# Novel View Synthesis in Tensor Space *

Shai Avidan Amnon Shashua

Institute of Computer Science
The Hebrew University
Jerusalem 91904, Israel
{avidan,shashua}@cs.huji.ac.il

## Abstract

*We present a new method for synthesizing novel views of a 3D scene from few model images in full correspondence. The core of this work is the derivation of a tensorial operator that describes the transformation from a given tensor of three views to a novel tensor of a new configuration of three views. By repeated application of the operator on a seed tensor with a sequence of desired virtual camera positions we obtain a chain of warping functions (tensors) from the set of model images to create the desired virtual views.*

## 1. Introduction

This paper addresses the problem of synthesizing a novel image, from an arbitrary viewing position, given a small number of model 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 model images without ever computing explicitly 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 photo-realistic images from viewpoints within a 60 degrees 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 multi-linear matching constraints, known as trilinearities, that were first introduced in [31]. The trilinearities provide a general (not subject to singular camera configurations) warping function from model 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 model cameras, yet is general and robust.

The core of this work is the derivation of a tensorial 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 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 model images (from which the seed tensor was computed) to create the desired virtual views. We also show that the process can start with two model views by having the "seed" tensor be comprised of the elements of the fundamental matrix of the model views.

### 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 computational-intensive process of acquiring a 3D model followed by rendering, and instead to use a number of model 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: (i) image interpolation, (ii) Off-line (Mosaic-based) synthesis, and (iii) On-line synthesis. The first class, image interpolation, is designed to create "in-between" images among two or more model images. This includes image morphing [5], direct interpolation from image-flows ("multi-dimensional morphing") [7, 27], image interpolation using 3D models instead of image-flow [9], and "physically correct" image interpolation [30, 39]. All but the last

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

two 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 model images.

Instead of flow-field interpolation among the model images, it is possible to interpolate directly over the plenoptic function [1] — a function which represents the amount of light emitted at each point in space as a function of direction. Levoy et al. [22] and Gortler et al. [14] interpolate between a dense set of several thousands of 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 model 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 [15, 10] in what is known as "QuickTime VR". Szeliski and Kang [37] create high-resolution mosaics from low-resolution video streams, and Peleg and Herman [25] 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 model images.

The major limitations of the aforementioned techniques is that a relatively large number of model images is required to represent an object. The third class, on-line synthesis, along the lines of this paper reduces the number of acquired (model) images by exploiting the 3D-from-2D geometry for deriving an on-line warping function from a set of model images to create novel views on-the-fly based on user specification of the virtual camera position. Laveau and Faugeras [20] 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 (like when the virtual camera center is collinear with the centers of the model 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 [24] 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 model images/mosaics [16]. 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 model images to useful algebraic structures that would suffice for generating new views. For example, it is likely that the base-line between model 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.

*Non-singular 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. [13]) 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 satisfy all three requirements. For example, [20] satisfy the first requirement at the cost of complicating the driving mode by specifying control points; using depth maps provide 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 implicit the scene and the camera parameters, does not suffer from singularities, and is governed by the prevalent driving mode used by most 3D viewers.

## 2 View Synthesis in Tensor Space

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.

## 2.1 The Trilinear Warping Function

The trilinear tensor concatenates together the camera transformation matrices (camera locations) across three views, as follows. Let $P$ be a point in 3D projective space projecting onto $p, p', p''$ in three views $\psi, \psi', \psi''$ respectively, represented by the two dimensional projective space. The relationship between the 3D and the 2D spaces is represented by the $3 \times 4$ matrices, $[I, 0]$, $[A, v']$ and $[B, v'']$, i.e.,

$$\begin{array}{rcl} p & = & [I,0]P \\ p' & \cong & [A,v']P \\ p'' & \cong & [B,v'']P \end{array}$$

Where $I$ is the identity matrix and $A$ and $B$ are homography matrices due to some plane in space from $\psi$ to $\psi'$ and $\psi''$, respectively (in particular, rotation is the homography matrix due to the plane at infinity in case the cameras are calibrated). The vectors $v'$ and $v''$ are known as epipolar points (represent the translational component of camera motion when the cameras are internally calibrated).

We may adopt the convention that $p = (x, y, 1)^{\top}$, $p' = (x', y', 1)^{\top}$, $p'' = (x'', y'', 1)^{\top}$ and $P = [x, y, 1, \rho]$. The coordinates $(x, y)$, $(x'y')$, $(x'', y'')$ are matching points across the three images.

The trilinear tensor is an array of 27 entries:

$$\alpha_i^{jk} = v'^j b_i^k - v''^k a_i^j. \quad i,j,k = 1,2,3 \qquad (1)$$

where superscripts denote contravariant indices (representing points in the 2D plane, like $v'$) and subscripts denote covariant indices (representing lines in the 2D plane, like the rows of $A$). Thus, $a_i^k$ is the element of the k'th row and i'th column of $A$, and $v'^k$ is the k'th element of $v'$ (see Appendix A on tensorial notations). The tensor $\alpha_i^{jk}$ forms the set of coefficients of certain trilinear forms that vanish on any corresponding triplet $p, p', p''$ (i.e., functions of views that are invariant to object structure).

$$p^i s_j^{\mu} r_k^{\rho} \alpha_i^{jk} = 0 \qquad (2)$$

where $s_j^{\mu}$ are any two lines ($s_j^1$ and $s_j^2$) intersecting at $p'$, and $r_k^{\rho}$ are any two lines intersecting at $p''$ (see Fig. 1).

Since each of the free indices $\mu, \rho$ is in the range 1,2, we have 4 trilinear equations which are unique up to linear combinations. If we choose the canonical form where $s$ and $r$ represent vertical and horizontal lines, then the four trilinear forms, refered to as trilinearities, are expanded as follows:

$$\begin{aligned} x''\boldsymbol{\alpha}_i^{13}p^i - x''x'\boldsymbol{\alpha}_i^{33}p^i + x'\boldsymbol{\alpha}_i^{31}p^i - \boldsymbol{\alpha}_i^{11}p^i &= 0, \\ y''\boldsymbol{\alpha}_i^{13}p^i - y''x'\boldsymbol{\alpha}_i^{33}p^i + x'\boldsymbol{\alpha}_i^{32}p^i - \boldsymbol{\alpha}_i^{12}p^i &= 0, \\ x''\boldsymbol{\alpha}_i^{23}p^i - x''y'\boldsymbol{\alpha}_i^{33}p^i + y'\boldsymbol{\alpha}_i^{31}p^i - \boldsymbol{\alpha}_i^{21}p^i &= 0, \\ y''\boldsymbol{\alpha}_i^{23}p^i - y''y'\boldsymbol{\alpha}_i^{33}p^i + y'\boldsymbol{\alpha}_i^{32}p^i - \boldsymbol{\alpha}_i^{22}p^i &= 0. \end{aligned}$$

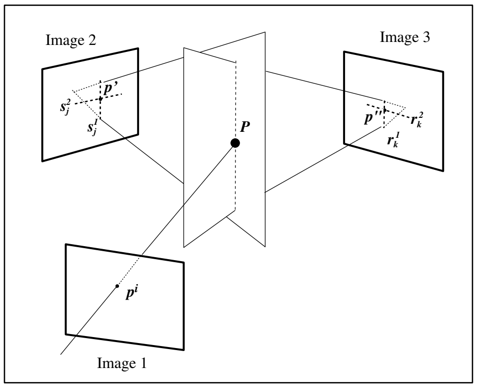


**Figure 1.** Each of the four trilinear equations describes a matching between a point $p$ in the first view, some line $s_j^{\mu}$ passing through the matching point $p'$ in the second view and some line line $r_k^{\rho}$ passing through the matching point $p''$ in the third view. In space, this constraint is a meeting between a ray and two planes.

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 $\alpha_i^{jk}$. These constraints first became prominent in [31] and the underlying theory has been studied intensively in [36, 18, 32, 12, 38, 19, 33].

One can readily see that given two views in full correspondence and the tensor (recovered using 7 matching points with a third view), the entire third view can be synthesized by means of (forward) warping: simply, from each trilinearity one can extract either $x''$ or $y''$, thus for every matching pair $p, p'$ we can obtain $p''$ and copy the appropriate brightness value (take the average from the two model images, for example). This process is referred to as “reprojection” in the literature. There are alternative ways of performing reprojection, but if we would like to do it without recovering first a 3D model of the scene, the trilinear tensor generally provides the best results (see [4, 31, 34]).

We have described so far the implementation of the reprojection paradigm via the trilinear equations. In other words, given two model views and a tensor, the third view is uniquely determined and can be synthesized by means of a warping function applied to the two model images. 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.

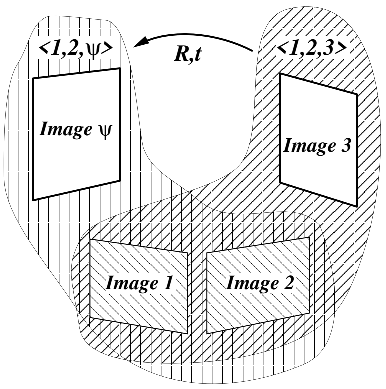


**Figure 2.** We generate tensor $< 1, 2, \psi >$, that relates images $1, 2$ with some novel image $\psi$, from the seed tensor $< 1, 2, 3 >$ and the virtual camera motion parameters $(R, t)$ from image 3 to image $\psi$. Tensor $< 1, 2, 3 >$ relates images $1, 2$ and $3$ and is computed only once at the pre-processing stage. Tensor $< 1, 2, \psi >$ is computed every time the user specifies a new $(R, t)$. We use tensor $< 1, 2, \psi >$ to render the novel image (image $\psi$) from model images $1, 2$.

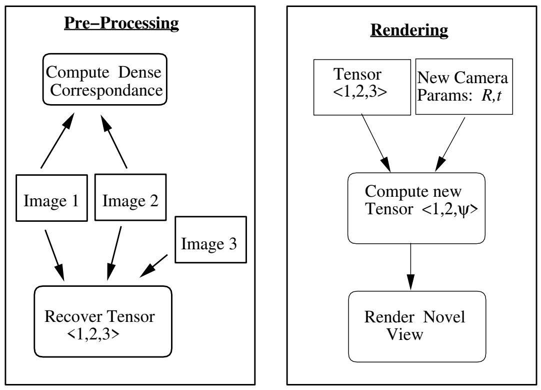


**Figure 3.** View synthesis is divided into two parts. The pre-processing stage, done only once and the actual rendering done for every image.

## 2.2 The basic Tensorial Operator

The basic tensorial 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).

Going back to the tensor of views $< 1, 2, 3 >$ described in eqn. (1), we note that the homographies $A$ and $B$ correspond to an arbitrary plane, and the choice of the plane does not change the tensor coefficients [31]. In particular, we may select the plane at infinity, i.e.,

$$A = M'R'M^{-1} \qquad\qquad B = M''R''M^{-1}$$

where $M, M', M''$ are the internal parameters matrices of the cameras and $R', R''$ are the rotational components. Assuming that the cameras are internally calibrated, i.e., that $M = M' = M'' = I$, the tensor of three *calibrated* model views is:

$$\alpha_i^{jk} = v'^j R''^k_i - v''^k R'^j_i. \tag{3}$$

Note that $v', v''$ are the translational components of camera motion — now that the camera is assumed internally calibrated.

Consider the tensor $\alpha_i^{jk}$ of the views $< 1, 2, 3 >$, and assume the user wishes to to apply an incremental change of position of the third image, i.e., rotate the third camera position by the $3 \times 3$ coordinate matrix $R$, and translate it by the $3 \times 1$ translation vector $t$ — this motion would result to a novel view, call it view $\psi$. Hence, the camera matrix from image 1 to image $\psi$ is $[R_l^k R''^l_i; v''^l R_l^k + t^k]$ and the tensor $\gamma_i^{jk}$ of the configuration of views $< 1, 2, \psi >$ is:

$$\begin{aligned}
\gamma_i^{jk} &= v'^j (R_l^k R''^l_i) - (v''^l R_l^k + t^k) R'^j_i \\
&= R_l^k v'^j R''^l_i - R_l^k v''^l R'^j_i - t^k R'^j_i \\
&= R_l^k \alpha_i^{jl} - t^k R'^j_i.
\end{aligned} \tag{4}$$

and we have arrived at a linear equation, which is the tensorial operator, that generates the tensor $\gamma_i^{jk}$ of views $< 1, 2, \psi >$ from the seed tensor $\alpha_i^{jl}$ of views $< 1, 2, 3 >$, the virtual camera parameters $R, t$ and the rotation matrix $R'$ between the two model images. No control points across the views are needed, only applying the tensorial operator. In appendix B we elaborate on how to recover $R'$ directly from the tensor $\alpha_i^{jl}$.

To summarize, starting from a seed tensor and dense correspondence between two of the model images, then by repeated application of eqn. (4) based on user-specified position $R, t$ of the virtual camera we obtain a sequence of tensors — each serving as a warping function from the model views onto the desired novel view (see Fig. 3).

## 3. The Seed Tensor of Two views

Given two acquired images we can construct a special tensor composed of the elements of the fundamental matrix [11] that can serve as a seed tensor that starts the chain of tensors, as follows. Consider a configuration of three views in which views 2,3 coincide, i.e., Eq. 1 becomes:

$$\alpha_i^{jk} = v'^j a_i^k - v'^k a_i^j \tag{5}$$

where $\alpha_i^{jk}$ is the tensor of the image triplet $< 1, 2, 2 >$. It can be readily verified that the elements of $\alpha_i^{jk}$ are composed of the fundamental matrix $F$, and $-F$, and the remaining (nine) elements vanish:

$$\begin{array}{lcll} \alpha_i^{jk} & = & f_{(6-k-j)i} & if \quad k = (j+1) mod 3 \\ \alpha_i^{jk} & = & -f_{(6-k-j)i} & if \quad j = (k+1) mod 3 \\ \alpha_i^{jk} & = & 0 & if \quad j = k \end{array} \tag{6}$$

where $f_{ji}$ means the element in the $j$-th row and the $i$-th column of the fundamental matrix $F$. As shown in [3], the rank of $\gamma_i^{jk}$ is 2 whereas the rank of the tensor of three distinct views is 4 — but otherwise all other properties remain and, in particular, $\alpha_i^{jk}$ can serve as the first tensor that starts the synthesis process described above [3].

In other words, the synthesis process can start with two model views and their fundamental matrix, as in [20], but the later steps follow the trilinear tensor machinery which ensures lack of singular configurations and provide a natural driving mode — thus satisfying the three requirements described in Section 1.

## 4. 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 model images or the fundamental matrix, in its tensor form, for two model 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 [23, 6], recovery of the trilinear tensor [31, 36, 12] and recovery of the fundamental matrix [11, 17]. Any of the methods described there can be used with our algorithm. Here we give our implementation.

### 4.1. Dense Correspondence

We obtain dense correspondence using a Lucas-Kanade style optical-flow [23] embedded in a hierarchical framework [6]. For every pixel we estimate its motion $(u, v)$ using the well known equation:

$$\begin{pmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_x I_t \\ -I_y I_t \end{pmatrix}$$

where $I_x, I_y, I_t$ are the sum of derivatives in the $x, y$-directions and time respectively in a $5 \times 5$ window centered around the pixel. We construct a Laplacian pyramid [8] and recover the motion parameters at each level, using the estimate of the previous level as our initial guess. Typically we have 5 levels in the pyramid and we perform 2 to 4 iterations per level. The initial motion estimation at the coarsest level is 0.

### 4.2. Robust Recovery of the Seed Tensor

The seed tensor is recovered from the model images and since the number of model images may be either two or three, a slightly different algorithms are needed. We describe the procedure for recovering the two-view tensor and inform the user when it deviates from the algorithm for recovering the three-view tensor. The steps for computing the tensor are:

- **Find Matching Points**

  For every pixel in the first image we compute its Hessian matrix
  $$\begin{pmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{pmatrix}$$
  where $I_x, I_y$ are the sum of derivatives in the $x-$ and $y-$directions respectively in a $5 \times 5$ window centered around the pixel, and extract its eigenvalues. We select points with their smaller eigenvalue above some predefined threshold. 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 are accepted.

- **Robust Estimation of the Tensor** The previous stage usually produces several hundreds points, refered to as "good" points, which are then normalized to ensure good numerical solution [17]. 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 [26], to avoid the effects of outliers. The robust estimation is done by a repetitive lottery of a subset of 8 "good" points (7 "good" points in case of three 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.

Finally, in case of two-view tensor, we need to convert the recovered fundamental matrix into a tensorial form using equation 6.

#### 4.2.1 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 equation 2 to obtain the coordinates of the point in the novel view. Let $s_j^m$ be the matrix,

$$s = \begin{bmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \end{bmatrix}$$

and, similarly, let $r_k^l$ be the matrix,

$$r = \begin{bmatrix} u'' & 0 & -x'' \\ 0 & u'' & -y'' \end{bmatrix}$$

and solve equation 2, in a least-squared manner, for $p'' = (x'', y'', u'')^T$. The image coordinate in the third image are then $(\frac{x''}{u''}, \frac{y''}{u''})$.

#### 4.2.2 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. [20, 28, 35]). 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 $\gamma_i^{jk}$ [36]. 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 [21], nevertheless, the procedure is fairly robust in practice.

## 5. Experimental Results

The tensor-based rendering method was implemented on synthetic and real images. Specifically, we concentrated on the case where only a pair of model images are given. 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 the second model frame. 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. An example is shown in Fig. 4. The two model images were captured using a standard indy camera at a resolution of $640 \times 480$ pixels.

We extended our approach to handle dynamic scene as well, by treating it as a series of static scenes in time. A pair of stationary cameras captured a flexible object (in this case facial expressions) and for each such pair a novel view was synthesized. The result is a fly-through around a "talking head". Figure 5 demonstrates some of the input images, as well as the synthesized views generated.

## 6. 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 tensorial 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). We are currently working on modeling non-rigid as well as rigid transformation under this framework [2].

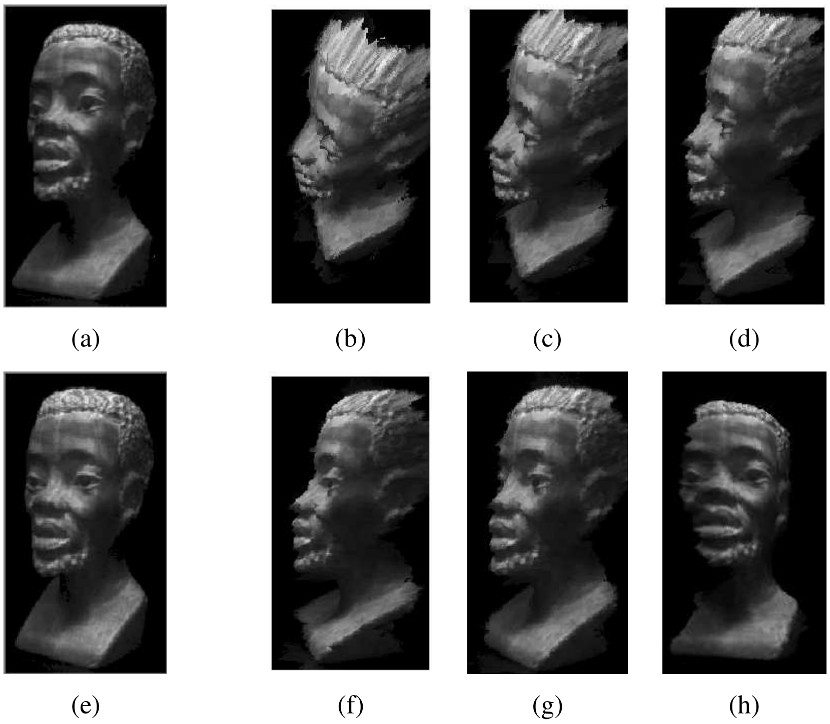

**Figure 4.** An example of image synthesis using optic-flow and a tensor. The original images are framed ((a) and (e)), the rest of the images are taken from the generated movie. They should be seen left to right, top to bottom.

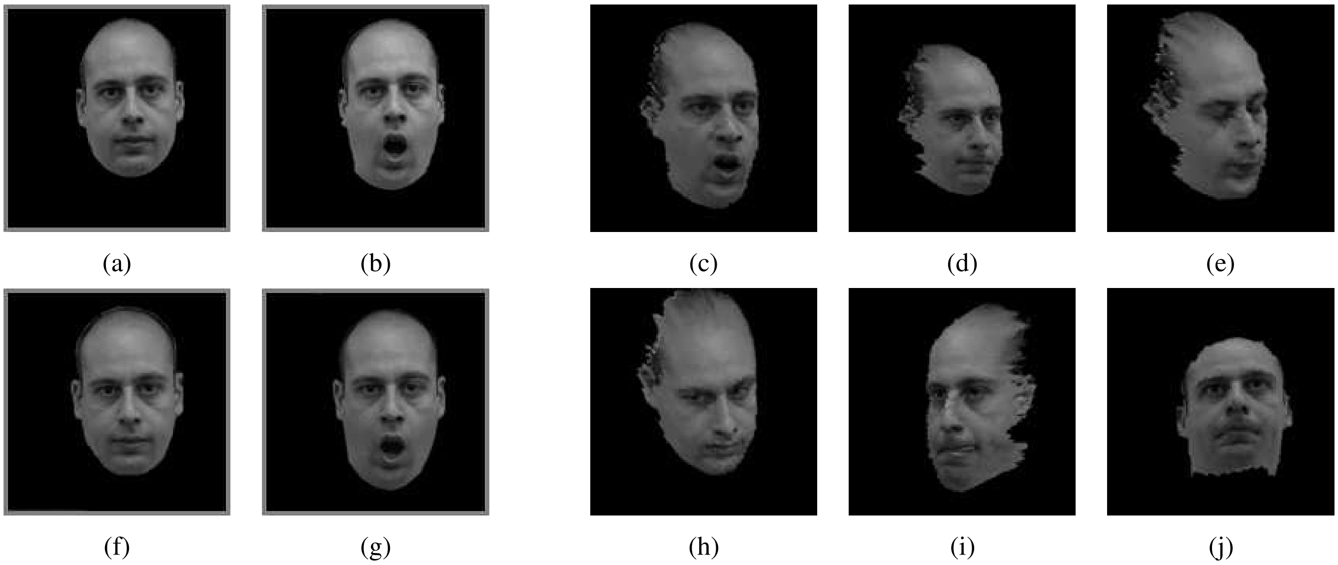

**Figure 5.** A sample from the virtual movie of the "talking head". The original image pairs are framed ((a),(f) and (b),(g)). The rest are taken from the virtual movie.

## References


[1] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*. The MIT Press, Cambridgem, Mass., 1991. Chapter 1.

[2] S. avidan, T. Evgeniou, A. Shashua, and T. Poggio. Image-based view synthesis. Technical report, A.I. Memo No. 1603, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1997.

[3] S. Avidan and A. Shashua. Unifying two-view and three-view geometry. In *ARPA, Image Understanding Workshop*, 1997.

[4] E. Barrett, P. Payton, and G. Gheen. Robust algebraic invariant methods with applications in geometry and imaging. In *Proceedings of the SPIE on Remote Sensing*, July 1995.

[5] T. Beier and S. Neely. Feature-based image metamorphosis. In *SIGGRAPH*, 1992.

[6] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, 1992.

[7] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. Technical report, A.I. Memo No. 1431, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[8] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31:532–540, 1983.

[9] S. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH*, 1993.

[10] S. E. Chen. QuickTimeVR - an image-based approach to virtual environment navigation. In *SIGGRAPH*, 1995.

[11] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision*, 1992.

[12] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proceedings of the International Conference on Computer Vision*, 1995.

[13] O. Faugeras and L. Robert. What can two images tell us about a third one? In *Proceedings of the European Conference on Computer Vision*, 1994.

[14] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[15] N. Greene. Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. IEEE Computer Graphics and Applications, 6(6):21–27, 1986.

[16] W. Grimson. Why stereo vision is not always about 3D reconstruction. Technical report, A.I. Memo No. 1435, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[17] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the International Conference on Computer Vision*, 1995.

[18] R. Hartley. A linear method for reconstruction from lines and points. In *Proceedings of the International Conference on Computer Vision*, 1995.

[19] A. Heyden. Reconstruction from image sequences by means of relative depths. In *Proceedings of the International Conference on Computer Vision*, 1995.

[20] S. Laveau and O. Faugeras. 3-d scene representation as a collection of images. In *Proceedings of the International Conference on Pattern Recognition*, 1994.

[21] S. Laveau and O. Faugeras. Oriented projective geometry for computer vision. In *Proceedings of the European Conference on Computer Vision*, 1996.

[22] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996.

[23] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. Proceedings IJCAI, pages 674–679, Vancouver, Canada, 1981.

[24] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, 1995.

[25] S. Peleg and J. Herman. Panoramic mosaic by manifold projection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[26] T. P.H.S., Z. A., and M. D. Motion clustering using the trilinear constraint over three views. Workshop on Geometrical Modeling and Invariants for Computer Vision, 1995.

[27] T. Poggio and R. Brunelli. A novel approach to graphics. Technical report, A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[28] M. I. Rakesh Kumar, P. Anandan, J. Bergen, and K. Hanna. Representation of scenes from collections of images. IEEE Workshop on Representation of Visual Scenes, 1995.

[29] B. Rousso, S. Avidan, A. Shashua, and S. Peleg. Robust recovery of camera rotation from three frames. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[30] S. M. Seitz and C. R. Dyer. Physically-valid view synthesis by image interpolation. IEEE Workshop on Representation of Visual Scenes, 1995.

[31] A. Shashua. Algebraic functions for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):779–789*, 1995.

[32] A. Shashua and P. Anandan. The generalized trilinear constraints and the uncertainty tensor. In *Proceedings of the ARPA Image Understanding Workshop*, 1996.

[33] A. Shashua and S. Avidan. The rank4 constraint in multiple view geometry. In *Proceedings of the European Conference on Computer Vision*, 1996. Also in CIS report 9520, November 1995, Technion.

[34] A. Shashua and S. Maybank. Degenerate $n$ point configurations of three views: Do critical surfaces exist? Technical report, Technical report, Hebrew University of Jerusalem, November 1996.

[35] A. Shashua and S. Toelg. The quadric reference surface: Applications in registering views of complex 3d objects. In *Proceedings of the European Conference on Computer Vision*, 1994.

[36] A. Shashua and M. Werman. On the trilinear tensor of three perspective views and its underlying geometry. In *Proceedings of the International Conference on Computer Vision*, 1995.

[37] R. Szeliski and S. Kang. Direct methods for visual scene reconstruction. IEEE Workshop on Representation of Visual Scenes, 1995.

[38] B. Triggs. Matching constraints and the joint image. In *Proceedings of the International Conference on Computer Vision*, 1995.

[39] T. Werner, R. Hersch, and V. Hlavac. Rendering real-world objects using view interpolation. In *Proceedings of the International Conference on Computer Vision*, 1995.

## A On Tensorial Notations

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 hyper-planes — 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. For example, if $p$ is a point incident to a line $s$ in $\mathcal{P}^2$, then $p^i s_i = 0$. Matrices have two indices and the transformation they represent depends on the covariant-contravariant positioning of the indices. $a_i^j$ is a mapping from points to points, and hyper-planes to hyper-planes, because $a_i^j p^i = q^j$ and $a_i^j s_j = r_i$ (in matrix form: $Ap = q$ and $A^\top s = r$); $a_{ij}$ maps points to hyper-planes; and $a^{ij}$ maps hyper-planes to points. When viewed as a matrix the row and column positions are determined accordingly: in $a_i^j$ and $a_{ji}$ the index $i$ runs over the columns and $j$ runs over the rows, thus $b_j^k a_i^j = c_i^k$ is $BA = C$ in matrix form. An outer-product of two vectors, $a_i b^j$, is a matrix $c_i^j$ whose $i, j$ entries are $a_i b^j$ — note that in matrix form $C = ba^\top$.

## B Direct Recovery of Rotations from Tensors

We recover the small-angles rotation matrix between two model images directly from the trilinear tensor, under the assumption that the cameras are calibrated.

$$\Omega_X = det \begin{pmatrix} \alpha_2^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K$$

$$\Omega_Y = det \begin{pmatrix} -\alpha_1^{j3} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K$$

$$\Omega_Z = det \begin{pmatrix} \alpha_1^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} / K$$

$$K = det \begin{pmatrix} \alpha_2^{j2} \\ \alpha_2^{j3} + \alpha_3^{j2} \\ \alpha_3^{j3} - \alpha_2^{j2} \end{pmatrix} \quad (7)$$

where $\alpha_2^{j2}$ stands for $(\alpha_2^{12},\ \alpha_2^{22},\ \alpha_2^{32})$, 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 $\alpha_i^{jk(n+1)} = R'^{l\,(n)}_j \alpha_i^{lk(n)}$, where $R'^{l\,(n)}_j$, $\alpha_i^{lk(n)}$ are the recovered rotation matrix in the previous iteration and the previous tensor, respectively. This method was first presented in [29] for the purpose of video stabilization.