Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Vision and Image Understanding 96 (2004) 345–366
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

# Tri-view morphing

Jiangjian Xiao*, Mubarak Shah

*Computer Vision Lab, School of Electrical Engineering and Computer Science University of Central Florida, Orlando, FL 32816, USA*

Received 1 May 2003; accepted 1 March 2004
Available online 29 July 2004

**Abstract**

This paper presents an efficient image-based approach to navigate a scene based on only three wide-baseline uncalibrated images without the explicit use of a 3D model. After automatically recovering corresponding points between each pair of images, an accurate trifocal plane is extracted from the trifocal tensor of these three images. Next, based on a small number of feature marks using a friendly GUI, the correct dense disparity maps are obtained by using our trinocular-stereo algorithm. Employing the barycentric warping scheme with the computed disparity, we can generate an arbitrary novel view within a triangle spanned by three camera centers. Furthermore, after self-calibration of the cameras, 3D objects can be correctly augmented into the virtual environment synthesized by the tri-view morphing algorithm. Three applications of the tri-view morphing algorithm are demonstrated. The first one is 4D video synthesis, which can be used to fill in the gap between a few sparsely located video cameras to synthetically generate a video from a virtual moving camera. This synthetic camera can be used to view the dynamic scene from a novel view instead of the original static camera views. The second application is multiple view morphing, where we can seamlessly fly through the scene over a 2D space constructed by more than three cameras. The last one is dynamic scene synthesis using three still images, where several rigid objects may move in any orientation or direction. After segmenting three reference frames into several layers, the novel views in the dynamic scene can be generated by applying our algorithm. Finally, the experiments are presented to illustrate that a series of photo-realistic virtual views can be generated to fly through a virtual environment covered by several static cameras.


* Corresponding author.
*E-mail addresses:* jxiao@cs.ucf.edu (J. Xiao), shah@cs.ucf.edu (M. Shah).

1077-3142/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2004.03.014

*Keywords:* Multi-image processing; Image-based rendering; Trifocal geometry; Trinocular-stereo; Augmented reality; 4D video; Dynamic view morphing; Multiple view morphing

## 1. Introduction

The techniques for rendering a novel view from a collection of sample images are called image-based rendering (IBR). They are broadly used to generate virtual cameras to visualize and navigate within virtual environments. Instead of using geometrical primitives as in traditional graphics, these approaches deeply discover the geometrical relationship between multiple sample images based on perspective principles. Moreover, ray-correspondences or point-correspondences are recovered to render a new virtual view by a back-tracing projection or forward warping.

Most IBR methods are based on plenoptic functions, which represent ray properties of a scene. Therefore, in order to record each ray from the scene, these approaches, such as light fields [12], lumigraphs [8], panorama mosaics [21], and visual hulls [14], require a large number of pictures for a static scene. Another limitation of existing approaches is that they require special markers in the images to calibrate cameras [5,6,8,12,14,28]. Some of them even need an explicit 3D model or geometric proxy to trace ray correspondences [5,6,8], and others require an assumption about depth [21].

Another class of approaches for interpolating two reference images is called image morphing and view morphing [25]. These approaches can transform a source view into a target view. Xiao et al. [26] presented a detailed comparison of three major methods in this area: shape blending [1,22], feature-based morphing [3,11], and view morphing [7,10,19]. These approaches use very few images compared to the ray tracing methods discussed above. However, since shape blending and feature-based morphing usually cannot retain strict geometrical properties of the scene, these two methods are mainly used for images of *different* scenes (e.g., morphing of an elephant into a deer). View morphing and view interpolation are based on perspective geometry principles, which can provide more realistic results when blending the images of the *same* scene. Another drawback is that all of these approaches can only be used to fill the gap between two views (i.e., navigation is limited to a straight line between two original views), therefore cannot be used to navigate in a 2D or 3D virtual environment.

In this paper, we follow the view morphing approach and propose a novel technique to synthesize a virtual view in a 2D space. First, based on three wide-baseline uncalibrated images, we automatically recover corresponding feature points between each pair of images and determine the epipolar geometry for each pair. Second, we refine these correspondences and extract the trifocal plane by trifocal tensor computation. Third, employing the user friendly GUI, a small number of feature lines are easily marked. Then, we automatically compute the disparity maps using our trinocular-stereo algorithm. As a result, we generate an arbitrary novel view located within a triangle spanned by three camera centers, and easily navigate through the scene

over a 2D space. After self-calibration of these three cameras, we even can accurately augment 3D objects in virtual scene.

We demonstrate three applications of the tri-view morphing algorithm. The first one is 4D video synthesis, which can be used to fill in the gap between a few sparsely located video cameras to synthetically generate a video from a virtual moving camera. This synthetic camera can be used to view the dynamic scene from a novel view instead of the original static camera views. The second application is multiple view morphing, where we can seamlessly fly through the scene over a 2D space constructed by more than three cameras. The last one is dynamic scene synthesis using three still images, where rigid objects may move in any orientation or direction. After segmenting three reference frames into several layers, the novel views in the dynamic scene can be generated by applying our algorithm.

The paper is organized as follows. Section 2 reviews the previous work related to view morphing algorithms. Section 3 presents an overview of our tri-view morphing algorithm. Section 4 introduces how to automatically determine corresponding feature points to recover the epipolar geometries and the trifocal plane implied in three views. Section 5 describes the trinocular-stereo algorithm to get the correct disparity maps. Section 6 deals with the view blending function, and presents two schemes: blending by barycentric coefficients, and blending by using linear ratios. Section 7 illustrates how to augment 3D objects in the virtual environment synthesized by tri-view morphing algorithm. Section 8 presents three major applications: 4D video synthesis, multiple view morphing, and dynamic tri-view morphing, and illustrates several results obtained by our approach.

## 2. Related work

Seitz and Dyer's [19] static view morphing algorithm consists of four main steps: determining the fundamental matrix, prewarping, morphing, and postwarping. First, eight or more corresponding points were manually selected to determine a fundamental matrix, $F$, by using a linear algorithm. Next, the two original images were warped into a plane parallel to the camera baseline using epipolar geometry. Following this, a user manually specified a set of corresponding line segments and then the Beier–Neely method was used to interpolate correspondences. After linearly interpolating the two parallel views based on the disparity map, a parallel morphing view was obtained. To obtain a realistic final view, a quadrilateral was used to determine the postwarping path for the final homography projection using linear interpolation. However, the postwarping path obtained by linear interpolation may cause shrinking problem as mentioned in [19,26].

Manning and Dyer [13] extended static view morphing to dynamic view morphing. However, in their scenario, the moving objects can only move along a straight line, and their motion is limited to only translation. Xiao et al. [26] relaxed this constraint and allowed an arbitrary motion. They showed that a rigid dynamic scene is equivalent to several static scenes with different epipolar geometries based on relative

motion. They also extended their method to articulated object motion, such as walking and arm gestures, and obtained photo-realistic results.

Avidan and Shashua [2] proposed their work on tri-view synthesis by using trifocal tensor. In their framework, an arbitrary novel view can be generated at any 3D view position based on three *small* baseline images, where the disparity can easily be determined by Lucas–Kanade optical-flow method. It will be almost impossible for Lucas–Kanade method to work for the *wide* baseline images. Pollard et al. [16] determined edge correspondences and used interpolation to generate a new view over trinocular images. However, since they cannot guarantee that the edge correspondences are correct, their disparity map was computed using the conventional edge-scanline algorithm, which is not clean. Therefore, their results contain a lot of artifacts due to some incorrect correspondences. Vedula et al. [24] proposed view interpolation over spatio-temporal domain, where 14–17 fully calibrated cameras (with small baseline) were used on the one side of the actor/actress to capture the events. In their approach, they used voxel coloring, 3D scene flow, and ray-casting algorithm to synthesize the novel view over these original image sequences. They removed the background layer, and only rendered the actor/actress layers. Their results contain some visible artifacts due to the errors in shape estimation, scene flow, etc.

Pollefeys and Van Gool [17] combined 3D reconstruction and IBR to render a new view from a sequence of images. They first determined the relative motion between consecutive images, and then recovered the structure of the scene. Next, employing unstructured light field rendering, they can generate a virtual view by using view-dependent texture. Using this sequence of images (small baseline), they accurately estimated dense surface of the scene, which can efficiently improve the visual effect of their results.

Recently, Zhang et al. [28] proposed to use feature-based morphing with light fields to obtain very realistic 3D morphing. In their approach, a large number of images (hundreds of pictures) were taken for each object using an array of calibrated cameras. Then, several feature polygons were manually determined employing a user interface. Using the corresponding feature polygons, they generated a 4D light field and grouped the corresponding ray bundles for reference images. Finally, a novel view was synthesized using blending and warping functions on reference images.

Our paper overcomes the limitation of previous view morphing methods, which are subject to a linear transformation between source and target views (or objects) and only can generate one visualization trajectory connecting these original frames. In particular, this paper makes the following contributions: First, we show that a virtual environment can be generated for 2D navigation based on only a few (three or four) wide baseline reference images. Second, using the trifocal plane extracted by the trifocal tensor, our morphing procedure can maintain geometrical correctness of synthesized novel views, which even can be correctly augmented with 3D objects. Third, we introduce three novel applications—4D video synthesis, multiple view morphing, and dynamic tri-view morphing, which can be efficiently implemented by using our tri-view morphing algorithm.

## 3. Algorithm overview

The input to tri-view morphing is three wide-baseline uncalibrated reference images as shown in Figs. 1A–C. A series of novel virtual views (Figs. 1D–F) from any arbitrary position in the trifocal plane can be synthesized to navigate through the scene based on the three original images without any knowledge of the scene. Our tri-view morphing algorithm is implemented using the following steps:

First, using a two-stage wide baseline matching algorithm method using edge corners [27], a number of corresponding points are automatically recovered to compute the fundamental matrix and epipolar geometry for each pair of reference images.

Second, a unique trifocal plane $E$ is determined employing their epipoles $e_{ij}(i,j \in \{1,2,3\}$ and $i \neq j)$ as shown in Fig. 2. Then, the three original images $I_1$, $I_2$, and $I_3$ are warped into a plane parallel to the trifocal plane to obtain rectified images $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$.

Third, our feature-based trinocular-stereo algorithm is used to automatically compute the correct disparity map between each pair of rectified images.

Finally, a tri-view blending function is determined according to the viewpoint position. Following the perspective geometrical principles, the morphed image, $\hat{I}_s$, is obtained by combining the blending function with the disparity maps. Then, a 5-point postwarping scheme is used to project the morphed image to a proper final position (Appendix C).

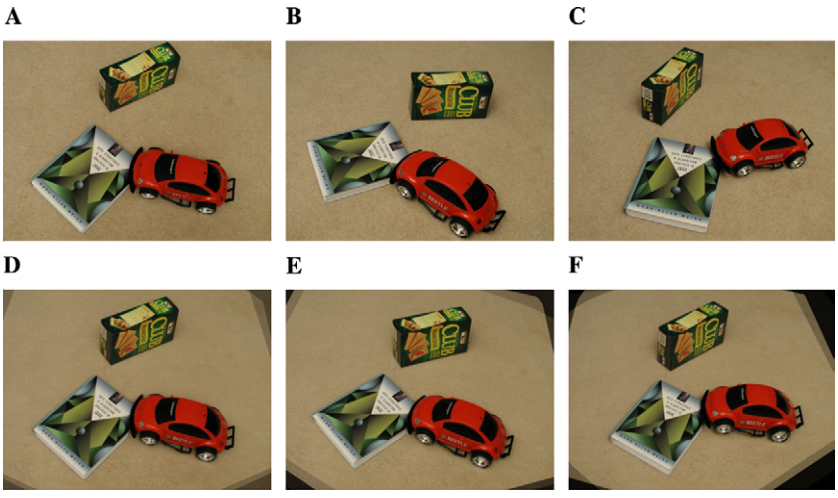


Fig. 1. A typical tri-view morphing scenario. (A–C) Three uncalibrated wide baseline reference images. (D–F) A series of synthesized virtual views.

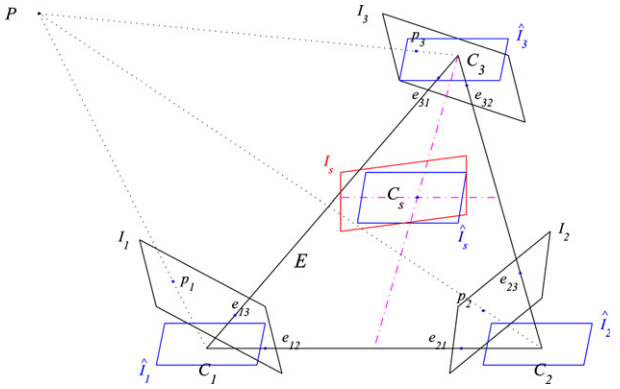


Fig. 2. Tri-view morphing procedure. After automatically determining a focal plane $E$, which is constructed by three camera centers $C_1$, $C_2$, and $C_3$, three original images $I_1$, $I_2$, and $I_3$ are warped into parallel views $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$. The morphing image, $\hat{I}_s$, is blended by using the rectified images with correct disparity maps. The final image $I_s$ at $C_s$ is postwarped from $\hat{I}_s$ by using the 5-point postwarping scheme.

## 4. Determining corresponding points and the trifocal plane

### 4.1. Determining corresponding points

The popular technique for determining correspondences over *small* baseline images is the well-known STK tracker [20], which uses an affine model to effectively compensate the motion over video frames. However, it fails to track or find correspondences if a large rotation or scaling (usually present in *wide* baseline images) between two frames is introduced. Pritchett and Zisserman [18] have also presented an approach to estimate reliable point correspondences based on local planar homographies. Nevertheless, it is very difficult to determine these homographies by parallelogram structures or using motion pyramids in most cases. Mikolajczyk and Schmid [15] proposed the recent work on affine invariant interest point detector using scale space. Since they used multi-scale space to determine feature points, their method worked very well for significant scaling case (homography case). However, their corner descriptors employing the high order derivatives cannot fully recover non-linear rotation transformation.

In this paper, we use our wide baseline matching method to determine corresponding points between each pair of images [27]. By decomposing the affine model into rotation matrix $R(\alpha)$, scaling matrix $S(\kappa)$, and stretch-shearing matrix $E$, the minimal image residue between the windows around two corresponding corners can be computed by a two-stage algorithm.

For each pair of images, first, we determine a large number of corners by edge-corner detector in two images. The edge-corner detector can efficiently detect the corners located at the intersection of multiple edges. Then, a set of reliable corresponding points are found using the two-stage matching algorithm [27]. Fig. 3 shows the corresponding points and several epipolar lines obtained by our algorithm for a pair of images.

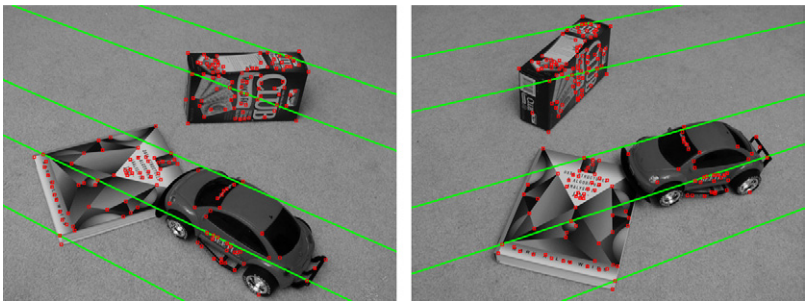

Fig. 3. One hundred and eighty two corresponding points between two car images ((B,C) in Fig. 1) were found. The green lines are epipolar lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

### 4.2. Determining the trifocal plane

To get geometrically correct morphing (Fig. 2), we first need to determine a trifocal plane, $E$, and warp the original images into parallel views. The focal plane is defined by the three camera centers. The epipole $e_{ij} = P_i C_j$, where $P_i$ is the projection matrix of camera $i$, and $C_j$ is the optical center of camera $j$. After computing the corresponding corners $m'_1$ and $m'_2$ between image $I_1$ and $I_2$, and corresponding corners $m''_1$ and $m''_3$ between image $I_2$ and $I_3$, we merge these two groups of corresponding points into one group and obtain new correspondences $m_1$, $m_2$, and $m_3$, where $m_1 \in (m'_1 \cap m''_1)$.

Using Hartley and Zisserman's [9] robust method, the outliers are eliminated and the trifocal tensor $T = [T_1, T_2, T_3]$ is determined. Then, the fundamental matrices $F_{12}$, $F_{23}$, and $F_{31}$ are extracted from the tensor. As a result, we can compute the trifocal plane by using the cross products of epipoles. For each camera, the trifocal plane normal is different. In camera $C_1$, the plane normal $N_{E_1} = e_{12} \times e_{13}$; in camera $C_2$, $N_{E_2} = e_{23} \times e_{21}$; and in camera $C_3$, $N_{E_3} = e_{31} \times e_{32}$.

After the trifocal plane is determined, the three original images are warped into parallel views using the prewarping algorithm (Appendix A) and employing the computed $N_{E_1}$, $N_{E_2}$, and $N_{E_3}$. Also, the epipoles are projected into infinity. As a result of this warping, all epipolar lines in the three rectified images are pairwisely parallel. Next, for each pair of rectified images, corresponding epipolar lines are rotated to make them parallel to scanline directions.

## 5. Trinocular-stereo

In this section, we propose a trinocular-stereo algorithm to compute the disparity map between each rectified image pair, which is based on a pixel-to-pixel dynamic scanline algorithm [4,23]. In our algorithm, the dissimilarity function uses three image intensity differences (SAD: Sum of Absolute Difference) of corresponding pixels in three rectified pairs of images (Eq. (1)).

Fig. 4 shows that the rectified images can be rotated to make scanlines parallel to different epipolar lines. For example, image $\hat{I}_{12}$ is obtained when the $x$ axis of $\hat{I}_1$ is rotated to make it parallel to $e_{12}$, and $\hat{I}_{13}$ is obtained when the $x$ axis of $\hat{I}_1$ is rotated to make it parallel to $e_{13}$. Images $\hat{I}_{ij}$ and $\hat{I}_{ji}$ are called a corresponding rectified pair, since the correspondences of these two images are always located on the same scanline.

Consider the corresponding scanlines $L$ and $R$ in $\hat{I}_{12}$ and $\hat{I}_{21}$, which start from $L_s$ and $R_s$, and end at $L_e$ and $R_e$, respectively. If the two pixels $x_{12}$ and $x_{21}$ match, the corresponding pixel, $x_3$, in original image $I_3$ is given as the intersection of the two epipolar lines:

$$x_3 = (F_{31}x_1) \times (F_{32}x_2),$$

where $x_1$ are the coordinates of the pixel $x_{12}$ in original image $I_1$, and $x_2$ is the coordinates of the pixel $x_{21}$ in original image $I_2$. Let $x_{13}$ be the projection of $x_1$ on $\hat{I}_{13}$, $x_{23}$ be the projection of $x_2$ on $\hat{I}_{23}$, and $x_{31}$ and $x_{13}$ be the projections of $x_3$ on $\hat{I}_{31}$ and $\hat{I}_{13}$, respectively.

The dissimilarity function $d(x_{12},x_{21})$ is computed by using three SADs over $3 \times 3$ window $N$.

$$d(x_{12},x_{21}) = \sum_N |\hat{I}_{12}(x_{12}) - \hat{I}_{21}(x_{21})| + \sum_N |\hat{I}_{23}(x_{23}) - \hat{I}_{32}(x_{32})| + \sum_N |\hat{I}_{31}(x_{31}) - \hat{I}_{13}(x_{13})| \quad (1)$$

Next, we use a pixel-to-pixel dynamic-scanline algorithm which uses an inter-scanline penalty to compute a dense disparity map for each corresponding rectified pair.

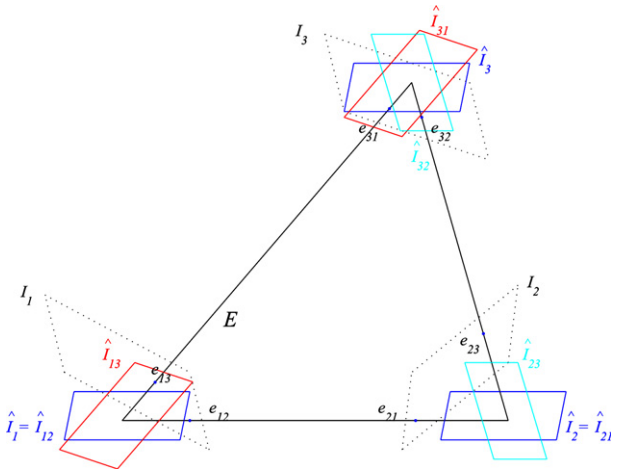


Fig. 4. The $x$ axis of rectified image $\hat{I}_i$ can be rotated to make it parallel to different epipolar directions. $\hat{I}_{12}$ and $\hat{I}_{21}$, $\hat{I}_{23}$ and $\hat{I}_{32}$, and $\hat{I}_{31}$ and $\hat{I}_{13}$ are three corresponding rectified pairs.

This stereo algorithm is reliable for the images with small occlusions and diffuse reflections. On the other hand, the stereo algorithm is not robust enough to handle some areas with large occlusion or specular reflection. Therefore, we have developed a user interface (GUI) to get some additional features to improve the disparity maps. This tool is based on trifocal geometry, which guarantees that the three corresponding feature lines match simultaneously. Once a user clicks a point in the first window, two epipolar lines will appear in the second and third window to guide the user to get the correct corresponding points. Fig. 5 shows how to find three corresponding feature points by only two clicks.

After marking the feature lines, there may be several corresponding feature pixels at scanlines $L$ and $R$ with certain order, such as $(L_s, L_1, \ldots, L_i, \ldots, L_e)$ and $(R_s, R_1, \ldots, R_i, \ldots, R_e)$. For each corresponding marked interval, $L_iL_{i+1}$ and $R_iR_{i+1}$, we apply the trinocular-stereo algorithm to compute the dense disparity inside this interval. Next, we link these intervals together and get the complete disparity map for these scanlines.

Using this tool, the geometrically correct features can be obtained easily. As a result, the dense disparities in these areas with specular reflections or occlusions

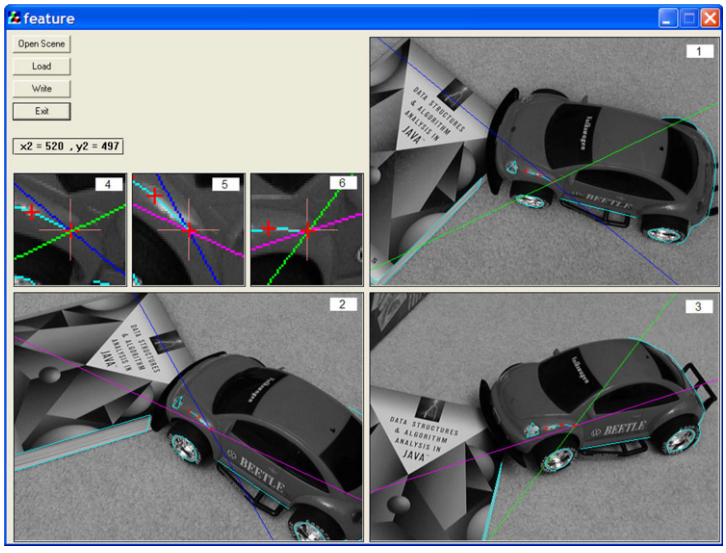


Fig. 5. GUI to mark feature lines for car frames. (1–3) Three reference image windows. (4–6) Zoomed windows for current corresponding feature points. The cyan lines are feature lines. The current feature points are located at the intersection of two epipolar lines with red '+.' The magenta, blue, and green lines are epipolar lines. After clicking in the first window, the UI will give two epipolar lines (blue and green) in this window, and one corresponding epipolar line (blue) in window (2), and one (green) in window (3). Then, the magenta lines will show the two corresponding points (the intersections with blue and green lines) in window (2,3). The user can move the mouse to drag the magenta lines and easily find these two corresponding feature points simultaneously with the second click. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

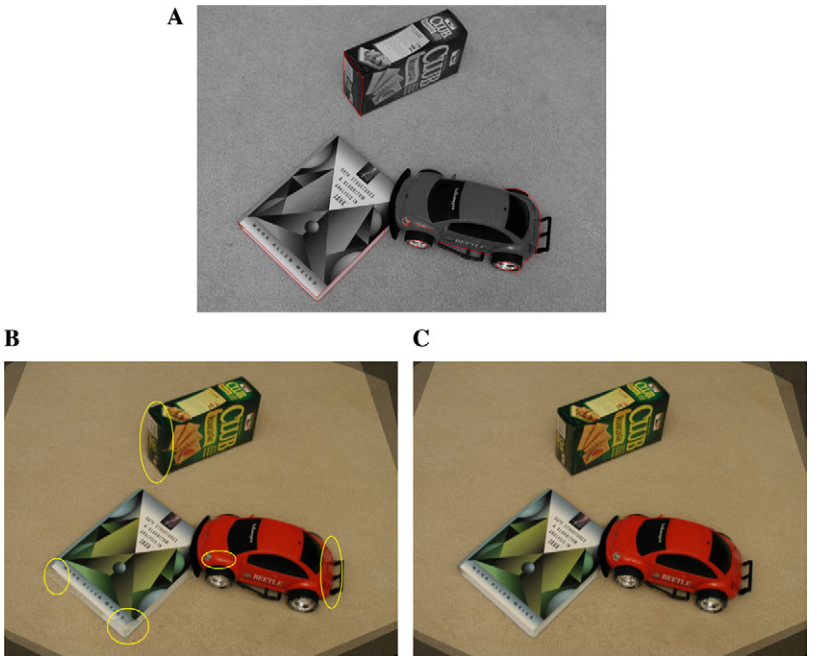


Fig. 6. Comparison of the morphing results using two different disparity maps. (B) The result obtained without using feature lines. *Note.* There are some sever artifacts at the left most edge of the box, headlight of the car, book boundaries (yellow circles), etc. (C) The result obtained with feature lines (red lines in (A)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

can be correctly computed and blended. Figs. 6(B–C) shows a comparison of the results with and without the GUI. In car images, 8 feature lines are marked on the area with occlusions or specular reflection, which required about less than ten minutes of manual work by employing the GUI. For other image sets in this paper, with the help of GUI, 10–20 feature lines are marked within 10–20 min, and the disparity error is dramatically reduced.

## 6. View blending function

The view blending function determines the contribution of each reference image to the morphed images. In traditional image morphing, the blending function works on the original images such as $I_1$ and $I_2$. Usually, it cannot maintain geometric properties as strictly as view morphing, which interpolates the rectified images, $\hat{I}_1$ and $\hat{I}_2$, according to perspective geometry principles [19]. Following this direction, we prove in Appendix B that any linear combination of three parallel views satisfies the perspective geometry property.

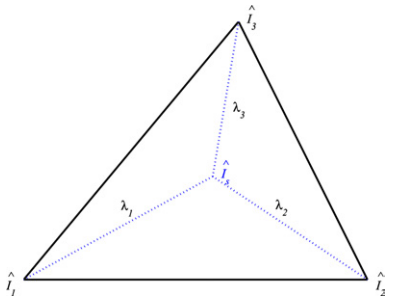


Fig. 7. Blending by barycentric coefficients $\lambda_1$, $\lambda_2$, and $\lambda_3$. $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$ are rectified images, $\hat{I}_s$ is the desired image morphed using these three images.

Let $W_{ij}$ be the warping function between images $\hat{I}_i$ and $\hat{I}_j$, which specifies the correspondences between $\hat{I}_i$ and $\hat{I}_j$.

$$W_{ij} = \hat{p}_i + d_{ij}, \tag{2}$$

where $d_{ij}$ can be obtained using the dense disparity map in Section 5 when $i \neq j$, and $d_{ij} = 0$ when $i = j$. Next, we create a new warping function $B_i$ to warp each of the images $\hat{I}_i$ to $\hat{I}_s$ and blend them together.

$$B_i = \sum_{i=1}^{3} \lambda_i W_{ij}, \tag{3}$$

$$\hat{I}_s = \sum_{i=1}^{3} \lambda_i B_i(\hat{I}_i), \tag{4}$$

where $\lambda_i B_i(\hat{I}_i)$ represents the warped image $\hat{I}_i$ into $\hat{I}_s$ with opacity value $\lambda_i$. It is easily verified that Eq. (4) has the linear blending property $\hat{p}_s = \sum_{i=1}^{3} \lambda_i \hat{p}_i$ by substitution of Eqs. (2) and (3).

Based on this result, we present a scheme to blend three rectified images, which is based on image $\hat{I}_s$'s barycentric coordinates $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, subject to $\lambda_i \geqslant 0$ and $\sum_{i=1}^{3} \lambda_i = 1$ (Fig. 7). Along each edge of the triangle, one of the three coordinates (corresponding to the opposite vertex) is zero. This property is very useful for continuous interpolation across the edges of a triangulation in multiple view morphing, such as Fig. 11. After blending the rectified image pairs, we reproject the morphing images to the final position by 5-point postwarping algorithm (Appendix C) and obtain the final novel views as shown in Fig. 1.

## 7. Augmenting 3D objects

In order to augment 3D objects in the virtual morphing environment, first we self-calibrate three cameras, and obtain the intrinsic and external parameters of these

cameras. Then, for any novel view, the new camera matrix can be interpolated using blending coefficient $\lambda$. Based on this camera model, the 3D objects can be correctly rendered and augmented into the scene during the scene navigation, which is a prerequisite for the interaction with the synthesized environment.

From the recovered trifocal tensor, $T$, the projective camera matrices $\Pi_{p_1}$, $\Pi_{p_2}$, and $\Pi_{p_3}$, can be extracted as follows:

$$\begin{aligned}
\Pi_{p_1} &= [\mathrm{I}|0], \\
\Pi_{p_2} &= [[\mathrm{T}_1, \mathrm{T}_2, \mathrm{T}_3]e_{31}|e_{21}], \\
\Pi_{p_3} &= [(e_{31}e_{31}^T - \mathrm{I})[\mathrm{T}_1^T, \mathrm{T}_2^T, \mathrm{T}_3^T]e_{21}|e_{31}].
\end{aligned}$$

Since we used one digital camera to capture these images, we can safely assume that only the focal length in the camera's intrinsic parameters was changed during taking the pictures. Then, we employ the self-calibration method [9] to recover a 3D homography $H$ and obtain a metric 3D construction of the sparse corresponding points, where the metric camera matrix $\Pi_{m_i} = \Pi_{p_i}H$. The metric camera matrix $\Pi_{m_i}$ also can be represented as:

$$\Pi_{m_i} = K_i[R_i|R_it_i] = K_i[R_z(\phi_i)R_y(\theta_i)R_x(\psi_i)|R_it_i],$$

where $K_i$ represents intrinsic parameters of camera $\Pi_{m_i}$, $R_i$ and $t_i$ are the rotation and translation components of the camera's external parameters. Each of rotation matrix, $R_i$, can be decomposed into three components $R_z(\phi_i)$, $R_y(\theta_i)$, and $R_x(\psi_i)$.

Therefore, the parameters of new virtual camera can be easily linearly interpolated by multiplying the coefficient $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, respectively.

$$\begin{aligned}
K_s &= \lambda_1K_1 + \lambda_2K_2 + \lambda_3K_3, \\
t_s &= \lambda_1t_1 + \lambda_2t_2 + \lambda_3t_3, \\
\phi_s &= \lambda_1\phi_1 + \lambda_2\phi_2 + \lambda_3\phi_3, \\
\theta_s &= \lambda_1\theta_1 + \lambda_2\theta_2 + \lambda_3\theta_3, \\
\psi_s &= \lambda_1\psi_1 + \lambda_2\psi_2 + \lambda_3\psi_3.
\end{aligned}$$

The new camera matrix is given by

$$\Pi_{m_s} = K_s[R_z(\phi_s)R_y(\theta_s)R_x(\psi_s)|R_st_s].$$

This new camera model is guaranteed to move on the trifocal plane due to the linear interpolation of the translation components of the cameras. Based on this new camera matrix $\Pi_{m_s}$, we use OpenGL to render the 3D object and augment the object into the view generated from the same viewpoint. In Fig. 8, we augment a 3D "Bunny" model on the top of the book. To generate the shadow, we simply recover a plane of book using sparse 3D points reconstructed from self-calibration method. Then, after assuming an approximate light source, the shadow is generated and projected on this plane and "Bunny" model by multiple pass rendering.

Fig. 8. Augmenting 3D object in tri-view morphing. The first row shows three original images with the augmented object ("Bunny" model from Stanford University). The 3D model is accurately augmented on the top of the book. The second row shows that a series of synthesized virtual views with the augmented object can maintain geometric correctness in the morphing procedure. We also generate the shadow of model and cast it on the book by assuming a light source. The bottom row shows the detail of one synthetic image.

## 8. Applications and results

In this section, we demonstrate three applications of tri-view morphing: 4D video synthesis, multiple view morphing, and dynamic tri-view morphing, and illustrate examples for each application. In our experiments, all of the static reference images were captured by a hand held Olympus digital camera C-3000.

### *8.1. 4D video synthesis*

Our tri-view morphing algorithm can be used for either static scenes or dynamic scenes with rigid objects (Section 8.3). If the scene contains a non-rigid object with a large deformation, it is very difficult to segment the object into several rigid parts as

discussed in [26]. To navigate through a real dynamic scene, we can set up several static uncalibrated video cameras to capture the scene over time. For each time, $T_i$, using the tri-view morphing algorithm, we can morph multiple frames and navigate through the virtual scene. Furthermore, we can generate a moving camera to navigate through a 4D space (3D Euclidean space + 1D time space), which is called 4D video synthesis as shown in Fig. 9.

In our 4D video synthesis, we only use three uncalibrated wide baseline video cameras to acquire original image sequences, and blend the foreground (moving objects) and background simultaneously during the view interpolation. Our approach can be implemented in three steps. First, the video frames amongst the three cameras are synchronized. For each time, $T_i$, we get three images taken at the same time. Second, applying the tri-view morphing algorithm, we compute dense disparity between each pair of images and generate an arbitrary novel view within a triangle spanned by three camera centers. Third, if the trajectory for the motion of the virtual camera is specified, we even can link the frames over time $T_i$ and generate a moving video camera instead of the original static video cameras.

A few images of the synthetic video for one specified trajectory are shown in the last two rows of Fig. 10. We placed three video cameras at three different locations and in different orientations to record a car collision event. The car was controlled by a remote controller to impact the box. For each video camera, we obtained 26 frames. The first frame of each camera was used to determine the trifocal plane using the algorithm proposed in this paper. Next, for each time $T_i$, disparity maps were computed between rectified images using our trinocular stereo algorithm. After an arbitrary trajectory for a virtual moving video camera is specified, a novel dynamic video is synthesized. A few images of video for one specified trajectory are shown in the second row of Fig. 10.

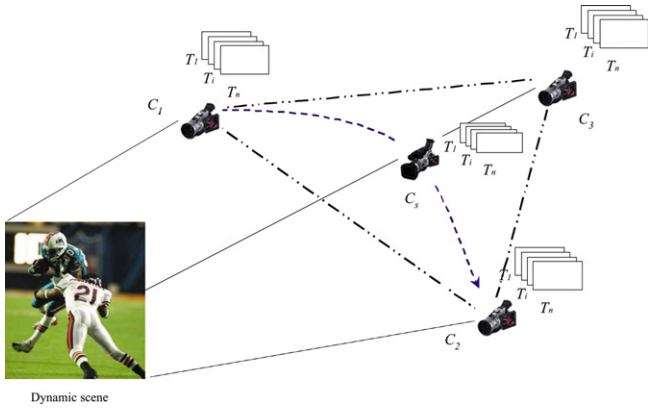


Fig. 9. 4D video synthesis. A dynamic scene is captured by three video cameras. For each time $T_i$, tri-view morphing can be used to generate a virtual video camera $C_s$ to navigate through the scene.

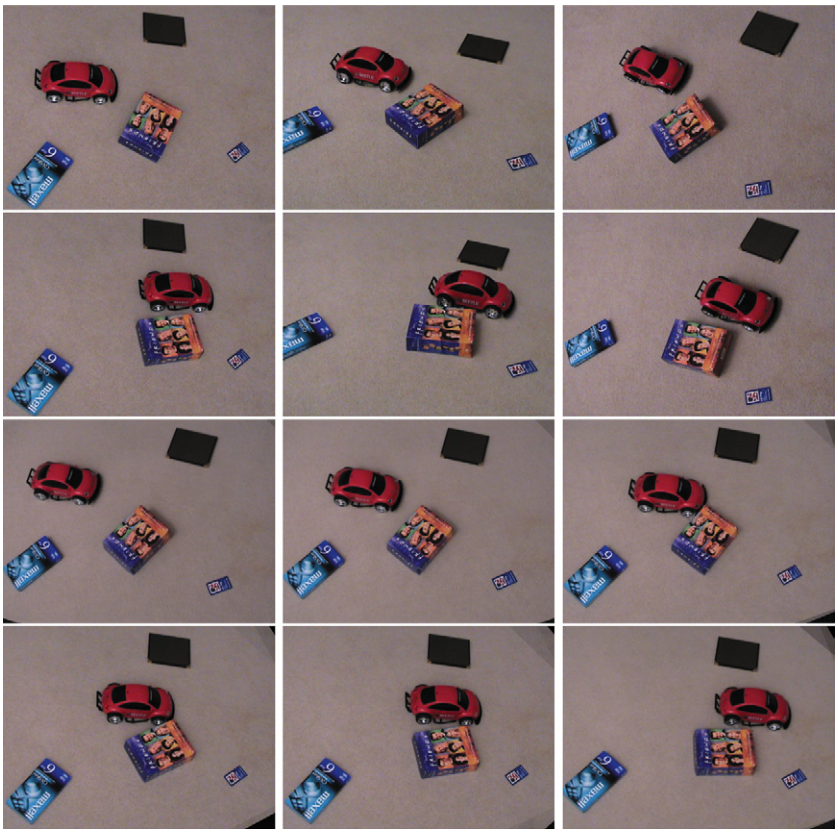

Fig. 10. 4D video synthesis for car collision. The first row shows the images captured at time $T_1$ by three video cameras. The second row shows the images captured at $T_2$ by these video cameras. All of the original cameras were fixed during the video capture. The last two rows show images synthesized by a virtual moving video camera to navigate through the dynamic event of a car collision.

### 8.2. Multiple view morphing using triangle tessellation

If a scene is captured by more than three cameras (the cameras do not necessarily have to lie in the same plane), we can use triangle tessellation to group each triple of neighboring cameras and generate a new view to seamlessly navigate through the scene. Fig. 11 shows a triangle tessellation for four view morphing. The four cameras are grouped into two triples ($C_1$, $C_2$, $C_3$) and ($C_4$, $C_3$, $C_2$). In Section 6, we introduced barycentric blending schemes to blend a new view, which can make a seamless connection over the triangle boundaries.

Fig. 12 shows the morphing results using four images of a skateboarding doll. These four images were acquired by a hand held digital camera without any camera calibration. The surface of the doll is quite complicated. It is very difficult to model it using a 3D textured model employing only these four images. We tessellated the four

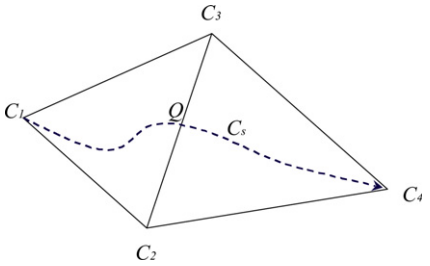


Fig. 11. Four view morphing. A virtual camera $C_s$ can seamlessly navigate the scene covered by four cameras over two triangles.

Fig. 12. Four view morphing of a doll. The first row shows four original uncalibrated images taken by a hand held camera. The second row shows a series of synthesized virtual views.

views into two triangles as shown in Fig. 11, and generated novel views using the view-independent blending scheme. In this case, a novel camera generates a continuous trajectory, which passes through the triangle boundaries, resulting in a seamless navigation.

### 8.3. Dynamic tri-view morphing

Our dynamic tri-view morphing is focused on a dynamic scene containing only rigid moving objects, which may rotate or translate. Xiao et al. [26] have shown that novel views can be generated for such dynamic scenes by using a separate fundamental matrix for each rigid object (layer) based on the relative motion between two

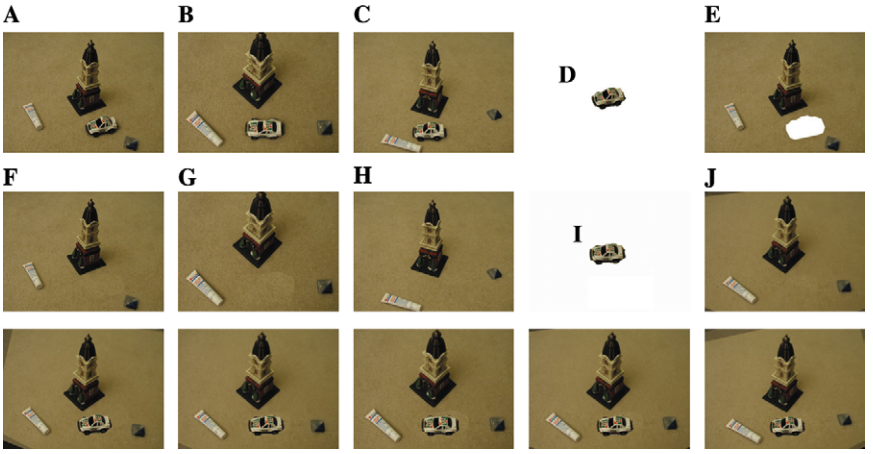


Fig. 13. A typical tri-view morphing scenario. (A–C) Three uncalibrated wide-baseline reference images. (D,E) Segmentation results of the first frame. (F–H) The background images corresponding to the three original views after filling in the gaps occupied by the car. (I,J) Morphing results of the car and background layers, respectively. The last row shows the virtual views obtained after compositing the car and background layers.

views. In this paper, we extend this idea to dynamic tri-view morphing. Our algorithm consists of three main steps: segmentation, tri-view morphing for each layer and multiple layer blending.

Fig. 13 shows a dynamic scene with a moving car. Three reference images were taken at different times and at different positions and orientations while the car was moving. We put an eye-drops tube in the scene to introduce a transparent material. Since the moving object (car) and the background (the rest of the image) have different epipolar geometries, we segment the scene into two layers: background (Fig. 13E) and car (Fig. 13D). Next, we compute a homography to fill in the hole in Fig. 13E using the pixels from the other two images, in order to obtain the background image (Fig. 13F) for the first frame. Then, the background (Fig. 13I) and car (Fig. 13J) are blended separately by the tri-view morphing algorithm. Finally, these two images are composited together and the final results are obtained as shown in the last row of Fig. 13. As a result, the car is moving along its own trajectory during the scene navigation.

*Note.* The true strength of the proposed approach can only be realized by looking at a video sequence. The high resolution video sequences of the above experiments are available on our web site: http://www.cs.ucf.edu/~vision/projects/triview/.

## 9. Conclusion and discussion

Using our proposed image-based approach, we can interactively navigate through a scene based on only three wide-baseline uncalibrated images without

the explicit use of a 3D model. Based on the decomposition of the affine matrix, a large number of corresponding points among the original images are automatically recovered to determine an accurate trifocal geometric relationship between three reference images. The disparity maps are computed to construct the blending functions using our feature-based trinocular-stereo algorithm. An arbitrary novel view located within a triangle on the trifocal plane is blended to obtain a photo-realistic image, which can be correctly augmented by 3D objects after camera self-calibration.

We demonstrated three applications of tri-view morphing: 4D video synthesis, multiple view morphing, and dynamic view morphing. All of the applications are useful for filling in gaps in multiple images, movie-making, etc. Our results successfully show that even using few images (3–4), we can generate photo-realistic images to navigate the virtual environment, where the virtual objects can interact with the 3D scene.

One limitation of our work is that our navigation currently is restricted in a 2D space constructed by three images. Even though an arbitrary novel view at any position can be synthesized from two or three images as claimed by Avidan [2], it usually cannot preserve the good-quality rendering results due to missing texture from the new viewpoint. Hence, the novel view along the baseline of two images or along the tri-focal plane of three images can avoid this problem and can provide the higher quality results. In order to navigate a 3D volume and obtain a realistic visual effect, four images captured by the non-coplanar cameras are minimal requirements. In the future, we will investigate this area and extend our method to 3D volume navigation. Also, if the strong specular reflections are present in the images, it is very hard to determine disparities only by the stereo algorithm without feature marks. How to remove the specular reflection and obtain stable disparity is another direction worth researching.

## Appendix A. Prewarping

The purpose of the prewarping procedure is to warp the three reference images into the trifocal plane and to obtain rectified images, $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$, which are used to compute dense disparity maps. This procedure is implemented in three steps.

Note that $N_{E_1} = e_{12} \times e_{13}$ in camera $C_1$, $N_{E_2} = e_{23} \times e_{21}$ in camera $C_2$, and $N_{E_3} = e_{31} \times e_{32}$ in camera $C_3$.

First, we rotate the original images to a plane parallel to the trifocal plane. Let $R_1$, $R_2$, and $R_3$ be the rotation matrices for images $I_1$, $I_2$, and $I_3$, respectively.

$$R_1^{-1} = \begin{bmatrix} r_{11}^T \\ r_{12}^T \\ r_{13}^T \end{bmatrix}, \quad R_2^{-1} = \begin{bmatrix} r_{21}^T \\ r_{22}^T \\ r_{23}^T \end{bmatrix}, \quad R_3^{-1} = \begin{bmatrix} r_{31}^T \\ r_{32}^T \\ r_{33}^T \end{bmatrix},$$

where $r_{11} = \frac{e_{12}}{|e_{12}|}$, $r_{13} = \frac{N_{E_1}}{|N_{E_1}|}$, $r_{12} = r_{11} \times r_{13}$, $r_{21} = \frac{-e_{21}}{|e_{21}|}$, $r_{23} = \frac{-N_{E_2}}{|N_{E_2}|}$, $r_{22} = r_{21} \times r_{23}$, $r_{31} = \frac{e_{31}}{|e_{31}|}$, $r_{33} = \frac{-N_{E_3}}{|N_{E_3}|}$, $r_{32} = r_{31} \times r_{33}$. After this step, all of the epipoles are projected

into infinity, and the epipolar lines passing through $e_{12}$ and $e_{21}$ are rotated to horizontal. Now the new $F_{12}$ can be represented as:

$$\tilde{F}_{12} = R_2 F_{12} R_1^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & b & c \end{bmatrix}.$$

The image $\hat{I}_2$ is adjusted by vertical scaling and translation by a matrix $T$ [19]. The prewarping homographies for images $I_1$, $I_2$ are $H_1 = R_1$, $H_2 = TR_2^{-1}$, respectively. Next, we rotate $\hat{I}_3$ by an angle $\phi$ and adjust the image by a scaling and shearing matrix $Q$, which guarantees that the sum of the three internal angles between projected epipoles equals 180°. The images $\hat{I}_1$ and $\hat{I}_2$ are equivalent to the corresponding rectified pair $\hat{I}_{12}$ and $\hat{I}_{21}$.

Similarly, we can rotate images to make different epipolar lines (passing $e_{ij}$ and $e_{ji}$) horizontal and get the other two corresponding rectified pairs, $\hat{I}_{23}$ and $\hat{I}_{32}$, $\hat{I}_{31}$, and $\hat{I}_{13}$.

## Appendix B. Linear blending

In this section, we will prove that any linear combination of three parallel views satisfies the perspective geometry property. Suppose that the trifocal plane is located at $Z = 0$, and the camera center $C_i = [C_{ix}\ C_{iy}\ 0]^T$. The projection matrices for the parallel view $C_i$ can be represented as:

$$\Pi_i = \begin{bmatrix} f_i & 0 & 0 & -f_i C_{ix} \\ 0 & f_i & 0 & -f_i C_{iy} \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

where $f_i$ is focal length of $C_i$. The new point $\hat{p}_s$ can be obtained by linearly blending the three corresponding points $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$, which are, respectively, the projections of a 3D point $P = [X\ Y\ Z\ 1]^T$ on images $\hat{I}_1$, $\hat{I}_2$, and $\hat{I}_3$.

$$\hat{p}_s = \lambda_1 \hat{p}_1 + \lambda_2 \hat{p}_2 + \lambda_3 \hat{p}_3 = \frac{1}{Z}(\lambda_1 \Pi_1 + \lambda_2 \Pi_2 + \lambda_3 \Pi_3) P = \frac{1}{Z} \Pi_s P,$$

where $\Pi_s$ is the linear interpolation of $\Pi_1$, $\Pi_2$, and $\Pi_3$, its focal length $f_s = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$, and the camera center $C_s = \lambda_1 C_1 + \lambda_2 C_2 + \lambda_3 C_3$. Therefore, any linear combination of three parallel views satisfies the perspective geometry property when $\sum_{i=1}^{3} \lambda_i = 1$.

## Appendix C. Postwarping

The postwarping procedure deals with the reprojection of the morphing results into a proper final position. In the original view morphing algorithm, this step was implemented by manually determining the final position of four control points or performing linear interpolation (which usually causes shrinking [19,26]).

If the cameras are self-calibrated using the computed correspondences and metric sparse 3D points are recovered (the case for augmenting 3D objects), the position of these points in the final view can be computed by transforming 3D points using the new interpolated camera matrix. Then, the reprojection can be implemented by simply finding a homography between the morphed image and the final view.

For the uncalibrated camera case, we can use the least distortion method [1,26] to approximate the final postwarping shape position to obtain smooth views transformation. First, we automatically select 5 corresponding points such that one is close to the centroid, and the other four points are located in four different directions near the image boundary in original images $I_1$, $I_2$, and $I_3$ as shown in Fig. 14. Then, based on the assumption that the centroid of the scene $p_{cs}$ moves following the linear combination of $p_{c_1}$, $p_{c_2}$, and $p_{c_3}$,

$$p_{c_s} = \sum_{i=1}^{3} \lambda_i p_{c_i},$$

where $p_{c_i}$ is the center point position in image $I_i$ (Fig. 14). In the 5-point scheme, the triangulation $T$ includes 4 triangles $t_i$ ($i = 1, \ldots, 4$). The intermediate postwarping position (blue lines in Fig. 14D) can be obtained by minimizing Eq. (C.1) by the least squares method.

$$\varepsilon_V = \sum_{t_i \in T} ||A_{\{t_i\}} - B_{\{t_i\}}||^2, \tag{C.1}$$

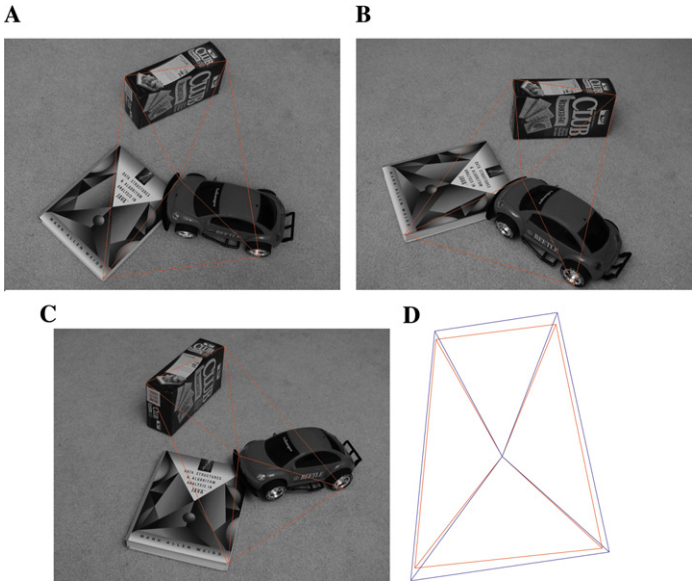


Fig. 14. Five-point postwarping. Five corresponding points are selected to construct four triangles in reference images (A–C). (D) Compares an intermediate shape generated by the least distortion method (blue) with linear method (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

where $A_{\{t_i\}}$ is an ideal shape of triangle $t_i$ obtained by the least distortion method over three images, $B_{\{t_i\}}$ is an actual shape of the triangle $t_i$, and $V$ is a set of the five points.

## References


[1] M. Alexa, D. Cohen-Or, D. Levin. As-rigid-as-possible shape interpolation, in: Proc. ACM SIGGRAPH 2000, 2000, pp. 157–164.
[2] S. Avidan, A. Shashua, Novel view synthesis by cascading trilinear tensors, IEEE Trans. Visual. Comput. Graph. (1998) 293–306.
[3] T. Beier, S. Neely, Feature-based image matamorphosis, in: Proc. ACM SIGGRAPH 1992, 1992, pp. 35–42.
[4] S. Birchfield, C. Tomasi, Depth discontinuities by pixel-to-pixel stereo, Int. J. Comput. Vis. 35 (1999) 269–293.
[5] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, Unstructured lumigraph rendering. in: Proc. ACM SIGGRAPH 2001, 2001, pp. 425–432.
[6] W. Chen, J. Bouguet, M. Chu, R. Grzeszczuk, Light field mapping: efficient representation and hardware rendering of surface light fields, in: Proc. ACM SIGGRAPH 2002, 2002, pp. 447–456.
[7] S. Chen, L. William, View interpolation for image synthesis, in: Proc. ACM SIGGRAPH 1993, 1993, pp. 279–288.
[8] S. Gortler, R. Grzeszczuk, R. Szeliski, M. Cohen. The lumigraph, in: Proc. ACM SIGGRAPH 1996, 1996, pp. 43–52.
[9] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.
[10] H. Huang, S. Nain, Y. Hung, T. Cheng, Disparity-based view morphing a new technique for image-based rendering, Proceedings of the ACM symposium on Virtual reality software and technology, 1998.
[11] S. Lee, G. Wolberg, S. Shin, Polymorph: morphing among mutltiple images, IEEE Comput. Graph. Appl. 18 (1998) 60–72.
[12] M. Levoy, P. Hanrahan, Light field rendering. in: Proc. ACM SIGGRAPH, 1996, pp. 31–42.
[13] R. Manning, C. Dyer. Interpolating view and scene motion by dynamic view morphing. in: Proc. CVPR, 1999, pp. 388–394.
[14] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, L. McMillan. image-based 3D photography using opacity hulls. in: Proc. ACM SIGGRAPH 2002, 2002, pp. 427–437.
[15] K. Mikolajczyk, C. Schmid, An affine invariant interest point detector, in: Proc. ECCV, 2002.
[16] S. Pollard, M. Pilu, S. Hayes, A. Lorusso, View synthesis by edge matching and transfer, IEEE Workhop Appl. Comput. Vis. (1998).
[17] M. Pollefeys, L. Van Gool, Visual modeling: from images to images, J. Visual. Comput. Animat. 13 (2002) 199–209.
[18] P. Pritchett, A. Zisserman, Wide baseline stereo matching, in: Proc. ICCV, 1998, pp. 754–760.
[19] S. Seitz, C. Dyer. View morphing. in: Proc. ACM SIGGRAPH 1996, 1996, pp. 21–30.
[20] J. Shi, C. Tomasi, Good features to track, in: Proc. CVPR'94.
[21] H. Shum, L. He, Rendering with concentric mosaics. in: Proc. ACM SIGGRAPH 1999, 1999, pp. 299–306.
[22] A. Tal, G. Elber, Image morphing with feature preserving texture, in: Proc. EUROGRAPHICS'99, 1999, pp. 339–348.
[23] G. Van Meerbergen, M. Vergauwen, M. Pollefeys, L. Van Gool, A hierarchical symmetric stereo algorithm using dynamic programming, Int. J. Comput. Vis. 47 (2002) 275–285.
[24] S. Vedula, S. Baker, T. Kanade, Spatio-Temporal View Interpolation Eurographics Workshop on Rendering, 2002.
[25] G. Wolberg, Image morphing: a survey, Vis. Comput. 1 (1998) 360–372.

[26] J. Xiao, C. Rao, M. Shah. View interpolation for dynamic scenes. in: Proc. EUROGRAPHICS'02, 2002, pp. 153–162.
[27] J. Xiao, M. Shah, Two-frame wide baseline matching. in: Proc. ICCV'03, 2003. Available from: <http://www.cs.ucf.edu/~vision/projects/widematching/>.
[28] Z. Zhang, L. Wang, B. Guo, H. Shum, Feature-based light field morphing. in: Proc. ACM SIGGRAPH 2002, 2002, pp. 457–464.