What Does the Scene Look Like from a Scene Point?

M.Irani[1], T.Hassner[1], and P.Anandan[2]

[1] Dept. of Computer Science and Applied Mathematics
The Weizmann Institute of Science
76100 Rehovot, Israel

[2] Microsoft Research
One Microsoft Way
Redmond, WA 98052-6399

Abstract. In this paper we examine the problem of synthesizing virtual views from scene points *within* the scene, i.e., from scene points which are imaged by the real cameras. On one hand this provides a simple way of defining the position of the virtual camera in an uncalibrated setting. On the other hand, it implies *extreme* changes in viewpoint between the virtual and real cameras. Such extreme changes in viewpoint are not typical of most New-View-Synthesis (NVS) problems.
In our algorithm the virtual view is obtained by aligning and comparing all the projections of each line-of-sight emerging from the "virtual camera" center in the input views. In contrast to most previous NVS algorithms, our approach does not require prior correspondence estimation nor any explicit 3D reconstruction. It can handle any number of input images while simultaneously using the information from all of them. However, very few images are usually enough to provide reasonable synthesis quality. We show results on real images as well as synthetic images with ground-truth.

Keywords: Novel-view synthesis, Synthesis *without* structure or motion.

1 Introduction

Consider a sporting event with several cameras taking pictures of the game. What does the scene look like from the point view of one of the players in the field? What does the ball "see"? In this paper we show how from a few images of the same scene we can synthesize a virtual view from a real physical scene point which is imaged by all the input cameras.

This problem can be seen as a special case of *New View Synthesis* (NVS). However, it is unique in two ways: (i) In the problem defined above, the virtual camera is located *within* the scene itself, whereas the real cameras view the scene from "outside". This implies an *extreme* change in viewpoint between the virtual camera and each of the real cameras. Such extreme changes in viewpoint are not typical of most NVS examples. (ii) On the other hand, the specification of the new view location is simplified here since this location is visible in the input images. This allows to naturally specify a *physically meaningful position* of the "virtual camera" with respect to the scene in an uncalibrated setting (i.e., without requiring Euclidean calibration of the scene).

In this paper, we describe an algorithm for synthesizing virtual views from scene points within the scene. Our algorithm requires *no prior correspondence estimation nor any 3D reconstruction*. The "epipole" of the virtual camera (the *virtual epipole*) in each of the real views is defined by the image of the physical scene point selected to be the location of the virtual camera. This point is visible in the input images. The color of each pixel in the virtual camera image is determined by aligning and comparing the projections of each line-of-sight (LOS) emerging from the virtual camera center in all the real images. This process does not require knowledge of the 3D structure of the scene and can be done without Euclidean calibration. This leads to a representation which bears some resemblance to the "generalized disparity space" of [16]. Our approach can handle any number of input images while simultaneously using the information available in all of those images.

Many algorithms for NVS have been proposed over the years. To better explain the benefits of our approach and to place it in the appropriate context, we briefly review existing approaches to NVS. These can broadly be classified into three classes of techniques:

The first class of techniques relies on 3D reconstruction of the scene followed by the rendering of the new view (e.g.,[9, 18, 14, 7, 12]). However, the 3D reconstruction process is error-prone. These errors can lead to significant distortions in the reconstructed image because geometric error criteria often used to optimize 3D reconstruction do not translate gracefully into errors in rendered appearance. Furthermore, the 3D reconstruction is optimized in one coordinate system whereas the virtual view is rendered in another coordinate system. Distortions in synthesis due to inaccuracies in 3D reconstruction are amplified in cases of severe changes in viewing position between the real and virtual cameras. Many of these methods thus require a large number of input images taken from a wide range of viewing directions (e.g., Kanade et. al [11, 12] report using 51 cameras placed on an elaborate rig surrounding the scene being reconstructed).

The second class of approaches avoids explicit 3D reconstruction, and instead utilizes dense correspondence estimation between the input views. These dense correspondences are then used to perform a "view transfer" from the input views to the virtual views (e.g., [1, 20]). For example, given two images and their dense correspondence field, a third view can be synthesized by using geometric constraints induced by the Trifocal tensor on the location of these points in the three views. While these methods avoid explicit 3D reconstruction, the errors in correspondence result in similar distortions in the new view. Moreover, the synthesis part involves a forward warping step, which leads to "hole-filling" problems at surface discontinuities. These effects are amplified in cases of severe changes in viewing position between the real and virtual cameras.

The approach proposed in this paper performs *direct* synthesis without an intermediate step of correspondence estimation or explicit 3D reconstruction. Because the analysis and synthesis are done directly in the coordinate system of the virtual view, our process involves only backward ("inverse") warping. Backward-warping does not generate holes in the synthesized view, and handles more gracefully large changes in viewpoint and in image resolution between the synthesized and real images. Furthermore, our synthesis method optimizes errors directly in the coordinate system of the virtual view, thus avoiding many of the optimization problems associated with the first two classes of NVS methods. Our method can thus synthesize views of the scene from significantly different viewing positions than those of the real cameras (as required in order to synthesize the scene from a point *within* the scene).

A third class of NVS methods exists which, like our method, also avoids the need for 3D reconstruction and correspondence estimation altogether. This family of methods is exemplified by the "lightfield" [8] and the "lumigraph" [4]. However, these methods use very dense sampling of the view-space. They require an extreme number of input camera views to generate 4D representations of the scene by storing the radiance observed at *each point* in the 3D world in *each direction*. Synthesis then proceeds by extracting 2D slices of this dense 4D data volume, corresponding to the light observed in the requested viewing direction from the requested viewpoint. Acquiring all the images for generating this large 4D volume at preprocessing is a practical limitation of these methods. This problem amplifies when dealing with large-scale complex scenes or with dynamic scenes (i.e., scenes which change frequently), as collecting the required amount of data at reasonable space-time costs becomes practically impossible.

Unlike this family of methods, our approach does not need a large number of input images. In fact, very few images (in some experiments less than ten) are enough to provide reasonable synthesis quality by our method, with image quality degrading gracefully with fewer images.

Our algorithm is embedded in the Plane+Parallax geometric framework [6, 5, 2, 17, 15, 13]. By aligning all the input images with respect to a real planar surface in the scene (e.g., the ground plane), the camera matrices simplify to the camera epipoles between the real views. We further show that for some practical scenarios the explicit estimation of the epipoles (which may be difficult between widely separated views) is not necessary. Thus, in those situations we can also deal with cases where the real input cameras are situated very far apart from each other viewing the scene from significantly different viewing directions. This is on top of the wide base-line between the virtual camera and each of the real input cameras, which is inherently dictated by our NVS problem.

The rest of the paper is organized as follows: In Section 2 we provide an overview of our approach for solving this problem. The formulation of the problem using the Plane+Parallax geometric framework is described in Section 3. Section 4 summarizes the algorithm. In Section 5 we show how the problem further simplifies in several practical scenarios. Results are shown in Section 6.

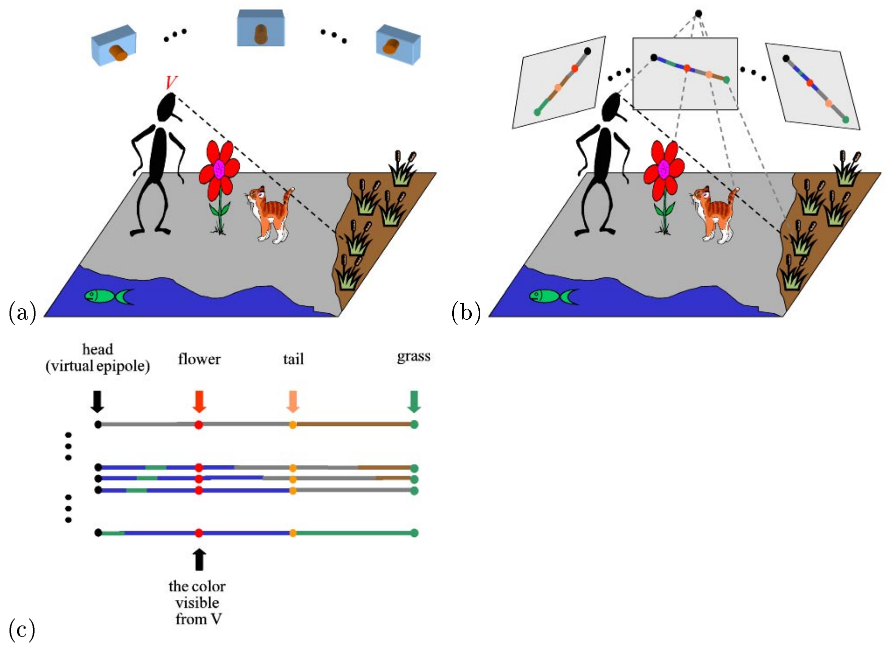

Fig. 1. Overview of the approach. (a) n cameras imaging a scene . The position of the virtual camera V is selected as the tip of the man's head. The *line of sight* (LOS) stretching from V to the patch of grass is shown as a thick dotted line. (b) The 3D LOS is projected via the n camera centers to 2D lines in the n input images. (c) The n projected lines associated with the 3D LOS are geometrically aligned and stacked. The leftmost (black) column corresponds to the virtual camera V in each of the images (i.e., the virtual epipoles). The color-consistent column closest to the projections of V is selected as the color visible by V when looking in the direction of the LOS (i.e. the man sees "red" when he looks in that direction).

2 Overview of the Approach

Figure 1 illustrates a scenario where n uncalibrated cameras image a 3D scene. We wish to synthesize a view of the scene from a scene point V (in this case - the tip of the man's head). To estimate what "color" (radiance) is visible from V when looking in any particular direction, one need only analyze the *line of sight* (LOS) stretching from V in that direction. Naturally, the first physical point (object) in the scene intersecting this LOS will be the point visible from V. For example, in our illustration, the LOS (shown as a thick dotted line in Figures 1.a and 1.b) stretching from V to the patch of grass on the ground, intersects both the flower and the cat's tail. Since the flower is the closest physical object to V on this LOS, its color (red) is the one visible from V in this viewing direction.

The 3D LOS is not available to us. Instead, assuming that the 3D LOS is not occluded from the real cameras, what we do have is the 2D projections of the 3D LOS in each of the input images (Figure 1.b). The physical points on the 3D LOS (e.g. head, flower, tail, grass) are thus projected onto the corresponding points on the 2D lines. However, because the 3D LOS is not fully occupied with physical objects these 2D projected lines also contain projections of points “behind” the 3D LOS which are visible through the “gaps” (such as the blue from the water, the green from the fish, etc.) Given one such 2D projection of the LOS obtained from one input image we cannot tell which of these colors originated from points on the LOS and which colors originated from points not on the LOS. However points on the LOS (i.e., the head, flower, tail and grass) will consistently appear in all projections of the LOS in *all* input images. This is not the case for points not on the LOS. Therefore, if we can identify where all the 2D projections of a single 3D LOS are, geometrically align them and stack them (see Figure 1.c), then physical points on the LOS will be distinguished from other points (seen through the “gaps”) as they will generate uniformly colored columns within the stacked lines (e.g. the black, red, ginger and green columns in Figure 1.c). Other columns (points) will not have consistent colors and can therefore be ruled out (e.g., the brown, green and blue colors of Figure 1.c). The leftmost (black) column corresponds to the virtual epipoles of the virtual camera V on each of the input images. Of all the color-consistent columns the one closest to the projections of the virtual camera V is selected as the color visible from V when looking in the direction of the LOS. In our example it is the red color (the flower’s leaf). Applying this logic to any viewing direction allows us to estimate the complete view of the scene from the virtual camera V.

3 Formulating the Problem Using “Plane+Parallax”

We next show how the detection and alignment of the lines shown in Figure 1 becomes simple using the “Plane+Parallax” geometric framework (e.g., [6, 5, 2, 17, 15, 13]). The 2D line projections of the 3D Line-of-Sight (LOS) L in Figure 1.b are in different coordinate systems of the different camera views. One point correspondence across these lines is known (the images of the virtual camera V). We next show that by aligning any planar surface in the scene across all these images, these lines transform to a single coordinate system forming a pencil-of-lines which emerge from a common axis point. In fact, this axis point is the piercing point of the LOS L with the planar surface. The axis point and the known virtual epipoles uniquely define these lines for any LOS L.

3.1 Pencil of Lines

Let $\Phi_0, \Phi_1, .., \Phi_n$ be images of a scene taken using cameras with unknown calibration parameters. Let Φ_0 denote the “reference image”. Let Π be a plane in the scene that is visible in all the images (e.g., could be the ground plane). We can align all the images with respect to the plane Π by estimating the homography $\mathcal{H}_i$ of Π between the reference image Φ_0 and each of the other images. Warping the images by those homographies yields a new set of images $\{\mathcal{I}_i\}_{i=0}^{n}$,

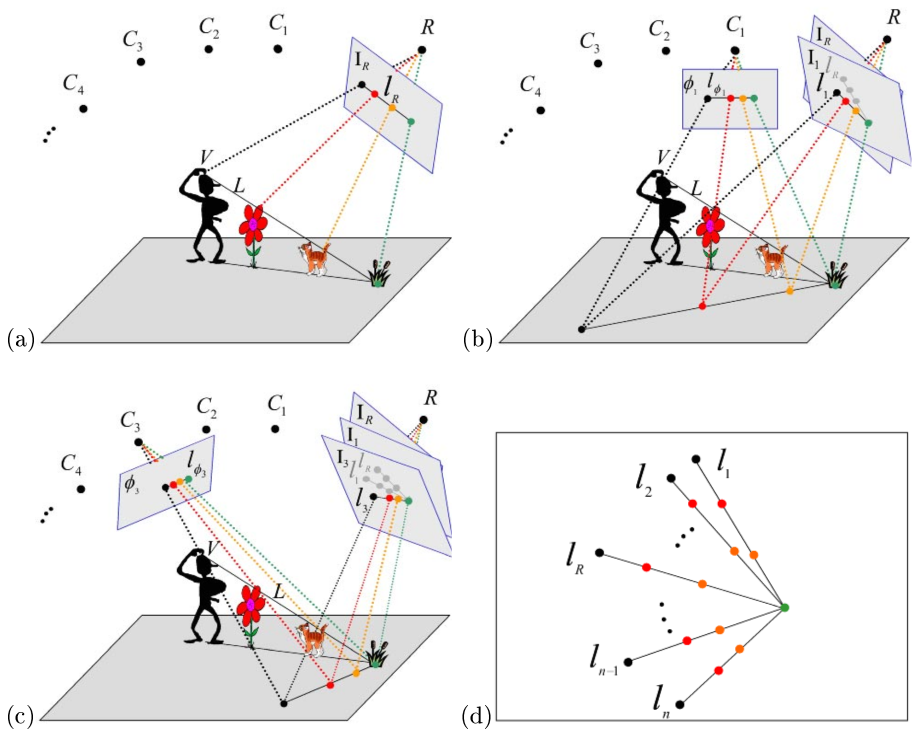

Fig. 2. The pencil of lines.

where for any pixel p in image $\mathcal{I}_i$: $\mathcal{I}_i(\mathbf{p}) = \Phi_i(\mathcal{H}_i\mathbf{p})$. Note that the reference image $\mathcal{I}_0 = \Phi_0$ remains unchanged. We will refer to it as $\mathcal{I}_R$, to emphasize that this is the reference image.

We will next show that all the 2D line projections of a 3D line-of-sight (LOS) form a *pencil of lines* in the plane-aligned images $\mathcal{I}_R, \mathcal{I}_1, .., \mathcal{I}_n$. In the generation of image $\mathcal{I}_i$ from Φ_i each pixel in Φ_i was warped by the homography $\mathcal{H}_i$. The geometric meaning of this operation is displacing each pixel in Φ_i as if its corresponding 3D point was on the plane Π, as shown in Figure 2.b. Points that are truly on Π (e.g., the green grass) will thus appear in $\mathcal{I}_i$ in their correct image position (i.e., will be aligned with their corresponding point in $\mathcal{I}_R$), whereas points not on Π (e.g., the tail, flower, or man's head) will be misaligned with their corresponding points in $\mathcal{I}_R$. (The farther a point is from Π, the more misaligned it will be.)

Let l_{Φ_i} denote the 2D projection of the 3D LOS L on image Φ_i (Figure 2). As a result of plane alignment, l_{Φ_i} is transformed by the homography $\mathcal{H}_i$ to a line l_i in image $\mathcal{I}_i$ (see Figures 2.b and 2.c). We can see from Figure 2.d that all these lines $\{l_i\}_{i=0}^n$ (one from each image $\{\mathcal{I}_i\}_{i=0}^n$) form a *pencil of lines*. The axis point of this pencil (the green point) corresponds to the image of the "piercing point"

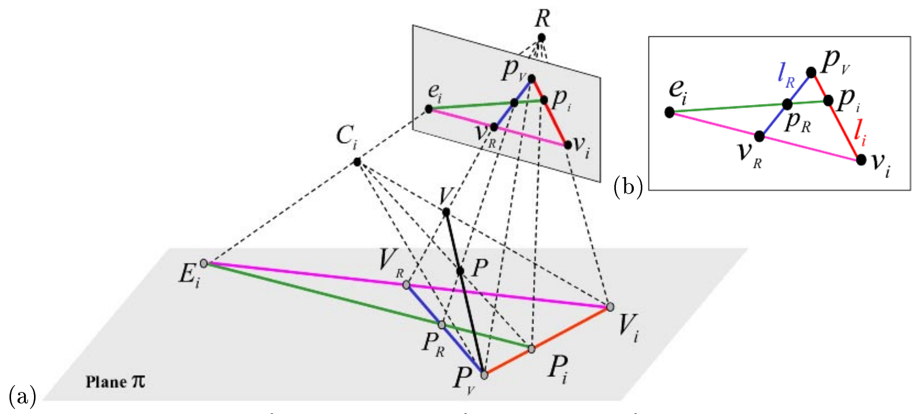

Fig. 3. The LOS line configuration.

of the LOS L with Π (in this case – the grass). The black points at the other end of the lines $\{l_i\}_{i=0}^n$ correspond to the "virtual epipoles" of the virtual camera V in $\mathcal{I}_R, \mathcal{I}_1, .., \mathcal{I}_n$. These lines ($\{l_i\}_{i=0}^n$) are the lines that need to be "aligned" to obtain the stacked representation of Figure 1.c.

The virtual epipoles in each image (the black points) are independent of the lines-of-sight L. Their image coordinates are known, since these points are defined by the user (see Section 4). As to the axis point of the pencil of lines: Every pixel in image $\mathcal{I}_R$ corresponds to a piercing point of a different LOS L with Π. Connecting that pixel to the known "virtual epipoles" uniquely defines the pencil of lines corresponding to the LOS L.

3.2 Aligning and Stacking the Lines

We next show how to bring the lines $\{l_i\}_{i=1}^n$ into alignment with l_R. This step is necessary so that they can be stacked and processed to find the first color-consistent column (that which is closest to the virtual epipoles), in order to determine which color is visible from V in the direction of the LOS L (see Figure 1.c).

We first define some notations. Let $R, C_1, ...C_n$ denote the camera centers of the real cameras. V denotes the 3D scene-point from which we wish to synthesize the virtual view, i.e., the camera center of the "virtual camera" (whose 2D projection in each of the images is known). Let L be any line-of-sight, and let P be a physical scene point on L. Every pair of the above-mentioned camera-centers or scene-points defines a 3D line. We denote by upper-case letters the piercing points of these lines with the plane Π (see Figure 3.a), and by corresponding lower-case letters the projections of these piercing points in the coordinate system of the reference camera R. Thus, for example, the piercing-point of the line passing through C_i and P is denoted by P_i, the piercing-point of the line passing through R and P is denoted by P_R, and the piercing-point of the line passing through V and P (which is the LOS L) is denoted by P_V.

Figure 3.b shows an enlargement of the projections of the piercing points to the coordinate system of the camera R. Note that $\{e_i\}_{i=1}^n$ are the (real) epipoles of cameras $\{C_i\}_{i=1}^n$ in image $\mathcal{I}_R$, v_R is the (virtual) epipole of the "virtual camera" V in image $\mathcal{I}_R$ (e.g., the black point in Figure 2.a), and $\{v_i\}_{i=1}^n$ are the (virtual) epipoles of the virtual camera V in the (plane-warped) images $\{\mathcal{I}_i\}_{i=1}^n$ (e.g., the black point in Figure 2.b or Figure 2.c). Further note that p_V is the projection of the piercing point of L with Π, which is also the axis point of the pencil-of-lines defined by the LOS L (e.g., the green point in Figure 2). This point is invariant to the positions of the cameras $C_1, .., C_n$ and is thus common to all the images $\mathcal{I}_R, \mathcal{I}_1, .., \mathcal{I}_n$.

We now proceed to show how for any given axis point p_V and for any camera C_i, we can compute the geometric coordinate transformation between the line l_R in image $\mathcal{I}_R$ (marked in blue) and the line l_i in image $\mathcal{I}_i$ (marked in red), which will bring the two lines into alignment.

In all the derivations below we assume that points are expressed in homogeneous coordinates. The "virtual epipoles" v_i and v_R are known up to an arbitrary scale factor (these are specified by the user as the images of the scene point from which to synthesize the virtual view; see Section 4). The real epipole e_i is also known (e.g., can be estimated from $\mathcal{I}_R$ and $\mathcal{I}_i$). For any point p_R along the blue (reference) line we can find its corresponding point p_i along the green line:

$$p_i \cong (v_i \times p_V) \times (e_i \times p_R) \tag{1}$$

This equation can be easily verified by noting from Figure 3.b that the point p_i is on the intersection of the red line with the green line. This formulation is similar to the one derived in [3] for predicting the position of a scene point in a third image given its location in two other images. Eq. (1) can be rewritten in the following matrix form:

$$p_i \cong M_i \; p_R \tag{2}$$

where M_i is a 3×3 matrix whose elements can be expressed in terms of the components of the vectors v_i, p_V and e_i.

To summarize, given all the real epipoles $\{e_i\}_{i=1}^n$ (even if only up to *arbitrary* individual scale factors), and the virtual epipoles $\{v_i\}_{i=1}^n$, then for any axis point p_V we can compute the projective transformations $\{M_i\}_{i=1}^n$ that bring the lines $\{l_i\}_{i=1}^n$ into alignment with their reference line l_R.

A real example of alignment and stacking of such lines is shown in Figure 6. Several input images are shown with the projections of a LOS L highlighted in red (Figures 6.(a-f)). These lines were aligned and stacked (Figure 6.g) and the first color-consistent column is marked (which corresponds to the color of the wooden-gate in front of the can and cubes). The resulting representation in Figure 6.g bears resemblance to 2D slices of the 4D generalized disparity space of [16]. However, our representation is obtained by projectively aligning lines rather than by globally discretizing the disparity space.

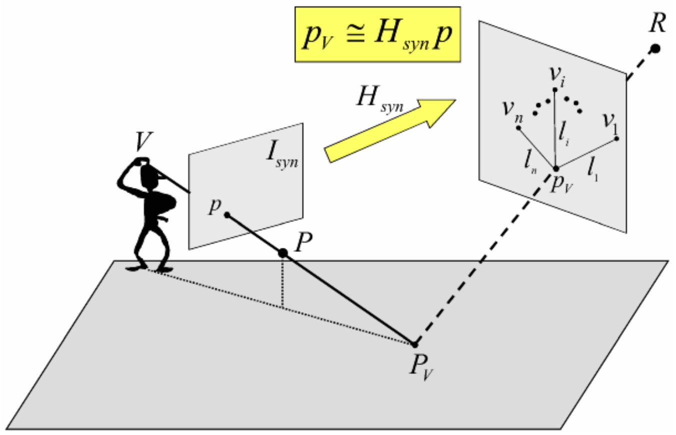

Fig. 4. Relating the virtual view to the reference view.

4 Synthesizing the Virtual View

In this section we provide a summary of our algorithm. Let $\mathcal{H}_{syn}$ be the homography relating the synthesized view with the reference view R. Since the position of the virtual camera is fixed (defined by the virtual epipoles), the only remaining degrees of freedom are the internal parameters of the "virtual camera" (e.g., its zoom) and its orientation (i.e., to which direction we wish the virtual camera to look). The user can select these by specifying $\mathcal{H}_{syn}$. For example, if $\mathcal{H}_{syn}$ is chosen to be the identity matrix, then the synthesized view would be recovered in the coordinate system of the plane-aligned views $\{\mathcal{I}_i\}_{i=0}^n$ (i.e., Π-aligned with the reference view $\mathcal{I}_R$), with the same internal calibration parameters as camera R.
For each pixel $\mathbf{p} = (x, y, 1)^T$ in the synthesized image $\mathcal{I}_{syn}$ do:

1. Let P denote the 3D scene point visible by the virtual camera center V at pixel $\mathbf{p}$. Then : $p_V \cong \mathcal{H}_{syn}\mathbf{p}$ (See Figure 4).
2. For each of the plane-aligned images $\mathcal{I}_1, ..., \mathcal{I}_n$, l_i is the line connecting p_V with v_i, $1 \leq i \leq n$ (see below). Align all these n lines with l_R, the line connecting p_V with p_R by using the line-to-line transformation M_i defined in Eq. (2), and stack them vertically as shown in Figure 1.c.
3. Find the first color-consistent column, i.e., the column which is closest to the column of the synthetic epipoles $v_R, v_1, \ldots, v_n$ (see below for more details). Let "color" denote the color of this column.
4. Assign $\mathcal{I}_{syn}(\mathbf{p})$:= "color".

Specifying the virtual epipoles $\{v_i\}_{i=1}^n$ can be done by the user in one of the following ways: (i) The user can "pin-point" the selected scene point in all the input images, thus determining v_i, $1 \leq i \leq n$ explicitly. (ii) The user can "pin-point" the selected scene point in *two* images, and geometrically infer its position in all other images using trifocal constraints (which are also simple epipole-based constraints after plane alignment [5]). (iii) The user can "pin-point" the selected scene point in one image and use correlation-based techniques to find its corresponding point in each of the other images. All three options provide a way for specifying a physically meaningful position of the virtual camera in an *uncalibrated* setting.

Color consistency within each column (step 3) is determined using color techniques proposed by [10]: Let $A_{(n+1)\times 3}$ be a column of colors represented in YIQ color space. Denote by $cov(A)$ the covariance matrix of A. We use the maximal eigenvalue of $cov(A)$ (denoted by λ) to indicate the degree of color consistency within that column. A high value for λ indicates large variance of colors in the column being evaluated, whereas a low value for λ indicates high color consistency (assuming a Lambertian model). A column whose λ is below a certain threshold is considered "color-consistent". Since λ (the variance) is almost always nonzero, the actual color we select as a representative of a "color-consistent" column is the median color of that column. To robustify the synthesis process we prefer color consistent columns whose position along the LOS is spatially consistent with the chosen column of neighboring pixels. This can be thought of as a local "smoothing" constraint.

Large homogeneous regions in the scene may pose a problem for our algorithm. Projections of different physical points in a uniform region may be interpreted as projections of the same point because they share the same color. To reduce the effect of uniform regions we use a flagging scheme similar to that used in [14] for scene reconstruction. We automatically flag pixels in images already used in the synthesis to prevent them from being used again: If a column contains pixels flagged as previously used, it is not selected even if it is color consistent. The pixel scanning order is defined in $\mathcal{I}_{syn}$ so that pixels corresponding to physical points closer to Π are evaluated before points farther away.

Since in many natural scenes the reference plane may contain large uniform regions (e.g. floor, grass etc.) we further add a preprocessing step which detects the ground plane after alignment. Thus, color consistent columns containing information from the ground plane will only be regarded as projections of a physical point on the LOS if they appear in the last column (i.e., the piercing point of LOS L with plane Π).

5 A Practical Scenario (Avoiding Epipole Estimation)

So far we assumed the real epipoles of the cameras $\{e_i\}_{i=1}^{n}$ are known. However, these may be difficult to estimate, especially when the cameras are widely separated and are looking at the scene from significantly different viewing positions. We next show how in some practical scenarios the need to estimate the inter-camera epipoles can be alleviated and replaced by simple image-based manipulations.

Such is the case when all the cameras are at the same height from the plane Π. In many real-world scenarios this may not be a very restrictive assumption. For example, consider a sports event. If we choose the ground plane to be the plane Π, then if all cameras are placed at the same height in the stadium (e.g., the same bench level) this assumption would be satisfied. Similarly, If a camera is attached to a mobile rig (e.g., a cart), and is moved around on the flat ground while taking pictures of the scene, then all these pictures would be taken from the same height (even if the camera changes its internal parameters and its orientation as the rig moves). The same assumption holds if the camera is mounted on an airplane flying at fixed altitude from the ground or if the camera is attached to a crane which is parallel to the plane Π (Π can also be a wall or any other plane in the scene).

In all these cases the "cameras" are all coplanar. Referring back to Figure 3.b, we note that the epipole e_i lies along the line connecting the two (known) virtual epipoles v_i and v_R. Thus, there is only one degree of freedom in the epipole e_i (namely, where it is along that line). When all the real cameras are coplanar, then all the epipoles $\{e_i\}_{i=1}^n$ lie on a single line in $\mathcal{I}_R$. This line is defined by the intersection of two planes: the plane defined by the real (coplanar) camera centers and the image-plane of the reference camera R. If we further *rectify* the image $\mathcal{I}_R$ of R, then this line of real epipoles would go to infinity, and all the real epipoles $\{e_i\}_{i=1}^n$ would now be uniquely defined by the virtual epipoles. For example, if e_i is known to be infinite (i.e., its third component $w_{e_i} = 0$), then:

$$e_i \cong w_{v_R} v_i - w_{v_i} v_R \tag{3}$$

where $v_i = [x_{v_i} y_{v_i} w_{v_i}]^T$, $v_R = [x_{v_R} y_{v_R} w_{v_R}]^T$, and $e_i = [x_{e_i} y_{e_i} w_{e_i}]^T$.

Substituting e_i in Eq. (1) with the expression of Eq. (3), we get a new matrix M_i whose 3rd row equals $(0, 0, 1)$. In other words M_i is now an affine transformation. This implies that the line-to-line transformations $\{M_i\}_{i=1}^n$ reduce to *simple linear stretching* of the lines $\{l_i\}_{i=1}^n$ relative to l_R.

The above result was obtained by rectifying the reference image $\mathcal{I}_R$. There are different possible ways of rectifying images. However, since in this particular case we assumed the cameras were of the same height from the plane Π, then the rectified image of $\mathcal{I}_R$ should be a "nadir view" of the plane Π. Such rectification can be achieved based on the visual information in the image $\mathcal{I}_R$ alone, without referring to any of the other images. The rectification can be obtained by manually imposing linear constraints that force parallel lines on the plane Π to become parallel after rectification of $\mathcal{I}_R$. This rectification step can be thought of as a type of weak scene-based calibration [19].

6 Results

In our experiments we avoided epipole estimation by maintaining constant camera height above the ground plane (see Section 5). We first tested our method on a synthetic scene with ground truth data. Figure 5.(a–c) shows three of the ten images rendered from a 3D graphic model (each of size 800×600). These images were used to synthesize a view of the scene from the tip of the green pencil standing on the floor, looking in the direction of the wooden gate. The synthesized view is shown in Figure 5.d. Note the extreme change in viewpoint between the positions of the real cameras and the "virtual camera" (the green pencil). In all input images only the ground is visible through the wooden gate, while in the reconstructed image (Figure 5.d) parts of the Coke-can and the toy cube are visible beneath the gate. For comparison Figure 5.e shows the ground truth image rendered directly from the 3D model from the same viewpoint. This shows the geometric correctness of our synthesis. The differences in image quality are due to the fact that the image our algorithm synthesized (Figure 5.d) was generated from low-resolution input images (e.g., Figures 5.(a–c)) whereas the ground truth image was rendered directly from a perfect 3D model. See figure 6 for an illustration of our synthesis process.

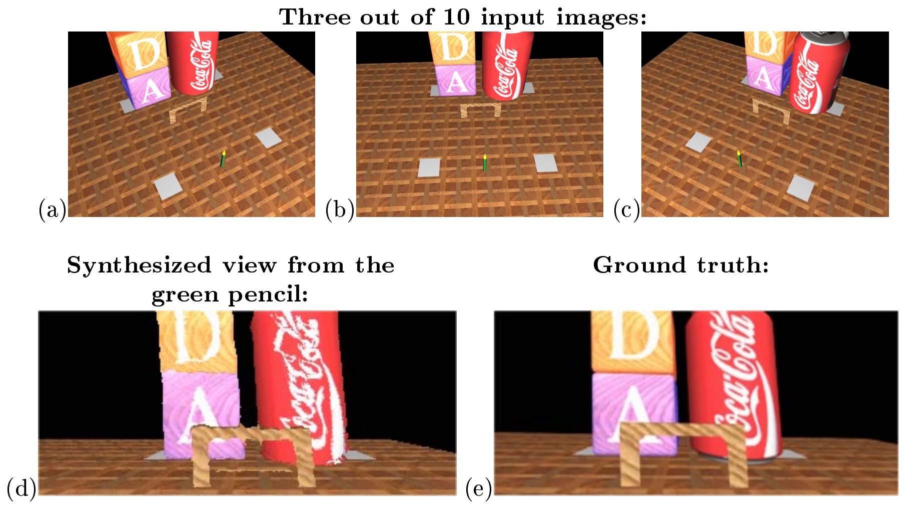
Three out of 10 input images:
(a)
(b)
(c)
Synthesized view from the green pencil:
Ground truth:
(d)
(e)

Fig. 5. Synthesis results for the Coke-Can sequence (see text).

Figure 7 shows the result of applying our algorithm to images of a real scene captured by an off the shelf digital camera. The camera was mounted on a tripod to guarantee constant height above the ground in all images. Three of the 14 input images (each of dimension 640 × 480) are shown in Figure 7.(a–c). The algorithm was used to synthesize virtual views of the scene from two different scene points: Figure 7.d shows a reconstructed view of the scene from the tip of the green dinosaur's nose using all 14 input images. Figure 7.e shows the reconstructed view of the scene from the tip of the purple dinosaur's nose created using only 11 of the input images. Although both the folder and the green-and-yellow cube on the floor are fully visible in all input images, they are partially occluded in the synthesized views: The cube appears over the left shoulder of the triangle in the green dinosaur's view, and over the right shoulder of the triangle in the purple dinosaur's view as can be expected. Both the triangle

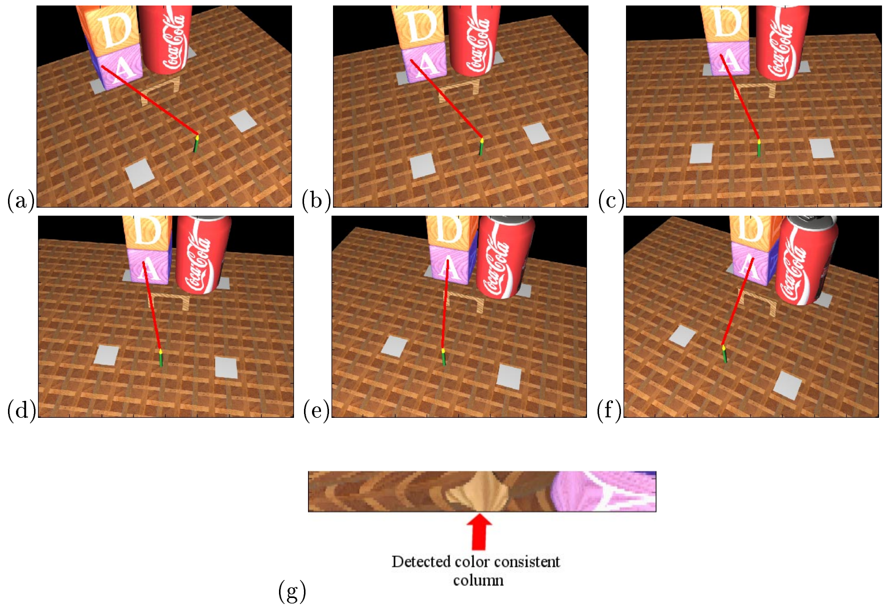

Fig. 6. LOS color analysis (the Coke-Can sequence). (a-f) Show the projection of a line of sight emerging from the virtual camera (at the tip of the green pencil) in 6 input images of the Coke-Can experiment (See Figure 5). The projected LOS is highlighted in red. (g) All ten lines were aligned and stacked. The first color-consistent column is indicated by a red arrow. Its color is that of the wooden gate, which is the first physical object along this LOS from the pencil.

and the cube occlude appropriate parts of the folder located behind them with respect to the viewpoint of the two dinosaurs.

Figure 8 shows another real example only this time one dinosaur is placed in front of the other. Eleven images (560×420 pixels each) were used to synthesize a view from the green dinosaur's nose (Figure 8.d), and only nine were enough to synthesize a view from the purple dinosaur's nose Figure (8.e). It can be seen that the purple dinosaur completely occludes both the cube and the puppet from the viewpoint of the green dinosaur (Figure 8.d). Also, the green and orange columns on the left almost totally occlude the red-yellow triangle (only the tip of its yellow side is visible). In contrast the purple dinosaur sees both clearly (Figure 8.e). The green smear on the floor at the bottom left side of the synthesized view in Figure 8.e is due to the fact that this floor-region was never visible in any of the input images. The puppet (Ernie) appears leaning backwards from the purple dinosaur's view because it is indeed leaning back as can be seen in Figures 8.(a,c).

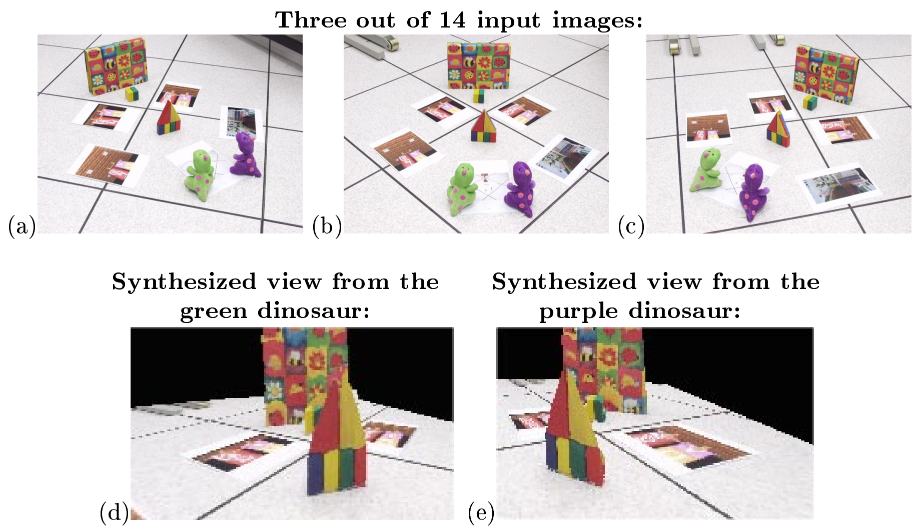
Three out of 14 input images:
(a)
(b)
(c)
Synthesized view from the green dinosaur:
Synthesized view from the purple dinosaur:
(d)
(e)

Fig. 7. Synthesis results for the Folder sequence (see text).

References

1. S. Avidan and A. Shashua. Novel view synthesis by cascading trilinear tensors. In *IEEE Transactions on Visualization and Computer Graphics*, 1998.
2. A. Criminisi, I. Reid, and A. Zisserman. Duality, rigidity and planar parallax. In *ECCV*, Freiburg, 1998.
3. O. Faugeras and L. Robert. What can two images tell us about a third one? In *ECCV*, pages 485–492, 1994.
4. S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.
5. M. Irani, P. Anandan, and D. Weinshall. From reference frames to reference planes: Multi-view parallax geometry and applications. In *ECCV*, Freiburg, June 1998.
6. R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: a parallax based approach. In *Proc 12th ICPR*, pages 685–688, 1994.
7. K. N. Kutulakos. Approximate n-view stereo. In *ECCV*, 2000.
8. M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, 1996.
9. P. J. Narayanan, P. W. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *ICCV*, 1998.
10. M. Orchard and C. Bouman. Color quantization of images. In *IEEE Transactions on Signal Processing*, volume 39, 1991.
11. P. Rander, P.J. Narayanan, and T. Kanade. Virtualized reality: constructing time-varying virtual worlds from real events. In *Proc. IEEE Visualization*, pages 277–283, October 1997.
12. H. Saito and T. Kanade. Shape reconstruction in projective grid space from a large number of images. In *CVPR*, 1999.
13. H. Sawhney. 3D geometry from planar parallax. In *CVPR*, 1994.
14. S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *CVPR*, 1997.

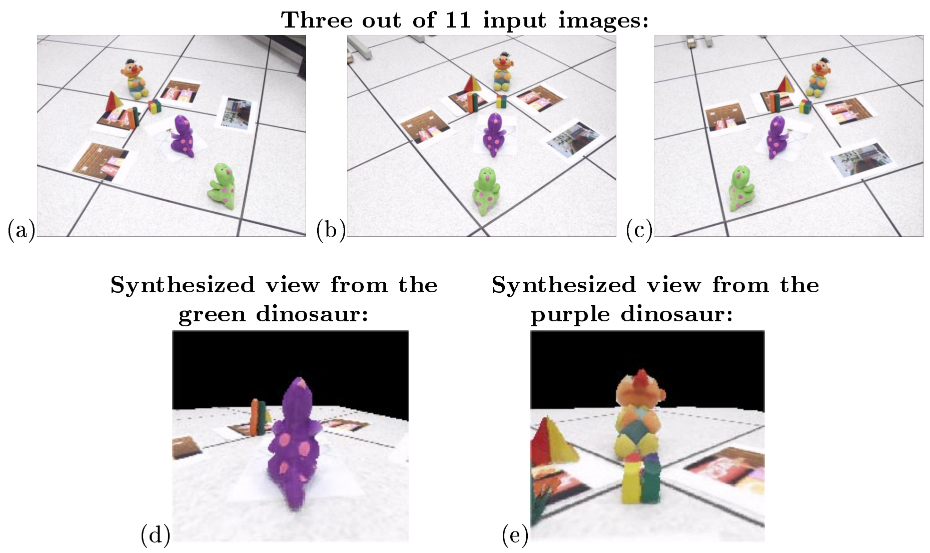
Three out of 11 input images:
(a)
(b)
(c)
Synthesized view from the green dinosaur:
Synthesized view from the purple dinosaur:
(d)
(e)

Fig. 8. Synthesis results for the Puppet sequence (see text).

15. A. Shashua and N. Navab. Relative affine structure: Theory and application to 3D reconstruction from perspective views. In *CVPR*, pages 483–489, 1994.
16. R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *ICCV*, pages 517–524, January 1998.
17. W. Triggs. Plane + parallax, tensors, and factorization. In *ECCV*, pages 522–538, June 2000.
18. S. Vedula, P. Rander, H. Saito, and T. Kanade. Modeling, combining, and rendering dynamic real-world events from image sequences. In *Proceedings of the International Conference on Virtual Systems and Multimedia*, 1998.
19. D. Weinshall, P. Anandan, and M. Irani. From ordinal to euclidean reconstruction with partial scene calibration. In *Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, Freiburg, June 1998.
20. T. Werner, R.D. Hersch, and V. Hlaváč. Rendering real-world objects using view interpolation. In *ICCV*, pages 957–962, June 1995.