Virtual View Specification and Synthesis in Free Viewpoint Television Application

Wenfeng Li[1], Jin Zhou[1], Baoxin Li[1], M. Ibrahim Sezan[2]

[1]Dept. of Computer Science & Engineering
Arizona State University, Tempe, AZ 85287
{wenfeng.li,jin.zhou,baoxin.li}@asu.edu

[2]SHARP Laboratories of America
5759 NW Pacific Rim, Camas, WA 98607
sezan@sharplabs.com

Abstract

In conventional TV, users can get only a single view of a real 3D world. The view is determined not by a user but by the camera position during acquisition. Free viewpoint television (FTV) is a new concept which aims at giving viewers the flexibility to select a novel viewpoint by employing multiple video streams as the input. Current proposed solutions for FTV include those based on ray-space sampling and interpolation. Most existing work makes the assumption that the cameras are calibrated. Moreover, virtual view specification is done in ways that would not be practical for actual use (e.g., some methods specify the virtual view by a rotation matrix and a translation vector). This paper proposes a framework for FTV based on image-based rendering, complete with key algorithms for color-based segmentation and correspondence, and multiple-image-based virtual view synthesis from uncalibrated cameras. Moreover, we propose an intuitive approach to specifying the virtual viewpoint, based on any two views chosen by the user. This makes the specification of the virtual view very intuitive and thus practical for the FTV application. Experiments with real video streams were performed to validate the proposed approaches.

1. Introduction

Television is probably the most important visual information system in past decades, and it has indeed become a commodity of modern human life. In conventional TV, the viewer's viewpoint is determined by that of the acquisition camera. Recently, a new concept, free viewpoint television (FTV) [8-9], has been proposed, which promises to bring a revolution to TV. The basic idea of FTV is to provide the viewer the freedom of choosing his/her own viewpoint, by incorporating multiple video streams captured by a set of cameras. In addition to home entertainment, the FTV concept can certainly apply to other related domains such as gaming and education. Note that, the user-chosen viewpoints do not need to coincide with those of the acquisition cameras, and thus this is not a simple view change by switching cameras (as possible with some DVD for a couple of preset views). Apparently, this FTV revolution demands a whole spectrum of efforts ranging from acquisition hardware, coding, bandwidth management, and standardization, etc. In this paper, we focus on one particular aspects of FTV: virtual view synthesis.

The problem of virtual view synthesis can be defined as follows: Given a set of images acquired from different viewpoints, construct a new image that appear to be acquired from a novel viewpoint. This problem is also known as image-based rendering (IBR), which has been extensively studied in the literature. For example, [10] and [11] are among the early papers, and more recent ones include [1,3,6,12,14]. While there exist alternative approaches to virtual view synthesis in FTV (e.g., the ray-space sampling and interpolation approach in [9]), IBR has the potential of requiring fewer cameras and loose constraints on the camera configuration. Our proposed approach belongs to this category.

In the FTV application, it is unlikely that the camera calibration information can be made available (e.g., imagine shooting a movie with multiple cameras which need to be calibrated each time they are moved). This immediately renders those IBR methods requiring full camera calibration inapplicable. Moreover, before virtual view synthesis, the virtual view needs to be specified. Existing IBR techniques use a variety of ways for this purpose. For example, in [3], the virtual view specification is trivial as the entire setup is fully calibrated; in [10], a method was given based on the user's manual pick of some points including the projection of the virtual camera center; in [6], this is determined by a rotation matrix and a translation vector with respect to a known camera. Apparently, none of these approaches can be easily adopted by the FTV application with uncalibrated cameras, where an ordinary user needs an intuitive way of specifying some desired (virtual) viewpoints.

In this paper, we propose a framework for the rendering problem in FTV based on IBR. Our approach uses multiple images from uncalibrated cameras as the input. Further, while a virtual view is synthesized mainly from two principal views chosen by a viewer, other views are also employed to improve the quality. Being able to start with two user-chosen views (which are assumed to be closer to

Proceedings of the Third International Symposium on
3D Data Processing, Visualization, and Transmission (3DPVT'06)
0-7695-2825-2/06 $20.00 © 2006 IEEE

the desired virtual views than others) contributes to the reduction in the number of required views, as these two views can be used heavily for synthesizing the new view while others are used for only improvement. These aspects of the proposed approach sets it apart from exist methods such as in [3,6,11-13]. Moreover, as the second major contribution of the paper, we propose an intuitive method for specifying the virtual view in uncalibrated cameras, and thus providing a practical solution to view specification in the FTV application without requiring either full camera calibration (as in [3]) or complicated user interaction (as in [6,10]), which are all impractical for FTV.

2. Problem Definition

In this section, we formally define the problem to be solved. The problem definition also reflects our view on how potentially the FTV application should configure the entire system including how (ideally) cameras should be positioned and how a user may interact with the system, if the proposed approach is used for rendering.

We assume that multiple synchronized views of the same scene are captured by a set of fixed but otherwise uncalibrated cameras[1]. We assume that the multiple video streams are available to a viewer. The viewer is to specify a virtual viewpoint and requests the system to generate a virtual video corresponding to that viewpoint.

In a typical IBR approach, since no explicit 3D reconstruction and re-projection is performed, in general the same physical point may have a different color in the virtual view than from any of the given views, even without considering occlusion. The differences among different views can range from little to dramatic, depending on the viewing angles, the illumination and reflection models etc. Therefore, IBR approaches, while being attractive, have a fundamental limitation: the virtual views cannot be too far from the given views, otherwise unrealistic color may entail.

With this consideration, we further assume that the cameras used in a FTV program are located strategically so that most potentially interesting viewpoint should lie *among* the given views. For the convenience of a viewer, this can be simplified to the following: the virtual view is defined as one between any two user-chosen views from the given multiple ones. The choice of the two views can be quite intuitive and transparent in practice: for example, a viewer may feel that view 1 is too to-the-left than the desired, while view 2 is too to-the right; then the desired virtual view should in somewhere between view 1 and view 2. (An implementation scheme will be presented in Sect. 4.)

Thus, our system should solve the following two problems to support the FTV application:

(1) Given the multiple video streams from uncalibrated cameras and any two user-chosen views, synthesize a virtual view in between the two views; and

(2) Provide the viewer an intuitive way of specifying the virtual viewpoint in relation to the given available views.

3. IBR for FTV: Proposed Approach

As defined above, we have a set of video streams with two that are the closest to the user's desired viewpoint (while the notion of closeness is not well-defined in uncalibrated views, in our problem this is given by the user's pick of the views. We will defer the specification of the virtual viewpoint to Sect. 4). Naturally, from the discussion in Sect. 2, we want to make maximum use of the two specified views although other views can be potentially helpful. In this sense, we will call the two user-chosen views as the basis images. Note that the basis images are dynamic based on the user's choice and this is different from, for example [12], where the basis images are defined by two specially-positioned cameras.

Our proposed approach to virtual view synthesis consists of the following major steps:

1. Pair-wise weak calibration of all views to support potentially any pair that a viewer may choose.[2]
2. Color-segmentation-based correspondence between the two basis views, where other views are taken into consideration.
3. Forward warping from basis views to the virtual view with verified disparity map.
4. For unfilled pixels, use an algorithm to do backward search on all auxiliary views to find a dominant and disparity consistent color.

We detail these steps in the following subsections.

3.1. Virtual view synthesis via weak calibration

We assume that n cameras are used in the system (n=8 in our experiments). We denote the basis views as basis camera 1 and basis camera 2. The remaining views are called auxiliary cameras 3 to n. Fundamental matrices between the basis and the auxiliary cameras are calculated with feature detector and the RANSAC algorithm and denoted as F_{13}, F_{23}, …F_{1n}, F_{2n}. The fundamental matrix between the basis cameras is F_{12}. Computation of fundamental matrices needs to be done once unless the cameras are moved. The fundamental matrices between the basis and the virtual views are denoted as F_{10} and F_{20}, respectively (which is obtained in Sect. 4, as we cannot use the same feature-detection-based method since the virtual view does not exist yet).

With fundamental matrices determined, for any point x

[1] In practice, moving cameras pose no theoretical problem if the weak calibration is done for very frame. Practically, it may be assumed that the cameras are fixed at least for a video shot and thus the weak calibration is needed only for each shot.

[2] In practice, pair-wise processing may not be needed if, for example, one view lies on the straight line between another two views.

Proceedings of the Third International Symposium on
3D Data Processing, Visualization, and Transmission (3DPVT'06)
0-7695-2825-2/06 $20.00 © 2006 IEEE

in camera 1, its corresponding point in camera 2, x', is constrained via the fundamental matrix by $x'^{T}F_{12}x=0$, which can be used to facilitate the search for the disparity d. A third corresponding point in an auxiliary camera k is denoted by x_k which is determined from $x_k^{T}F_{1k}x=0$ and $x_k^{T}F_{2k}x'=0$. Once the correspondence between x and x' is determined, a virtual view pixel x'' can be determined by forward mapping, where x'' satisfies both $x''^{T}F_{10}x=0$ and $x''^{T}F_{20}x'=0$. These relationships are illustrated in Fig. 1.

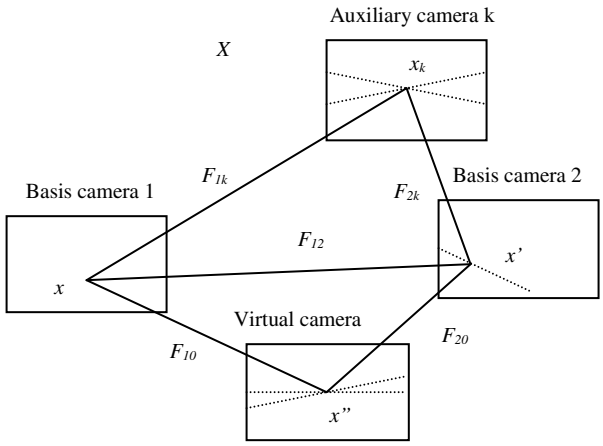

Figure 1. Illustration of the relationship among the views through fundamental matrices.

3.2. Segmentation-Based Correspondence

In Sect. 3.1, even with the epipolar constraint, we still need to search along an epipolar line for the disparity for a given point x. To establish the correspondence between x and x', we first use graph-cut-based segmentation [4] to segment each of the basis views. For all pixels within each segment, we assume that they have the same disparity, i.e. on the same front parallel plane. It is obvious that over-segmentation is favored for more accurate modeling, and similar to [3] we limit each segment to be no wider and higher than 15 pixels, which is a reasonable value for a traditional NTSC TV frame with pixel resolution of 720× 480.

Each segment is warped to another image by the epipolar constraint described above (also see Fig. 1). Instead of using the common sum-of-squared-difference (SSD), or sum-of-absolute-difference (SAD) criteria as matching scores, we simply count the number of corresponding pixel pairs whose relative difference (with respect to the absolute value) is less than 0.2 (i.e.$|R_1-R_2|/R_1<0.2$, similar for G and B), and this number, normalized by the number of pixels in the segment, is used as the matching score, denoted $m_{ij}(d)$ for any possible d and for j-th segment in basis image i. This measure was found to be more robust to lighting condition changes in our experiments.

In addition to using the matching score from the other basis image, we incorporate all the auxiliary images by computing the final matching score for a segment S_j in basis image i (denoted as S_{ij}) with disparity d as

$$m_{ij}(d) = \max_k \{m_{ijk}(d)\} \tag{1}$$

where $m_{ijk}(d)$ is the matching score of segment S_{ij} in any other basis or auxiliary camera k. (Note that, the d is always for the basis views, and searching in other auxiliary views is equivalent to checking which d is able to give arise to the most color consistency among the views whose relation is given in Fig. 1).

Furthermore, instead of deciding on a single d based on the above matching score, we now use that score in the following iterative optimization procedure, which is similar to cooperative algorithm [2]. The basic idea is to update the matching score of each color segment based on its neighboring segments of similar color in order to enforce disparity smoothness:

$$S_{ij}^{\,0}(d) = m_{ij}(d),\ r_{ij}^{\,k}(d) = \sum_{\phi} \sum_{d_0 \in (d-\Delta, d+\Delta)} S_{ij}^{\,k}(d_0)$$

$$S_{ij}^{\,k+1}(d) = S_{ij}^{\,0}(d) \left(\frac{r_{ij}^{\,k}(d)}{\sum_{d \in (d_{\min}, d_{\max})} r_{ij}^{\,k}(d)} \right)^{\beta} \tag{2}$$

where ϕ is the set of neighbor segments with similar color (defined by Euclidian color distance under a pre-determined threshold), β is the inhibition constant (set to 2 for computational simplicity) controlling the convergence speed, and k the iteration index. We use the following stopping criteria: at any iteration k, if for any d, S_{ij} exceeds the threshold, the updating process for this segment will stop at next iteration; the entire procedure will terminate until it converges (i.e., no segments need to be updated). Our experiments show that the algorithm converges typically after 10 iterations and thus we fix the number of iteration to 10.

The above procedure is performed for both basis views, and the disparity map is further verified by left-right consistency check, and only those segments with consistent results are used for synthesizing the virtual view (thus some segments may not be used, resulting in an incomplete disparity map). In Fig. 2, we show two examples of the color-segmentation results together with the resultant disparity map.

3.3. Forward Warping

Using the verified disparity map and the two basis views, an initial estimate of the virtual view can be synthesized by forward warping. For a pixel x in basis view 1 and x' in basis view 2, their corresponding pixel on the virtual view will be x'' whose color is computed as

$$RGB(x'') = (1-\alpha)RGB(x) + \alpha RGB(x') \tag{3}$$

with α being a coefficient controlling the contribution of the basis views (which may be set to the same α to be

Proceedings of the Third International Symposium on
3D Data Processing, Visualization, and Transmission (3DPVT'06)
0-7695-2825-2/06 $20.00 © 2006 IEEE

defined in Sect. 4).

Forward warping can preserve well texture details and it can easily be implemented in hardware, making real-time rendering possible. Figure 3 shows an intermediate image obtained after forward warping.

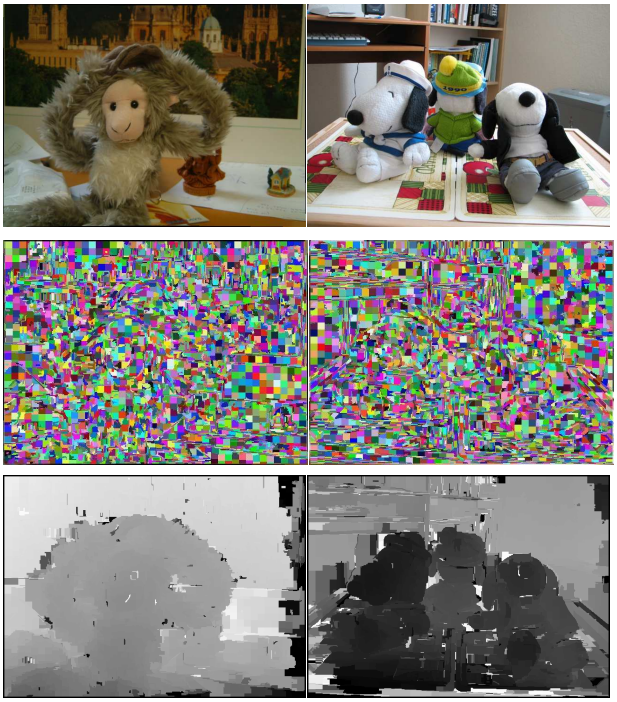

Figure 2. Color segmentation and disparity maps of the monkey scene and the snoopy scene. Top row: original images. Center row: color-based segmentation results shown as pseudo colors. Bottom row: computed disparity maps.

3.4. Backward Searching and Propagation

In the initial virtual view given by forward warping, it is not uncommon to see many uncovered pixels, which we name as "black holes". These black holes are due to incomplete disparity map (see Sect. 3.2) (among others, occlusion is one reason for this). For each black-hole pixel, we check its neighbor for a pixel that has been assigned a color value from the initial synthesis. The disparity of that pixel is then used for backward search on all the images. Unlike other similar disparity or depth searching algorithms that do exhaustive search on the entire disparity space, we search for a small range within the disparity of the "valid" neighbors (those with assigned color). The search objective function is defined as:

$$F(d) = \min_{d \in [d_n - \Delta, d_n + \Delta]} \{\lambda \cdot Dist_{color}(p_{dn}, \overline{p}) + (1-\lambda) \cdot Dist_{disp}(d_n, d)\} \quad (4)$$

where d_n is the disparity of a valid neighbor pixel and p_{dn} is its color; $\overline{p} = \{p_1, p_2\}$ are colors from two basis views corresponding to d; $Dist_{disp}$ and $Dist_{color}$ are two distance functions defined on disparity and color; and λ is a weight coefficient. The combination of the differences of color and the disparity is intended for the smoothness of both texture (color) and depth. In reality, $F(d)$ is set as the minimum one obtained from all the valid neighbor pixels. A new disparity will be accepted only when the resulting $F(d)$ is below a predetermined value. If the search fails after all possible d is tested on all valid neighbors, the corresponding pixel is left empty until propagation is reached from other pixels. Otherwise it is assigned a color based on the blending method of Eqn. (3) and is labeled as valid. New search then continues for other black-hole pixels.

Figure 3. Virtual view after forward warping with original two basis views on the top row (the basis views are shown in reduced size to save space).

Figure 4. Complete virtual view after the entire process.

3.5. Using Multiple Views

Even after the search and propagation processes, there may still be "black holes" left when the points cannot be seen in both basis cameras. To address this, the same search and propagation method as in the above is used but with $\overline{p} = \{p_i\}, i \neq 1, 2$. This means that we assume that the pixel may be (for example) occluded in either or both of views and thus both of them are excluded. But we may be able to obtain the information from other views. Since there

is no information for any preference for any of the auxiliary views, a dominant color found from the views is taken to fill the black holes. While it may appear to be computationally expensive to search in multiple images if the number of views n is large, considering that the number of uncovered pixels is relatively small after the previous steps, this search is quite fast in practice.

It should be noted that there is no guarantee that all pixels can be covered by the above procedure. For example, the problem may be caused by a few isolated noisy pixels, or maybe the scene is not covered by all the cameras. A simple linear interpolation can handle the former situation while the latter situation can be alleviated by constraining the free viewpoint range, which is already part of out assumption (i.e., the virtual view is always between two views, and the cameras are strategically positioned).

A complete virtual view obtained by following the entire process is shown in Figure 4.

4. Viewpoint Specification

In this section, we propose an intuitive way for virtual view specification based on only uncalibrated views. Essentially, we provide a viewer with the capability of varying a virtual view gradually between any two chosen views. The virtual view can thus be determined by, for example, conveniently pushing a +/- button (or similar) until the desired viewpoint is shown, similar to controlling color or contrast of a TV picture via a remote control button (similarly, a joystick on remote or a game console can be used to implement the idea).

4.1. Basic Idea

A viewpoint can be specified by a translation vector and a rotation matrix with respect to any given view to determine its position and direction. But it is unrealistic to ask a TV viewer to do this. A practical method is to start with a real view and let the viewer move to a desired viewpoint in reference to that view. This *relative viewpoint moving,* in an interactive manner, is much more convenient for the user. Thus we have designed our system to be able to interpolate continuous virtual views from one view to another. The interpolation can be controlled by a single parameter α. When $\alpha=0$, the basis view 1 is the current view; and with α increasing to 1, the viewpoint changes gradually to another view 2, which is determined by the button pushed. A mockup user interface is illustrated in Figure 5 for an illustration of this idea, where the left-right arrow buttons control the viewpoint change from two underlying basis views, and the result is shown immediately on the screen as visual feedback to the viewer. (In reality, we may display the two basis views on the screen as well.) The up-down arrow buttons can add variability of the views along a path between the two basis views (as explained in detail in the below).

Figure 5. Virtual view specification: A mockup illustrating the main idea. Pushing the arrows would lead to a virtual view that "moves away" from the current view.

4.2. Viewpoint Interpolation: Calibrated Case

We begin with the calibrated case as it is instructive, although our ultimate goal is to deal with the uncalibrated case. Note that in this case, potentially one can use either of the methods described in [1] and [6]. However, as discussed previously, these methods are not practical to use (e.g., it is unlikely that we can ask a TV viewer to input a rotation matrix and a translation vector). We now consider how to use an interface such as that of Fig. 5 to support intuitive virtual view specification. Suppose we have two camera matrices for the two basis views respectively:

$$P_1 = K_1 R_1 [I \mid -C_1], \quad P_2 = K_2 R_2 [I \mid -C_2] \tag{5}$$

Since we will handle uncalibrated case later, we are concerned with only relative relationship between the two views. By applying the following homography transform to each of the projection matrices,

$$P_i' = P_i H$$

where

$$H = H_c H_R, \; H_c = \begin{bmatrix} I & C_1 \\ 0^T & 1 \end{bmatrix}, \; H_R = \begin{bmatrix} R_1^{-1} & 0 \\ 0^T & 1 \end{bmatrix} \tag{6}$$

we convert the cameras to canonical form as:

$$\begin{cases} P_1' = K_1 R_1' [I \mid -C_1'] = K_1 [I \mid 0] \\ P_2' = K_2 R_2' [I \mid -C_2'] \end{cases} \text{ with } \begin{cases} R_2' = R_2 R_1^{-1} \\ C_2' = R_1 (C_2 - C_1) \end{cases} \tag{7}$$

i.e., the first camera's center is the origin, and camera 2 is related to camera 1 by rotation R_2 and translation C_2'.

We can specify the virtual view based on the canonical form. Suppose the camera matrix for the virtual view is:

$$P_0' = K_0' R_0' [I \mid -C_0'] \tag{8}$$

We use α to parameterize the path between basis views 1 and 2. Eq. (8) becomes

$$P_0'(\alpha) = K_0'(\alpha) R_0'(\alpha) [I \mid -C_0'(\alpha)] \tag{9}$$

For the camera intrinsic matrix, the gradual change from view 1 to view 2 may be viewed as camera 1 changing its focus and principal points gradually to those of camera 2 (if the two cameras are identical, then this will not have any effect, as desired). Thus, we can interpolate the intrinsic

matrix and obtain $K_v'(\alpha)$ as:

$$K_0'(\alpha) = (1-\alpha)K_1 + \alpha K_2 \tag{10}$$

For $R_0'(\alpha)$, suppose

$$R_i' = [r_i, s_i, t_i]^T \tag{11}$$

where r_i, s_i and t_i represent the x-axis, y-axis and z-axis, respectively. We construct $R_0'(\alpha) = [r_0(\alpha), s_0(\alpha), t_0(\alpha)]$ as follows:

$$\begin{aligned} t_0(\alpha) &= ((1-\alpha)t_1 + \alpha t_2) / \| (1-\alpha)t_1 + \alpha t_2 \| \\ s' &= (1-\alpha)s_1 + \alpha s_2 \\ r_0(\alpha) &= (s' \times t_0(\alpha)) / \| s' \times t_0(\alpha) \| \\ s_0(\alpha) &= t_0(\alpha) \times r_0(\alpha) \end{aligned} \tag{12}$$

The first step in Eqn. (12) means that we construct the new z-axis as the interpolation of two original z axes. Then we interpolate a temporary y-axis as s'. Note that s' may not be perpendicular to the new z-axis. But with it, we can construct a new x-axis ($r_0(\alpha)$) with the new z-axis and a temporary y-axis. Finally, we construct the new y-axis as the cross product of the new z-axis and x-axis.

Finally, we can construct the new camera center using linear interpolation:

$$C_0'(\alpha) = (1-\alpha)C_1' + \alpha C_2' \tag{13}$$

From Eqn. (13), the new camera center is on the line connecting the two camera centers, resulting in degeneracy for the epipolar constraint and thus we cannot use it for virtual view synthesis (see Fig. 1). While methods exist to handle the degeneracy (e.g., [7]) without using the epipolar constraint, we are interested in keeping the benefits derived from the constraint and thus want to avoid the degeneracy so that the fundamental matrix based method is still applicable. Thus we need to move the path away from the exact line between the two views. This can be achieved by simply increase slightly the y components of the virtual camera center computed from Eqn. (13). In implementation, by increasing/decreasing the y component, we can further achieve the effect of changing the viewpoint perpendicular to the first direction. Suppose that $C_v(\alpha)=[x_v,y_v,z_v]$, we get a new $C_v'(\alpha)$ as

$$C_v'(\alpha) = [x_v, \quad y_v + \gamma, \quad z_v].$$

This entire process is illustrated in Fig. 6, where the

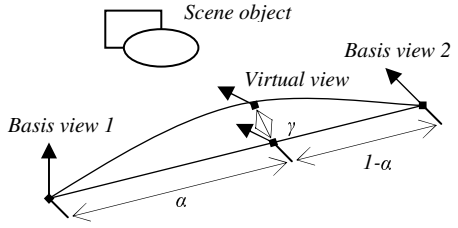

Figure 6. The virtual view as a function of the basis views through two parameter α and γ, which can be controlled by the left-right and up-down arrows of Fig. 5 respectively.

With the interpolated P_0, the corresponding fundamental matrices can be calculated easily (see, for example, [5]) and then used for virtual view synthesis.

4.3. Viewpoint Interpolation: Uncalibrated Case

Now we consider the uncalibrated case, i.e., how we can achieve similar results of Sect. 4.2 from only the fundamental matrices. Given a fundamental matrix F_{12}, the corresponding canonical camera matrices are:

$$P_1 = [I \mid 0], \quad P_2 = [[e']_\times F_{12} + e'v^T \mid \lambda e'] \tag{14}$$

where *e'* is the epipole on image 2 with $F_{12}{}^T e' = 0$, *v* can be any 3-vector, and λ is a non-zero scalar. Note that the reconstructed P_2 is up to a projective transformation. Apparently, a randomly-chosen *v* cannot be expected to give us a reasonable virtual view if the fundamental matrix is based on a P_2 defined by such a *v*. Here we present an approach to obtaining the *P*'s from an *approximately estimated* essential matrix. We first estimate the essential matrix by a simple approximation scheme. The essential matrix has the form:

$$E_{12} = K_2^{-1} F_{12} K_1 \tag{15}$$

For unknown camera matrices *K*, although auto-calibration can recover the focal length at the expense of tedious computation, it is not a practical option for the FTV application (unless the information is obtained at the acquisition stage). As an approximation, we set the parameters of the camera matrix based on the image width *w* and height *h*:

$$f = (w+h)/2, \; p_x = w/2, \; p_y = h/2 \tag{16}$$

So *K* becomes:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

Further, we assume that both cameras have similar configuration and use the same *K* to get the essential matrix E_{12}. An essential matrix can be decomposed into a skew-symmetric matrix and rotation matrix as:

$$E_{12} = [t]_\times R \tag{18}$$

where *R* and *t* can be viewed as the relative rotation and translation matrix of camera 2 relative to 1. Now we have

$$P_1 = K[I \mid 0], \quad P_2 = K[R \mid t] \tag{19}$$

and thus the corresponding fundamental matrices can be recovered (see, for example, [5]). This approach proved to be effective with multiple sets of data that we have experimented with, as will be illustrated in Sect. 5, even if we base on only an estimate in Eqn. (16) without knowing the actual camera internal matrices.

We would like to point out that, although it seems that we are going back to the calibrated case by estimating the essential matrix, the scheme is totally different from *true* full calibration. This is because one cannot expect to use the

approximation of Eqn. (16) for estimating the true rotation and translation that are needed for specifying the virtual as in the calibrated case (e.g., as in [3,6]). However, it is reasonable to use the approximation in the interpolation scheme in Sect. 4.3 (Eqns. (12) and (13)).

5. Experimental Results

We first verified our viewpoint specification method by showing a simulated free viewpoint moving path by using data that is made available on the Internet by the authors of [1] since the cameras are calibrated and the associated path has Euclidian meaning in real world. Fig. 7 shows two paths with viewpoint moving from camera 67 to 74 over a parabola and continuing to camera 80 following a piecewise linear curve (the latter is simply the result of Sect. 4.2 while the parabola illustrates the possibility of using other schemes such as a quadratic function for interpolation between the two views).

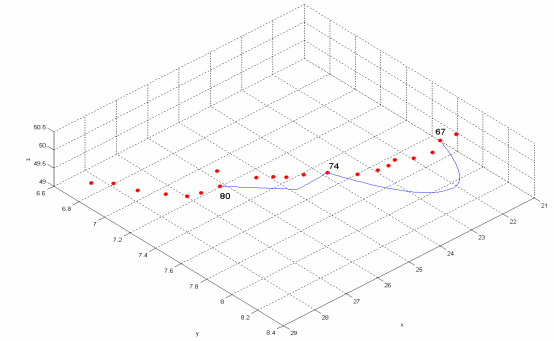

Figure 7. Simulated virtual viewpoint moving paths. The red dots are the given camera positions.

As examples of our experiments, we present results obtained from the following two data sets: the "monkey scene" from [1] and a "snoopy scene" captured by us. For the monkey scene, there are 89 views in this data set and we picked views 67 and 74, which have a proper distance, as basis views. We randomly picked 6 other views as auxiliary views. One hundred virtual views are synthesized with viewpoint moving gradually between the two basis views. Fig. 8 (left) shows the two basis views and three examples of synthesized images. In this test, we use the camera calibration information provided with the data to compute the fundamental matrices, and thus the results are pretty good, illustrating that the remaining procedures (other than the weak calibration stage for computing the fundamental matrices) are performing correctly.

For the "snoopy scene", the camera parameters are not known, and we relied on our proposed algorithms to solve the entire problem from purely the uncalibrated views. The results are shown in Fig. 8 (right). During acquisition, the camera was set to auto-focus and thus the internal matrices may vary in the images. However, with fixed parameters (for both the monkey scene and the snoopy scene), the results are reasonable, indicating that our method in Sect. 4.3 is working even with the assumption of Eqn. (16).

The results in Fig. 8 (right) is slightly worse than the monkey scene (left). For example, view 30 still contains some problematic regions. Among others, we attribute this to the potential inaccuracy in feature-detection-based procedure for estimating the fundamental matrices.

It needs to pointed out that the basis views in Fig. 8 are not very close although they appear to be so. The viewing angle difference between the basis views are in fact roughly 30 degrees. This can be seen more clearly from the background of the views (e.g. looking at the snoopy's nose in relation to the computer keyboard in the background).

6. Conclusion and Discussion

We have presented a new virtual view specification and synthesis approach for the FTV application. Our experimental results using multiple data sets have proved the correctness and the feasibility of the proposed approach. The proposed approach may therefore provide a practical scheme for realizing IBR-based FTV.

In terms of the visual quality of the generated virtual views, our current experiments do not indicate that our results are better than the leading approaches reported in the literature. However, our approach is capable of purely working from uncalibrated views without using any pre-calibration information, rendering it as a viable approach for practical FTV. Further visual quality improvements are expected as we further improve the components, especially the weak-calibration stage.

Future work includes analyzing the impact of the various steps of the proposed schemed on the overall quality of the virtual view. The current approach is assumed to be applied on a frame-by-frame basis, and we need to study the how temporal information in video may impact on some of the stages such as feature detection (e.g., feature tracking could be used) and visual quality related issues (e.g., spatio-temporal artifacts). Also, our development thus far has been focused on feasibility study of the key algorithms, and the time complexity of an integrated system has yet to be studied.

References

[1] A. W. Fitzgibbon, Y. Wexler and A. Zisserman, "Image-based rendering using image-based priors", ICCV 2003: 1176-1183.

[2] C.L. Zitnick and L. Kanade, "A cooperative algorithm for stereo matching and occlusion detection", PAMI 22(7): 675-684, 2000.

[3] L. Zitnick, S.B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation", SIGGRAPH 2004: 600-608.

[4] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient graph-based image segmentation", IJCV 59(2): 167-181, 2004.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, UK, 2000.

[6] K.R. Connor and I.D. Reid, "Novel view specification and synthesis", Proc. British Machine Vision Conf., 2002.

[7] A. Shashua and M. Werman, "Trilinearity of three perspective views and its associated tensor", ICCV 1995: 920-925.

[8] N. Bangchang, T. Fujii, M. Tanimoto, "Experimental system of free viewpoint Television", Proc. IST/ SPIE Symposium on Electronic Imaging, Vol. 5006-66, pp. 554-563, Jan 2003.

[9] M. Tanimoto, T. Fujii, "FTV: Achievements and Challenges", ISO/IEC JTC1/SC29/WG11 M11259, Oct. 2004.

[10] S. Laveau and O. Faugeras, "3-D Scene Representation as a Collection of Images", Proc. of Int. Conf. on Pattern Recognition, Oct. 1994. (INRIA Technical Report 2205).

[11] D. Scharstein, "Stereo vision for view synthesis", Proc. IEEE CVPR, 1996.

[12] Y. Ito and H. Saito, "Free-viewpoint image synthesis from multiple-view images taken with uncalibrated moving cameras", Proc. IEEE ICIP 2005.

[13] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, "Unstructured Lumigraph Rendering", SIGGRAPH 2001.

[14] S. Wurmlin, E. Lamboray, M. Washchbusch, P. Kaufmann, A. Smolic, M. Gross, "Image-space Free-viewpoint Video", Proc. Vision, Modeling, and Visualization, 2005.

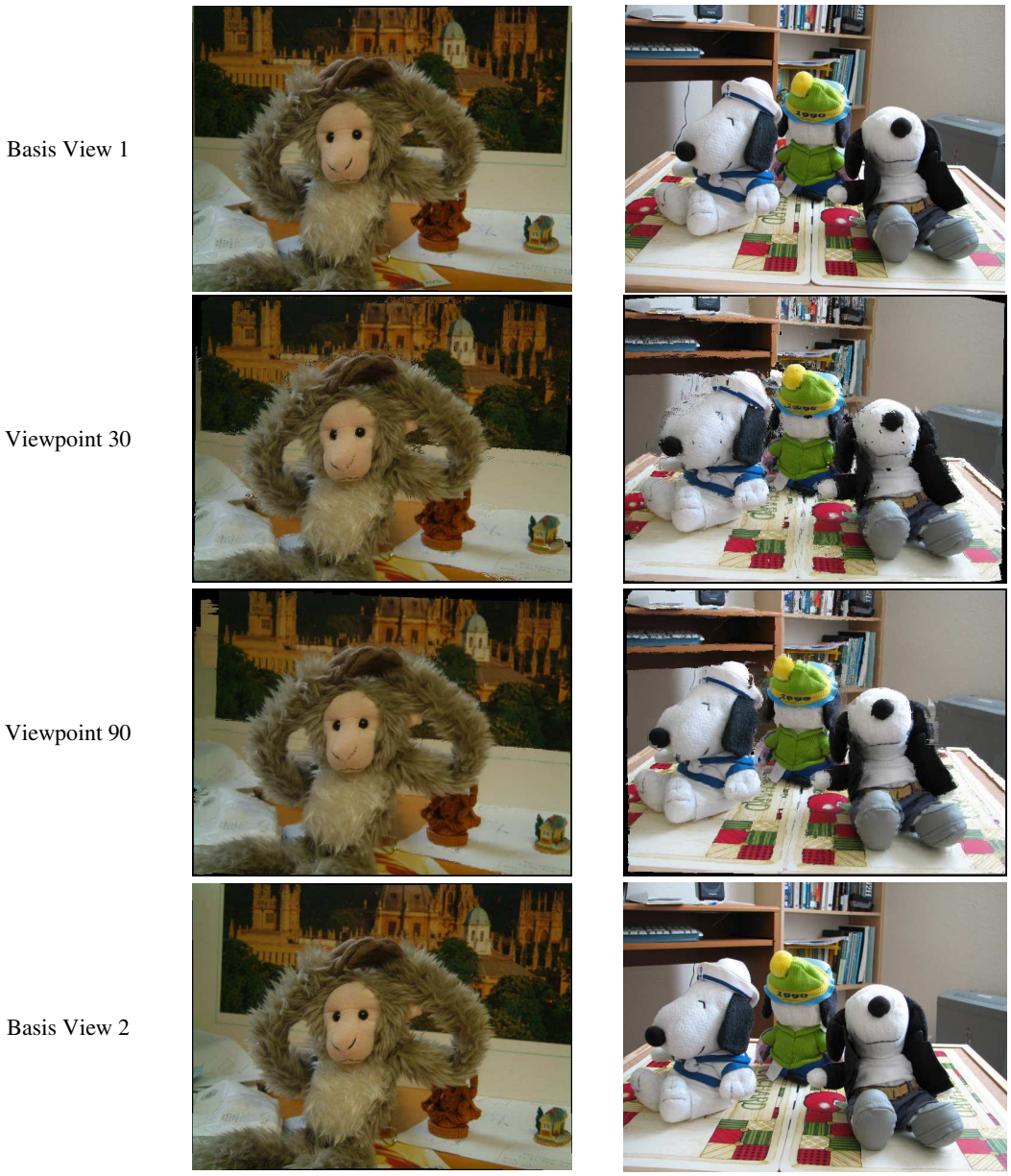

Figure 8. Synthesized and basis views. Left column: monkey scene. Right column: snoopy scene

Proceedings of the Third International Symposium on
3D Data Processing, Visualization, and Transmission (3DPVT'06)
0-7695-2825-2/06 $20.00 © 2006 IEEE