the detected ground area is larger than 10% of the image height, we instead set the top of detected ground area as the bottom of the rear wall.

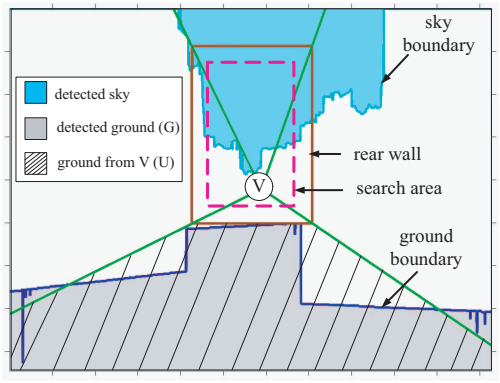

Figure 6: Determination of the best vanishing point.

After determining four boundaries for the rear wall, the next step is to find the vanishing point within the rear wall. A less-carefully-placed vanishing point might cause terrible distortion when walking through the model. Thus, we have to define criterion for selecting the vanishing point to avoid such distortion. For an assumed vanishing point $\mathbf{V}$, we can form two lines by connecting $\mathbf{V}$ to points 1 and 2 respectively (Figure 5). We define $\mathbf{U}$ as the set of pixels under these two lines and $\mathbf{G}$ as the set of pixels which belong to the detected ground (Figure 6). We then calculate the area α of the region $\mathbf{U} \cap \mathbf{G} - \mathbf{U} \cap \overline{\mathbf{G}}$. We prefer a larger α since it means a better match to the detected ground area. Notice that $\mathbf{U} \cap \overline{\mathbf{G}}$ is also considered because treating non-ground area as ground causes terrible distortion during rendering. Thus, we want to minimize this kind of misclassification as well. On the other hand, misclassification of the area $\overline{\mathbf{U}} \cap \mathbf{G}$ often won't cause serious distortion and it is thus ignored. In addition, to avoid a bizarre vanishing point, we restrict the search within a search area whose center is at the center of the rear wall and whose size is 75% of the rear window, as denoted as the magenta dashed line in Figure 6.

To avoid exhaustive search for the vanishing point within the search area, we use a multi-scale approach to find the position with the maximal α. Multi-scale techniques are well known in the computer vision and image processing communities. This technique starts by computing an image pyramid. At the smallest level of the pyramid, we find the best value by exhaustive search in the search area. The optimal position is then projected to its parent level as the initial point. A local search for the optimal value is performed only within a small neighborhood around the projected point. The process is repeated until returning to the base level of the pyramid. In this way, the vanishing point can be efficiently and accurately determined. Algorithm 2 sketches the pseudo code for our multi-scale vanishing point search algorithm.

After automatically determining the 2D positions of the rear wall and the vanishing point, a cuboid model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ and the initial camera pose $\left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]$ can be estimated using the method proposed by Cao *et al.* [4]. Since the focal length is assumed to be 1, we can also obtain the corresponding camera projection matrix Π. In addition to those, we also estimate the 3D coordinates $\mathbf{X}^{\mathrm{I}}_i$ for each 2D feature $\mathbf{x}^{\mathrm{I}}_i$. Since the projection matrix Π and the initial camera pose $\left[\mathbf{R}^0_{\mathrm{I}\to\mathrm{J}}|\mathbf{T}^0_{\mathrm{I}\to\mathrm{J}}\right]$ are given, we know that $\mathbf{X}^{\mathrm{I}}_i$ must be along the

Algorithm 2 (x,y)=**VPSearch**$(\mathbf{G},\Omega)$. Given a ground mask $\mathbf{G}$ and the coordinates of the rear wall Ω, determine the 2D position (x,y) of the vanishing point $\mathbf{V}$.

if image is not small enough **then**
  (x',y') = `VPSearch`$(\mathbf{G}/2, \Omega/2)$
  $(x,y) = (2x', 2y')$
  for $(\Delta x, \Delta y) \in \{-1,0,1\} \times \{-1,0,1\}$ **do**
    determine $\mathbf{U}$ assuming $(x+\Delta x, y+\Delta y)$ is the vanishing point
    $\alpha(x+\Delta x, y+\Delta y) = \left|\mathbf{U} \cap \mathbf{G} - \mathbf{U} \cap \overline{\mathbf{G}}\right|$
  end for
  return $(x+\Delta x, y+\Delta y)$ with maximum α
else
  set search area $\Psi = 0.75 * \Omega$
  for all (x,y) within Ψ **do**
    determine $\mathbf{U}$ assuming (x,y) is the vanishing point
    $\alpha(x,y) = \left|\mathbf{U} \cap \mathbf{G} - \mathbf{U} \cap \overline{\mathbf{G}}\right|$
  end for
  return (x,y) with maximum α
end if

line defined as follows:

$$\mathbf{X}_i^{\mathrm{I}} \sim \left(\Pi \left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]\right)^{-1} \mathbf{x}_i^{\mathrm{I}}.$$

Additionally, we assume that features are placed on the rear wall in 3D. Thus, $\mathbf{X}_i^{\mathrm{I}}$ can be uniquely determined by intersecting the above line with the rear wall. The procedure of Section 3.2 is denoted as: $(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \Pi, \left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right], \mathbf{X}_i^{\mathrm{I}}) =$ `SceneModeling`$(\mathrm{I}, \mathbf{x}_i^{\mathrm{I}})$. Given the source image I and matched features' positions $\mathbf{x}_i^{\mathrm{I}}$, the procedure `SceneModeling` returns a cuboid model $(\mathrm{I}, \mathbf{x}_i^{\mathrm{I}})$, the camera project matrix Π and the initial extrinsic camera parameters $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$.

3.3 Camera alignment

After the cuboid model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ is created and the initial camera pose $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$ is found, the next step is to decide the final camera pose $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ so that we can switch from browsing I to browsing J as seamlessly as possible. Thus, we would like to find the camera pose so that the rendering of $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ looks as similar to J as possible. To speed up the estimation, instead of using all pixels, we only measure the discrepancy between matched features. That is, we find the final pose $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ by the following optimization,

$$[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}] = \arg\min_{\mathbf{R},\mathbf{T}} \sum_i \left(\Pi \left[\mathbf{R}|\mathbf{T}\right] \mathbf{X}_i^{\mathrm{I}} - \mathbf{x}_i^{\mathrm{J}}\right)^2, \qquad (2)$$

where $\Pi \left[\mathbf{R}|\mathbf{T}\right] \mathbf{X}_i^{\mathrm{I}}$ is the projected 2D position of $\mathbf{X}_i^{\mathrm{I}}$ under a specific camera pose $[\mathbf{R}|\mathbf{T}]$. Here, we want the projected position to be close to its corresponding feature position $\mathbf{x}_i^{\mathrm{J}}$ in the target image.

Equation 2 is a nonlinear function to the optimizing parameters, $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$, where t_x, t_y and t_z are translation offsets along x, y and z axes, and θ_x, θ_y and θ_z are rotation angles for each axis. Thus, the above optimization problem becomes a nonlinear least square fitting problem. Levenberg-Marquardt algorithm and Gauss-Newton algorithm [15] are used to find the optimal camera parameters. We run both algorithms simultaneously and select the answer with the smaller error. The initial guess for both algorithms is $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$. To make the camera path more similar to a walk-through through the scene, two camera parameters (θ_z and t_y) are set fixed as their values in $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$. Thus, only four parameters are estimated. The above optimization process for finding the final camera parameters is denoted as $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}] =$ `CameraAlign`$(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \mathbf{X}_i^{\mathrm{I}}, \mathbf{x}_i^{\mathrm{J}}, \Pi)$. At the bottom of Figure 3, the left and right red pyramids represent the virtual cameras corresponding to $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$ and $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ respectively.

This section describes the core procedure of our system. Given two images I and J, the procedure automatically creates a cuboid model and estimates the initial and final camera poses. The overall procedure of this section can be summarized as Algorithm 3.

Algorithm 3 $(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right], [\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}])$=**Model**$(\mathrm{I}, \mathrm{J})$. Given two images I and J, find a cuboid model $\mathbf{M}_{\mathrm{I}\to\mathrm{J}}$ for I, the initial camera pose $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$ and the final camera pose $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$.

$(\mathbf{x}_i^{\mathrm{I}}, \mathbf{x}_i^{\mathrm{J}})$ = `FeatureMatching`(I, J)
$(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \Pi, \left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right], \mathbf{X}_i^{\mathrm{I}})$ = `SceneModeling`$(\mathrm{I}, \mathbf{x}_i^{\mathrm{I}})$
$[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ = `CameraAlign`$(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \mathbf{X}_i^{\mathrm{I}}, \mathbf{x}_i^{\mathrm{J}}, \Pi)$
return $(\mathbf{M}_{\mathrm{I}\to\mathrm{J}}, \left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right], [\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}])$

4. SLIDESHOW GENERATION

This section describes the procedure to automatically generate a camera path for travel photo navigation. Using the procedure described in Section 3, our system decides which photographs can be stitched together for navigation and determines a best camera path going through them (Section 4.1). Section 4.2 describes how to control speed and align with music. Section 4.3 describes the post-processing for improving visual quality.

4.1 Photo ordering

Given a collection of photographs, our system automatically determines which photographs should be stitched into a trace and generates a camera path for them. For a collection of unordered photographs, we first discover their spatial relationships by computing pairwise feature matching. To speed up the feature matching process, we use down-sampled images. Let $S(\mathrm{I}, \mathrm{J})$ record the number of matched features between I and J. Algorithm 4 summarizes our procedure to find the best image to follow an image I from the input photograph collection $\mathbf{C}$. We first drop images which have less than 15 matched features with I. For each of the remaining images J, we use the procedure of Section 3 to find the initial camera pose $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$ and the camera motion $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$ which brings us from I to J. Let $t_x^0, t_z^0, \theta_x^0, \theta_y^0$ be x-offset, z-offset, x-rotation, y-rotation of $\left[\mathbf{R}_{\mathrm{I}\to\mathrm{J}}^0 | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}^0\right]$ and $t_x, t_z, \theta_x, \theta_y$ be the ones of $[\mathbf{R}_{\mathrm{I}\to\mathrm{J}} | \mathbf{T}_{\mathrm{I}\to\mathrm{J}}]$. We then calculate the priority of each transition from I to J by evaluating the corresponding camera motion type.

We often prefer the "walking-through" camera motion. Thus, our system assigns the highest weight w_{tz} to the translation along z to encourage a camera motion spending more time on "moving forward". The next preferred camera motions are panning and panoramic motion, corresponding to translation along x-axis and rotation with respect to y-axis. We prefer panning more than the panoramic motion because it is often more challenging to transit from one image to another using a panoramic camera motion. Hence,

we assign the second highest weight w_{tx} to panning motion and a smaller weight w_{ry} to panoramic motion. The final remaining motion is rotation with respect to x-axis, corresponding camera motion of looking up and down. Since this is a rare motion, we assign the lowest weight w_{rx} to it. To sum up, we assign different weights to four kinds of camera motions, *walking-through* (w_{tz}), *panning* (w_{tx}), *panoramic motion* (w_{ry}) and *looking-up-and-down* (w_{rx}) in the order of preference, $w_{tz} > w_{tx} > w_{ry} > w_{rx}$ to encourage camera motion spending more time on our preferred motion components. Assume that the preferred velocity for the four camera parameters $t_x, t_z, \theta_x, \theta_z$ are $v_x, v_z, \omega_x, \omega_z$ respectively. The portion of each motion type is estimated by its corresponding duration, $\frac{|t_z - t_z^0|}{v_z}, \frac{|t_x - t_x^0|}{v_x}, \frac{|\theta_y - \theta_y^0|}{\omega_y}, \frac{|\theta_x - \theta_x^0|}{\omega_x}$ respectively. Their weighted sum is then used as the priority of the transition from I to J. Such a priority function will encourage camera motion in the order of walking-through, panning, panoramic motion and looking-up-and-down.

With the procedure of Algorithm 4, one usage scenario is to let the user specify a photograph as the starting point. Our system then automatically selects an ordered sequence of photographs starting from that photograph and generates the camera path. Given a collection of unordered photographs, our system can also automatically find all possible series of photographs and generate a camera path for each of them by building a directed graph and finding all connected components. Note that the photo ordering algorithm is used instead of just using time tags for the following reasons. First, spatially close photos were not necessarily taken closely in time. Second, users may take many photos which are close in space. Our ordering algorithm can pick up the best ones to stitch together. Finally, the ordering algorithm can deal with photos from other travel partners' cameras.

Algorithm 4 J_{best}=**FindNext**(I, **C**). Given a photograph I and a photograph collection **C**, find the best image J_{best} in **C** to follow I.

```
max = 0
for each J ∈ C do
  if S(I, J) ≥ 15 then
    (M_{I→J}, [R^0_{I→J} | T^0_{I→J}], [R_{I→J} | T_{I→J}]) = Model(I, J)
    ρ = w_tz |t_z − t^0_z|/v_z + w_tx |t_x − t^0_x|/v_x + w_ry |θ_y − θ^0_y|/ω_y + w_rx |θ_x − θ^0_x|/ω_x
    if ρ > max then
      max = ρ; J_best = J;
    end if
  end if
end for
return J_best
```

4.2 Music beat alignment and speed control

Once we have determined a camera trace for a series of photographs, our system can render a series of images to fly through this set of photographs according to a specific walking speed. Instead of walking in a constant speed, varying speed gives people better perception as "tour into the picture." Our system increases speed in first three quarters of time interval and decreases speed in the last quarter of time interval. This speed change applies to all camera parameters. Figure 7 shows examples for speed control. Based on the speed control function, our system interpolates camera poses for each time instance accordingly and then generates

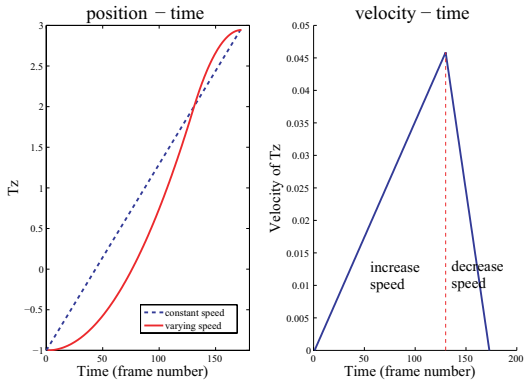

Figure 7: Examples of speed control functions.

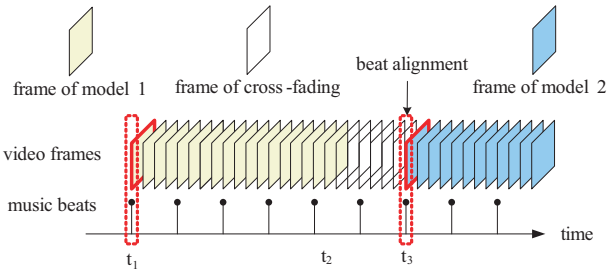

Figure 8: Beat alignment and cross fading.

the corresponding frame by rendering the current cuboid model with the interpolated camera pose.

When switching from the source image to the next image, a cross-fading effect is applied for a smoother transition. To even improve the browsing experience, the rendered video is accompanied with a music as proposed in tiling slideshow [5]. To harmonize the audio-visual effect, touring speed is fine-adjusted to be aligned with the nearest beat of the music. Music beat is detected using the method by Dixon [6]. Figure 8 demonstrates such an alignment.

4.3 Post-processing

Our system uses a post-processing to further improve the visual quality. The post-processing includes three parts: cross-fading, cross-blending, and unknown area filling as shown in Figure 9. Unknown area filling happens when the rendering of the model under the current pose does not cover the whole image. When this happens, we estimate a suitable translation for the next image to fill in the unknown area. To hide the boundary of the rendered area and the filled area, cross-blending is performed along their boundary. The width of cross blending is set to 100 pixels. Cross-fading happens at the transition time from the source image to the next image, from t_2 to t_3, as shown in Figure 8. During the transition, the rendered view is cross-faded to the next image gradually. In our current implementation, the cross-fading period is set to one second long.

5. EXPERIMENTAL RESULTS

Five sets of unordered photos taken by different amateurs were used for testing and evaluating the proposed Photo

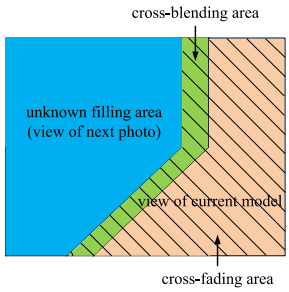

Figure 9: Area of each post-processing.

input	#photos	category	video length
set A (Figure 10)	4	landscape	29s
set B (Figure 11)	3	landscape	17s
set C (Figure 12)	16	landmark	118s
set D (Figure 13)	8	night view	57s
set E (Figure 14)	20	campus	165s

Table 1: Information of the tested photo sets and produced video clips.

Navigator. Detailed information for the photo sets and produced video clips are listed in Table 1. All results are available at `http://www.cmlab.csie.ntu.edu.tw/navigator`.

The input photo set A contains photographs taken when climbing a mountain. Photo set B was taken in a forest. Both belong to the category of natural landscape. The numbers of photos in set A and set B are much less than other types. Generally, the scenes in personal travel photographs cannot be found easily elsewhere. Therefore, photo-tourism-like systems are not suitable to be applied to this kind of personal travel photograph sets.

Figure 10 shows the result that Photo Navigator produced for photo set A. Figure 10(a) shows a frame of the animation of camera shuttering with audio effects, used to represent the photo capturing moment. Figures 10(b,d,g,i) show the input photographs within photo frames. These effects bring enjoyment and enhance the differences between 2D photographs and 3D animations. Figures 10(b–g) illustrate the panorama effect produced by our system. Figure 10(h) shows the view of three-dimensional walk-through between photographs. Note that, for further enhancing the effects, some foreground objects (i.e. people) are manually cut out from the scene to make "pop-ups" [9] in our results. The same technique is also applied to the following examples.

Figure 11 illustrates results of Photo Navigator on photo set B. This example is quite challenging for feature matching since it consists of natural scenery with only a limited number of noticeable objects. Although spatial relationships between some of those photographs are not obvious to viewers, Photo Navigator successfully enhances the spatial relations in the output video with compelling walk-through effects. Furthermore, Photo Navigator can display captions supplemented by users on the photographs to enhance slideshows with semantic information as shown in Figure 11(a) and (d).

Photo set C belongs to the category of landmark photographs, taken in Chiang Kai-Shek memorial hall. Figure 12 shows the novel renderings when walking through the

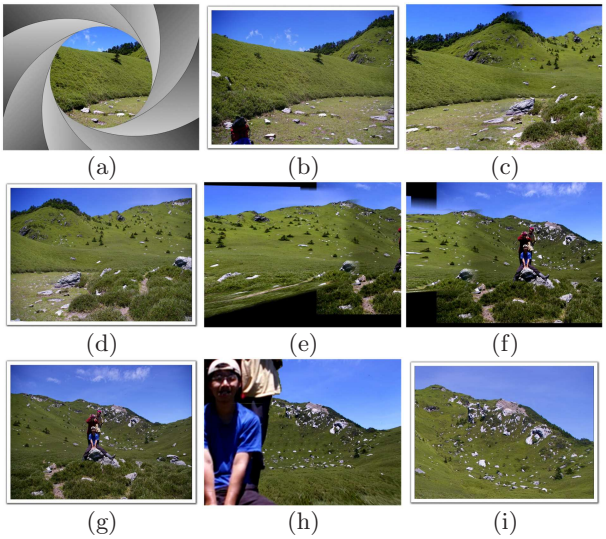

Figure 10: Sample frames of slideshow A.

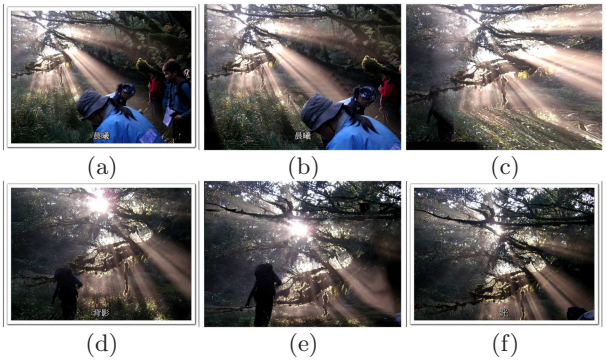

Figure 11: Sample frames of slideshow B.

scene. Due to the limitation of 3D scene modeling, objects with more prominent shapes, such as architectures, usually result in more obvious artifacts compared to the natural landscape. Therefore, a varying speed of camera motion is required to bring a little perceptual motion blur to effectively reduce the artifacts.

Photo set D is a night view for a street scene. Similar to photo sets A and B, it belongs to a personal photograph collection. Thus, it is not easy to find external sources and Photo-tourism-like systems won't work well with such a sparse set of photos. Since it is dark in most areas of night view photos, this scene is more challenging for feature matching. Nevertheless, the artifacts of 3D scene model reconstruction are not obvious to users, as shown in Figure 13.

In Figure 14, Photo Navigator is applied to photo set E, a set of pictures taken at National Taiwan University. This set belongs to the category of campus landscape. This is an example which contains buildings and trees in the same scene. Figures 14(d) and (e) show novel views that are not part of the input images.

The proposed system is implemented with Matlab 7.0 on a machine with Intel Core-2 1.86 GHz CPU, 2GB memory, and Microsoft Windows XP. On average, the processing time

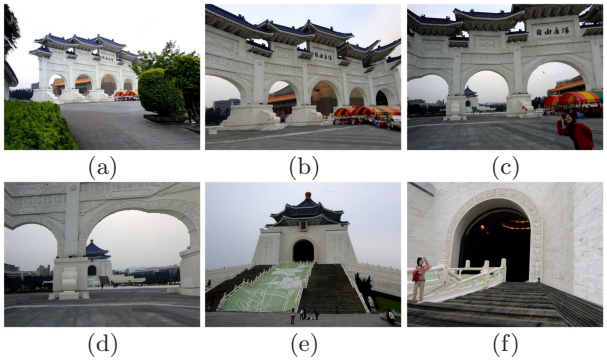

Figure 12: Sample frames of slideshow C.

Figure 13: Sample frames of slideshow D.

for a set of 15 photos (with a resolution of 2253×1690) is about 19,637 seconds, and the time is roughly quadric to the number of photos. Note that, if the photos are first scaled down to the size of 800×600, the processing time can be effectively reduced to around 5,778 seconds. The percentage of time spent in each underlying component is shown in Figure 15. It is clear from the figure that feature matching is the most expensive component, consuming roughly 3/4 of time.

6. EVALUATION

This section shows evaluation of Photo Navigator. Since objective evaluation of the proposed system is difficult, subjective experiments were taken to evaluate the performance of our system. We compare the slideshows of Photo Navigator to the ones produced automatically by ACDSee [1] and the ones generated manually with Photo Story [2] in terms of satisfaction. Five photo sets mentioned in Section 5 were used for the evaluation. The incidental music clips for sets A and C are pop music and those for others are pure music.

Fifteen evaluators interested in tour and photography were invited for the user study. The slideshows produced by ACDSee, Photo Story, and Photo Navigator were presented to them by projecting onto a 60-inch screen. Evaluators were requested to grade from 1 to 10 to show their satisfactions (higher scores for better satisfactions) with respective to each of the following perspectives:

- Reality. How do they feel about the reality of the virtual walking-through?

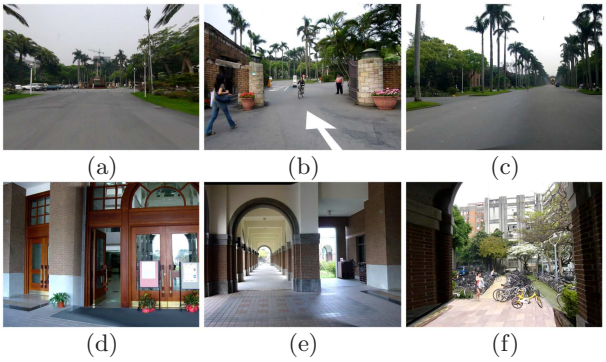

Figure 14: Sample frames of slideshow E.

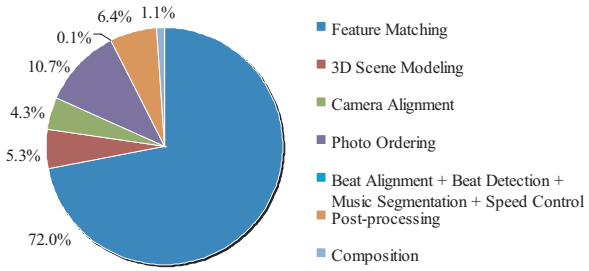

Figure 15: The percentage of computation time spent for each of the system components.

- Visual perception. How do they like the novel views between photographs?
- Smoothness. How do they think about the smoothness of the transitions?
- Spatiality. How strong sense of space does this sequence offer them after watching the slideshow?
- Acceptance. How do they feel about the overall system?
- Experience. Do they think that the slideshow helps them experience this travel and encourages them to visit?

The results of the subjective evaluation are illustrated in Figure 16. The differences between Photo Story and Photo Navigator in terms of visual perception and smoothness are relatively small. Possible reasons are that the scene visualization in 3D is not good enough, and the camera path for photograph navigation doesn't always act as users expect. Generally, Photo Navigator has significantly better satisfactions than others in spatiality.

The average satisfaction scores among three systems for the five photo sets are illustrated in Figure 17. Clearly, Photo Navigator has much higher satisfaction scores than others. In particular, our system was given an average score of 7.68, which is 14% better than Photo Story and 25% superior to ACDSee for the five photo sets. The difference of scores between Photo Navigator and Photo Story for landscape photographs are smaller than other types of photographs. It is because that the number of photographs

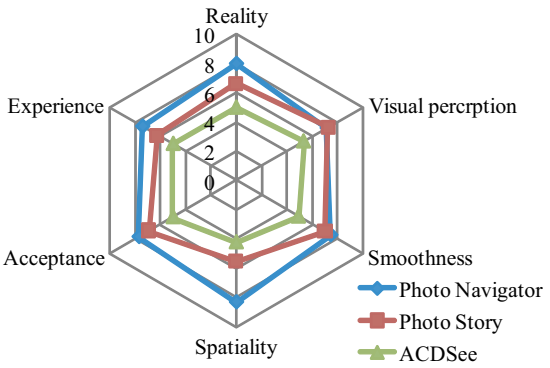

Figure 16: Comparisons of Photo Navigator, Photo Story, and ACDSee on six perspectives.

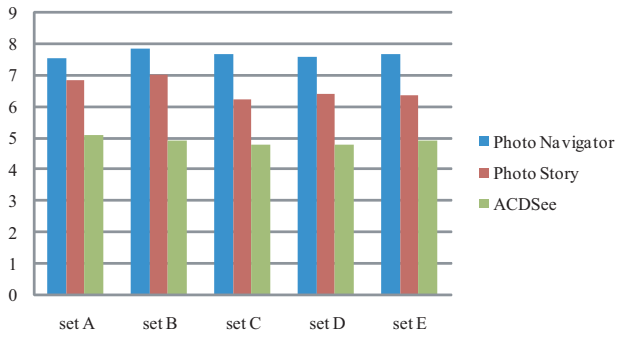

Figure 17: The results of subjective evaluation for three types of slideshows on the five tested sets.

in this category is less than other categories. On the other hand, the zooming and panning transition effects of Photo Story behave similarly to the 3D effects of Photo Navigator because of the lack of noticeable objects in sight. Therefore, the difference between Photo Story and Photo Navigator for the examples with few landscape photographs is relatively small. However, it is apparent that Photo Navigator is suitable for various types of photo sets.

7. CONCLUSION

This paper proposes Photo Navigator, a system which provides a new type of slideshows for travel photos by making visually seamless 3D presentations for user's own photos. Simple cuboid models and a smooth camera path are automatically discovered to provide users a 3D navigation experience for photographs with limited artifacts. Compared to Photo Story, our system reveals more sense of space and offers more enjoyment of watching the slideshows. Compared to Photo Tourism, our system can work with a sparse set of photographs and is more suitable for personal travel photo slideshows. Many aspects of our system can be improved. For example, automatic algorithms for creating the "pop-up" foregrounds are worth of further investigation. In addition, more efficient algorithms for feature matching would greatly speed up our system.

Acknowledgments

This work was supported by the National Science Council of Taiwan, R.O.C., under contracts NSC95-2622-E-002-018 and NSC96-2622-E-002-002. It was also supported by National Taiwan University under grant NTU95R0062-AE00-02. The author would like to thank reviewers for their helpful suggestions.

8. REFERENCES

[1] ACDSee: `http://www.acdsystems.com/`.

[2] Photo Story: `http://www.microsoft.com/`.

[3] Picasa: `http://picasa.google.com/`.

[4] Z. Cao, X. Sun, and J. Shi. Tour into the picture using relative depth calculation. In *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 38–44, 2004.

[5] J.-C. Chen, W.-T. Chu, J.-H. Kuo, C.-Y. Weng, and J.-L. Wu. Tiling slideshow. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 25–34, 2006.

[6] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[8] R. I. Hartley. In defence of the 8-point algorithm. In *Proceedings of ICCV 1995*, pages 1064–1070, 1995.

[9] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, 24(3):577–584, 2005.

[10] Y. Horry, K.-I. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of ACM SIGGRAPH 1997*, pages 225–232, 1997.

[11] X.-S. Hua, L. Lu, and H.-J. Zhang. Automatically converting potograic series into video. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 708–715, 2004.

[12] X.-S. Hua, L. Lu, and H.-J. Zhang. Photo2video–a system for automatically converting photographic series into video. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(7):803–819, 2006.

[13] C.-C. Liu, C.-H. Huang, W.-T. Chu, and J.-L. Wu. ITEMS: intelligent travel experience management system. In *Proc. 9th ACM Intl. Workshop on Multimedia Information Retrieval (MIR'07)*, pages 291–298, 2007.

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

[15] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[16] A. Saxena, M. Sun, and A. Y. Ng. Learning 3-D scene structure from a single still image. In *Proceedings of ICCV 2007*, pages 1–8, 2007.

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