A unified approach for motion analysis and view synthesis*

Alex Rav-Acha   Shmuel Peleg
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
Email: {alexis,peleg}@cs.huji.ac.il

Abstract

Image based rendering (IBR) consists of several steps: (i) Calibration (or ego-motion computation) of all input images. (ii) Determination of regions in the input images used to synthesize the new view. (iii) Interpolating the new view from the selected areas of the input images. We propose a unified representation for all these aspects of IBR using the Space-Time (x-y-t) volume. The presented approach is very robust, and allows to use IBR in general conditions even with a hand-held camera.

To take care of (i), the Space-Time volume is constructed by placing frames at locations along the time axis so that image features create straight lines in the EPI (epipolar plane images). Different slices of the Space-Time volume are used to produce new views, taking care of (ii). Step (iii) is done by interpolating between image samples using the feature lines in the EPI images.

IBR Examples are shown for various cases: sequences taken from a driving car, from a hand held camera, or when using a tripod.

1 Introduction

When input images are taken with a camera translating along a straight line, the *x-y-t* Space-Time volume is used as a unified representation for all aspects of view synthesis. For 2D camera motion on a plane, the *x-y-t* space-time volume is replaced with the *x-y-u-v* light field representation. Space-Time representation for a camera translating on a line is simple and elegant, and is therefore used in most of the discussions. But since perspective views can not be synthesized from a camera moving along a line, 2D camera motion on a plane is used as well.

The Space-Time approach can also be extended to handle camera rotations. Setups describing camera motions which are applicable to this work are shown in Fig. 1.
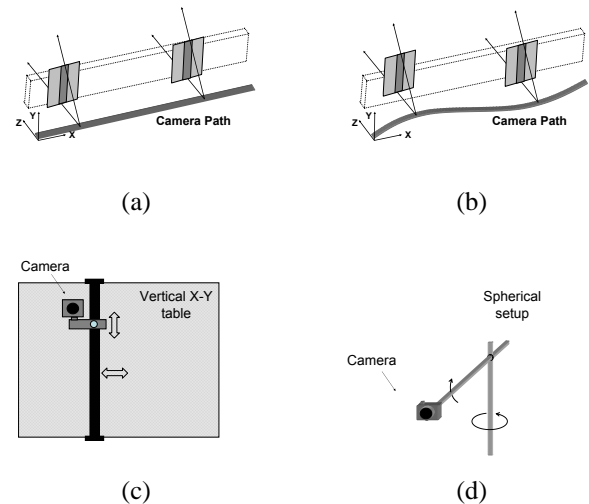

Figure 1: Arrangements for 1D and 2D camera motion.
(a) 1D motion - The camera moves along a straight line.
(b) 1.5D motion - Like (a), with some vertical jumps.
(c) 2D motion - Traditional light field capturing device.
(d) 2D motion - Camera moves on a surface of a sphere.

1.1 Relation to Previous Work

The Space-Time volume (or the *epipolar volume*), constructed by stacking all input images into an *x-y-t* volume, was introduced by Bolles et. al. [2]. They used *x-t* slices of the *x-y-t* volume (EPI) to extract 3D information.

Later, this representation became a popular tool for many robust algorithms of scene recovery and scene augmentation [14, 11, 19]. The EPI representation is also used as a simple representation of image sequences [8], under the assumption of a known 3D camera motion.

The EPI representation is usually used when the camera's translation is parallel to the image plane and is perpendicular to the viewing direction. However, it can also be used with other viewing directions after image rectifica-
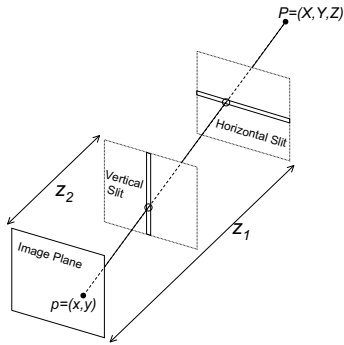
Figure 2: In the X-Slits projection a ray passes through a point and two linear slits. In this figure the slits are orthogonal to each other and parallel to the image plane.

Figure 3: New views are generated as *x-t* slices in the Space-Time volume. (a) Changing the orientation of the slice moves the vertical slit inside and outside the scene. The central slice gives a pushbroom image (a "traditional" mosaic). (b) Sliding parallel slices in the Space-Time volume results in different viewing directions of oblique pushbroom images.

tion [9], and even in the case of forward motion [15].

Recently, the Space-Time volume was used for view synthesis [20] when the camera motion is known. In addition, the sampling of the Space-Time volume along the x axis was assumed to be very dense, so a trivial interpolation was sufficient to produce nice looking views.

In this paper the camera motion does not have to be known, since it is computed from the input images using a "Time Warping" method. In addition, the need for a dense sampling is relaxed by improving the interpolation of slices in the Space-Time volume.

The presented work is closely related to many image-based rendering (IBR) techniques. In these techniques, rays from a set of input images are collected and a new image is rendered by resampling the stored rays [13, 7, 18]. Several techniques use sequences taken from a hand-held camera [5, 12]. Our approach is unique in using the same representation for motion computations and for view synthesis. A unified representation has both appealing features and practical advantages. Moreover, the motion computation is robust as it is not feature-based, and does not require calibration.

We begin by describing the "Crossed-Slits" (X-slits) projection [20], showing the close relations between a slicing of the Space-Time volume and View Synthesis.

1.2 X-Slits Projection

The X-Slits projection, shown in Fig. 2, involves two slits which are two different lines in $\mathcal{R}^3$, together with an image plane that does not contain any of the slits. For every 3-D point there is a single ray which connects the point with both slits simultaneously. The intersection of this ray with the image plane defines the projected image of the 3-D point. Fig. 2 shows a special case of the X-Slits projection, where the two slits are perpendicular to each other and are parallel to the image plane. It was proposed in [20] to use the X-slits projection in IBR as an approximation of the
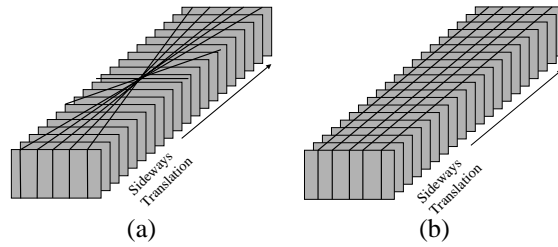
perspective projection.

Since there are no X-slits cameras, X-slits images are generated by mosaicing. Mosaicing of strips taken from the same location in each input image corresponds to a slice of the Space-Time (x-y-t) volume parallel to the y-t plane (Fig. 3.b). Such an image is generated from rays on parallel planes in the x direction, and corresponds to an X-slits projection with the vertical line in infinity (a pushbroom projection). This projection can be generalized to X-slits projections with any desired location of the vertical slit [20] by slicing the Space-Time volume diagonally as demonstrated in Fig. 3.a. Diagonal slices correspond to rays on planes that intersect in a vertical line corresponding to closer viewing locations.

2 View Synthesis in Space-Time Volume

For clarity of presentation, we describe our approach using the x-y-t Space-Time volume, where the camera motion is restricted to a straight line. The motion is later generalized to 2D motion, using the light field representation.

We propose the following scheme for view synthesis. First, the camera motion is computed using a "Time Warping". In time-warping, the t axis is resampled so that image features in the EPI planes reside on straight lines.

After the Space-Time volume was correctly warped in time, and assuming dense input images, view synthesis could be accomplished by taking y-t slices of the Space-Time volume using the "Crossed-Slit" projection. When the images are not dense enough, interpolation to densely complete the space-time volume is needed as a pre-processing. Since this is not practical due to memory limitations, only the slices in the synthesized views should be interpolated. For fast computation this interpolation uses pixels taken from the images themselves. In the remainder of this section we describe in more details the "Time Warping" and the interpolation.

2.1 Time Warping: From Time To Location

When the camera's velocity and frame rate are constant, the time of frame capture is proportional to the location of the camera along the camera path. In this case, the image features are arranged in the EPI plane (an *x-t* slice of the *x-y-t* volume) along straight lines, since the projections of each 3D point are only along a straight line in this plane. Each straight line represents a different image feature corresponding to a point in the 3D world, and the slope of this line is inversely proportional to the depth of that point. Points at infinity, for example, will create straight lines parallel to the t axis, since their projection into the image is constant, and does not change with camera translation. Closer points move faster in the image, and the straight line representing them will have a small angle with the x axis.

When the velocity or the frame rate of the camera vary, the time of frame capture is no longer proportional to the location of the camera. Image features are no longer arranged along straight lines in the EPI plane.

The lines in the EPI plane can be straightened by "Time Warping". In time warping, the image location along the time axis is replaced with the camera's location along the x axis. When the camera locations along the x axis are unknown, any time warping that will make the EPI lines straight must have time spacing which is proportional to the camera locations along the x axis.

In Section 4 we describe the implementation of our time-warping, addressing both cameras translating along a straight line, as well as two dimensional camera motion on a plane. Rotations can also be handled.

2.2 View Interpolation

Perfect view interpolation is difficult as in general it requires the estimation of the depth (or the optical flow) for each image point [17, 6]. When the interpolation is needed only for producing continuous and nice looking images, sequences with infinitesimal displacements between neighboring frames can be used. In this case, depth-independent interpolation can produce good results [13, 7, 3]. This kind of interpolation is similar to the blending scheme used for mosaicing [4, 15].

When the sampling of the Space-time volume is not very dense, better interpolation is needed for the generation of new views. This can be obtained by scaling the patches taken from each frame. Fig. 4 demonstrates the view interpolation in the EPI domain. In this figure it can be seen that scaling the strip from its width in the original image to its width in the new view is necessary to produce seamless images. This scaling is proportional to the slopes of the EPI lines. Theoretically, the slopes of the EPI feature lines can vary for each *x-t* EPI plane. A practical approach is to use a simple parameterization of the scale. In our experiments we found that even a constant scale factor per frame
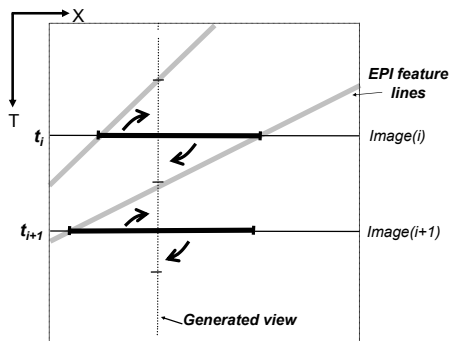

Figure 4: View interpolation in the EPI domain. The intersection of a slice in the Space-Time volume (corresponding to a new view) with the EPI image is a line. The Space-Time volume is sparsely sampled, so the gaps in the intersection line should be interpolated. This can be done by "bending" strips from each image onto the new view. As can be seen from the figure, to produce a continuous result strips should be scaled to fit the slopes of the EPI feature lines.

gave a significant improvement over the naive interpolation. Determining the optimal scale factor for each frame is discussed in Section 4.

3 2D Camera Motion: 4D Light Field

Given a set of images whose optical centers reside on a plane, all images can be represented as a set of rays [13, 7]. Each ray can be determined by its intersection with two planes. Commonly, one plane is the camera plane *u-v*, and the second plane is the image plane *s-t*.

A perspective image is a set of rays which intersect the camera plane in a single point. When the optical axis of the camera is perpendicular to the camera plane (i.e - a frontal view), the camera plane is expressed by the 3D coordinates: $(u, v, 0)$, and the image plane is expressed by the 3D coordinates: (s, t, f). Each image can be considered as a sampling of a 2D slice in the continuous 4D function $L(u, v, s, t)$.

In our notation each image has its own *x-y* coordinate system, and the 4D volume is represented by *x-y-u-v*. Using the image coordinate system is a natural generalization of the Space-Time volume *x-y-t*. Also, in local image coordinates the slopes of the epipolar lines are inversely proportional to the depth. This is equivalent to the light-field notations with the *s-t* plane at infinity.

3.1 Analysis of the Light Field Space

Let I_n be the nth frame, and let (u_n, v_n) be its optical center. The 3D point $P = (X, Y, Z)$ is projected to its image coordinates:

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X - u_n \\ Y - v_n \end{pmatrix} \qquad (1)$$

Without loss of generality, $(u_0, v_0) = (0, 0)$ and thus:

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \frac{f}{Z} \begin{pmatrix} u_n \\ v_n \end{pmatrix} \qquad (2)$$

From Eq. 2 it can be shown that the projections of the point P onto these images reside on a plane in the 4D light field space. The parameterized representation of this plane is given by:

$$\begin{pmatrix} x \\ y \\ u \\ v \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ 0 \\ 0 \end{pmatrix} + u \begin{pmatrix} -\frac{f}{Z} \\ 0 \\ 1 \\ 0 \end{pmatrix} + v \begin{pmatrix} 0 \\ -\frac{f}{Z} \\ 0 \\ 1 \end{pmatrix} \qquad (3)$$

The slope of this plane is identical in both the u and the v directions as both are determined by f/Z, and only a single slope parameter is needed to describe the plane.

Taking an *x-u* or an *y-v* slice of the 4D space will produce an EPI image which is constructed respectively from the rows of the images taken under a translation along the u axis, or from the columns of the images taken under a translation along the v axis. This is a reduction to the traditional representation of the epipolar plane image.

We will describe the use of this 4D representation to recover the 2D motion of a camera moving in a plane. This is done by "shifting" the estimated u, v coordinates of each frame so that the image features will be arranged in the 4D volume *x-y-u-v* along 2D planes. When this goal is achieved after warping the u and v axes, the estimated u, v coordinates of the frames become proportional to the 2D locations of the optical centers of the camera. In the case of 1D motion this is called "Time Warping".

3.2 Synthesizing a Novel View

View synthesis with 2D camera motion is very similar to the synthesis described for the 1D case (the Crossed-Slit), but now both x and y coordinates are used in the computations. Alternatively, the methods proposed in [13, 7] can be used. When the sampling of the light field is not dense, interpolation of the generated view becomes similar to the 2D mosaicing methods [16]. In this case, a Voronoi tessellation is used to determine the regions in the generated view coming from each input image. The above 2D mosaicing methods address only camera rotations where no parallax is involved. To handle parallax, image patches should be scaled as shown in Section 4.

3.3 Spherical Camera Motion

When a sequence is taken by a hand-held camera or by using a tripod, the camera's motion is constrained to a spherical surface, rather than to a plane. In this case the camera motion includes both translation and rotation. However, since the rotation is proportional to the translation in all directions, the EPI feature lines are almost straight. As a result, it is adequate to treat the motion identically to the case of a camera translating on a plane, with the two unknowns u and v. When the camera can rotate about the z axis, e.g. when the camera is hand-held, a rotational component α can be added as a third unknown.

4 Ego-Motion by Time Warping

As usual in motion computation, the alignment process uses both motion parameters and shape parameters. The motion parameters are the translation and rotation of the camera, which vary for each frame. The shape parameters are represented by the slopes of the lines in the EPI plane for a camera translating along a straight line, or the slopes of the planes in the light field space for a camera translating in a plane. The slopes of the lines and the planes in the EPI domain are inverse proportional to the depth of the corresponding 3D points, and thus they remain constant for each scene point at all frames.

To compute the locations of the optical centers of the input cameras, such that image features will reside on straight lines (or on planes), we used the following scheme, alternating between the estimation of shape and the estimation of motion:

1. Choose a frame I, and initial from it a set $S = \{I\}$. Assume that the shape parameters corresponding to this image are spatially uniform.

2. Compute motion parameters (translation components and optionally rotation components) by aligning a new frame to the existing set S.

3. Add the registered frame to the set S.

4. Estimate the shape parameters (the slopes of EPI lines or the slope of EPI planes) for this set S.

5. Return to 2. Repeat until reaching the last frame of the sequence.

Fig. 5 demonstrates this scheme for the case of a camera translating along a straight line.

4.1 Estimating the Shape Parameters

The shape parameters are needed only for a subset of image points, as they are used to compute only a few motion parameters. The process can be formulated in the following way: Let k be the index of the frame for which we estimate the shape and let $T_{n,k} = (u_n - u_k, v_n - v_k)^t$ be the translation of the optical center of the camera between the nth and the kth frames.
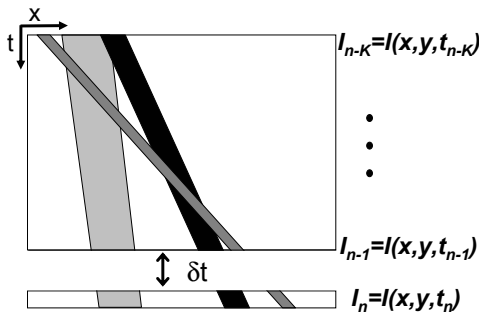
Figure 5: Time warping: When the scene has depth variations there is only a single placement t_n of the new frame I_n for which the EPI lines remain straight.

Following [10], The shape parameter $d = d(x, y, k)$ in the image point (x, y) minimizes the error function:

$$Err(d) = \sum_{n \neq k} w_n^d \cdot \sum_{x,y \in W} (d \cdot \bigtriangledown I^t \cdot T_{n,k} + I_n - I_k)^2, \quad (4)$$

Where $\bigtriangledown I$ is the gradient of the image I_k in the point (x, y), and W is a small window around (x, y). (We used a 5x5 window). The minimum of this quadratic equation is obtained by:

$$d = -\frac{\sum_{n \neq k} w_n^d \cdot \sum_{x,y} \bigtriangledown I^t \cdot T_{n,k} \cdot (I_n(x,y) - I_k(x,y))}{\sum_{n \neq k} w_n^d \cdot \sum_{x,y} (\bigtriangledown I^t \cdot T)^2}$$
$$(5)$$

The weights w_n^d determine the influence of each frame on the shape estimation. Most of the weights are set to zero, except for frames which are close in time or in space (currently we use the five closest frames).

For each window in I_k, the computation described above is repeated iteratively until convergence, where in each iteration, the relevant regions in all the frames $\{I_n\}$ with $w_n^d \neq 0$ are warped back towards I_k according to $T_{n,k}$ and the current estimate of d.

As we do not need to estimate the shape parameters for every pixel, only the best points are used:

1. We do not use points with a small gradient in the direction of motion. The threshold is selected according to the desired number of points to use.

2. We do not use points for which the iterative shape computation algorithm fails to converge.

4.2 Depth Invariant Alignment

The alignment concept is demonstrated in Fig. 5. The motion parameters should align an input image with the lines or planes formed by image features in the preceding frames. We use a slight modification of the Lucas-Kanade direct 2D alignment as described in [1].

Assume that all the images $I_0 \ldots I_{k-1}$ have already been aligned, and let the kth frame be the new frame to be aligned. We also know of the shape parameters $d(x, y, n)$ for $n < k$. To compute the motion parameters of the new frame, we minimize the error function: (Sometimes the term I_t is used to denote the difference between images).

$$Err(p, q) = \sum_{n \neq k} w_n^a \cdot \sum_{x,y} (p\frac{\partial I_n}{\partial x} + q\frac{\partial I_n}{\partial y} + I_n - I_k)^2, \quad (6)$$

where the displacement p, q of each point is given by:

$$p(x, y, n) = (u_n - u_k) \cdot d(x, y, n)$$
$$q(x, y, n) = (v_n - v_k) \cdot d(x, y, n). \quad (7)$$

Note the use of the derivatives $\frac{\partial I_n}{\partial x}$ and $\frac{\partial I_n}{\partial y}$ which are estimated from I_n rather then from I_k, since we haven't computed $d(x, y, k)$ yet, and therefore we must align frame I_k to the rest of the images.

The coefficients w_n^a are also used to weight the importance of each frame in the alignment. For example, frames which are far off, or contain fewer information should receive smaller weights. For each image whose location u_n, v_n is unknown we set $w_n^a = 0$.

Currently we align a new frame using about three preceding frames. When the camera is translating on a plane we use several additional frames which are not neighbors in time but whose optical centers are close. In this way we reduce the drift in the motion computations.

4.2.1 Handling rotations

When the camera can also rotate, image displacements are a combination of the translational component, which is depth dependent, and the rotational component which is depth independent. Assuming small camera rotations and using the approximation $cos(\alpha) \approx 1$ and $sin(\alpha) \approx \alpha$ the following motion model is obtained:

$$p(x, y, n) = (u_n - u_k) \cdot d(x, y, n) + a - \alpha \cdot y$$
$$q(x, y, n) = (v_n - v_k) \cdot d(x, y, n) + b + \alpha \cdot x. \quad (8)$$

a and b denote the small pan and tilt which induce an approximately uniform displacement in the image. α denotes small camera rotation about the z axis. For larger rotations, or when the focal length is small, full rectification can be used.

Using Eq. 8 with the error function in Eq. 6, and setting to zero the derivative with respect to the motion parameters (camera shift u, v and rotational components α, a, b), gives a set of five linear equations with five unknowns.

If the camera is restricted to translate along a straight line (without the loss of generality this line is horizontal), then $v_n = v_k = 0$, and we are left with fewer unknowns - one unknown for translation only, and four unknowns for translation plus rotation.

5 Interpolating Generated Views

As stated before, in order to obtain a seamless synthesized view, the slices from each image should be scaled before being used. For 2D camera motion the scaling is performed in all directions, while for 1D horizontal motion the slices are scaled only horizontally. The scaling is determined by the slopes of the EPI features lines (or planes), as demonstrated by Fig. 4. The slopes of the EPI lines are determined in a similar manner to the slope estimation described in Eqs. 4-5. Here, however, we do not estimate the slope at each point, but rather a single slope (or a parametric one) for the entire slice. Another difference is that the scaling need not be very accurate, but rather it should fit the closest neighbors. Hence, we use only the preceding and succeeding frames. For constant scale we receive:

$$scale = -\frac{\sum_{i=-1,1} \cdot \sum_{x,y \in R} \nabla I^t \cdot T_{k+i,k} \cdot (I_{k+i} - I_k)}{\sum_{i=-1,1} \cdot \sum_{x,y \in R} (\nabla I^t \cdot T_{k+i,k})^2},$$
(9)

where R is the set of pixels from the kth frame which participate in the synthesized view.

Note that the scaling factor is calculated relative to the original image. For a synthesized view which is not located at infinity, there is an additional scale that is not depth dependent, but is determined by the z coordinate of the view. These two scales should be added to a single unified scale, to obtain a seamless synthesized image.

6 Examples

Figure 6-10 show view synthesis from images taken by a moving camera in various cases: In Fig. 6 the original images were taken by a camera mounted on a panning tripod as shown in Fig. 1.d. In Fig. 7 the tripod was also tilting. In Fig. 8 the sequence was taken by a hand-held camera. In Fig. 9 the camera laid on a table and was pushed by hand, and in Fig. 10 the images were taken by a camera in a moving car.

7 Concluding Remarks

All steps of Image-Based Rendering can be done in a single framework of the Space-Time volume. This includes motion computation, which is carried out by re-spacing the input images along the time axis such that image features reside on straight lines. Determining the image segments to be used in the synthesized view can be done by slicing the Space-Time volume, and the interpolation of the synthesized view from its parts can be done using the slopes of the curves in the Space-Time Volume.
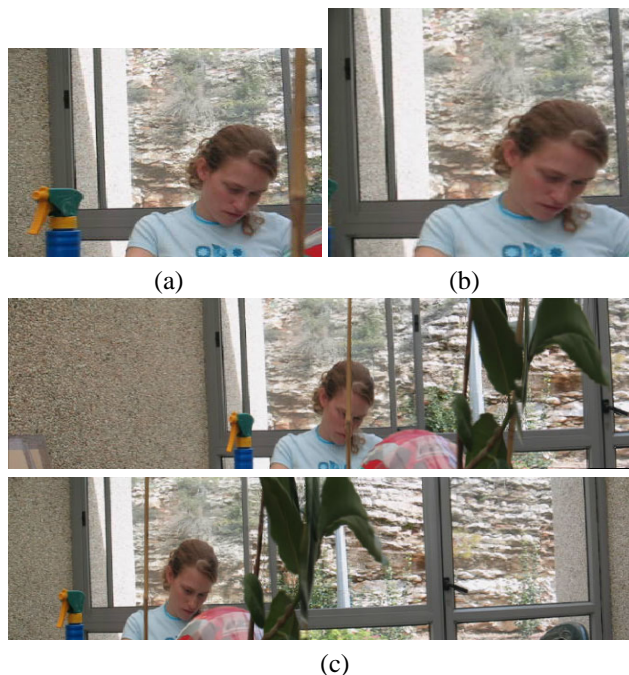

(a)　　　　　　　　　　　(b)

(c)

Figure 6: View Synthesis from a camera on a circular trajectory as demonstrated in Fig. 1.d. The sequence consists of 145 frames. a) An image from the sequence. (b) A close-by view (Note the disappearance of occluding objects.) (c) Panoramic left and right views.

Figure 7: Synthesized views (upper-left, upper-right, lower-left, lower-right) from a camera moving on a surface of a sphere. The sequence consists of 308 frames.

This process was found to be very robust, and can even handle input images taken by a hand-held camera. Such images can rotate, and translate in x and y.

While the cases handled by the approach presented here are limited, they represent a very large class of situations.

(a)   (b)

(c)

Figure 10: Synthesizing views from images taken by a translating camera (moving in a car). This setup is problematic for many IBR methods, which require a better control on the scene.
(a) Image from the original sequence which consists of 300 frames. (b) Synthesizing closer views enables us to "enter" the gaps between the cars. (Note the hidden car and the tires which are revealed). (c) Simulating a faraway synthesized view gives a panorama of the scene.

Figure 8: Left and right views synthesized from a hand-held camera. (Total of 240 frames.) Hand-held cameras undergo significant rotations and vertical vibrations.
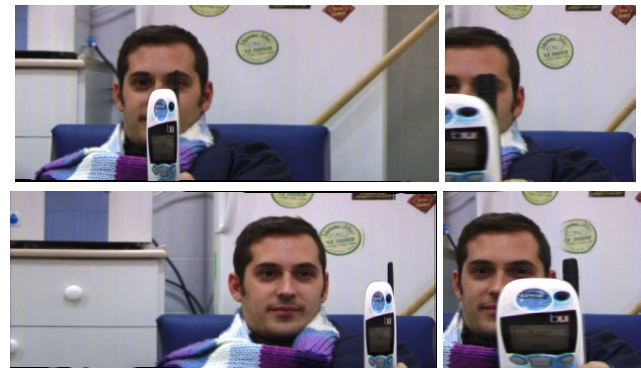
These include images taken from a side window of a moving car, images taken by a stationary person scanning a scene, etc.

Acknowledgment
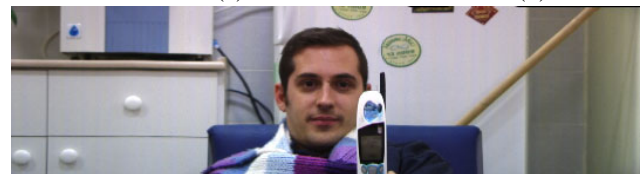
References

[1] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252, 1992.

(a)   (b)

(c)

Figure 9: The source sequence was taken from a translating camera (pushed by hand). It consists of 148 frames. Few synthesized images are shown: Right and left views(a), Two close-by views(b), and a faraway view(c).

[2] R. Bolles, H. Baker, and D. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–56, 1987.

[3] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured lumigraph rendering. *SIGGRAPH*, pages 425–432, 2001.

[4] J. Burt and E. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, 2(4):217–236, 1983.

[5] J. Chai, S. Kang, and H. Shum. Rendering with non-uniform approximate concentric mosaics. In *SMILE00*, pages 94–107, 2000.

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH*, 30:11–20, 1996.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. *SIGGRAPH*, 30:43–54, 1996.

[8] M. Halle. Multiple viewpoint rendering. *Computer Graphics*, 32(Annual Conference Series):243–254, 1998.

[9] R. Hartley. Theory and practice of projective rectification. *IJCV*, 35(2):1–16, November 1999.

[10] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *PAMI*, 24(11):1528–1534, November 2002.

[11] S. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR01*, pages I:103–110, 2001.

[12] R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *ICCV99*, pages 585–591, 1999.

[13] M. Levoy and P. Hanrahan. Light field rendering. *SIGGRAPH*, 30:31–42, 1996.

[14] M. Okutomi and T. Kanade. A multiple-baseline stereo. *PAMI*, 15(4):353–363, April 1993.

[15] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet. Mosaicing on adaptive manifolds. *PAMI*, 22(10):1144–1154, October 2000.

[16] H. S. Sawhney, S. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *ECCV*, pages 103–119, 1998.

[17] H. Shum and R. Szeliski. Construction and refinement of panoramic mosaics with global and local alignment. In *ICCV98*, pages 953–958, 1998.

[18] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. *SIGGRAPH*, 33:299–306, 1999.

[19] Z. Zhu, G. Xu, and X. Lin. Panoramic epi generation and analysis of video from a moving platform with vibration. In *CVPR99*, pages II: 531–537, 1999.

[20] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall. Mosaicing new views: The crossed-slits projection. *PAMI*, pages 741–754, June 2003.