CENTER FOR
MACHINE PERCEPTION

CZECH TECHNICAL
UNIVERSITY

REPRINT

Efficient 3-D Scene Visualization by Image Extrapolation

Tomáš Werner
Tomáš Pajdla
Václav Hlaváč

Reference

Tomas Werner, Tomas Pajdla, and Vaclav Hlavac. Efficient 3-D scene visualization by image extrapolation. In *European Conf. Computer Vision*, 1998.

March 13, 1998

Czech Technical University, Faculty of Electrical Engineering
Department of Control Engineering, Center for Machine Perception
121 35 Prague 2, Karlovo náměstí 13, Czech Republic
fax +420 2 24357385, phone +420 2 24357465, http://cmp.felk.cvut.cz

Efficient 3-D Scene Visualization by Image Extrapolation*

Tomáš Werner, Tomáš Pajdla, Václav Hlaváč

Center for Machine Perception
Faculty of Electrical Engineering, Czech Technical University
121 35 Prague 2, Karlovo náměstí 13, Czech Republic
http://cmp.felk.cvut.cz

Abstract. Image-based scene representation is believed to be an alternative to the 3-D model reconstruction and rendering. In attempt to compare generality of image-based and model-based approaches we argue that it is plausible to distinguish three approaches to 3-D scene visualization: image interpolation, image extrapolation, and 3-D model reconstruction and rendering. We advocate that image extrapolation is a useful trade-off between simple but limited interpolation and general but difficult 3-D model reconstruction and rendering. Image extrapolation is able to visualize correctly the part of a 3-D scene that is visible from two reference images. In fact, it is equivalent to reconstructing a projective 3-D model from two reference images and rendering it. In the second part of the work, we present an algorithm for rendering a projective model. Our approach is more efficient than the ray-tracing-like algorithm by Laveau and Faugeras [6]. We show that visibility can be solved by z-buffering, and that virtual images can be synthesized by transferring triangles from a reference image via a homography or an affinity. Such algorithms are often supported by hardware on graphics work stations, which makes a step towards the real-time synthesis. The results are presented for real scenes.

1 Introduction

This work considers visualization of a real 3-D scene described by a set of intensity *reference images* taken from different viewpoints. The novel images of the scene, not captured by any real camera, are called *virtual images.* Traditional scene visualization proceeds in two steps: (i) reconstruction of a 3-D model, (ii) rendering of the model.

Image-based scene representation is believed to be an alternative. It proposes to display a real 3-D scene from any viewpoint without using its 3-D model. The scene is represented only by a collection of 2-D reference images. A virtual image is then constructed directly from the reference images using correspondences

* This research was supported by the Grant Agency of the Czech Republic, grants 102/97/0480 and 102/97/0855, by European Union, grant Copernicus CP941068, and by the Czech Ministry of Education, grant VS96049.

among them. The aim of such a procedure is to avoid the difficult reconstruction of a consistent 3-D model.

It was thought [1, 6, 8, 10, 15, 5] that displaying a 3-D scene from 2-D reference images is quite a different process from rendering a 3-D model. The difference seemed to follow from the observation of Ullman [14] who has proposed that the objects could be recognized just on the basis of linear combination of corresponding points in their orthographic images. It was in deep contrast to recognition based on verification of a 3-D model projected to the image.

Ullman's approach has attracted new attention since Shashua showed that a trilinear function replaces the linear one [11] for a perspective camera, and since Laveau and Faugeras [6] and Hartley [4] made clear that any projective reconstruction of the scene suffices for the visualization itself. Tedious calibration of the camera has been thus avoided in the case of visualization.

Seitz and Dyer [10] have stressed that visualizing an object by interpolating close views is a well-posed process and therefore a perfect correspondence algorithm and removal of hidden surfaces are not ultimately needed for certain limited tasks. Other works have demonstrated that even quite complicated scenes can be visualized by interpolating between the reference views [8, 15].

The above results make an impression that virtual images of a general scene can be constructed by an image-based approach. However, knowledge about geometry of isolated points is not sufficient for constructing virtual images because *occlusion* must be also considered.

To display the scene by a camera revolving around it in a circle, quite many reference images were needed in [15] to make visual effect realistic enough. This is caused by the principal deficiency of image interpolation, namely by its *inability to show a general object from an arbitrary view point using the images and the correspondences obtained from a sparse set of viewpoints.* This deficiency is caused by occlusion. Surprisingly, no object, not even a convex polyhedron, can be completely visualized by mere interpolating between finite number of reference images.

Consider the situation illustrated in Fig. 1 when the reference views $C_1, \ldots, C_6$ are located around a simple convex polyhedron. The images taken by the virtual camera C, lying in the segment C_2C_3, cannot be constructed by interpolating between reference views C_2 and C_3 since the camera C_2 does not see both sides of the polyhedron which are seen by the camera C. It will not help to move one of the reference cameras, e.g. C_3, closer to B_1 in hope to avoid the malfunction of the view synthesis. By moving C_3 to B_1, the same problem appears on the segment C_3C_4. The only solution would be to increase the density of views near B_1 to infinity. Indeed, the algorithm for automatic finding the optimal sparse set of reference views [16] has tended to select many reference images near places like B_1.

In the light of this observation, we believe that the following questions must be clarified: How general are image-based approaches? Can an arbitrary virtual image of an arbitrary scene be constructed by an approach avoiding 3-D model reconstruction?

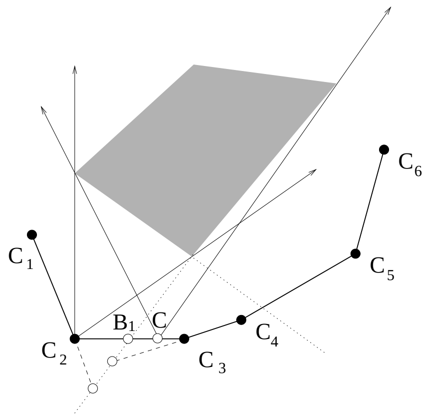


Fig. 1. Virtual view C cannot be constructed by interpolation from the views C_2 and C_3 but can be extrapolated from C_3 and C_4.

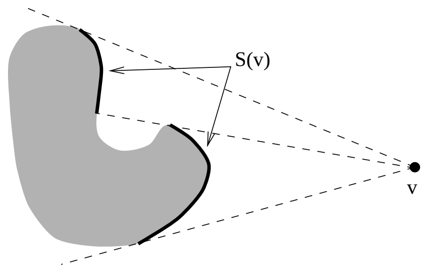


Fig. 2. The set $\mathcal{S}(v)$ of scene points visible from the viewpoint v.

In the first part of the work, Section 2, we attempt to compare generality of image-based and model-based approaches. We show that it is plausible to distinguish three qualitatively different approaches to constructing virtual images. The simplest one, *image interpolation*, is similar to Seitz's and Dyer's approach [10]. In *image extrapolation*, visibility must be solved explicitly. *3-D Model reconstruction and rendering* is the most general approach, the only one which can be used if both the scene and the position of virtual camera are general. It requires fusing parts of the virtual image transferred from different sets of reference images.

Since the applicability of image interpolation is limited and 3-D model reconstruction is difficult, we advocate *image extrapolation* as a trade-off useful in practice. The application can be, e.g., observing a scene described by a pair of uncalibrated photographs. In Section 3 we present an efficient algorithm for synthesizing virtual images by image extrapolation. The algorithm is similar to rendering a Euclidean 3-D model, however, no Euclidean reconstruction is needed and visibility can be solved in image planes. In contrary to Laveau and Faugeras [6], the rendering algorithm allows using hardware-supported graphics functions available on some work stations. Therefore, the synthesis of virtual images can be faster.

2 Analysis of Approaches to 3-D Scene Visualization

In this section the following questions are addressed: *Can an image-based approach be used to construct an arbitrary virtual image of an arbitrary 3-D scene? If not, for which configurations of the scene and virtual and reference viewpoints can it be used?*

2.1 Used Concepts

A *scene* is one or more rigid opaque bodies in a fixed relative position. *Scene points* are the points on the surface of the scene. *Viewpoints* are points that are not inside the scene nor on its surface. We will denote the *set of scene points visible from the viewpoint* v by $\mathcal{S}(v)$. The set $\mathcal{S}(v)$ for an example configuration is shown in[1] Fig. 2. Each viewpoint is a possible center of a camera. If it is a real camera, the viewpoint is called the *reference* viewpoint. If it is a center of a non-existing camera for which a novel view of the scene is to be created, it is called the *virtual* viewpoint.

We assume that an algorithm able to find corresponding pairs of pixels in two reference images is available. We further assume that there exists an algorithm that allows to compute the position of an image point in the virtual image from the positions of corresponding image points in some subset of reference images (this subset usually contains only two images). This algorithm is often called *transfer*. Approaches to transfer can be found in [12, 4, 2, 13].

2.2 Three Approaches to Constructing Virtual Images

The considered task is to construct virtual images of a scene from a captured set of reference images. The virtual viewpoint will be denoted by v and two reference viewpoints by v', v''.

If the scene were a cloud of isolated points we could construct an arbitrary virtual image by transferring all corresponding points from a pair of arbitrary reference images to the virtual image (in fact, this is image interpolation described below). Since scene surfaces occlude each other this simple approach cannot be used in some situations.

Possible configurations of the scene, the reference viewpoints and the virtual viewpoint can be divided into three qualitatively different classes. A different approach to constructing virtual images is suitable for each class so that:

- The virtual image is *complete*, that is, no part of the scene is missing in it.
- Possible *hidden surfaces are correctly removed* in the virtual image.
- The approach is as *simple and efficient* as possible.

The three classes of approaches are summarized in Table 1 and described in the following three sections. For each class, necessary and sufficient condition for configuration of the scene and the viewpoints is formulated in terms of $\mathcal{S}(v), \mathcal{S}(v'), \mathcal{S}(v'')$.

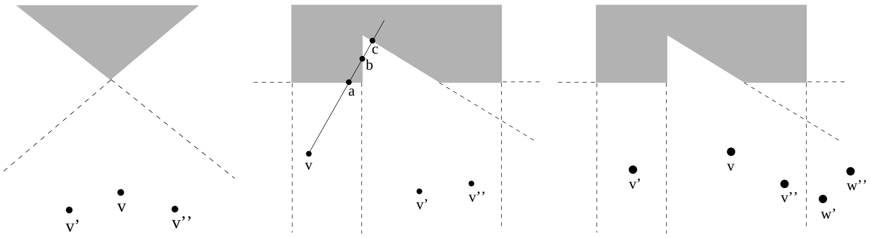
Image Interpolation. Image interpolation allows synthesizing the virtual image from two reference images of the scene by transferring points in an arbitrary order from the reference images to the virtual one.

Once we find the correspondence, the synthesis of the virtual image is simple and fast. The following algorithm can perform image interpolation:

[1] In this figure, as well as in other figures in the paper, the scene is shown as two-dimensional for simplicity.

Table 1. Three approaches to constructing virtual images.

Approach	Condition	Solving visibility necessary?	Fusion necessary?
image interpolation	$\mathcal{S}(v) = \mathcal{S}(v') \cap \mathcal{S}(v'')$	no	no
image extrapolation	$\mathcal{S}(v) \subset \mathcal{S}(v') \cap \mathcal{S}(v'')$	yes	no
model reconstruction and rendering	$\mathcal{S}(v) \not\subset \mathcal{S}(v') \cap \mathcal{S}(v'')$	yes	yes



a *b* *c*

Fig. 3. Configurations of the scene and the virtual and reference images corresponding to image interpolation (a), image extrapolation (b), and model reconstruction and rendering (c).

1. Find the correspondences in the two reference images by a stereo algorithm. Sparse correspondences of salient features are sufficient.
2. For each pair of corresponding points in the reference images, compute the position of the corresponding point in the virtual image by a transfer.
3. Construct a triangulation on the set of transferred points in the virtual image. Warp the interiors of the triangles from one reference image to the virtual image. This is similar to morphing, well-known in computer graphics.

Let v be the virtual viewpoint and v', v'' the reference viewpoints. The necessary and sufficient condition for image interpolation to yield a correct and complete virtual image is:

$$\mathcal{S}(v) = \mathcal{S}(v') \cap \mathcal{S}(v'') \; . \tag{1}$$

It means that exactly the part of the scene visible from both v' and v'' has to be visible from v. Figure 3a shows an example configuration when the virtual image that can be correctly constructed by image interpolation. Viewpoints v, v', v'' must not leave the domain bounded by dashed lines.

This method is similar to "image interpolation" by Seitz and Dyer [10].

Image Extrapolation. The example in Fig. 3b shows scene points a, b, c transferred to the same image point in the virtual image. One has to decide which of them is visible. If visibility is not explicitly solved (like in image interpolation), an incorrect virtual image can obtained.

An algorithm for image extrapolation is image interpolation augmented with an explicit removal of hidden surfaces. The removal can be done in a way similar to ray tracing [6] or, more efficiently, by z-buffering distances from the epipole, described in Section 3.3. In contrary to image interpolation, sparse correspondences of salient features need not be sufficient. The reason is that a part of the scene can be occluded by a surface with no salient features.

The necessary and sufficient condition for image extrapolation to yield a complete virtual image is the following:

$$\mathcal{S}(v) \subset \mathcal{S}(v') \cap \mathcal{S}(v'') \ , \tag{2}$$

i.e. the part of the scene visible from the viewpoint v has to be a *subset* of the parts visible from both v' and v''.

3-D Model Reconstruction and Rendering. Sometimes, two reference images do not allow to construct a complete virtual image because some parts of the scene visible from the virtual viewpoint are not visible from both reference viewpoints. In this case, we have to use other pairs of reference images and *fuse* multiply reconstructed image parts. The fusion makes this approach practically equal to the reconstruction and rendering of a consistent (projective) 3-D model[2].

Images taken from reference viewpoints v' and v'' do not suffice to construct the complete virtual image from the viewpoint v if

$$\mathcal{S}(v) \not\subset \mathcal{S}(v') \cap \mathcal{S}(v'') \ , \tag{3}$$

that means, a part of the scene visible from v is invisible from either v' or v''. An example configuration is in Fig. 3*c*.

Since two reference images are not enough, it is necessary to use another pair of reference viewpoints, e.g. w' and w'' in Fig. 3*c*. If

$$\mathcal{S}(v) \subset [\mathcal{S}(v') \cap \mathcal{S}(v'')] \cup [\mathcal{S}(w') \cap \mathcal{S}(w'')] \tag{4}$$

the transfer from v' and v'' to v combined with the transfer from w' and w'' to v yields the complete virtual image (otherwise, other pairs of reference images have to be added). A difficulty occurs if

$$[\mathcal{S}(v') \cap \mathcal{S}(v'')] \cap [\mathcal{S}(w') \cap \mathcal{S}(w'')] \neq \emptyset \tag{5}$$

because some parts of the scene will be constructed *twice* in the virtual image. Since these two parts will not be exactly the same due to noise, they have to be fused. Thus we have to cope with fusion of multiply reconstructed scene parts, which is one of the most difficult issues in 3-D model reconstruction.

[2] This approach seems to work with a view-centered representation whereas a model-based approach works with an object-centered representation. But this is not significant because (i) there is no qualitative difference between transfer and projective reconstruction–reprojection, and (ii) it is convenient to express reconstructions from different subsets of reference images in a common projective coordinate system in order to construct a single virtual image v, which is in fact an object-centered representation.

2.3 Comparison of the Three Approaches

Which approach should be used for a given application? In order to visualize a general scene from a general viewpoint, we showed that only 3-D model reconstruction/rendering yields correct and complete virtual images.

Image interpolation is on the opposite end of the spectrum. It has nice properties in its well-posedness [10], relative insensitivity to errors in correspondence, and no care about possible hidden surfaces. However, the domain of its validity is limited and it is difficult to find this domain automatically for a general scene.

Image extrapolation is a trade-off between the two limit approaches. For a general scene and general viewpoints, it does not ensure completeness (i.e., some part of the scene can be missing in the virtual image) yet it guarantees reconstructed parts to be visualized with correctly solved visibility. Obviously, image extrapolation is the most general method for visualizing a scene described by two reference images only.

While it is reasonable to call image interpolation an image-based approach, and 3-D model reconstruction/rendering a model-based one, we could hesitate about image extrapolation. This may indicate that using the concepts "image-based" and "model-based" can sometimes cause misunderstanding.

3 Image Extrapolation from Uncalibrated Images

Three issues are involved in image extrapolation:

1. Reconstruction of a partial 3-D model from several (often only two) reference images. The model is partial because only scene parts that are visible from all these reference images *at the same time* are reconstructed. This is important, as it distinguishes image extrapolation from the reconstruction of a complete consistent 3-D model from a large number of reference images.
2. Virtual camera positioning.
3. Rendering the partial 3-D model.

This section presents an efficient algorithm for the issue 3. However, we will also briefly introduce issues 1 and 2 to have a necessary background.

We will consider only uncalibrated reference images. In that case, at most *projective* 3-D model can be reconstructed [9]. This situation is important especially when the reference images cannot be calibrated, e.g., if we want to visualize scene parts visible in a set of photographs taken by an unknown cameras.

We will denote camera centers by C, image planes by π, image points by $\mathbf{u}_i = [u_i \ v_i \ w_i]^\top$, scene points by $\mathbf{X}_i = [X_i \ Y_i \ Z_i \ W_i]^\top$, and 3×4 camera projection matrices by $\mathbf{M}$. The points are elements of a projective space and represented by homogeneous coordinates. Entities related to the virtual camera and the reference cameras will be distinguished by primes. Thus, C is a center of the virtual camera and C', C'' are centers of the first and second reference camera; π, π' and π'' are respectively the virtual image plane, the first reference and the second reference image plane; etc.

3.1 Reconstructing a Projective Model

We assume that a reconstructed projective model is available and that it is represented by the following data:

- The *set of corresponding pairs [image point, scene point]*, $\{[\mathbf{u}'_i, \mathbf{X}_i],\ i = 1, \ldots, N\}$. This is the result of a projective reconstruction (e.g., [9]) from the set of corresponding image points $\{[\mathbf{u}'_i\ \mathbf{u}''_i],\ i = 1, \ldots, N\}$. It is

$$\rho'_i \mathbf{u}'_i = \mathbf{M}'\mathbf{X}_i\ , \quad \rho''_i \mathbf{u}''_i = \mathbf{M}''\mathbf{X}_i\ , \tag{6}$$

  where $\mathbf{M}', \mathbf{M}''$ are camera matrices of the reference cameras, $\{\mathbf{X}_i,\ i = 1, \ldots, N\}$ is the set of reconstructed scene points, and $\rho'_i \neq 0$, $\rho''_i \neq 0$. Points $\mathbf{X}_i$ differ from an underlying Euclidean structure by an unknown projective transformation.
- The *triangulation* on the set $\{\mathbf{X}_i,\ i = 1, \ldots, N\}$. The triangulation approximates the underlying scene surface. Since only the surface that is visible from both reference cameras can be reconstructed, there is a one-to-one mapping between this surface and the reference image plane π'. Therefore the triangulation can conveniently be done in π' rather than in the projective space of the scene.
- *The reference image* captured from the viewpoint C'. It is used to store *texture* which will be warped to the the virtual image during the rendering process. (The texture from the other reference images is currently not used.)

3.2 Virtual Camera Positioning

The reconstructed points $\mathbf{X}_i$ are projected to the virtual image as

$$\rho_i \mathbf{u}_i = \mathbf{M}\mathbf{X}_i\ , \tag{7}$$

where $\rho \neq 0$. The camera matrix $\mathbf{M}$ specifies the virtual view, in other words, it *positions the virtual camera*. If Euclidean reconstruction were available, $\mathbf{M}$ could be easily computed from extrinsic and intrinsic camera parameters. Virtual camera positioning in projective space is a difficult task, which has not yet drawn too much attention, however. It has a connection to the theory of self-calibration and Kruppa equations [3]. We will assume that a matrix $\mathbf{M}$ specifying the required virtual image is available.

3.3 Rendering a Projective Model

The rendering of the projective model involves two problems to be solved: (i) explicit removal of invisible parts of the scene, (ii) correct and efficient filling pixels in the virtual image plane. Let us discuss them in more detail.

A part of the scene visible in reference images (and therefore reconstructed) can be invisible in the virtual image for the following reasons:

- *This part is behind the virtual camera.* If the projection to the virtual image plane π is done exactly according to (7), the virtual camera cannot distinguish between scene points in front of it and behind it, both can be projected into an image point in π. To model a real situation correctly, the scene points behind π must be explicitly distinguished from the scene points in front of π, and their rendering must be suppressed. The solution has been suggested using oriented projective geometry [7].
- *This part is occluded by another scene part.* Scene points $\mathbf{X}_1, \mathbf{X}_2$ projected in two image points $\mathbf{u}'_1, \mathbf{u}'_2$ distinct in π' can be projected in a common point $\mathbf{u}_1 = \mathbf{u}_2$ in π. This means that either $\mathbf{X}_1$ occludes $\mathbf{X}_2$ or *vice versa* if the scene is observed from C. The decision which of these two cases occurs can be done according to the distance of points $\mathbf{u}'_1, \mathbf{u}'_2$ from the epipole $\mathbf{e}'$ in π' [6, 7]. Equation (10) below gives the exact formulation. We will use this distance to remove occluded surfaces by z-buffering.

Let us focus on how to fill pixels in the virtual image plane π. A naive way is to transfer pixels in a random order from π' to π. The drawback is that some pixels in π remain unfilled because of different sampling frequencies in π and π'.

Ray tracing remedies this drawback. It allows warping texture and solving visibility correctly. It is interesting that the rays can be traced directly in image planes [6]. Ray tracing is time-consuming since an intersection of the scene and rays going through all pixels in π must be found.

A more efficient approach is *transfer by triangles*. First, the vertices of each triangle are transferred from π' to π. Second, the interior of each triangle is transferred by an appropriate transformation. Visibility can be solved by z-buffering. We will show that this approach can result in a very efficient implementation.

Transferring Triangles via Homography. Let us consider two corresponding triangles T and T'. The triangle T lies in the virtual image plane π and has the vertices $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$. The triangle T' lies in the reference image plane π' and has the vertices $\mathbf{u}'_1$, $\mathbf{u}'_2$, $\mathbf{u}'_3$. We know $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ from (7) and $\mathbf{u}'_1$, $\mathbf{u}'_2$, $\mathbf{u}'_3$ from (6). T is rendered by transferring the interior points from T' to T. The interiors of T and T' are related via a homography:

$$\mathbf{u} \simeq \mathbf{H}\mathbf{u}' \tag{8}$$

where $\mathbf{u}$ lies inside T and $\mathbf{u}'$ lies inside T'.

We need four corresponding point pairs to determine $\mathbf{H}$. We show that we can use the pairs $[\mathbf{u}_1, \mathbf{u}'_1], [\mathbf{u}_2, \mathbf{u}'_2], [\mathbf{u}_3, \mathbf{u}'_3]$, and $[\mathbf{e}, \mathbf{e}']$. The first three pairs can be used obviously as they are vertices of T and T'. Figure 4 illustrates why $[\mathbf{e}, \mathbf{e}']$ can be used as the fourth pair. The triangle $\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$ corresponds[3] to T in π and also to T' in π'. The plane $\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$ and the line CC' intersect in the point $\mathbf{X}_4$. The epipoles $\mathbf{e}$ and $\mathbf{e}'$ are the corresponding images of $\mathbf{X}_4$ and therefore the fourth corresponding pair. In projective space, $\mathbf{X}_4$ exists even if the plane $\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3$ and the line CC' are parallel.

[3] The situation is simplified by one dimension in Fig. 4, therefore $\mathbf{X}_3$ is not shown.

$\mathbf{H}$ is used for computing intensities of pixels inside T in the virtual image (i.e., transferring texture from T' to T) and removing hidden pixels by z-buffering.

Transferring texture. The intensity function[4] in the image point $\mathbf{u}'$ in the reference image is denoted by $I'(\mathbf{u}')$ and in the corresponding image point $\mathbf{u}$ in the virtual image by $I(\mathbf{u})$. Currently, we assume that $I(\mathbf{u}) = I'(\mathbf{u}')$ for each pair $[\mathbf{u}', \mathbf{u}]$ (that is, we neglect changes due to photommetry). Synthesis of the virtual image means computing an intensity in each its pixel $\mathbf{u}$. This intensity $I(\mathbf{u})$ is obtained as

$$I(\mathbf{u}) = I'(\mathbf{u}') = I'(\mathbf{H}^{-1}\mathbf{u})\,. \tag{9}$$

Transferring depth. The visibility in the virtual view can be solved by z-buffering in π. Let us assign values of the following function $d'(\mathbf{u}')$ to points in π':

$$d'(\mathbf{u}') = o(\mathbf{e}')\ \sigma\left(\left|\left|\left[\tfrac{u'}{w'}\ \tfrac{v'}{w'}\right] - \left[\tfrac{e'}{g'}\ \tfrac{f'}{g'}\right]\right|\right|^2\right) \tag{10}$$

where $\mathbf{u}' = [u'\ v'\ w']^\top$, $\mathbf{e}' = [e'\ f'\ g']^\top$, $||[\frac{u'}{w'}\ \frac{v'}{w'}] - [\frac{e'}{g'}\ \frac{f'}{g'}]||$ is the distance of the point $\mathbf{u}'$ from the epipole $\mathbf{e}'$, and σ is an arbitrary monotonic function. The function $o(\mathbf{e}')$ equals 1 resp. -1 if C lies in front resp. from behind of π' (see [7] for explanation). The values of $z(\mathbf{u})$ in the z-buffer[5] for each pixel $\mathbf{u}$ in π are obtained in terms of $d'(\mathbf{u}')$:

$$z(\mathbf{u}) = d'(\mathbf{u}') = d'(\mathbf{H}^{-1}\mathbf{u})\,. \tag{11}$$

Approximating the Homography by Affinity. Equations (9) and (11) allow to synthesize virtual images in a similar manner as a Euclidean 3-D model is rendered in computer graphics. This rendering can be very fast especially on work stations with hardware-supported texture warping and z-buffering (e.g. Silicon Graphics). We want to enable utilizing these hardware-supported functions.

Sometimes only warping via an affine transformation is supported, rather than via a homography as in (9) and (11). If T is small compared to the whole image and the maximum difference in depth of the interior points of T is also small, $\mathbf{H}$ can be approximated by an affinity. Since an affine transformation leaves length ratios unchanged, the relation $\mathbf{u}' = \mathbf{H}^{-1}(\mathbf{u})$ can be conveniently approximated using barycentric coordinates, $\mathbf{u}' = \alpha_1\mathbf{u}'_1 + \alpha_2\mathbf{u}'_2 + \alpha_3\mathbf{u}'_3$, where α_i are determined by $\mathbf{u} = \alpha_1\mathbf{u}_1 + \alpha_2\mathbf{u}_2 + \alpha_3\mathbf{u}_3, \quad \alpha_1 + \alpha_2 + \alpha_3 = 1$. Then (9) changes to

$$I(\mathbf{u}) = I'(\mathbf{u}') = I'(\alpha_1\mathbf{u}'_1 + \alpha_2\mathbf{u}'_2 + \alpha_3\mathbf{u}'_3)\,. \tag{12}$$

[4] For simplicity, the notation $I(\mathbf{u})$ is used for intensity in an image point even if $\mathbf{u} = [u\ v\ w]^\top$ is in projective coordinates and the function I usually takes points in Cartesian coordinates. $I(\mathbf{u})$ can be considered as an abbreviation for $I([\frac{u}{w}\ \frac{v}{w}]^\top)$.

[5] Unlike in rendering a Euclidean model for which z-buffer is usually used, $z(\mathbf{u})$ does not mean the Euclidean depth in $\mathbf{u}$ here but rather an increasing function of it.

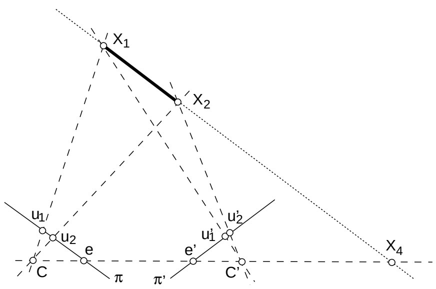


Fig. 4. The fourth point needed to determine $\mathbf{H}$. The situation is simplified by one dimension, therefore $\mathbf{X}_3$ is not shown.

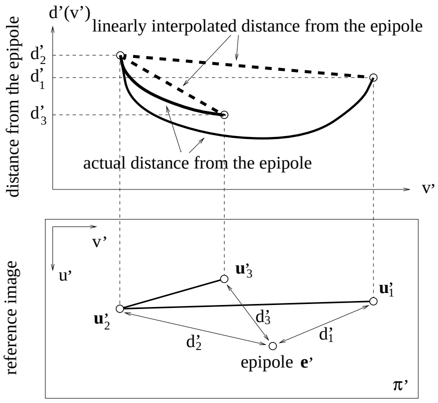


Fig. 5. The error in visibility due to the approximation of (11) by (14). The segment $\mathbf{u}_2'\mathbf{u}_3'$ is incorrectly determined as visible.

Similarly, we can approximate (11):

$$z(\mathbf{u}) = d'(\mathbf{u}') = d'(\alpha_1\mathbf{u}_1' + \alpha_2\mathbf{u}_2' + \alpha_3\mathbf{u}_3') \ . \tag{13}$$

However, this requires evaluating (10) for each pixel in π. A more efficient way is to further approximate (13) in a way similar to Goraud shading well-known from computer graphics:

$$z(\mathbf{u}) = \alpha_1 d'(\mathbf{u}_1') + \alpha_2 d'(\mathbf{u}_2') + \alpha_3 d'(\mathbf{u}_3') \ . \tag{14}$$

Note that (14) becomes identical to the exact relation (13) if $d'(\mathbf{u}')$ is *linear* in coordinates of $\mathbf{u}'$.

Thus, (12) and (14) are approximations of the exact relations (9) and (11) and allow using hardware-supported warping via an affine transformation.

Artifacts Due to the Approximations. The approximation of (9) by (12) causes the texture inside T to be distorted. This distortion becomes more significant near the middle of T and increases with the size of T and with the maximal depth difference of its points. It will vanish for orthographic cameras.

We will show that the approximation of (11) by (14) can cause determining the visibility of T incorrectly on some conditions. In Fig. 5, there are two triangles, $\mathbf{u}_1'\mathbf{u}_2'$ and $\mathbf{u}_2'\mathbf{u}_3'$. Since the situation is simplified by one dimension, these triangles are illustrated as line segments. The correct decision whether the segment $\mathbf{u}_2'\mathbf{u}_3'$ occludes $\mathbf{u}_1'\mathbf{u}_2'$ or *vice versa* should be done according to the actual distance of the segments' interior points from the epipole $\mathbf{e}'$. When this actual distance ($d'(\mathbf{u}')$ in (11), solid-line plots in Fig. 5) is approximated by a linear interpolation from end points ($d'(\mathbf{u}')$ in (14), dashed-line plots), the segment $\mathbf{u}_2'\mathbf{u}_3'$ is incorrectly determined as visible.

This error can be avoided by an appropriate choice of the function σ in (10) so that $d'(\mathbf{u}')$ is *linear* in coordinates of $\mathbf{u}'$ because then (13) and (14) will be equal. It is not obvious, however, if expressing $d'(\mathbf{u}')$ as a linear function in $\mathbf{u}'$ is possible and when. We will show that it is possible if $\mathbf{e}'$ lies outside the convex hull of all image points $\mathbf{u}'$. Then we can replace (10) with

$$d'(\mathbf{u}') = \mathbf{n}'^\top \mathbf{u}' \tag{15}$$

where $\mathbf{n}'$ is a vector such that $\mathbf{n}'^\top(\mathbf{u}'_i - \mathbf{e}')$ has the same sign for all image points $\mathbf{u}'_i$. Such a vector exists only if $\mathbf{e}'$ lies outside the image. This sign has to be positive, resp. negative if $\mathbf{C}$ lies in front of π' resp. behind π'. In fact, (15) describes a projection of the ray $\mathbf{e}'\mathbf{u}'$ to a vector that is "parallel enough" to the image of the optical axis of the reference camera. This projection can be written as $d'(\mathbf{u}') = \mathbf{n}'^\top(\mathbf{u}' - \mathbf{e}')$, hence (15) after omitting the constant $-\mathbf{n}'^\top\mathbf{e}'$.

4 Experimental Results

We will describe experiments with image extrapolation on two real scenes. The first scene (Teeth) was a plaster cast of human teeth arch, its dimensions were approx. $8 \times 8 \times 8$ cm. The second scene (Doll) was a small ceramic statue of about the same dimensions.

Figure 6 shows the reference images. Figs. 6*a, c* show images in π', b, d show images in π''. Image sizes are 250×160 pixels for the Teeth scene and 384×278 pixels for the Doll scene.

Reconstruction of the partial projective model. We found sparse correspondences of the two reference images by tracking edges in a dense image sequence between the reference images [17]. The edges were obtained in rows using Deriche edge detector with subpixel precision. Any standard feature-based stereo algorithm would also suffice. The matched edges for the first scene are shown in Fig. 6*e, f*.

Projective reconstruction was done, using the corresponding edge points.

Next, triangulation of the projective model was constructed by using Delaunay triangulation on the set of the matched points in π'. There were 4183 triangles for the Teeth scene (Fig. 6*g*) and 12408 triangles for the Doll scene.

Virtual camera positioning. We set the elements of the virtual camera matrix $\mathbf{M}$ (see (7)) experimentally so that it yielded an acceptable virtual image. Resolving the front-behind ambiguity, i.e. marking a single image point in π as well as the epipole $\mathbf{e}'$ in π' as projected from front or from behind, was specified by the user.

Rendering the projective model. Equation (12) was used to warp texture from π' (i.e., Fig. 6*a* or *c*) to π. Removing hidden pixels by z-buffering was done according to (14). We used for computing $d'(\mathbf{u}')$ (15) if $\mathbf{e}'$ was outside the convex hull of $\mathbf{u}'_i$ and (10) otherwise. The function σ was linear.

We used hardware-supported z-buffering on a Silicon Graphics Indigo 2 work station. About 4 grey-scale images for the Teeth scene and 2 color images for the

Doll scene were synthesized per second, image sizes being about 500×500 pixels. We hope that if also texture warping were supported by hardware (as is the case for some other workstations) real-time synthesis could have been achieved.

The resulting virtual images are shown in Fig. $6h$–l. Incorrectly reconstructed triangles due to errors in the correspondence can be noticed in $6b$, c. The cusp on the doll's nose in $6l$ is caused by the existence of only one salient feature (edge) on the tip of the nose. Figure $6m$ shows a virtual image with the following error in visibility: C was incorrectly marked in (10) as lying behind π' instead of in front of π'.

5 Conclusion

This paper presented two main results. First, image-based visualization was shown to be closely related to model-based one, because a consistent 3-D model must be used for a perfect visualization of a general scene from a general viewpoint. Fusion and visibility must be explicitly solved like in 3-D model reconstruction/rendering. We distinguished three approaches to visualization: (i) image interpolation (simplest, correct only for a special configuration of a scene and viewpoints), (ii) image extrapolation (correctly removing hidden surfaces but some parts of the scene can be missing), and (iii) 3-D model reconstruction/rendering (difficult, correct in a general situation).

Second, image extrapolation was introduced and advocated as a useful tradeoff between simplicity and generality. It was shown to be equivalent to the reconstruction and rendering of a partial projective model. An efficient algorithm for rendering a projective model was presented. It includes transfer from the reference images to the virtual one by triangles via a homography and solving visibility by z-buffering. The derived relations were approximated so that the resulting algorithm could use warping via affine transformation. We showed that qualitative errors in visibility can be caused by these approximation and when they can be avoided.

Examples of virtual images of two real scenes were presented. The synthesis was close to real time when using triangle warping supported by hardware on a Silicon Graphics Indigo workstation.

References

1. Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.
2. O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between N images. In *Proceedings of the 5th International Conference on Computer Vision*, pages 951–956, Boston, USA, June 1995. IEEE.
3. Olivier D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *European Conference on Computer Vision*, pages 321–333, 1992.

4. R. I. Hartley. A linear method for reconstruction from lines and points. In *Proceedings of the 5th International Conference on Computer Vision*, pages 882–887, Boston USA, June 1995. IEEE Computer Society Press.
5. R. Kumar, P. Anadan, M. Irani, J. Bergen, and K. Hanna. Representation of scenes from collection of images. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 10–17. IEEE Computer Society Press, 1995.
6. S. Laveau and O. Faugeras. 3-D scene representation as a collection of images. In *Proc. of 12th International Conf. on Pattern Recognition, Jerusalem, Israel*, pages 689–691, October 9–13 1994.
7. S. Laveau and O. Faugeras. Oriented projective geometry for computer vision. In *Proceedings of the 4th European Conference on Computer Vision'96*, volume I, pages 147–156. Springer-Verlag, April 1996.
8. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In Robert Cook, editor, *SIGGRAPH'95 Conference Proceedings*, pages 39–46. Addison Wesley, August 1995.
9. Charlie Rothwell, Gabriela Csurka, and Olivier Faugeras. A comparison of projective reconstruction methods for pairs of views. Technical Report 2538, INRIA, Sophia-Antipolis, France, April 1995.
10. S. M. Seitz and C. R. Dyer. Physically-valid view synthesis by image interpolation. In *Proceedings of the Visual Scene Representation Workshop, Boston, MA., USA, June 24*, pages 18–27. IEEE Computer Society Press, 1995.
11. A. Shashua. On geometric and algebraic aspects of 3D affine and projective structures from perspective 2D views. Technical Report AI Memo No. 1405, Massachusetts Institute of Technology, Artifical Intelligence Laboratory, July 1993.
12. A. Shashua and M. Werman. Trilinearity of three perspective views and its associated tensor. In *Proceedings of the 5th International Conference on Computer Vision*, pages 920–925. IEEE Computer Society Press, May 1995.
13. M. Spetsakis and J. Alomoinos. A unified theory of structure from motion. In *Proceedigs ARPA Image Understanding Workshop*, pages 271–283, Pittsburg, PA, USA, 1990.
14. S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 13(10):992–1005, October 1991.
15. T. Werner, R.D. Hersch, and V. Hlaváč. Rendering real-world objects using view interpolation. In *Proceedings of the 5th International Conference on Computer Vision*, pages 957–962, Boston, USA, June 1995. IEEE Computer Society Press.
16. T. Werner, V. Hlaváč, A. Leonardis, and T. Pajdla. Selection of reference views for image-based representation. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume I – Track A: Computer Vision, pages 73–77, Vienna, Austria, August 1996. IEEE Computer Society Press, Los Alamitos, CA., USA.
17. Tomáš Werner, Tomáš Pajdla, and Václav Hlaváč. Correspondence by tracking edges in a dense sequence for image-based scene representation. In *Proceedings of the Czech Pattern Recognition Workshop '97*, pages 64–68, Milovy, Czech Republic, Feb 1997. Czech Pattern Recognition Society. Available from `http://cmp.felk.cvut.cz`.

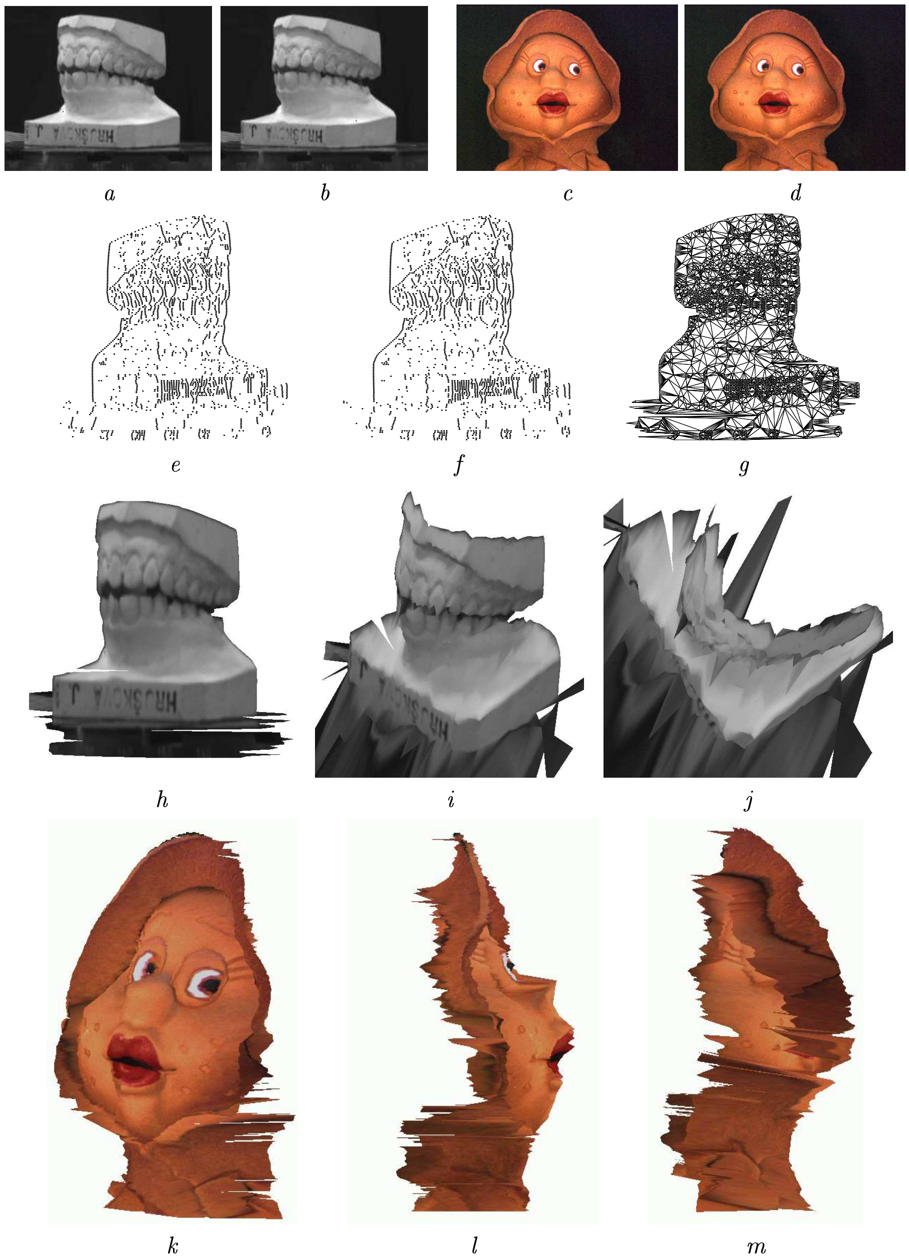


Fig. 6. The reference images for the Teeth scene (a,b) and for the Doll scene (c,d). Matched edge elements in the reference images of the first scene are in e,f. Subfigure g shows the triangulation on the set of edgels in e. The synthesized virtual images of the Teeth scene (h, i, j) and of the Doll scene k,l. Subfigure m shows the result if visibility is solved incorrectly.