The Epipolar Occlusion Camera

Paul Rosen[*]
Purdue University

Voicu Popescu[+]
Purdue University

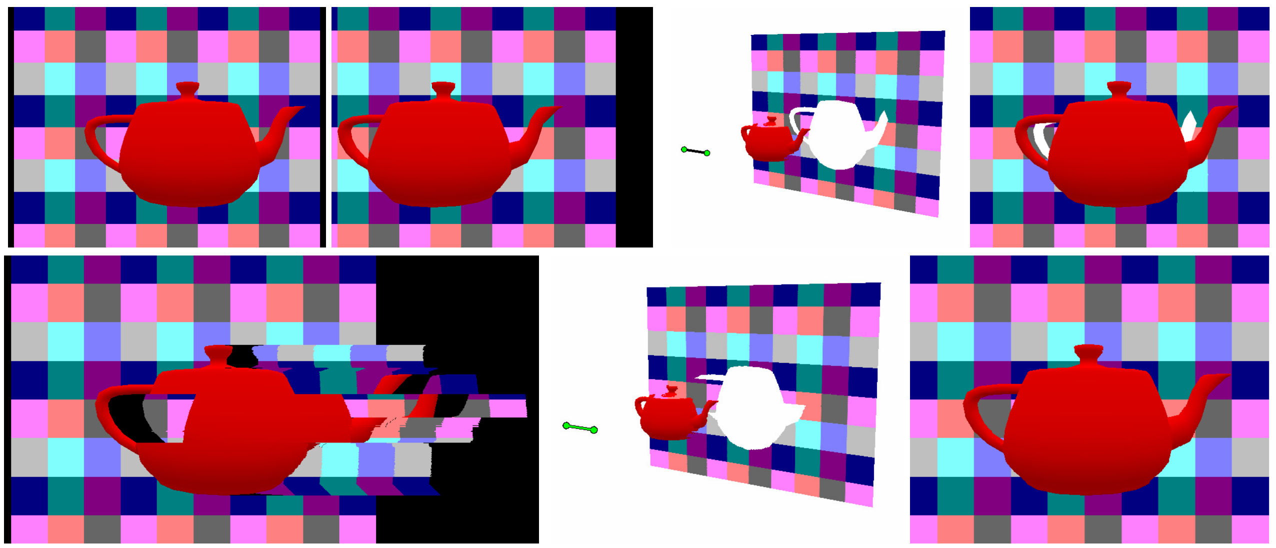


Figure 1 *Top:* Even when two regular depth images are used (*left*, down-sampled here due to space constraints), one for each end of a segment of viewpoints (*middle*), important disocclusion errors occur for an intermediate viewpoint (*right*). *Bottom:* A single epipolar occlusion camera image (*left*) captures all samples visible from the viewpoint segment (*middle*), and avoids disocclusion errors (*right*).

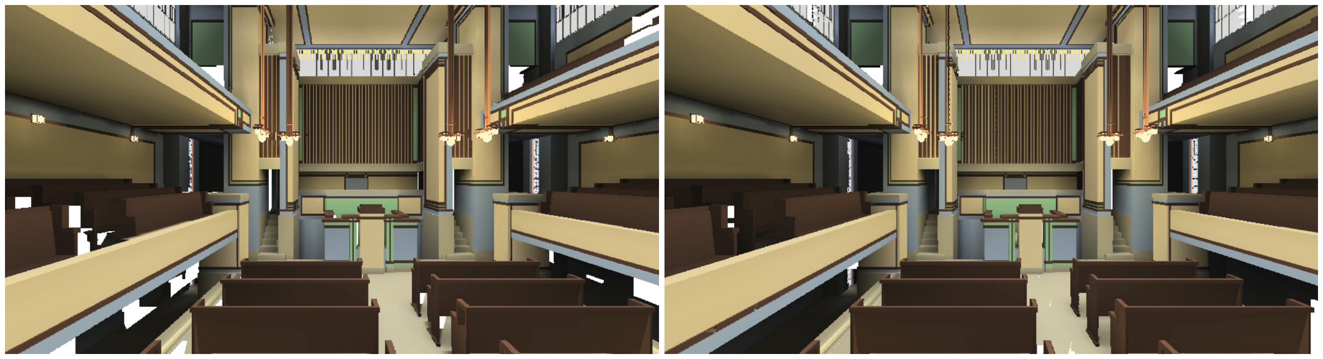


Figure 2 Output image with viewpoint on a segment *LR* reconstructed once from two depth images with viewpoints *L* and *R* (*left*) and once from an epipolar occlusion camera image with segment of viewpoints *LR* (*right*).

Abstract

A depth image constructed with a pinhole camera suffers from disocclusion errors: even a minimal viewpoint translation exposes samples not visible from the original viewpoint. The conventional solution to employ additional depth images is inefficient. A recent approach is to render the depth image with an occlusion camera, a non-pinhole that also gathers samples not seen from the reference viewpoint but needed for nearby viewpoints.

We introduce the epipolar occlusion camera (EOC), which overcomes disadvantages of earlier occlusion cameras. An EOC is a non-pinhole which gathers samples of a 3D scene visible from a segment of viewpoints. The EOC is constructed by expressing the disocclusion events in the (2D) image as the sum of independent disocclusion events along (1D) epipolar lines. The EOC image has a single layer, is non-redundant, and is constructed efficiently by directly rendering the 3D scene with the EOC in feed-forward fashion, through projection followed by rasterization.

CR Categories: I.3.3. [Computer Graphics]—Three-Dimensional Graphics and Realism.

Keywords: non-pinhole camera, disocclusion errors.

[*]email: rosen@purdue.edu
[+]email: popescu@cs.purdue.edu

1. Introduction

A depth image, defined as an image with per-pixel depth, is an important computer graphics primitive. One use of depth images is in the context of image-based rendering where they replace geometry in order to reduce rendering cost (e.g. 3D warping [McMillan 1995], impostors [Maciel 1995, Popescu 2006b], or relief texture mapping [Oliveira 2000, Policarpo 2005]). Depth images are also used to determine visibility from the light in shadow computation. In the context of compression of rendered imagery depth image key frames are automatically mapped to intermediate frames by 3D warping, reducing the amount of data residual images need to store [Aliaga 2005]. In the context of 3D displays, depth images can be used as an intermediate representation that accelerates the rendering of the 3D image and reduces the bandwidth to the 3D display [Popescu 2006a].

An important challenge is posed by disocclusion errors. A depth image is typically rendered with a pinhole camera and only stores samples visible from the camera's viewpoint. Even minimal viewpoint translations expose samples that were not visible from the original viewpoint, which causes disocclusion errors. In image-based rendering this leads to highly objectionable "holes" in the output image. In shadow mapping, the single viewpoint constraint limits the approach to hard shadows. In compression, the disocclusion errors lead to high residuals, lowering compression performance (i.e. compression ratio). In the context of a 3D display, disocclusion errors limit image quality when the display is seen from additional viewpoints (i.e. second-eye viewpoint in a single-user scenario, and additional user viewpoints in a multi-user scenario).

The classic approach for alleviating the problem of disocclusion errors is to use additional depth images, rendered from nearby viewpoints, in the hope of filling in disocclusion errors. However, such an approach is inefficient. First, disocclusion errors occur throughout the volume of the scene, and one cannot find a small set of additional depth images that provide all missing samples. Second, the additional depth images have considerable overlap, which causes costly redundancy. Third, the output image is rendered from a variable number of depth images, so the advantage of bounded rendering cost is lost.

Disocclusion errors occur because the depth image is constructed with a pinhole camera which samples the scene from a single viewpoint, yet the depth image is asked to support additional viewpoints; this is only possible when the scene is trivial and the additional viewpoints do not require additional samples. We have recently introduced a novel approach for alleviating the problem of disocclusion errors based on innovation at the camera model level [Mei 2005, Popescu 2006c]. Instead of constructing the depth image with a pinhole camera, the depth image is constructed with a non-pinhole called an occlusion camera. In addition to the samples seen from a reference viewpoint, an occlusion camera also gathers samples that are likely to become visible from nearby viewpoints. Moreover, although occlusion cameras are non-pinholes, they do provide closed-form projection and therefore occlusion camera images can be constructed efficiently in feed-forward fashion, by projecting and rasterizing scene triangles.

Occlusion cameras offer an elegant solution to the problem of disocclusion errors. Like a regular depth image, an occlusion camera image has a single layer, it does not store redundant samples, and it is rendered in hardware. Unlike regular depth images, an occlusion camera image also stores samples needed in nearby views. However, the occlusion cameras developed so far suffer from the following disadvantages. The single-pole occlusion camera (SPOC) model [Mei 2005] is too simple to tackle complex scenes. The depth-discontinuity occlusion camera (DDOC) [Popescu 2006c] works by pulling barely hidden samples from underneath occluders and thus can handle complex scenes, but only under the condition that sufficient space exists between occluders. The DDOC constructor attempts to disocclude according to parameters set by the application, but when two occluder edges compete for the same image region, the disocclusion amount has to be reduced. This reduces the range of viewpoint translation supported by the DDOC image. Furthermore the application has no easy way of knowing the set of viewpoints for which the DDOC image has sufficient samples.

To overcome these disadvantages we introduce the epipolar occlusion camera (EOC). Given a 3D scene, a planar pinhole camera *PPHC,* and a segment of viewpoints *LR*, an EOC is a non-pinhole that gathers samples *PPHC* would see when translating from *L* to *R*. While a pinhole gathers the samples visible from a *point*, an EOC gathers the samples visible from a *segment*. The EOC model leverages epipolar constraints to avoid disocclusion errors on individual epipolar lines. The EOC image has a single layer, yet occluder spacing is not a problem. The EOC provides efficient projection so its image is constructed directly by rendering the scene with the EOC, with hardware support, and *not* by combining several pinhole camera images.

The EOC concept is illustrated in Figure 1 (also see color plate). Both pinhole depth images miss back wall samples that are visible from intermediate viewpoints. For example, some of the samples occluded by the teapot handle in the left image are then occluded by the teapot body in the right image. These samples are visible in the intermediate view and their absence causes the white gap around the handle in the top right image. The EOC is constructed from the pinhole *PPHC* of the left image, the segment of viewpoints *LR*, and the scene. The goal is for the EOC to gather the samples visible by *PPHC* as it translates on the viewpoint segment. Due to epipolar geometry constraints, the disocclusion events that occur can be conveniently described as the sum of independent disocclusion events on individual epipolar lines. To take advantage of this fact, EOC rows are defined according to the epipolar lines induced by *LR* on *PPHC*. For each row, the EOC gathers additional samples as needed, according to the occluders crossed by the epipolar line of the row. In the case shown, *LR* is parallel to the *PPHC* rows, thus the epipolar lines are the *PPHC* image rows, and consequently the EOC has the same rows as *PPHC*. The EOC image has more samples than the *PPHC* image on rows where more disocclusion events occur (i.e. rows that cross the handle, the body and the spout), and the same number of samples on rows without disocclusion events (i.e. top and bottom rows). Figure 2 shows the EOC at work on a complex scene. Figure 8 shows an EOC built to support forward translation.

The remainder of this paper is organized as follows. The next section briefly reviews prior work, including a detailed comparison between the prior DDOC and the novel EOC. Section 3 gives the algorithm for constructing an EOC. Section 4 describes the algorithm for computing EOC images by rendering the scene with the EOC. Like regular depth images, the EOC image can serve as infrastructure in many applications. Section 5 explores several possibilities. Section 6 presents conclusions and sketches possible directions for future work.

2. Prior work

We first review prior multiple-depth-image solutions to the problem of disocclusion errors, then we discuss prior non-pinhole camera models, and we conclude with a detailed comparison between the EOC and the prior occlusion camera models.

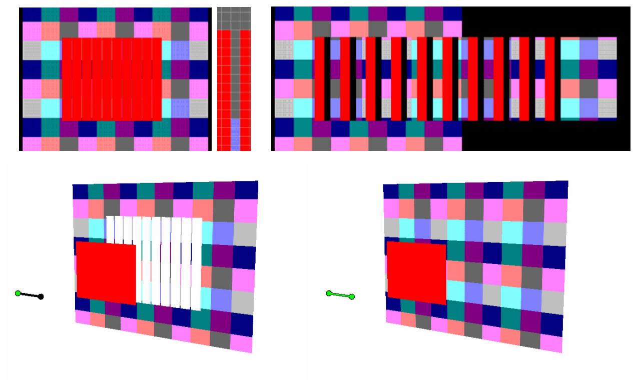


Figure 3 *Top*: Regular depth image and magnified fragment (*left*), and EOC image (*right*). *Bottom*: The depth image misses a considerable part of the back wall (*left*) while the EOC does not, preventing disocclusion errors.

2. 1. Multiple-depth-image solutions

The conventional solution to the disocclusion error problem is to rely on additional images to provide the missing samples. Early solutions combined the samples of multiple depth images at run time [McMillan 1995, Mark 1997, Popescu 2000]. The problem of finding where in the output image disocclusion errors occur is challenging, so these early methods were simply processing several depth images with viewpoints near the output image viewpoint. In order to shield the application from the cost of processing multiple depth images on the fly, subsequent techniques moved to pre-combining depth images in layered representations as a preprocess (e.g. multi-layered z-buffers [Max 1995], layered depth images (LDIs) [Shade 1998], and LDI trees [Chang 1999]). The large pre-processing cost makes these methods unsuitable for dynamic scenes. Since the layered representations are built off-line, a larger number of depth images can be considered, which increases the chance that all needed samples are gathered in the layered representation. However, the layered representations remain non-conservative.

A conservative run-time sample selection technique is the vacuum buffer [Popescu 2001], which keeps track of which subvolumes of the view frustum of the output image could potentially be missing samples. Note that finding where in the output image disocclusion errors occur cannot be reduced to simply finding the image regions with no samples; disocclusion errors can occur over samples that the missing samples should hide. Finding disocclusion errors conservatively has to examine the volume of the view frustum. Additional depth images are considered until all vacuum buffer subvolumes are either proven to be empty or their samples are revealed. The method has the merit of conservatively avoiding all disocclusion errors, but it is expensive since it has to process multiple depth images for every frame.

2. 2. Prior non-pinholes

Instead of filling in the gaps caused by disocclusion errors, our method is based on a novel non-pinhole camera model which avoids disocclusion errors in the first place. Non-pinhole camera models have been relatively little explored, which we attribute to misconceptions such as the belief that the images non-pinholes produce are not useful since they are different from the images the human visual system acquires, or that rendering with non-pinholes is inherently inefficient. Images are often used as intermediate representations and in many such cases deviations from the rules of single-perspective are not only acceptable but also desirable. Regarding efficiency, we will show that rendering with non-pinhole cameras can be fast if the camera model is designed such that it provides efficient projection.

Several non-pinhole camera models were studied in computer vision to model complex lens and catadioptric systems, including the pushbroom camera [Gupta 1997], the two-slit camera [Pajdla 2002], and their generalization, the general linear camera [Yu 2004], but these models are not sufficiently powerful to capture complex occlusion patterns in 3D scenes.

In computer graphics, one prior use of non-pinholes is in the context of replicating distorted-perspective effects employed in some paintings for aesthetic and artistic purposes. Another use of non-pinholes is in the context of image-based rendering. A light field [Levoy 1996] or a lumigraph [Gortler 1996] can be seen as a non-pinhole defined by a matrix of pinholes. They do solve the disocclusion error problem for a region of the 3D scene, but they have high construction and storage cost.

The multiple-center-of-projection camera (MCOP) [Rademacher 1998] acquires samples along a user-specified path with a one-column pinhole. An MCOP can gather all samples needed along the path, but it requires user guidance, it is expensive to construct, and it stores redundant data. Consider generating an MCOP that gathers all samples needed for the segment of viewpoints and the scene in Figure 1. Although the path is given (i.e. the viewpoint segment), the user has to pan the column camera to gather all samples around the teapot silhouette. Simply maintaining a view direction perpendicular to the back wall produces an MCOP equivalent to an orthographic projection image, which misses many samples. Even if the user interactively pans the column camera to get all samples, some samples will be acquired more than once. Lastly, the scene needs to be rendered for each position along the path, which is inefficient.

2. 3. Prior occlusion cameras

The epipolar occlusion camera builds upon two earlier members of the occlusion camera class. The single-pole occlusion camera (SPOC) [Mei 2005] is constructed by radially 3D distorting a planar pinhole camera around a pole defined by the occluder's center. The SPOC has the merit of introducing the occlusion camera concept but the simple camera model only works for single relatively simple objects. The depth-discontinuity occlusion camera (DDOC) [Popescu 2006] disoccludes samples by distorting them along the direction perpendicular to the occluder edge, and away from the occluder. The distortion is specified per pixel with a distortion map which provides the flexibility needed to handle complex scenes.

However, the DDOC makes room for the disoccluded samples by reducing the resolution on a band parallel to the occluder edge. The visible samples do not use the entire band anymore which makes room for the disoccluded samples. Although when there is enough room between occluders this approach only implies halving the resolution close to occluder edges, when occluders are close together, the DDOC's disocclusion capability is greatly reduced. In Figure 3 the picket fence has a distance between pickets of 1 pixel and there is no room for disoccluding the back wall samples hidden by the pickets, which are needed for intermediate viewpoints. In this case, a DDOC would not perform better than the depth image shown, whereas the EOC makes room for the pickets by shifting samples on epipolar lines. The EOC shifts the pickets sufficiently as to sample the background at the appropriate resolution, as dictated by the segment of viewpoints. The only course of action for the DDOC would be to make the

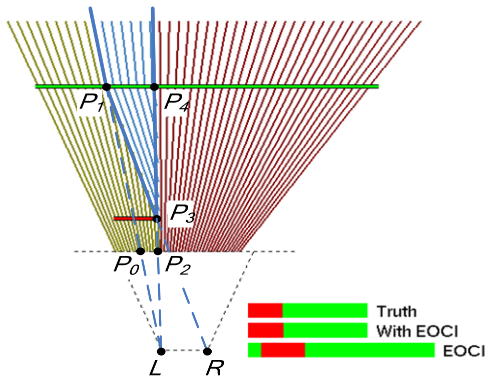


Figure 4 Visualization of an EOC row.

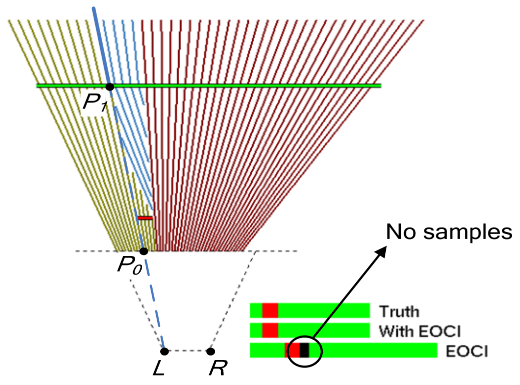


Figure 5 Visualization of an EOC row with a narrow occluder.

gap between pickets larger by increasing the resolution of the entire image. The required distance between pickets is 32 pixels, which leads to a prohibitive 32x32 increase in resolution (i.e. more than 900 times more pixels), whereas the EOC image is only about twice as big as the original pinhole image. Even a solution of compromise that widens the gap between pickets to 4 pixels and then proceeds with asymmetrical distortion, still implies a 4x4=16 fold increase in the image size. Moreover the asymmetrical distortion leads to a 4/32=1/8 resolution reduction.

A final point in the comparison between the DDOC and the EOC considers the set of viewpoints for which each camera gathers sufficient samples. The DDOC is designed to support viewpoint translations in all directions, whereas the EOC only supports translations along a line segment. However, for a DDOC, the application intended range of translations can be overruled by the constructor which has to arbitrate competing occluders, as described earlier. The EOC on the other hand provides good support for the given segment of viewpoints.

3. Camera model construction

The EOC model is constructed from a 3D scene *S*, a planar pinhole camera *PPHC*, and a segment of viewpoints *LR*. The rays of the EOC are line segments which do not pass through a common point. The EOC model is encoded with *PPHC*, *LR*, a projection map *PM*, and a ray map *RM*. *RM* stores the ray segments of the EOC and *PM* enables projection.

3. 1. EOC construction algorithm

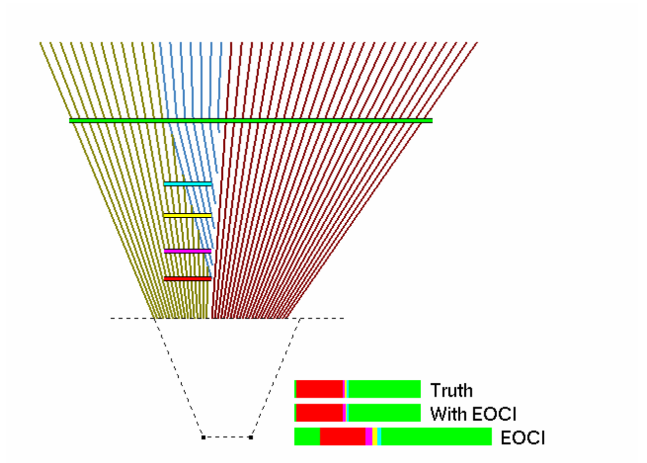
The EOC is constructed with the following steps.

```
EOC::EOC (S, PPHC, LR) {
1. Render S with PPHC from L to create depth buffer $ZB_0$
2. For every epipolar line $e_v$ defined by LR in $ZB_0$, and for every pixel u on $e_v$
    If u is not a depth discontinuity
        a. Update RM with ray (PPHC.Ray($e_v(u)$))
    If u is a depth discontinuity
        a. Compute additional set of rays $A_i$ from (v, $ZB_0$, u, $e_v$)
        b. Update RM with rays $A_i$
        c. Update PM using first and last rays $A_1$, and $A_n$
}
```

The first step creates a planar pinhole camera depth buffer ZB_0 of the scene, as seen from the left endpoint of the viewpoint of segments. The rows of the projection and ray maps correspond to the epipolar lines induced in ZB_0 by the translation *LR*. The second step walks on epipolar lines and sets the two maps one row at the time. While no depth discontinuity is found, the *PPHC* ray is simply added to *RM*. For a depth discontinuity, *PM* and *RM* are updated to introduce a set of additional rays. Figure 4 (also see color plate) shows the rays of a row of an EOC constructed for a scene with a single foreground object (*red*) occluding a background object (*green*). The row has the rays of *PPHC* located at *L* except for the additional set of rays (*blue*) which are created to handle the near to far depth discontinuity encountered when stepping off the foreground object, from P_3 to P_4. The corresponding EOC image row (*EOCI*) stores all background object samples needed as *PPHC* translates from left to right. *With EOCI* and *Truth* show the rows of a reconstruction from an intermediate viewpoint obtained from the samples of the EOCI and from geometry, respectively.

The set of additional rays is computed as follows. Let z_f be the depth at P_4. P_1 is computed as the intersection of RP_3 with the plane $z=z_f$. P_0 is the *PPHC (L)* projection of P_1 onto the image plane. For clarity, the image plane is chosen here to be coincident with the hither plane. P_2 is the image plane projection of P_3. The number of additional rays is the length of segment P_0P_2 in pixels. The additional rays are defined as pairs of rays (r_L, r_R), where r_L/r_R passes through *L/R*. The far point of r_R is the same as the near point of r_L, and they are defined by intersecting *R* rays with the $z=z_f$ plane. The additional rays shorten the original *L* rays between P_0 and P_2, but only if the intersection with the blue ray is not visible from *L*. In other words, the blue rays are not allowed to reemerge from underneath the occluder—their function is to sample the occluded regions. Figure 5 (also see color plate) shows a case where the occluder is narrow and the viewpoint segment is long enough to reveal all hidden samples. Some r_R blue rays are clipped by P_0P_1, do not reach z_f, and do not have a r_L pair. In the case shown here, these rays do not find any samples, and they correctly leave a black gap in the EOCI. Similar black gaps can be observed in Figure 1 and Figure 3 to the right of the teapot handle and of the pickets, respectively.

The ray map *RM* is updated with the set of additional rays A_i by simply appending the rays A_i to the rays that are already in the current row. Each *RM* location has room for a (r_L, r_R) pair of rays. Appending the additional rays effectively extends the row of the EOC image to accommodate hidden samples. Sufficient *RM* locations are used to provide the samples at full resolution, as dictated by *PPHC*.



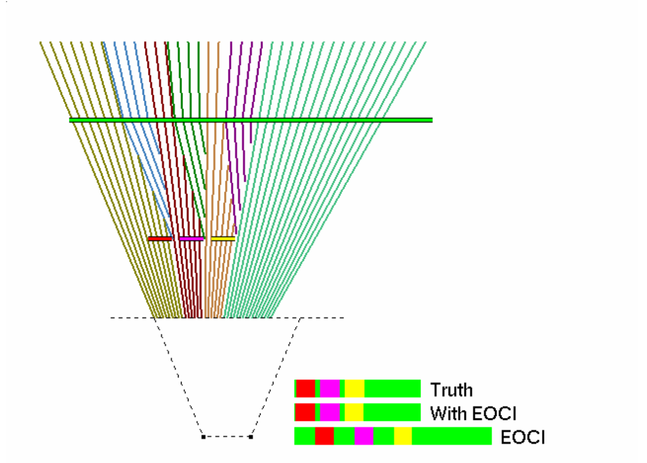


Figure 6 Visualization of an EOC row for a stack of occluders hidden by the closest one (*top*), and for side by side occluders separated by small gaps (*bottom*).

The function of *PM* is to enable projecting 3D points in the EOC image where rays have been inserted. *PM* has the same resolution as *PPHC*. A *PM* location stores two offsets o_0 and o_1 and two depth values z_0 and z_1. These four values model the EOC rays visible at a *PPHC* pixel. For example, using Figure 4 again, at the location u_{P0} corresponding to P_0, *PM* stores 0, u_{P2}-u_{P0}, z_f, and z_f, for o_0, o_1, z_0, and z_1, respectively. For P_2, *PM* stores 0, u_{P2}-u_{P0}, z_n, and z_f. For an intermediate ray (not shown), z_1 would be a value between z_f and z_n and the other 3 values remain the same. In order to project a 3D point *P* with the EOC, *P* is first projected with *PPHC* to find its corresponding *PM* location (o_0, o_1, z_0, z_1) along epipolar line e_v at pixel *u* (row *v* and column *u*). If the depth z_P of *P* is less than z_0, then the EOC image projection of *P* is (u+o_0, v). If z_P is greater than z_1, *P* projects to (u+o_1, v). If z_P is in between z_0 and z_1, *P* projects to the interval [(u+o_0, v), (u+o_1, v)].

Once an additional set of rays is inserted into *RM*, *PM* is updated with the first and last of the additional rays. In Figure 4, the *PM* locations that have to be updated are those corresponding to the segment P_0P_2. For a given location, a candidate z_0 value is computed by intersecting the location *L* ray with ray A_1. The candidate value replaces z_0 if closer than z_0. The candidate z_1 value is given by z_f and it replaces z_1 if farther. Offset o_1 is updated to o_0+ u_{P2}-u_{P0} if this candidate value is larger. Offset o_0 does not change.

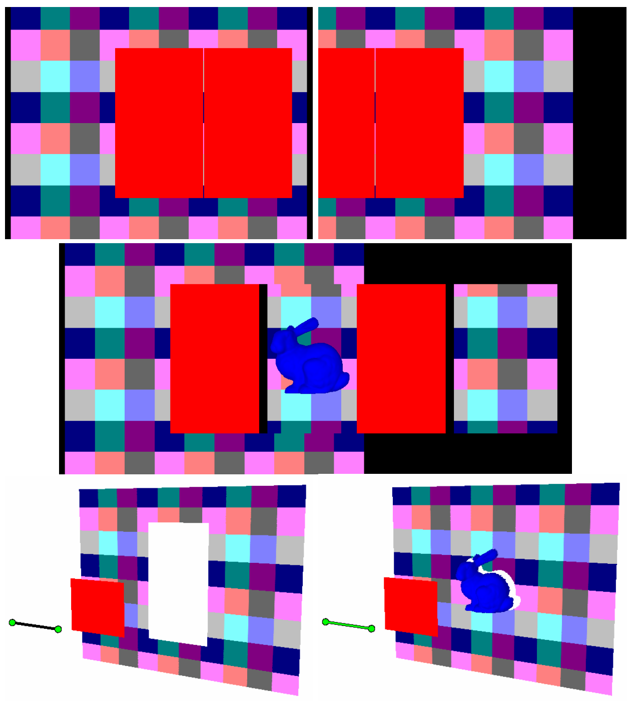


Figure 7 Left and right depth images (*top*), EOCI (*middle*) and third view visualization of the samples gathered by the two depth images (*bottom left*) and the EOC (*bottom right*).

3. 2. Performance

The EOC rays sample the space fully and non-redundantly resulting in good disocclusion capability. The EOC is not conservative, since it addresses only depth discontinuities that appear in *L*. However, the rays inserted handle additional “unforeseen” disocclusion errors well. For example, in the top image in Figure 6 (also see color plate), the blue, yellow and pink occluders are hidden by the red occluder and are not visible from *L,* so they are not addressed explicitly. Yet the disocclusions they cause are handled well by the additional rays introduced to handle the depth discontinuity caused by the red occluder. The bottom case corresponds to the picket fence shown in Figure 3.

The EOC makes room on epipolar lines for hidden samples that are needed as the viewpoint translates. We define the width of the EOC image (also the width of the ray map) as the maximum width of any of its rows. The width *w* of a row verifies the following inequality

$$w < w_0 + Ns,$$

where w_0 is the width of *PPHC*, *N* is the number of depth discontinuities on the row, and *s* is the maximum shift caused by one depth discontinuity. *s* can be computed as follows

$$s = LR \cdot (z_F - z_N) / z_F,$$

where *LR* is the length of the viewpoint segment, and z_F (z_N) is the farthest (closest) z_f (z_n) over all depth discontinuities. In practice, the width of an EOCI image is typically less than $2w_0$.

A projection map location stores 2 short integers and 2 floats for a total of 12 bytes. A ray map location stores 2 short integers and 4 floats for a total of 20 bytes. Consequently the memory size of an EOC constructed for a *PPHC* with resolution w_0 x h_0 is

$$12 \cdot w_0 h_0 + 20 \cdot 2 \cdot w_0 h_0 = 52 \cdot w_0 h_0 \text{ bytes.}$$

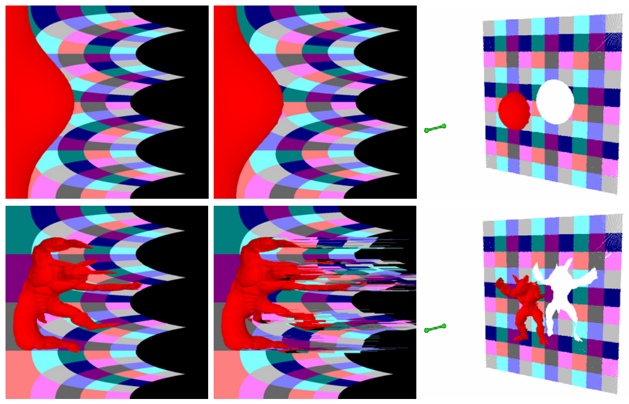


Figure 8 (*Top* and *bottom*) EOC images constructed for a short (*left*) and a long (*middle*) viewpoint segment aligned with the *PPHC* view direction, and the samples gathered by the EOC shown from a third view (*right*). As before, the EOC disoccludes on rows, now defined by radial epipolar lines.

However, as we will see later, once the EOC image is rendered, the EOC is not needed anymore.

Our current implementation computes the EOC model on the CPU with 4 independent steps: read back depth buffer ZB_0, compute depth discontinuities, compute ray map *RM*, and compute projection map *PM*. The average running times for each of these steps on a Pentium 4 Xeon 3.2GHz 2GB workstation with an Nvidia 7950GTX graphics card are 5, 32, 32, and 27ms, respectively, which yield an overall EOC model construction performance of 10Hz.

4. Rendering with the epipolar occlusion camera

The EOC is a non-pinhole, which poses two challenges for feed-forward rendering. First, one has to be able to project, functionality provided by the projection map in the case of the EOC. Second, rasterization is complicated by the fact that for a non-pinhole straight lines do not necessarily project to straight lines anymore. This implies that the edges of a polygonal primitive (i.e. triangle) are not necessarily line segments after projection, and that the bounding box of the projected triangle is not necessarily the bounding box of its projected vertices anymore. In the case of the SPOC, the simple distortion model allowed establishing a conservative bounding box for projected triangles and rasterizing directly in the output domain, which produced the correct curved triangle edges. In the case of the DDOC these problems were overcome by subdividing the scene triangles until conventional straight edge rasterization becomes an acceptable approximation. In the context of the EOC, the epipolar decomposition allows rasterization on segments with tight bounding boxes and output domain rasterization.

4. 1. EOC rendering algorithm

Given the 3D scene *S* and the *EOC*(*PPHC, LR, PM, RM)*, the epipolar occlusion camera image *EOCI* is computed by processing each triangle *t* in *S* with the following algorithm.

```
1. Project $t$ with $PPHC$ to $t'$
2. For every epipolar span $e_0e_1$ of $t'$
    Project $e_0 e_1$ with $PM$ into $RM$ to $e_0'e_1'$
    Rasterize $e_0'e_1'$ by intersecting $e_0e_1$ with rays $RM(e_0' \dots e_1')$
```

The triangle is split into spans according to the epipolar lines induced by *LR* into *PPHC*. A span e_0e_1 is projected with the EOC to find the *RM* segment $e_0'e_1'$ that stores the EOC rays that could intersect e_0e_1. A segment P_0P_1 is projected with an EOC by first projecting its endpoints P_0 and P_1 as described in the previous section. Let $[P_{00}, P_{01}]$ and $[P_{10}, P_{11}]$ be the projections of P_0 and P_1 (a point projects with an EOC to a segment or to a point, so P_{00} and P_{01}, and P_{10} and P_{11} could be identical). The projection of P_0P_1 is the bounding segment $[P_{00}, P_{11}]$. For every *RM* location e_j between e_0' and e_1' the span is intersected with the rays at e_j. If an intersection is found, the sample is written in the EOCI at the location corresponding to e_j.

Figure 7 gives another example of an EOCI. The bunny is not visible from the left and right viewpoints. The EOC opens the doors to sample the bunny which is needed as the viewpoint translates from left to right. The EOC image avoids disocclusion errors which are considerable when the two regular depth images are used, since they do not sample the bunny. Like in the case in Figure 3, the small gap between the two red occluders precludes any significant disocclusion when a DDOC is used. Figure 8 shows a case when the epipole (i.e. the intersection between the *PPHC* image plane and the segment of viewpoints *LR*) is inside the *PPHC* image, which causes radial epipolar lines.

4. 2. Performance

The EOC rendering algorithm is implemented on the GPU. A vertex shader implements the projection of the vertices of a triangle with *PPHC*, a geometry shader subdivides the triangle into spans and projects the spans with the projection map, and a pixel shader intersects the span with the one or two rays of each pixel of the projected span. For the *teapot* (1ktris , Figure 1), the *Unity* (110Ktris, Figure 2) the *picket fence* (1Ktris, Figure 3), the *double door* (16Ktris, Figure 7), the *sphere* (1Ktris, Figure 8, *top*), and the *Armadillo* (346Ktris, Figure 8, *bottom*), scenes the EOC image rendering times were 19, 1,793, 16, 36, 17, 1,130 ms, respectively. Performance depends on the number and size of triangles which determines the number of spans, and on the number and magnitude of depth discontinuities which determines the tightness of the EOC projection and implicitly the amount of overdraw (i.e. empty ray span intersections). Modifying the level of detail of the bunny in the *double door* scene to 69, 16, 4, and 1 ktris, the EOC image rendering times are 68, 25, 20, and 15 ms, respectively.

The EOC image stores color (4 bytes) and depth (1 float, or 4 bytes) samples, and a small amount of additional data needed to transform the depth and color data into 3D points. The additional data is either a short integer offset per sample which allows computing the *PPHC u* coordinate of the sample, or the depth discontinuities in the *PPHC* image from *L*. The second option has the advantage of compactness (2 short integers and 2 floats per depth discontinuity) but has the disadvantage of a slower conversion to 3D points. The first option increases the sample size from 8 to 10 bytes but allows a trivial unprojection to 3D points along *PPHC* rays. Consequently the memory footprint of an EOCI image for a *PPHC* with resolution w_0 x h_0 is 10 x 2 x w_0 x h_0 bytes, where we used again a typical upper bound for the EOC width of $2w_0$.

5. Applications

Like a regular depth image, an EOC image has a single layer, it is non redundant, and it can be rendered efficiently by projection followed by rasterization with hardware support. Unlike a regular image, the EOC has samples for a segment of viewpoints and not

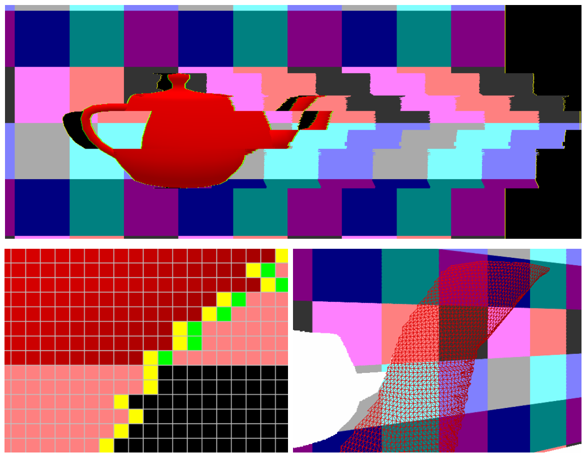


Figure 9 EOC image fragment (*top*), magnified fragment from the spout region with span start/end points highlighted with green/yellow and pixel grid (*bottom left*), and wireframe visualization of the reconstruction mesh (*bottom right*).

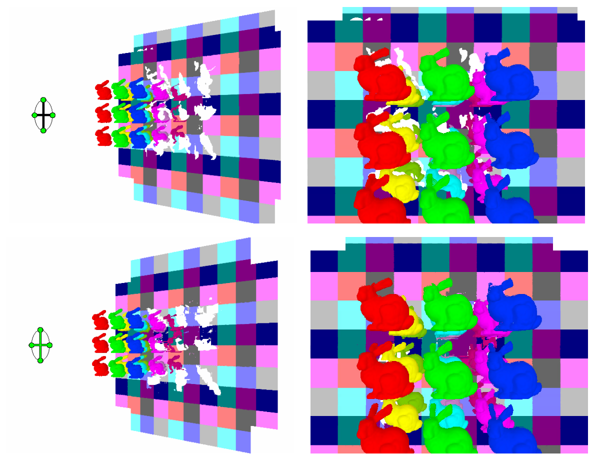


Figure 10 (*Top*) Four depth images rendered from the corners of the cross (*left*) leave important disocclusion errors (*right*). (*Bottom*) Even though the two EOCs do not sample the entire back wall (*left*) most disocclusion errors are avoided (*right*).

just for a single viewpoint. We sketch below several possible EOC applications, inherited from regular depth images.

5. 1. Geometry replacement

Depth images are a convenient solution to the challenging problems of level-of-detail adaptation and of occlusion culling, as work in image-based rendering by 3D warping [McMillan 1995], impostors [Maciel 1995, Popescu 2006], and geometric detail texturing [Oliveira 2000] has shown. However, the occlusion culling provided by a pinhole depth image is too aggressive and it quickly becomes obsolete as the viewpoint translates, which leads to the problem of disocclusion errors. For all these applications, substituting the pinhole depth images with EOC depth images brings the advantage of avoiding disocclusion errors. The EOC computes an occlusion culling that remains valid as the viewpoint translates along the viewpoint segment.

An EOC image sample is converted to a 3D point in one of two ways according to the data it stores. If each image sample (u, v, z) is enhanced with an offset, the 3D point is computed by unprojection with *PPHC*. If instead the EOC image stores depth discontinuities, the EOC model is recreated and the 3D point is computed by unprojection along the pair of rays (r_L, r_R) stored by the ray map. Once the EOC image is converted into 3D points, point-based rendering techniques (e.g. surfels [Pfister 2000], qsplat [Rusinkiewicz 2000]) can be used to produce high-quality output images at interactive rates. Even though the rows of the EOC image are misaligned, sufficient coherence remains to enable a straight forward triangulation by connecting corresponding spans on adjacent rows. Figure 9 shows how the triangulation of corresponding spans realigns the spout samples to form a mesh.

In order to increase the range of translations supported, multi EOC image configurations can be used. A combination of two EOC images—one with a vertical and one with a horizontal viewpoint segment is particularly effective (Figure 10).

5. 2. Compression

Image stream compression algorithms leverage data redundancy to minimize size while maximizing fidelity. One of the challenges is to devise mechanisms for easily locating data redundancy. In the case of images enhanced with per pixel depth, the depth data provides just such a mechanism. The samples of a key frame can be easily mapped to an intermediate frame by 3D warping, which reduces the size of the residual error image that needs to be stored for the intermediate frame. Depth is readily available for computer graphics imagery, and it can be approximated through proxy geometric scene models in the case of video. Disocclusion errors reduce the match between the warped key frame and the intermediate frame, which leads to increased residuals.

Using an EOC image for the key frame avoids disocclusion errors and leads to better compression performance. For example, in Figure 1, the depth image reconstruction misses 2,544 samples, which have to be encoded in the residual, whereas the EOC does not miss any sample. Just like in the case of regular depth images, the EOC image assumes a perfectly diffuse scene and color variation due to deviations from this ideal reflectance model have to be encoded in the residuals. In addition to small residuals, another important requirement is decoding efficiency. Since rendering from an EOC image is very efficient, this requirement is met as well. Note that the camera model is only needed for encoding and not for decoding. Encoding with an EOC is also efficient since the camera model construction is amortized over all the frames of the sequence.

5. 3. 3D display rendering acceleration

Most displays used in 3D computer graphics are 2D, which creates two important problems. First, the graphics system has to know the view desired by the user, which is typically done by burdening the user with unintuitive interfaces and cumbersome trackers. Second, the user sees the same image with both eyes which precludes depth cues. An ideal solution to these problems is a display that is truly 3D and that allows users to see freely a high quality 3D image. An overview of existing 3D display technologies is beyond the scope of this paper. However, an important common challenge is the computation and bandwidth costs required to render and display a 3D image. When the user(s) that view the 3D image are at known position(s), a possible solution to these problems is replacing the scene with a depth image rendered from the viewpoint(s) of the users. A pinhole depth image is insufficient. Even the parallax implied by the interpupillary distance of a single user creates disocclusion errors. Using an EOC image can avoid these problems. Figure 11 shows photographs of an autostereoscopic volumetric display which shows an EOC image in 3D. Although the display does not render opacity, the figure suggests that the EOC image provides sufficient samples for users seated in front of the display with views between those shown on the top row.

6. Conclusions and future work

We have presented a novel non-pinhole camera model designed to gather samples visible along a segment. A second important design criterion is fast projection, such that the image can be rendered efficiently in feed forward fashion. Additional rays are inserted at depth discontinuities to gather samples that become exposed for intermediate viewpoints. Unlike the single-pole and the depth-discontinuity occlusion cameras, the EOC does not restrict scene complexity, and provides a convenient way for the application to control the targeted set of viewpoints.

Immediate future work goals include accelerating the construction of the EOC model by off-loading it from CPU to GPU, and developing the applications sketched here. For efficiency we only considered depth discontinuities in the left image. When off-line EOC rendering is acceptable, conservative depth maps can be obtained by considering all depth discontinuities revealed as the viewpoint translates. The EOC generalizes the (0D) viewpoint to a (1D) *viewsegment*. We will investigate devising camera models that capture most samples seen as the viewpoint translates in 2D. Such a camera would produce a non-redundant light field. The ultimate goal is to design a camera model gathers all samples needed as the viewpoint translates in 3D (e.g. inside a tetrahedron). Another line of work that we consider is developing occlusion cameras models that can be constructed from sets of pinhole camera images, without the need of geometry. Designing a camera model with optimal properties for the problem at hand could be applied in other computer graphics contexts.

7. Acknowledgments

This work was supported by the NSF through grant CNS-0417458, by Intel and Microsoft through equipment and software donations, by the Tellabs Foundation through an equipment grant, and by the Computer Science Department of Purdue University.

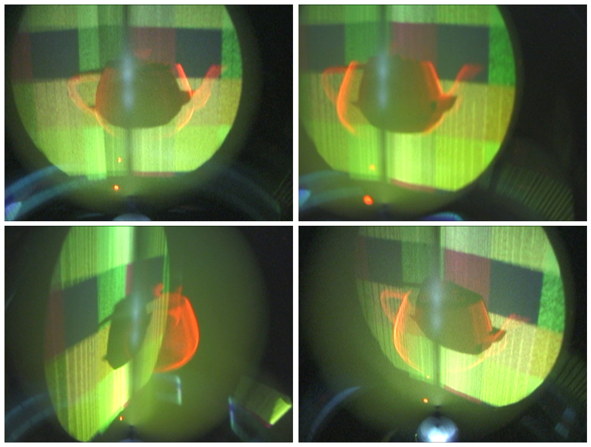


Figure 11 Photographs of a 3D display showing the samples captured by an EOC image similar to the one in Figure 1.

References

ALIAGA, D., AND CARLBOM, I. *A Spatial Image Hierarchy for Compression in Image-Based Rendering*. In proc. of IEEE International Conference on Image Processing (ICIP), 2005.

CHANG, C. F., BISHOP G., AND LASTRA A.. *LDI Tree: A Hierarchical Representation for Image-Based Rendering*. In proc. of SIGGRAPH'99.

GORTLER S., GRZESZCZUK R., SZELISKI R. AND COHEN M.. *The Lumigraph*. In proc. of SIGGRAPH 96, 43-54.

GUPTA, R. AND HARTLEY, R.I. 1997: *Linear Pushbroom Cameras*. IEEE Trans. Pattern Analysis and Machine Intell. vol. 19, no. 9 (1997) 963–975.

LEVOY M., AND HANRAHAN P. *Light Field Rendering*. In proc. of SIGGRAPH 96, 31-42 (1996).

MACIEL P., AND SHIRLEY P. *Visual Navigation of Large Environments Using Textured Clusters*, Symposium on Interactive 3D Graphics pp 95-102, 1995

MARK W., MCMILLAN L., AND BISHOP G. *Post-Rendering 3D Warping*. In proc. of 1997 Symposium on Interactive 3D Graphics (Providence, Rhode Island, April 27-30, 1997).

MAX, N., AND OHSAKI, K. *Rendering trees from precomputed z-buffer views*. In Rendering Techniques '95: Proc. of the Eurographics Rendering Workshop, 45–54, June 1995.

MCMILLAN, L., AND BISHOP, G. *Plenoptic modeling: An image-based rendering system*. In proc. SIGGRAPH '95, 39-46.

MEI, C., POPESCU V., AND SACKS E. *The Occlusion Camera*. In proc. of Eurographics 2005, Computer Graphics Forum, vol. 24, issue 3, sept 2005.

OLIVEIRA, M., BISHOP, G., AND MCALLISTER, D. *Relief Texture Mapping*. In proc. of SIGGRAPH 2000, 359-368.

PAJDLA, T *Geometry of Two-Slit Camera*. Research Report CTU–CMP–2002–02.

PFISTER, H., ZWICKER M., BAAR, J. VAN, AND GROSS M.. *Surfels: Surface Elements as Rendering Primitives*. In proc. of SIGGRAPH 2000, 335-342.

POLICARPO, F., OLIVEIRA, M., AND COMBA, J. *Real-Time Relief Mapping on Arbitrary Polygonal Surfaces*. In proc. of ACM Symposium on Interactive 3D graphics and Games, 155-162, 2005.

POPESCU, V., ROSEN, P., AND ALIAGA, D. *Three-Dimensional Display Rendering Acceleration Using Occlusion Camera Reference Images*. In IEEE/OSA Journal of Display Technology, vol. 2, no. 3, 274-283, 2006a.

POPESCU, V. ET AL. *Reflected Scene Impostors for Realistic Reflections at Interactive Rates*. In proc. of Eurographics 2006, 313-322, 2006b.

POPESCU, V. AND ALIAGA, D. *The Depth-Discontinuity Occlusion Camera.*. In proc. of ACM Symposium on Interactive 3D Graphics and Games, 139-143, 2006c.

POPESCU, V., AND LASTRA, A. *The Vacuum Buffer*. In proc. of ACM Symposium on Interactive 3D Graphics, 2001.

POPESCU V., et al. *The WarpEngine: An Architecture for the Post-Polygonal Age*. In proc. of SIGGRAPH 2000.

RADEMACHER P, AND BISHOP, G. *Multiple-center-of-Projection Images*. In proc of SIGGRAPH '98, 199–206.

RUSINKIEWICZ, S., AND LEVOY, M. *Qsplat: A Multi-Resolution Point Rendering System for Large Meshes*. In proc. of SIGGRAPH 2000.

SHADE J. et al. *Layered Depth Images*, In proc. of SIGGRAPH 98, 231-242.

YU J., AND MCMILLAN L. *General Linear Cameras* In 8th European Conference on Computer Vision (ECCV), 2004, Volume 2, pp 14-27.