Novel View Synthesis by Cascading Trilinear Tensors

Shai Avidan and Amnon Shashua, *Member*, *IEEE*

Abstract—We present a new method for synthesizing novel views of a 3D scene from two or three reference images in full correspondence. The core of this work is the use and manipulation of an algebraic entity termed *the trilinear tensor* that links point correspondences across three images. For a given virtual camera position and orientation, a new trilinear tensor can be computed based on the original tensor of the reference images. The desired view can then be created using this new trilinear tensor and point correspondences across two of the reference images.

Index Terms—Image-based rendering, trilinear tensor, virtual reality, image manipulation.

1 Introduction

This paper addresses the problem of synthesizing a novel image from an arbitrary viewing position given two or three reference images (registered by means of an optic-flow engine) of the 3D scene.

The most significant aspect of our approach is the ability to synthesize images that are far away from the viewing positions of the sample reference images without ever explicitly computing any 3D information about the scene. This property provides a multi-image representation of the 3D object using a minimal number of images. In our experiments, for example, two closely spaced frontal images of a face are sufficient for generating photorealistic images from viewpoints within a 60 degree cone of visual angle—further extrapolation is possible, but the image quality degrades.

We propose a new view-synthesis method that makes use of the recent development of multilinear matching constraints, known as trilinearities, that were first introduced in [42]. The trilinearities provide a general (not subject to singular camera configurations) warping function from reference images to novel synthesized images governed directly by the camera parameters of the virtual camera. Therefore, we provide a true multi-image system for view synthesis that does not require a companion depth map nor the full reconstruction of camera parameters among the reference cameras, yet is general and robust.

The core of this work is the derivation of a tensor operator that describes the transformation from a given tensor of three views to a novel tensor of a new configuration of three views. Thus, by repeated application of the operator on the *seed* tensor of the reference images with a sequence of desired virtual camera positions (translation and orientation), we obtain a chain of warping functions (tensors) from the set of reference images (from which the seed tensor was computed) to create the desired virtual views. We also show that the process can start with two reference views by having the "seed" tensor be comprised of the elements of the fundamental matrix of the reference views. A shorter version of this paper appeared in [4].

1.1 Novelty Over Previous Work

The notion of image-based rendering systems is gaining momentum in both the computer graphics and computer vision communities. The general idea is to avoid the computationally intensive process of acquiring a 3D model followed by rendering and, instead, to use a number of reference images of the object (or scene) as a representation from which novel views can be synthesized *directly* by means of image warping.

The work in this area can be roughly divided into three classes:

1) image interpolation,
2) off-line (Mosaic-based) synthesis, and
3) on-line synthesis.

The first class, image interpolation, is designed to create "in-between" images among two or more reference images. This includes image morphing [8], direct interpolation from image-flows ("multidimensional morphing") [10], [37], image interpolation using 3D models instead of image-flow [12], and "physically correct" image interpolation [40], [41], [55]. All but the last three references do not guarantee to produce physically correct images and all cannot extrapolate from the set of input images—that is, create novel viewing positions that are outside of the viewing cone of the reference images. For example, Seitz and Dyer [41] have shown that one can interpolate along the base-line of an image pair and obtain physically correct images (unlike flow-based interpolation [10], [37]). Their approach proceeds by first rectifying the images, interpolating along the epipolar lines (which are parallel after the rectification), and, then, inverting the rectification for the final rendering. Unfortunately, only images along the line connecting the two model images can be generated in this way and the user is not allowed to move freely in 3D space.

Instead of flow-field interpolation among the reference images, it is possible to interpolate directly over the plenoptic

• *The authors are with the Institute of Computer Science, The Hebrew University, Jerusalem 91904, Israel. E-mail: {avidan, shashua}@cs.huji.ac.il.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 106982.

function [1]—a function which represents the amount of light emitted at each point in space as a function of direction. Levoy and Hanrahan [31] and Gortler et al. [19] interpolate between a dense set of several thousand example images to reconstruct a reduced plenoptic function (under an occlusion-free world assumption). Hence, they considerably increase the number of example images to avoid computing optical flow between the reference images.

In the second class, the off-line (mosaic-based) synthesis, the synthesis is not created at run time—instead, many overlapping images of the scene are taken and then "stitched" together. The simplest stitching occurs when the camera motion includes only rotation—in which case the transformation between the views is parametric and does not include any 3D shape (the transformation being a 2D projective transformation, a homography). This was cleverly done by [13] in what is known as "QuickTime VR." Szeliski and Kang [51] create high-resolution mosaics from low-resolution video streams, and Peleg and Herman [35] relax the fixed camera constraint by introducing the projection manifold. A drawback of this class is that one cannot correctly simulate translational camera motion from the set of reference images.

The major limitation of the aforementioned techniques is that a relatively large number of reference images is required to represent an object. The third class, on-line synthesis, along the lines of this paper, reduces the number of acquired (reference) images by exploiting the 3D-from-2D geometry for deriving an on-line warping function from a set of reference images to create novel views on-the-fly based on user specification of the virtual camera position. Laveau and Faugeras [30] were the first to use the epipolar constraint for view synthesis, allowing them to extrapolate, as well as interpolate, between the example images. Epipolar constraints, however, are subject to singularities that arise under certain camera motions (such as when the virtual camera center is collinear with the centers of the reference cameras), and the relation between translational and rotational parameters of the virtual camera and the epipolar constraint is somewhat indirect and, hence, requires the specification of matching points. The singular camera motions can be relaxed by using the depth map of the environment. McMillan and Bishop [33] use a full depth map (3D reconstruction of the camera motion and the environment) together with the epipolar constraint to provide a direct connection between the virtual camera motion and the reprojection engine. Depth maps are easily provided for synthetic environments, whereas, for real scenes, the process is fragile, especially under small base-line situations that arise due to the requirement of dense correspondence between the reference images/mosaics [20]. The challenges facing an "optimal" on-line synthesis approach are, therefore:

Implicit Scene Modeling: To reduce, as much as possible, the computational steps from the input correspondence field among the reference images to useful algebraic structures that would suffice for generating new views. For example, it is likely that the base-line between reference views would be very small in order to facilitate the correspondence process. Thus, computing the full set of camera parameters (or, equivalently, the depth map of the scene) is not desirable as it may produce unstable estimates, especially for the translational component of camera motion (the epipoles). It is thus desirable to have the camera parameters remain as much as possible implicit in the process.

Nonsingular Configurations: To rely on warping functions that are free from singularities under camera motion. For example, the use of the fundamental matrix, or concatenation of fundamental matrices, for deriving a warping function based on epipolar line intersection (cf. [18]) is undesirable on this account due to singularities that arise when the camera centers are collinear.

Driving Mode: The specification of the virtual camera position should be intuitively simple for the user. For example, rotation and translation of the camera from its current position is prevalent among most 3D viewers.

None of the existing approaches for on-line synthesis satisfies all three requirements. For example, [30] satisfies the first requirement at the cost of complicating the driving mode by specifying control points; using depth maps provides an intuitive driving mode and lack of singularities but does not satisfy the implicit scene modeling requirement.

We propose an approach relying on concatenating trilinear warping functions that leave the scene and the camera parameters implicit, does not suffer from singularities, and is governed by the prevalent driving mode used by most 3D viewers.

2 Cascading Tensors

The view synthesis approach is based on the following paradigm: Three views satisfy certain matching constraints of a trilinear form, represented by a tensor. Thus, given two views in correspondence and a tensor, the corresponding third view can be generated uniquely by means of a warping function, as described below in more detail. We then derive a "driver" function that governs the change in tensor coefficients as a result of moving the virtual camera. We begin with basic terminology; more advanced details can be found in the Appendix.

2.1 Notations

A point $\boldsymbol{x}$ in the 3D projective space $\mathcal{P}^3$ is projected onto the point p in the 2D projective space $\mathcal{P}^2$ by a 3×4 camera projection matrix $\mathbf{A} = [A, v']$ that satisfies $p \cong \mathbf{A}\boldsymbol{x}$, where $\cong$ represents equality up to scale. The left 3×3 minor of $\boldsymbol{A}$, denoted by A, stands for a 2D projective transformation of some arbitrary plane (the reference plane), and the fourth column of $\boldsymbol{A}$, denoted by v', stands for the epipole (the projection of the center of camera 1 on the image plane of camera 2). In a calibrated setting, the 2D projective transformation is the rotational component of camera motion (the reference plane is at infinity), and the epipole is the translational component of camera motion. Since only relative camera positioning can be recovered from image measurements, the camera matrix of the first camera position in a sequence of positions can be represented by $[I; 0]$.

In the case of three views, we adopt the following convention: The relationship between the 3D and the 2D spaces is represented by the 3×4 matrices, $[I, 0]$, $[A, v']$, and $[B, v'']$, i.e.,

$$p = [I, 0]\boldsymbol{x}$$

$$p' \cong [A, v']\boldsymbol{x}$$

$$p'' \cong [B, v'']\boldsymbol{x},$$

where $p = (x, y, 1)^\top$, $p' = (x', y', 1)^\top$, $p'' = (x'', y'', 1)^\top$ are matching points with image coordinates (x, y), $(x'y')$, (x'', y'').

We will occasionally use tensor notations as described next. We use the covariant-contravariant summation convention: A point is an object whose coordinates are specified with superscripts, i.e., $p^i = (p^1, p^2, ...)$. These are called contravariant vectors. An element in the dual space (representing hyperplanes—lines in $\mathcal{P}^2$), is called a covariant vector and is represented by subscripts, i.e., $s_j = (s_1, s_2, ...)$. Indices repeated in covariant and contravariant forms are summed over, i.e., $p^i s_i = p^1 s_1 + p^2 s_2 + ... + p^n s_n$. This is known as a contraction. An outer-product of two 1-valence tensors (vectors), $a_i b^j$, is a 2-valence tensor (matrix) c_i^j whose i, j entries are $a_i b^j$—note that, in matrix form, $C = ba^\top$.

A vector can be represented by its symbol, say p', or by its tensor form p'^j (the range of the index is assumed known by context). An element of a vector can be represented by its designated symbol (if it exists), say $p' = (x', y', 1)^\top$, or by its tensor form $p'^j = (p'^1, p'^2, p'^3)$. Likewise, a matrix can be represented by its symbol, say B, or by its tensor form b_i^k, and its elements by designating values to the indecis: b_2^3 is a scalar and b_2^k is the kth row of B.

2.2 The Trilinear Tensor

The trilinear tensor is a $3 \times 3 \times 3$ array of 27 entries described by a bilinear function of the camera matrices **A**, **B**:

$$\mathcal{T}_i^{jk} = v'^j b_i^k - v''^k a_i^j, \tag{1}$$

where a_i^j, b_i^k are the elements of the homographies A, B, respectively, and v', v'' are the epipoles of the first image in the second and third images, respectively (see the Appendix for derivation).

The Fundamental matrix $F = [v']_x A$, where $[]_x$ is the skew-symmetric matrix defining the cross-product operation, can also be embedded in a trivalent tensor

$$\mathcal{F}_i^{jk} = v'^j a_i^k - v'^k a_i^j = \epsilon^{ljk} F_{li}, \tag{2}$$

where F_{li} are the elements of F and ϵ^{ljk} is the cross-product tensor $\epsilon^{ljk} u^j v^k = u \times v$. Further details can be found in the Appendix.

2.2.1 The Trilinear Tensor for Reprojection

Let s_j be any line coincident with p', i.e., $s_j p'^j = 0$, for example, the horizontal $(-1, 0, x')$ and vertical lines $(0, -1, y')$ span all other lines coincident with p'. Let r_k be any line coincident with p''. Then, the tensor acts on the triplet of matching points in the following way:

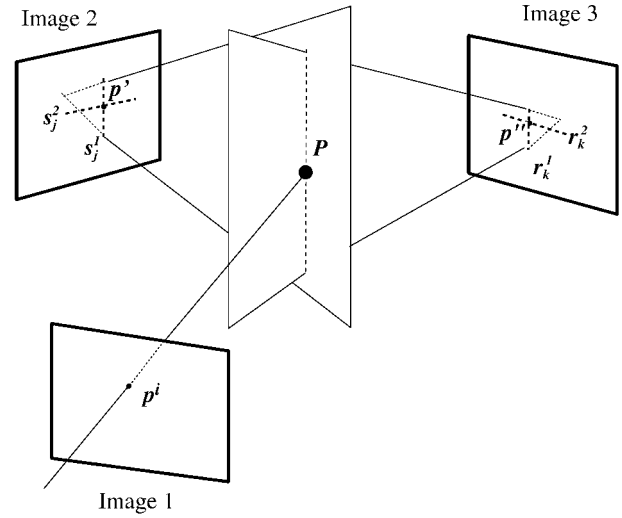


Fig. 1. Each of the four trilinear equations describes a matching between a point p in the first view, some line s_j^μ passing through the matching point p' in the second view and some line r_k^ρ passing through the matching point p'' in the third view. In space, this constraint is a meeting between a ray and two planes.

$$p^i s_j^\mu r_k^\rho \mathcal{T}_i^{jk} = 0, \tag{3}$$

where s_j^μ are two arbitrary lines (s_j^1 and s_j^2) intersecting at p', and r_k^ρ are two arbitrary lines intersecting p''. Since the free indices are μ, ρ each in the range 1, 2, we have four trilinear equations (unique up to linear combinations), as can be seen in Fig. 1.

The tensor consists of 27 coefficients and, since each matching point contributes four linearly independent equations, one needs at least seven matching points across three images to linearly recover the trilinear tensor. Once recovered, the tensor can be used for reprojection because, given two reference images and a tensor, the third image is uniquely determined and can be synthesized by means of a warping function applied to the two reference images, as follows. Let p, p' be given, then, since $p^i s_j^\mu \mathcal{T}_i^{jk}$ is a point that coincides with all lines passing through p'', then

$$p^i s_j^\mu \mathcal{T}_i^{jk} \cong p''^k, \tag{4}$$

which provides a set of four equations for p'' (i.e., a redundant set). This process is referred to as "reprojection" in the literature. There are alternative ways of performing reprojection without recovering a 3D model, such as by intersecting epipolar lines using the Fundamental matrix [18]; however, those are sensitive to degenerate situations (like when the three camera centers are collinear). The tensor reprojection approach is free from singularities of camera positions and is, therefore, a preferred choice. Comparative studies of various approaches for reprojection using algebraic invariants can be found in [7], [42], where [7] concludes that all approaches do well under "favorable" viewing geometry (low amount of noise and camera centers are far from being collinear)—and, in challenging situations, the tensor reprojection approach performs the best.

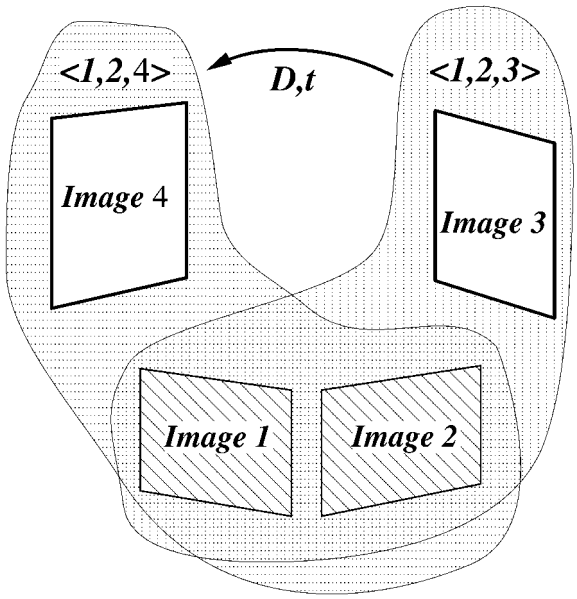


Fig. 2. We generate tensor <1, 2, 4>, that relates images 1, 2 with some novel image 4, from the seed tensor <1, 2, 3> and the virtual camera motion parameters [*D*, *t*] from image 3 to image 4. Tensor <1, 2, 3> relates images 1, 2, and 3 and is computed only once at the preprocessing stage. Tensor <1, 2, 4> is computed every time the user specifies a new [*D*, *t*]. We use tensor <1, 2, 4> to render the novel image (image 4) from reference images 1, 2.

In image-based rendering, we would like to obtain the tensor (the warping function) via user specification of location of virtual camera, rather than by the specification of (at least) seven matching points. If one knows (or recovers) the full relative orientation (rotation and translation) between the first two views 1, 2, then insertion of relative orientation between views 1 and the desired view 3 in the tensor equation (1) would provide the desired tensor for warping views 1, 2 onto 3 (see, for example, [14]). One can, however, create the desired tensor without knowing the full motion between views 1, 2 by introducing the "tensor operators" described next.

2.3 The Basic Tensor Operator

The basic tensor operator describes how to modify (transform) a tensor so as to represent a new configuration of three cameras. We are particularly interested in the case where only one camera has changed its position and orientation. Thus, by repeated application of the operator on a seed tensor with a sequence of desired virtual camera positions (translation and orientation), we obtain a chain of warping functions (tensors) from the set of acquired images (from which the seed tensor was computed) to create the desired virtual views (see Fig. 2).

Consider four views generated from camera matrices $[I; 0]$, $[A; v']$, $[B; v'']$, and $[C; v''']$, i.e., the 3D projective representation of the object space remains unchanged or, in other words, the homography matrices A, B, C correspond to the same reference plane π (if at infinity, then A, B, C are the rotational component of camera motion). The tensor of views 1, 2, 3 is $\mathcal{T}_i^{jk}$ and we denote the tensor of views 1, 2, 4 as $\mathcal{G}_i^{jk}$,

$$\mathcal{G}_i^{jk} = v'^j c_i^k - v'''^k a_i^j. \tag{5}$$

We wish to describe $\mathcal{G}$ as a function of $\mathcal{T}$ and the incremental camera motion from camera 3 to 4. Let the motion parameters from camera 3 to 4 be described by a 3×3 homography matrix D (from image plane 3 to image plane 4) and translation t. Due to the group property of homography matrices (since the reference plane π is fixed), $C = DB$ and, hence, we have:

$$\begin{aligned}\mathcal{G}_i^{jk} &= v'^j\left(d_l^{jk} b_i^l\right) - v'''^k a_i^j \\ &= d_i^k \mathcal{T}_i^{jl} + \left(d_l^k v''^k - v'''^k\right) a_i^j\end{aligned}$$

and, since $t = Dv'' - v'''$, we have the following result:

$$\boxed{\mathcal{G}_i^{jk} = d_l^k \mathcal{T}_i^{jl} + t^k a_i^j}. \tag{6}$$

Given a "seed" tensor $\mathcal{T}_i^{jk}$ and a user specified camera motion D, t from the last view to the desired view which is *compatible* with the projective representation of the object space (i.e., the matrix D and A are due to the same reference plane), then the formula above will generate a new tensor $\mathcal{G}_i^{jk}$ that can be used to reproject the first two model views (views 1, 2) onto the desired novel view. The seed tensor can be a trilinear tensor of three views, or the tensor embedding $\mathcal{F}_i^{jk}$ of the Fundamental matrix. In other words, the process can start with two model views or with three model views—once it has started, the subsequent tensors are of three views (the first two views and the novel views).

What is left to consider is how to ensure that the homographies A and D are due to the same plane. There could be two approaches. One approach is to have the user specify where some physical plane seen in the two model views should be in the novel view. A possible algorithm can be as follows:

1) Compute the seed tensor $\mathcal{T}$.
2) Accept from the user four coplanar points defining some (virtual or physical) plane π and use them to compute the homography matrix A from the system of linear equations $Ap \cong p'$ for each pair of matching points.
3) Accept from the user the translation vector t and the projections of four coplanar points from the plane π on the novel view. The four points, are enough to construct the homography D and, as a result, recover the new tensor $\mathcal{G}$ from (6).
4) Use the new tensor $\mathcal{G}$ together with the dense correspondence between the two model images to synthesize (reproject) the novel view.

This algorithm has the advantage of avoiding the need to calibrate the cameras at the expense of assuming the existence of a plane in the 3D scene and a somewhat indirect user interface. Similar methods for specifying the novel camera position by means of specifying few image control points were suggested by [30], [41].

The second alternative, which is the one we preferred, is to try and estimate the plane at infinity, i.e., the rotation, between the two model images (to be described later). As a result, the homography matrix D becomes the rotational component of camera motion and the process of specifying

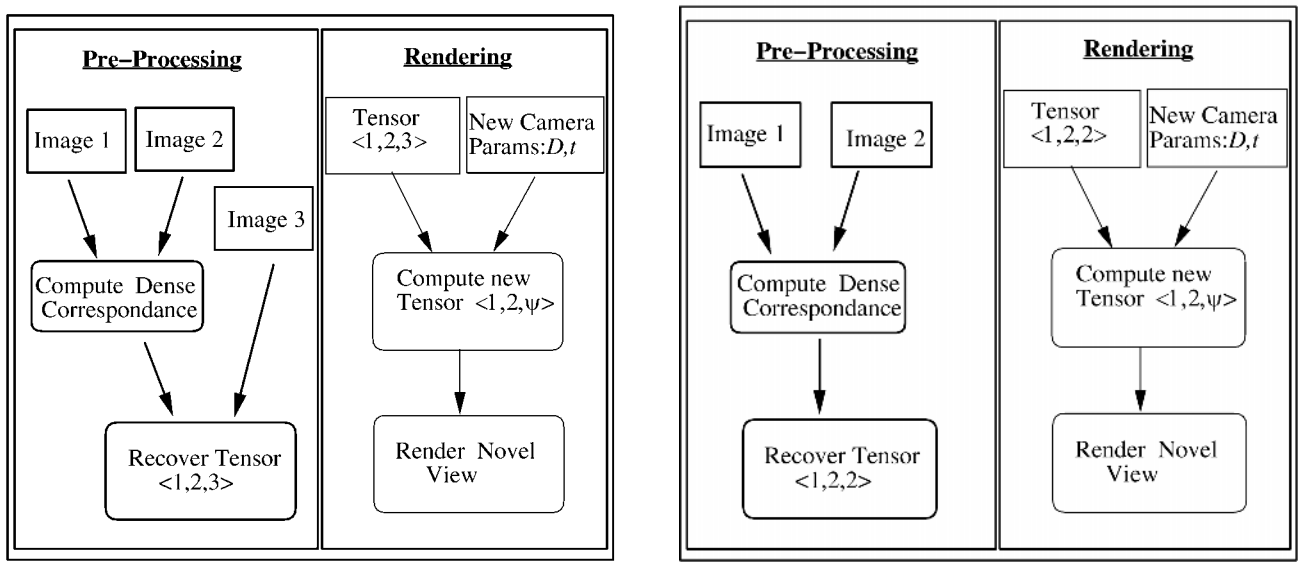


Fig. 3. View synthesis for two or three reference images. In both cases, the process is divided into two parts. The preprocessing stage is done only once and the actual rendering is done for every image. (a) Three model images. (b) Two model images.

the position of the novel image is simplified and more intuitive for the user. Our assumptions of the internal camera parameters are mild (the principal point is assumed to be at the center of the image and the focal length is assumed to be the image width), yet the algorithm is robust enough to produce "Quasi-Euclidean" [34] settings which result in plausible novel views.

To summarize, we start with the seed tensor of the reference images and modify it according to the user specified position D, t of the virtual camera position. The modified tensors, together with the dense correspondence between two of the model images, are used for rendering the novel images, as can be graphically seen in Fig. 3.

3 Implementation

To implement the method presented in this paper, one needs several building blocks:

- dense correspondence between a pair of images,
- robust recovery of the seed tensor (either the trilinear tensor for three reference images or the fundamental matrix, in its tensor form, for two reference images),
- a correct reprojection mechanism,
- handling Occlusions.

A large body of work is devoted to the problem of finding dense correspondence between a pair of images [32], [9], recovery of the trilinear tensor [42], [47], [17], [52], [25], [15], and recovery of the fundamental matrix [16], [23]. Any of the methods described there can be used with our algorithm. Here, we give our implementation.

3.1 Dense Correspondence

We obtain dense correspondence using a Lucas-Kanade style optical-flow [32] embedded in a hierarchical framework [9]. For every pixel, we estimate its motion (u, v) using the well known equation:

$$\begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_xI_t \\ -I_yI_t \end{pmatrix},$$

where I_x, I_y, I_t are the sum of derivatives in the x, y-directions and time, respectively, in a 5×5 window centered around the pixel. We construct a Laplacian pyramid [11] and recover the motion parameters at each level, using the estimate of the previous level as our initial guess. In each level, we iterate several times to improve the estimation. This is done by warping the first image toward the second image, using the recovered motion parameters, and then repeating the motion estimation process. Typically, we have five levels in the pyramid and we perform two to four iterations per level. The initial motion estimation at the coarsest level is zero.

3.2 Robust Recovery of the Seed Tensor

The seed tensor is recovered from the reference images and, since the number of reference images may be either two or three, slightly different algorithms are needed. We describe the procedure for recovering the Fundamental matrix and inform the reader when it deviates from the algorithm for recovering the trilinear tensor. The steps for computing the Fundamental matrix/tensor are:

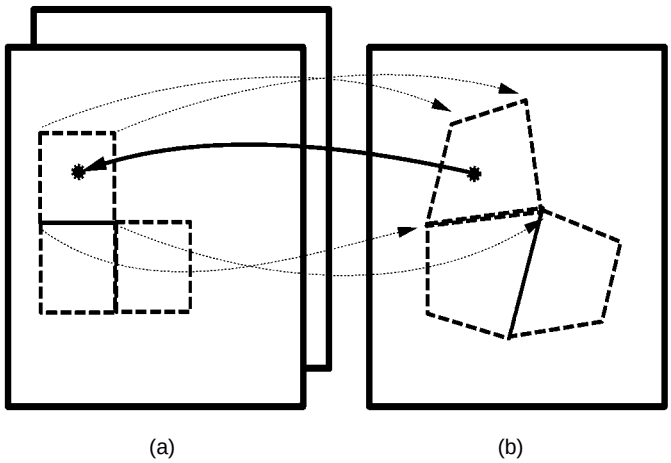
- **Find Matching Points:** The method we use is a variant of Harris corner detector [21]. For every pixel in the first image, we compute the "optic-flow" matrix

$$\begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix},$$

where I_x, I_y are the sum of derivatives in the x– and y–directions, respectively, in a 5×5 window centered around the pixel, and extract its eigenvalues. We select points with their smaller eigenvalue above some predefined threshold. (Usually, we set the threshold to

be about seven gray-level values. Since we sum the square values over a 5×5 window, we require the smallest eigenvalue to be larger than $1{,}225 = 7^2 * 5 * 5$.) We track them from the first image to the second image and back to the first image. (In case of three model images, we track the points from the first image to the third, via the second image and vice-versa.) Points that return, at the end of the loop, to their original position (up to distance of one pixel) are accepted.

- **Robust Estimation of the Trilinear Tensor/Fundamental Matrix:** The previous stage usually produces several hundreds points, referred to as "good" points, which are then normalized to ensure good numerical solution [23]. This normalization consists of applying a similarity transformation so that the transformed points are centered at the origin and the mean distance from the origin is $\sqrt{2}$. The normalized "good" points are used for computing the tensor in a robust statistics manner [36] to avoid the effects of outliers. The robust estimation is done by a repetitive lottery of a subset of seven "good" points (in case of three model images) or eight "good" points (in case of two model images), recovering the fundamental matrix (trilinear tensor, for three model images), and verifying the quality of the fundamental matrix (trilinear tensor) on the remaining set of good points. The quality of the fundamental matrix is measured by the distance of the points in the second image from their epipolar line (for three model images, the quality of a tensor is measured by the distance of the projected third point and the actual third point). Only points with error of less than a specified threshold (typically, one pixel) are considered. The best fundamental matrix (trilinear tensor) is the one with the largest number of supporting points, and we compute it again in a usual least-squared manner, using all the points that supported it.



(a) (b)

Fig. 4. Forward mapping is decomposed into smaller backward mapping problems. The corners of a rectangle in the source image are mapped to a quadrilateral in the destination image and a backward map is computed for all the pixels in the quadrilateral. (a) Source images. (b) Destination image.

Finally, in case of only two model images, we need to convert the recovered fundamental matrix into a tensor form, as described in the Appendix.

3.3 Recovering the Rotation

We recover the small-angles rotation matrix between two model images directly from the tensor, under the assumption that the principal point is in the center of the image and that the focal length is equal to the width of the image. This assumptions proved to be sufficient for our method to recover a reasonable approximation to the correct rotation matrix (one can use either $\mathcal{T}$ or $\mathcal{F}$ depending on whether two or three model views are used).

$$\Omega_X = det\begin{pmatrix} \mathcal{T}_2^{j3} \\ \mathcal{T}_2^{j3} + \mathcal{T}_3^{j2} \\ \mathcal{T}_3^{j3} - \mathcal{T}_2^{j2} \end{pmatrix} \Big/ K$$

$$\Omega_Y = det\begin{pmatrix} -\mathcal{T}_1^{j3} \\ \mathcal{T}_2^{j3} + \mathcal{T}_3^{j2} \\ \mathcal{T}_3^{j3} - \mathcal{T}_2^{j2} \end{pmatrix} \Big/ K$$

$$\Omega_Z = det\begin{pmatrix} \mathcal{T}_1^{j2} \\ \mathcal{T}_2^{j3} + \mathcal{T}_3^{j2} \\ \mathcal{T}_3^{j3} - \mathcal{T}_2^{j2} \end{pmatrix} \Big/ K$$

$$K = det\begin{pmatrix} \mathcal{T}_2^{j2} \\ \mathcal{T}_2^{j3} + \mathcal{T}_3^{j2} \\ \mathcal{T}_3^{j3} - \mathcal{T}_2^{j2} \end{pmatrix}, \tag{7}$$

where $\mathcal{T}_2^{j2}$ stands for $\left(\mathcal{T}_2^{12}, \mathcal{T}_2^{22}, \mathcal{T}_2^{32}\right)$, etc., and the vector $\Omega = (\Omega_X, \Omega_Y, \Omega_Z)^\top$ is the rotation axis, and the magnitude of the vector is the magnitude of the rotation around this axis. Large angles can be recovered by iteratively computing $\mathcal{T}_i^{jk^{(n+1)}} = A_j^{l^{(n)}} \mathcal{T}_i^{lk^{(n)}}$, where $A_j^{l^{(n)}}, \mathcal{T}_i^{lk^{(n)}}$ are the recovered rotation matrix in the previous iteration and the previous tensor, respectively. This method was first presented in [39] for the purpose of video stabilization.

3.4 Reprojection Process

The reprojection process is performed every time we wish to generate a novel view. First, we compute the new tensor, of the first two model images and the novel view and, then, use (4),

$$p^i s_j^\mu \mathcal{T}_i^{jk} \cong p''^k$$

to obtain the coordinates of the point in the novel view, which is a set of four equations with three unknowns (since $p'' = [ux, uy, u]^T$ is 2D homogeneous coordinate) which we solve in a least-square manner.

To overcome the forward mapping problem, we split the problem into smaller backward mapping tasks as follows: We map rectangles of size $n \times n$ pixels in the source images to quadrilaterals in the target image and then compute the backward mapping from the destination to the source image [56] as can be seen if Fig. 4. This method gives a nice trade-off between speed and quality by changing the size of the rectangle. Fig. 5 demonstrates the quality of the results for $n = 1, 2, 5, 10$.

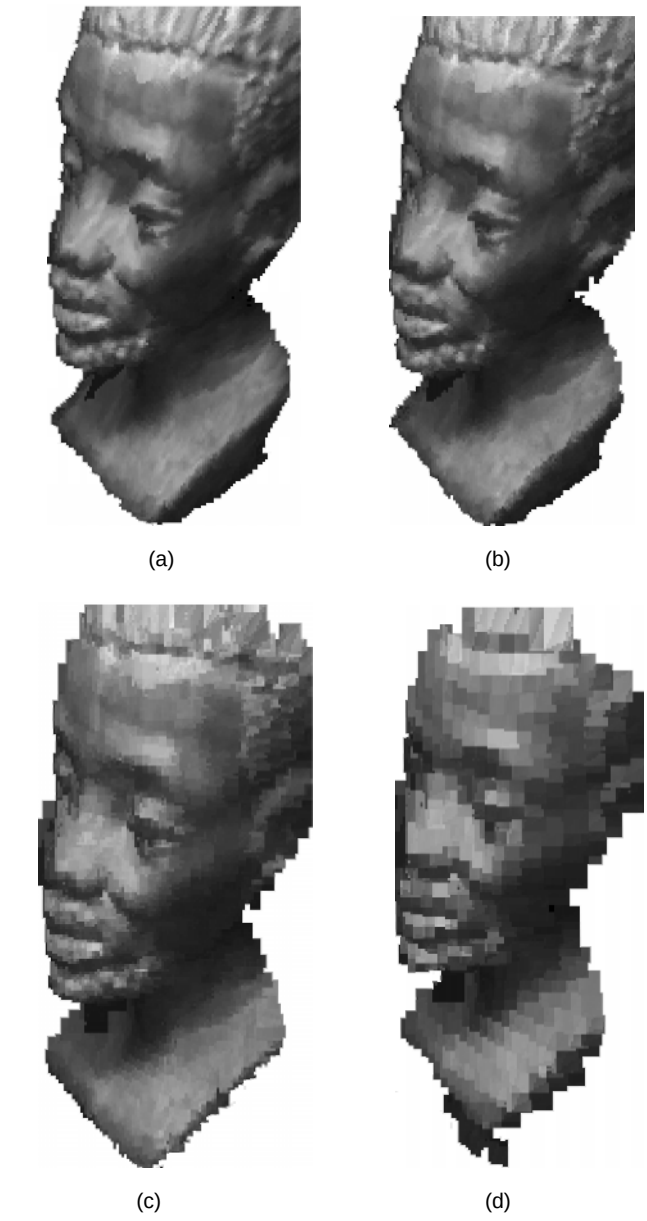


Fig. 5. Performing forward mapping with rectangles of size $n \times n$ pixels. The larger n is, the faster the reprojection performs. See text for more details. (a) $n = 1$, (b) $n = 2$, (c) $n = 5$, (d) $n = 10$.

3.5 Handling Occlusions

The image-based rendering algorithm described above does not handle visibility issues, i.e., the surface is assumed transparent. It is possible to use a simple "projective Z-buffering" procedure for enforcing the constraint that the surface is opaque (cf. [30], [38], [46]). If two pixels are reprojected onto the same cell, we simply choose the one close to the epipole in image 1 corresponding to the projection of the new center of projection onto image 1. The epipolar points can be recovered linearly from the tensor $\mathcal{T}_i^{jk}$ [47]. Note that we recover the epipolar points only for resolving visibility problems, not for reprojection, thus, inaccuracy in the epipoles would not affect the accuracy of reprojection. It is also important to note that the "projective Z-buffering" is not always guaranteed to correctly resolve visibility problems, as a more rigorous treatment is needed for the general case [29], nevertheless, the procedure is fairly robust in practice.

4 Experiments

4.1 Capturing the Images

In the examples below, we followed the following guidelines: The two images were taken with a single camera that was moving a few centimeters between shots. The motion of the camera is designed to avoid creating occluded areas between the images. The object is placed at about 50 cm from the camera and fills most of the viewing area. We found that short base-line between the images greatly enhances the quality of the correspondence and that our method is robust enough to generate synthesized images that are far away from the original viewing cone. Lighting conditions are normal and include either daylight or even fluorescent light. The camera types used varied from the standard indy-cam to scanned images from a 35mm camera.

4.2 Experimental Results With Real Images

The tensor-based rendering method was implemented on real images of both artificial and natural objects. In each example, a "movie" was created by specifying a set of key frames, each by a rotation and translation of the virtual camera from one of the model images. The parameters of rotation and translation were then linearly interpolated (not the images, only the user-specified parameters) to the desired number of frames. Also, we handled visibility problems, to obtain better results. In all the cases presented, we assumed the principal point of the camera to be at the center of the image and the focal length to be the width of the image. In the first example, two images of an African statue (260×480 pixels each) were captured using a standard indy-cam without calibration procedure or controlled light. The object is about 10 cm in height and was placed some 30 cm in front of the camera. We ran the preprocessing stage, which takes about 30 seconds on an SGI Indy machine, and then defined a set of key frames by moving the virtual camera in 3D space. Next, we interpolated between the parameters of the key frames to obtain the parameters for the entire "movie." We computed the novel trilinear tensor for each novel image and then reprojected the virtual image. The reprojection process takes about 5 seconds with rectangle size of 1×1 pixels for the warping stage. Some of the generated images can be seen in Fig. 6. We repeated the process with two human subjects. For the first subject, the camera was at about 80 cm from the subject and the size of the captured images was 576×576 We repeated the same procedure described for the African statue and present the result in Fig. 7. For the second subject, the camera was at about 50 cm from the person and the captured images were of size 230×288. The results can be seen in Fig 8. In another test, we have downloaded a pair of images from the CMU/VASC image database (the images courtesy of Hoff and Ahuja [28]). No information about the camera internal or external parameters is known or used. The images are 512×512 pixels each and an example for a synthesized image can be seen in Fig. 9.

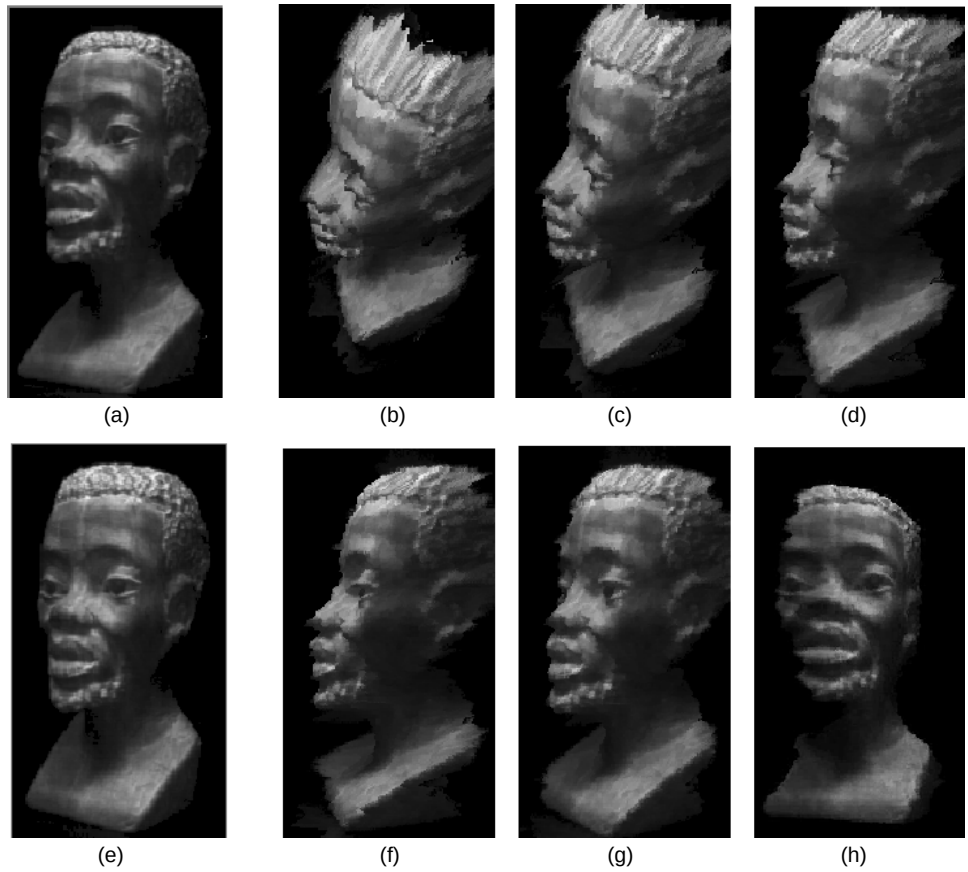


Fig. 6. An example of image synthesis using optic-flow and a tensor. The original images are (a) and (e), the rest of the images are taken from the generated movie. They should be seen left to right, top to bottom.

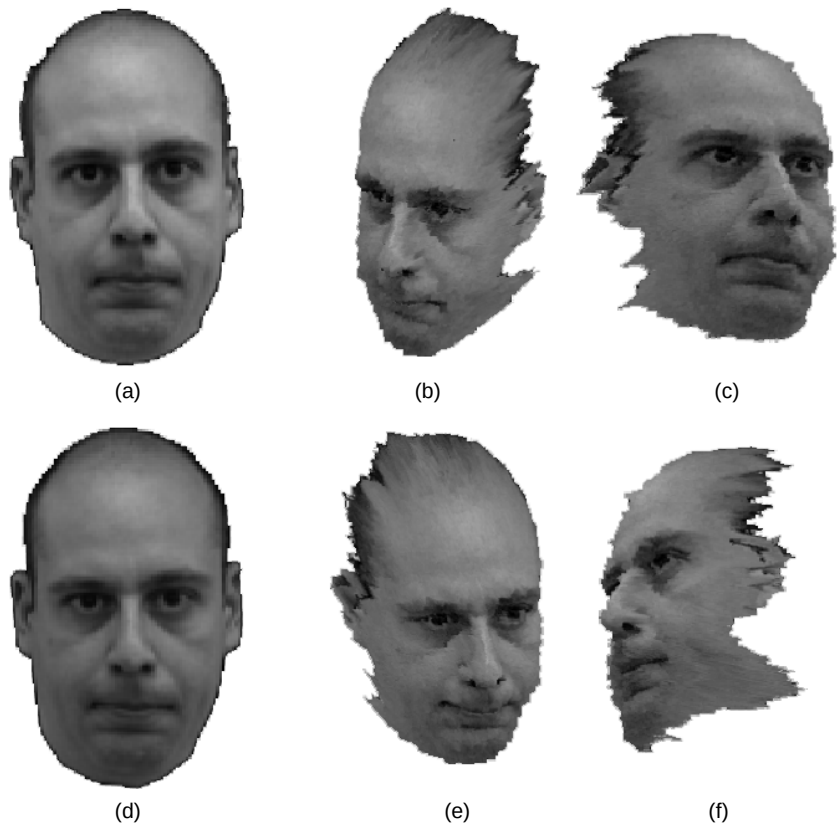
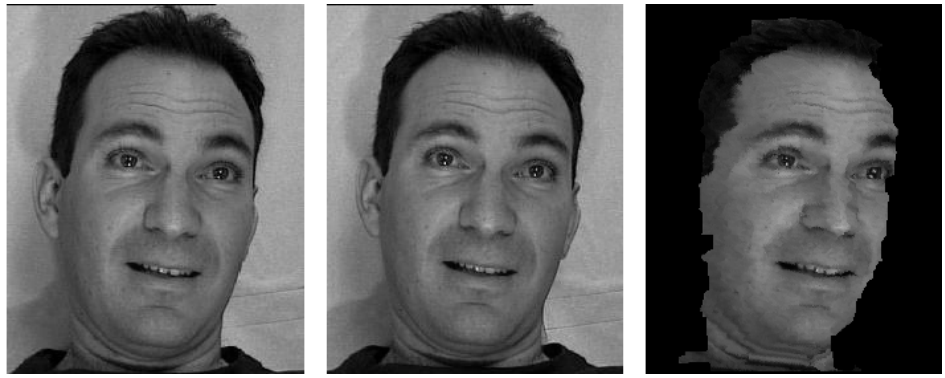


Fig. 7. The original two images (a) and (d). Novel views of the face (b), (c), (e), (f).



(a) (b) (c)

Fig. 8. The original images (a), (b) are used to synthesize the novel view (c).

(a) (b) (c)

Fig. 9. The original images (a), (b) are used to synthesize the novel view (c).

We have also measured the sensitivity of the reprojection process to errors in correspondence. This was done by adding increasing levels of noise to the correspondence values. The noise was added independently to the x and y-components of the correspondence and was uniformly distributed in the range $[-n, n]$, where n goes up to five pixels. Fig. 11 presents the results—from which we notice a graceful degradation of the rendering process despite the fact the distance between the model views is very small. In other words, the robustness of the synthesis method allows us to extrapolate considerably from the original viewing cone.

We extended our approach to handle dynamic scene as well by treating it as a series of static scenes in time. A pair of synchronized, stationary cameras captured a flexible object (in this case, facial expressions). Since the cameras are stationary, we compute the seed tensor only once. For every pair of images, we compute dense correspondence, generate a novel tensor from the seed tensor and the user specified parameters, and reproject the novel view. The result is a fly-through around a "talking head," as can be seen in Fig. 10. Notice that all the examples contain a considerable degree of extrapolation (i.e., views that are outside the viewing cone of the original two model views).

5 Summary

We have shown the use of the trilinear tensor as a warping function for the purpose of novel view synthesis. The core of this work is the derivation of a tensor operator that describes the transformation from a given tensor of three views to a novel tensor of a new configuration of three views. During the entire process, no 3D model of the scene is necessary nor is it necessary to recover camera geometry or epipolar geometry of the model images. In addition, the synthesis process can start with only two model views and their fundamental matrix, but the later steps follow the trilinear tensor machinery which ensures lack of singular configurations and provide a natural driving mode. Experiments have demonstrated the ability to synthesize new images from two closely-spaced model images, where the viewing range of the synthesized images far exceed the viewing cone of the model images (extrapolation of viewing position, rather than interpolation).

The limitations of the technique are mainly concerned with the correspondence problem. The fact that our method accepts closely spaced images (typically, the cameras are few centimeters apart) greatly enhances the quality of the optical-flow procedure. In addition, we are currently investigating methods for automatically extracting the focal length directly from the trilinear tensor, thus removing the need to assume some predefined value for the focal length.

Indeed, the view synthesis problem can be solved by means of 3D reconstruction using the epipolar geometry of a pair of images. Basically, there are two issues at hand. First, why use the tensor formulation rather than Fundamental matrices? Second, why not use depth maps as an intermediate step?

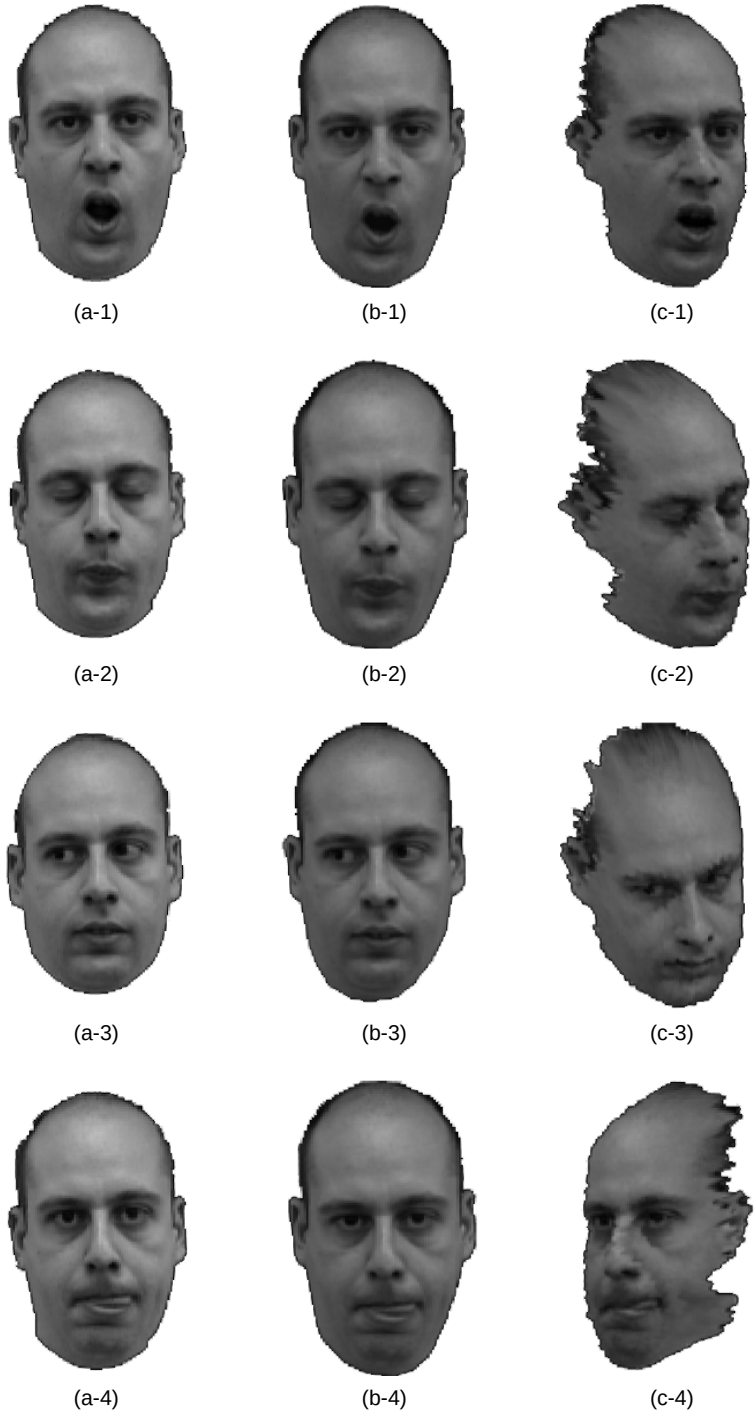
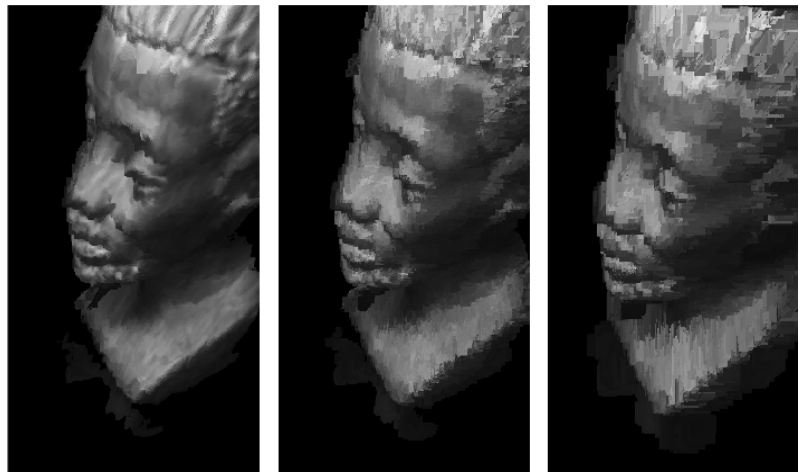


Fig. 10. The left two columns show some of the facial expression captured with the stationary pair of cameras. The rightmost column shows some virtual views. The preprocess stage of computing the seed tensor was done only once for the upper pair of images. For each additional pair, we start from the same seed tensor.

Regarding the use of Tensors, the tensor formulation on one hand does not suffer from singular motions and generalizes the formulation of the Fundamental matrix (see Section 1.2 on the tensor embedding of the Fundamental matrix). On the other hand, the collection of Fundamental matrices does not always contain the full representation of the 3D-from-2D geometry (like when the camera centers are collinear [27]), and preliminary work on degeneracy indicates that the family of "critical (ambiguous) surfaces" is at most a curve or a finite set of points under the tensor formulation, compared to ambiguous surfaces under Fundamental matrix formulation [45]. Furthermore, as was mentioned in the Introduction, the tensor formulation satisfies the three requirements necessary for an on-line synthesis system, such as intuitively simple "driving mode" and singular-free motions. Therefore, even though the complete story of the 3D-from-2D geometry is still unfolding, the tensor formulation appears as a viable alternative for representing 3D manipulations from collection of views.



(a) (b) (c)

Fig. 11. We measure the sensitivity of the reprojection process to errors in the correspondence by adding increasing levels of noise to the correspondence values. (a), (b), and (c) show an extrapolated view of the statue with noise levels of 0, 1.5, and 5 pixels, respectively.

Fig. 12. Reprint from [2]. On the left are the four input images. For every expression, we have a pair of images from different view points. We combine classical techniques of image morphing to handle the nonrigid transformation of the model and our tensor-based approach to generate novel view points.

Regarding the use of depth maps, we simply show that one can do without them and, moreover, one can do without explicitly recovering the camera translation between the model views (which is the most error sensitive component of motion estimation). Thus, having most of the unnecessary elements (camera translation, depth maps) remain implicit, we stand a better chance of focusing only on the relevant elements. We have included a sensitivity test to demonstrate the robustness of the rendering process in the presence of added noise to the image correspondences. The results show that, even with the small baseline we have worked with, the process degrades very gracefully.

In addition, we believe that, by working in the image domain and avoiding depth maps, one can extend current 2D image morphing techniques with the aid of the trilinear tensor. An example for such a combination appeared in [2], where we combine nonrigid facial transformations using classic view morphing techniques with our trilinear-tensor approach to handle rigid transformations. For example, we show in Fig. 12 an example of combining rigid and nonrigid transformations in a single framework.

Finally, we are currently working [6] on an extension of the method to handle an arbitrary number of images by constructing a *consistent* camera trajectory between all the model images and using the recovered tensor parameters for view synthesis.

APPENDIX A.1
THE TRILINEAR TENSOR OF THREE IMAGES

Consider a single point $\boldsymbol{x}$ in space projected onto three views with camera matrices $[I; 0]$, $\boldsymbol{A}$, $\boldsymbol{B}$ with image points p, p', p'', respectively. Note that $\boldsymbol{x} = (x, y, 1, \lambda)$ for some scalar λ. Consider $\boldsymbol{A} = [A; v']$, where A is the 3×3 principle minor of $\boldsymbol{A}$ and v' is the fourth column of $\boldsymbol{A}$. Consider $p' \cong \boldsymbol{A}\boldsymbol{x}$ and eliminate the scale factor:

$$x' = \frac{\boldsymbol{a}_1^\top x}{\boldsymbol{a}_3^\top x} = \frac{a_1^\top p + \lambda v_1'}{a_3^\top x + \lambda v_3'} \tag{8}$$

$$y' = \frac{\boldsymbol{a}_2^\top x}{\boldsymbol{a}_3^\top x} = \frac{a_1^\top p + \lambda v_2'}{a_3^\top x + \lambda v_3'}, \tag{9}$$

where $\boldsymbol{a}_i$ is the ith row of $\boldsymbol{A}$. These two equations can be written more compactly as follows:

$$\lambda \boldsymbol{s}'^\top \boldsymbol{v}' + \boldsymbol{s}'^\top A p = 0 \tag{10}$$

$$\lambda \boldsymbol{s}''^\top \boldsymbol{v}' + \boldsymbol{s}''^\top A p = 0, \tag{11}$$

where $\boldsymbol{s}' = (-1, 0, x)$ and $\boldsymbol{s}'' = (0, -1, y)$. Yet, in a more compact form, consider $\boldsymbol{s}', \boldsymbol{s}''$ as row vectors of the matrix

$$s_j^\mu = \begin{bmatrix} -1 & 0 & x' \\ 0 & -1 & y' \end{bmatrix},$$

where $j = 1, 2, 3$ and $\mu = 1, 2$. Therefore, the compact form we obtain is described below:

$$\lambda s_j^\mu v'^j + p^i s_j^\mu a_i^j = 0, \tag{12}$$

where μ is a free index (i.e., we obtain one equation per range of μ).

Similarly, let $\boldsymbol{B} = [B; v'']$ for the third view $p'' \cong \boldsymbol{B}\boldsymbol{x}$ and let r_k^ρ be the matrix,

$$r_k^\rho = \begin{bmatrix} -1 & 0 & x'' \\ 0 & -1 & y'' \end{bmatrix}.$$

And, likewise,

$$\lambda r_k^\rho v''^k + p^i r_k^\rho b_i^k = 0, \tag{13}$$

where $\rho = 1, 2$ is a free index. We can eliminate λ from (12) and (13) and obtain a new equation:

$$\left(s_j^\mu v'^j\right)\left(p^i r_k^\rho b_i^k\right) - \left(r_k^\rho v''^k\right)\left(p^i s_j^\mu a_i^j\right) = 0,$$

and, after grouping the common terms:

$$p^i s_j^\mu r_k^\rho \left(v'^k b_i^k - v''^k a_i^j\right) = 0,$$

and the term in parentheses is a trivalent tensor we call the *trilinear tensor*:

$$\boxed{\mathcal{T}_i^{jk} = v'^j b_i^k - v''^k a_i^j. \quad i, j, k = 1, 2, 3}. \tag{14}$$

And the tensor equations (the 4 trilinearities) are:

$$\boxed{p^i s_j^\mu r_k^\rho \mathcal{T}_i^{jk} = 0}. \tag{15}$$

Hence, we have four trilinear equations (note that $\mu, \rho = 1, 2$). In more explicit form, these trilinearities look like:

$$x''\mathcal{T}_i^{13}p^i - x''x'\mathcal{T}_i^{33}p^i + x'\mathcal{T}_i^{31}p^i - \mathcal{T}_i^{11}p^i = 0,$$
$$y''\mathcal{T}_i^{13}p^i - y''x'\mathcal{T}_i^{33}p^i + x'\mathcal{T}_i^{32}p^i - \mathcal{T}_i^{12}p^i = 0,$$
$$x''\mathcal{T}_i^{23}p^i - x''y'\mathcal{T}_i^{33}p^i + y'\mathcal{T}_i^{31}p^i - \mathcal{T}_i^{21}p^i = 0,$$
$$y''\mathcal{T}_i^{23}p^i - y''y'\mathcal{T}_i^{33}p^i + y'\mathcal{T}_i^{32}p^i - \mathcal{T}_i^{22}p^i = 0.$$

Since every corresponding triplet p, p', p'' contributes four linearly independent equations, then seven corresponding points across the three views uniquely determine (up to scale) the tensor $\mathcal{T}_i^{jk}$. Equation (14) was first introduced in [42] and the tensor derivation leading to (15) was first introduced in [43].

The trilinear tensor has been well-known in disguise in the context of Euclidean line correspondences and was not identified at the time as a tensor but as a collection of three matrices [48], [49], [54] (a particular contraction of the tensor known as correlation contraction). The link between the two and the generalization to projective space was identified later by Hartley [22], [24]. Additional work in this area can be found in [47], [17], [53], [26], [44], [3], [4], [50].

A.2 Tensor Embedding of the Fundamental Matrix

We return to (14) and consider the case where the third image coincide with the second. The camera matrices for both images are $\mathbf{A} = [A; v']$ and this special tensor can be written as:

$$\mathcal{F}_i^{jk} = v'^j a_i^k - v'^k a_i^j, \tag{16}$$

which is composed of the elements of the fundamental matrix, as the following lemma shows.

LEMMA 1. *The two-view-tensor $\mathcal{F}_i^{jk}$ is composed of the elements of the fundamental matrix:*

$$\mathcal{F}_i^{jk} = \epsilon^{ljk} F_{li}$$

where F_{li} is the fundamental matrix and ϵ^{ljk} is the cross-product tensor.

PROOF. We consider (14) with $\mathcal{F}_i^{jk} = \epsilon^{ljk} F_{li}$ to derive the following equalities:

$$p^i s_j r_k \mathcal{F}_i^{jk} =$$
$$p^i s_j r_k \left(\epsilon^{ljk} F_{li}\right) =$$
$$p_i \underbrace{\left(s_j r_k \epsilon^{ljk}\right)}_{p'^l} F_{li} = 0. \tag{17}$$

□

The two-view-tensor is an admissible tensor that embodies the fundamental matrix in a three-image-framework. The algorithm that works with the trilinear tensor of three views can work with this tensor as well. Further details can be found in [5].

ACKNOWLEDGMENTS

An on-line demo can be found on the World Wide Web at http://www.cs.huji.ac.il/labs/vision/demos/demo.html. Amnon Shashua would like to acknowledge US-IS BSF contract 94-00120 and the European ACTS project AC074.

REFERENCES

[1] E.H. Adelson and J.R. Bergen, "The Plenoptic Function and the Elements of Early Vision," *Computational Models of Visual Processing* M. Landy and J.A. Movshon, eds., Chapter 1. Cambridge, Mass.: MIT Press, 1991.

[2] S. Avidan, T. Evgeniou, A. Shashua, and T. Poggio, "Image-Based View Synthesis By Combining Trilinear Tensors and Learning Techniques," *VRST*, 1997.

[3] S. Avidan and A. Shashua, "Tensorial Transfer: On the Representation of $n > 3$ Views of a 3D Scene," *Proc. ARPA Image Understanding Workshop*, 1996.

[4] S. Avidan and A. Shashua, "Novel View Synthesis in Tensor Space," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997.

[5] S. Avidan and A. Shashua, "Unifying Two-View and Three-View Geometry," *Proc. ARPA Image Understanding Workshop*, 1997.

[6] S. Avidan and A. Shashua, "Threading Fundamental Matrices," *Proc. European Conf. Computer Vision*, June 1998.

[7] E.B. Barrett, P.M. Payton, and G. Gheen, "Robust Algebraic Invariant Methods with Applications in Geometry and Imaging," *Proc. SPIE Remote Sensing*, July 1995.

[8] T. Beier and S. Neely, "Feature-Based Image Metamorphosis," *Proc. SIGGRAPH*, 1992.

[9] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, "Hierarchical Model-Based Motion Estimation," *Proc. European Conf. Computer Vision*, 1992.

[10] D. Beymer, A. Shashua, and T. Poggio, "Example Based Image Analysis and Synthesis," Technical Report A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Inst. of Technology, 1993.

[11] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.*, vol. 31, pp. 532-540, 1983.

[12] S.E. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proc. SIGGRAPH*, 1993.

[13] S.E. Chen, "QuickTimeVR—An Image-Based Approach to Virtual Environment Navigation," *Proc. SIGGRAPH*, 1995.

[14] T. Evgeniou, "Image-Based Rendering Using Algebraic Techniques," master's thesis, Massachusetts Inst. of Technology, June 1996.

[15] O. Faugeras and T. Papadopoulo, "A Nonlinear Method for Estimation the Projective Geometry of 3 Views," *Proc. Int'l Conf. Computer Vision*, 1998.

[16] O.D. Faugeras, "What Can Be Seen in Three Dimensions with an Uncalibrated Stereo Rig?" *Proc. European Conf. Computer Vision*, 1992.

[17] O.D. Faugeras and B. Mourrain, "On the Geometry and Algebra of the Point and Line Correspondences Between N Images," *Proc. Int'l Conf. Computer Vision*, 1995.

[18] O.D. Faugeras and L. Robert, "What Can Two Images Tell Us About a Third One?" *Proc. European Conf. Computer Vision*, 1994.

[19] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The Lumigraph," *Proc. SIGGRAPH*, pp. 43-54, 1996.

[20] W.E.L. Grimson, "Why Stereo Vision Is Not Always About 3D Reconstruction," Technical Report A.I. Memo No. 1435, Artificial Intelligence Laboratory, Massachusetts Inst. of Technology, 1993.

[21] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf. Intelligence*, pp. 189-192, 1988.

[22] R. Hartley, "Lines and Points in Three Views—A Unified Approach," *Proc. ARPA Image Understanding Workshop*, 1994.

[23] R. Hartley, "In Defence of the 8-Point Algorithm," *Proc. Int'l Conf. Computer Vision*, 1995.

[24] R. Hartley, "A Linear Method for Reconstruction from Lines and Points," *Proc. Int'l Conf. Computer Vision*, 1995.

[25] R. Hartley, "Minimizing Algebraic Error in Geometric Estimation Problems," *Proc. Int'l Conf. Computer Vision*, 1998.

[26] A. Heyden, "Reconstruction from Image Sequences By Means of Relative Depths," *Proc. Int'l Conf. Computer Vision*, 1995.

[27] A. Heyden and K. Astrom, "Algebraic Varieties in Multiple View Geometry," *Proc. European Conf. Computer Vision*, Apr. 1996.

[28] W. Hoff and N. Ahuja, "Surfaces from Stereo: Integrating Feature Matching, Disparity Estimation, and Contour Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 121-136, Feb. 1989.

[29] S. Laveau and O. Faugeras, "Oriented Projective Geometry for Computer Vision," *Proc. European Conf. Computer Vision*, 1996.

[30] S. Laveau and O.D. Faugeras, "3-D Scene Representation as a Collection of Images," *Proc. Int'l Conf. Pattern Recognition*, 1994.

[31] M. Levoy and P. Hanrahan, "Light Field Rendering," *Proc. SIGGRAPH*, pp. 31-42, 1996.

[32] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. IJCAI*, pp. 674-679, Vancouver, Canada, 1981.

[33] L. McMillan and G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System," *Proc. SIGGRAPH*, 1995.

[34] A. Zisserman P.A. Beardsley, and D.W. Murray, "Sequential Updating of Projective and Affine Structure from Motion," *Int'l J. Computer Vision*, vol. 23, no. 3, pp. 235-260, 1997.

[35] S. Peleg and J. Herman, "Panoramic Mosaic by Manifold Projection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997.

[36] P.H.S. Torr A. Zisserman, and D. Murray, "Motion Clustering Using the Trilinear Constraint Over Three Views," *Proc. Workshop Geometrical Modeling and Invariants for Computer Vision*, 1995.

[37] T. Poggio and R. Brunelli, "A Novel Approach to Graphics," Technical Report A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Inst. of Technology, 1992.

[38] M. Irani, R. Kumar, P. Anandan, J. Bergen, and K. Hanna, "Representation of Scenes from Collections of Images," *Proc. IEEE Workshop Representation of Visual Scenes*, 1995.

[39] B. Rousso, S. Avidan, A. Shashua, and S. Peleg, "Robust Recovery of Camera Rotation from Three Frames," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996.

[40] S.M. Seitz and C.R. Dyer, "Physically-Valid View Synthesis By Image Interpolation," *Proc. IEEE Workshop Representation of Visual Scenes*, 1995.

[41] S.M. Seitz and C.R. Dyer, "View Morphing," *Proc. SIGGRAPH*, pp. 21-30, 1996.

[42] A. Shashua, "Algebraic Functions for Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 779-789, Aug. 1995.

[43] A. Shashua and P. Anandan, "The Generalized Trilinear Constraints and the Uncertainty Tensor," *Proc. ARPA Image Understanding Workshop*, 1996.

[44] A. Shashua and S. Avidan, "The Rank4 Constraint in Multiple View Geometry," *Proc. European Conf. Computer Vision*, 1996. Also in CIS report 9520, Technion, Nov. 1995.

[45] A. Shashua and S.J. Maybank, "Degenerate n Point Configurations of Three Views: Do Critical Surfaces Exist?" Technical Report TR 96-19, Hebrew Univ. of Jerusalem, Nov. 1996.

[46] A. Shashua and S. Toelg, "The Quadric Reference Surface: Applications in Registering Views of Complex 3D Objects," *Proc. European Conf. Computer Vision*, 1994.

[47] A. Shashua and M. Werman, "On the Trilinear Tensor of Three Perspective Views and Its Underlying Geometry," *Proc. Int'l Conf. Computer Vision*, 1995.

[48] M.E. Spetsakis and J. Aloimonos, "Structure from Motion Using Line Correspondences," *Int'l J. Computer Vision*, vol. 4, no. 3, pp. 171-183, 1990.

[49] M.E. Spetsakis and J. Aloimonos, "A Unified Theory of Structure from Motion," *Proc. ARPA Image Understanding Workshop*, 1990.

[50] G. Stein and A. Shashua, "Model Based Brightness Constraints: On Direct Estimation of Structure," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997.

[51] R. Szeliski and S.B. Kang, "Direct Methods for Visual Scene Reconstruction," *Proc. IEEE Workshop Representation of Visual Scenes*, 1995.

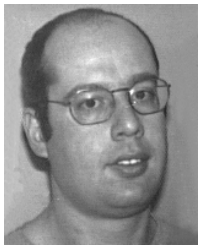
[52] P.H.S. Torr and A. Zisserman, "Robust Parametrization and Computation of the Trifocal Tensor," *Image and Vision Computing*, vol. 15, pp. 591-607, 1997.

[53] B. Triggs, "Matching Constraints and the Joint Image," *Proc. Int'l Conf. Computer Vision*, 1995.

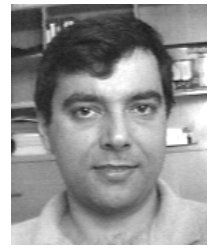
[54] J. Weng, T.S. Huang, and N. Ahuja, "Motion and Structure from Line Correspondences: Closed Form Solution, Uniqueness and Optimization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 3, Mar. 1992.

[55] T. Werner, R.D. Hersch, and V. Hlavac, "Rendering Real-World Objects Using View Interpolation," *Proc. Int'l Conf. Computer Vision*, 1995.

[56] G. Wolberg, *Digital Image Warping*. Los Alamitos, Calif.: IEEE CS Press, 1992.



Shai Avidan received the BSc degree in mathematics and computer science from Bar-Ilan University, Ramat-Gan, Israel, in 1993. Currently, he is a PhD candidate at the Hebrew University, where his research interests are in computer vision and computer graphics. In addition, he has been working for the past 10 years in the industry in the fields of CAD, GIS, and photogrammetry.



Amnon Shashua received the BSc degree in mathematics and computer science from Tel-Aviv University, Tel-Aviv, Israel, in 1986; the MSc degree in mathematics and computer science from the Weizmann Institute of Science, Rehovot, Israel, in 1989; and the PhD degree in computational neuroscience, working at the Artificial Intelligence Laboratory, from the Massachusetts Institute of Technology, in 1993.Dr. Shashua is a senior lecturer at the Institute of Computer Science, The Hebrew University of Jerusalem,

His research interests are in computer vision and computational modeling of human vision. His previous work includes early visual processing of saliency and grouping mechanisms, visual recognition, image synthesis for animation and graphics, and theory of computer vision in the areas of three-dimensional processing from a collection of two-dimensional views.