Solving Visibility with Epipolar Geometry

Tien-Tsin Wong

The Chinese University of Hong Kong

Abstract

In this paper, we use a graphically explaination on how epipolar geometry can be used to solve the visibility. This is paper is prepared mainly for teaching purpose.

1 Introduction

Traditional geometry-based computer graphics requires significant amount of time to render complex scenery due to the dependency of the rendering time on the scene complexity. Even with the state-of-the-art graphics accelerator, interactive rendering is still far from satisfactory. Image-based computer graphics provides an alternative to render complex scene within a short period of time. Several image-based approaches have been proposed during the last few years. In this paper, we focus on solving the visibility problem when warping a given image (reference image) to generate an image (desired image) from a new viewpoint.

Given an image with depth, image as viewed from another viewpoint can be synthesized by reprojecting each pixel. Since multiple pixels may be mapped to the same location in the new image, visibility has to be resolved. The most straightforward method is depth-buffering. However, in some cases, the depth information may not be available or not accurate. This is especially common for real-world photographs. In that case, only the correspondences or optical flow information are determined. We cannot resolve the visibility by depth-buffering.

McMillan [5, 4] proposed a clever solution to the visibility problem. Once the mapping of pixels from the reference image to the desired image is known (either by pixel reprojection [1] or by finding the point correspondences [3] or optical flow [6]), the image can be warped correctly. *No* depth-buffering is needed. The visibility is solved by mapping pixels in a specific order. McMillan's original algorithm only works for pixel-sized entities. Fu, Wong and Heng [2] generalized the drawing order from pixel-sized entities to triangles.

2 Epipolar Geometry

In this paper, we describe some basics of epipolar geometry. Consider a planar perspective image I_c captured with the center of projection at $\dot{c}$. We use the dot notation $\dot{a}$ to denote a 3D point and the arrow notation $\vec{a}$ to denote a 3D directional vector. A desired image I_e is generated with a new center of projection at $\dot{e}$. Figure 1 shows the geometry in both 3D and 2D.

Each pixel i in the image I_c stores the radiance along the ray $\vec{L}$ which is fired from $\dot{c}$ passing through the pixel window associated with i. Now, let's choose an arbitrary pixel i_1 from image I_c. A ray $\vec{L_1}$ is associated with it. The intersection point $\dot{p_1}$ associated with i_1 must lie somewhere on the ray $\vec{L_1}$. To generate a new

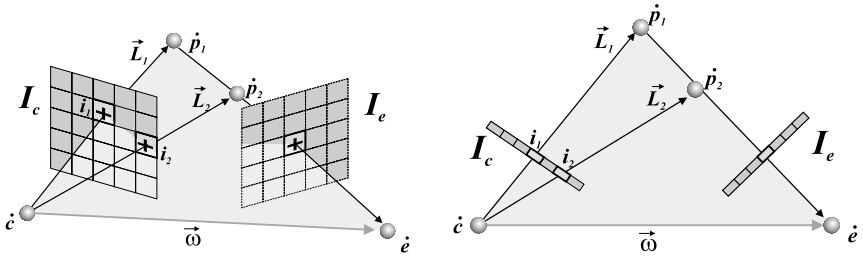


Figure 1: The geometry of two cameras (a) in 3D and (b) in 2D.

view from $\dot{e}$, $\dot{p_1}$ has to be reprojected onto I_e. The plane constructed by $\dot{c}$, $\dot{e}$ and $\dot{p_1}$ is known as *epipolar plane* in computer vision literature. The vector, $\vec{\omega}$, originated from $\dot{c}$ pointing towards $\dot{e}$ is called *positive epipolar ray* while the vector, $-\vec{\omega}$, originated from $\dot{c}$ pointing to the opposite direction is called *negative epipolar ray*.

Now let's choose another pixel i_2 from image I_c. Occlusion happens only when $\dot{p_1}$ and $\dot{p_2}$ are reprojected onto the same 2D location in I_e. If $\dot{p_2}$ does not lie on the epipolar plane associated with $\dot{p_1}$, then $\dot{p_1}$ and $\dot{p_2}$ will never occlude each other (see Figure 2). Hence occlusion happens only when $\dot{c}$, $\dot{e}$, $\dot{p_1}$ and $\dot{p_2}$ all lies on the same plane. Moreover the necessary condition of $\dot{p_2}$ occluding $\dot{p_1}$ is $\dot{e}$, $\dot{p_1}$ and $\dot{p_2}$ are collinear and $\dot{p_2}$ is in between $\dot{p_1}$ and $\dot{e}$, as illustrated in Figures 1 and 4.

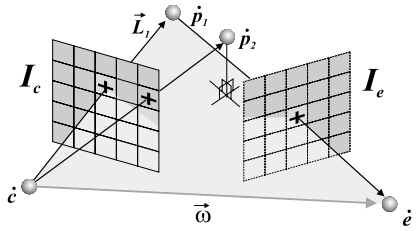


Figure 2: Since $\dot{p_2}$ does not lie on the epipolar plane associated with $\dot{p_1}$, $\dot{p_2}$ and $\dot{p_1}$ will never occlude each other.

From Figure 1, we know that $\dot{p_2}$ will never be occluded by $\dot{p_1}$ as viewed from $\dot{e}$ no matter where the exact positions of $\dot{p_1}$ and $\dot{p_2}$ are. Therefore, if we always draw $\dot{i_1}$ before $\dot{i_2}$ during reprojection, the visibility problem is solved without knowing or comparing their depth values. And hence, if we can identify those pixels whose intersection points may occlude each other and derive the drawing order, the visibility problem can be solved without depth-buffering.

To identify the pixels which may occlude each other, we first intersect the epipolar plane with the planar projection manifold (image I_c). The intersection line is called the *epipolar line*. Figure 3(a) illustrates the terminologies graphically. When the positive epipolar ray $\vec{\omega}$ intersects with the projection manifold I_c, the intersection point on the projection manifold is known as *positive epipole*. Figure 3 denotes it by a positive sign. On the other hand, if the negative epipolar ray intersects with the projection manifold, the intersection point is known as *negative epipole* and denoted by a negative sign. Note all epipolar lines pass through the epipole (either positive or negative). When the epipolar rays parallel to the planar projection manifold, no intersection point is found on the plane. All epipolar lines are in parallel.

All pixels in I_c that lie on the same epipolar line have a chance to occlude each other. Figure 4 shows

two pixels, i_1 and i_2, lying on the same epipolar line. Their associated intersection points $\dot{p_1}$ and $\dot{p_2}$ are coplanar and may occlude each other. And $\dot{p_1}$ will never occlude $\dot{p_2}$ as $\dot{p_1}$'s angle of derivation θ_1 is greater than, θ_2 of $\dot{p_2}$. In other words, if i_2 is closer to the positive epipole on the epipolar line than i_1, i_1 will never occlude i_2. Hence we should always draw i_1 first. The arrow on the epipolar line in Figure 4 indicates the drawing order of pixels. On the other hand, if i_2 is closer to the negative epipole on the epipolar line than i_1, i_2 will never occlude i_1. By intersecting all of the epipolar planes with the image I_c (Figure 3(b)), we obtain pictures of the drawing order (Figure 5). Note that once $\dot{c}$ and $\dot{e}$ are known, the picture of drawing order is already determined. It is not necessary to define the epipolar planes explicitly. Hence no depth information is required in constructing the drawing order.

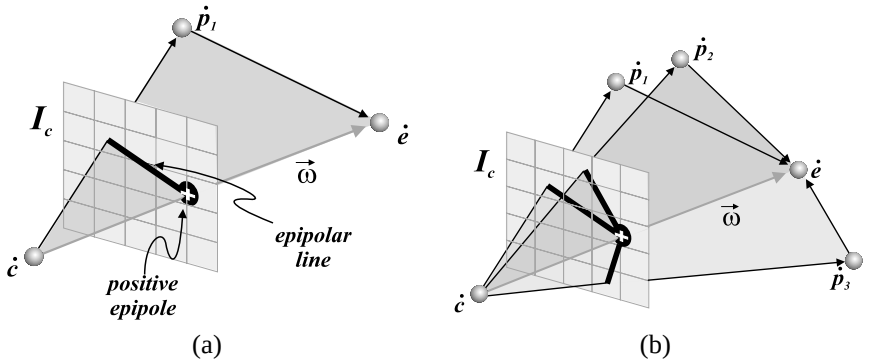


Figure 3: The epipolar line is the intersection of the projection manifold and the epipolar plane.

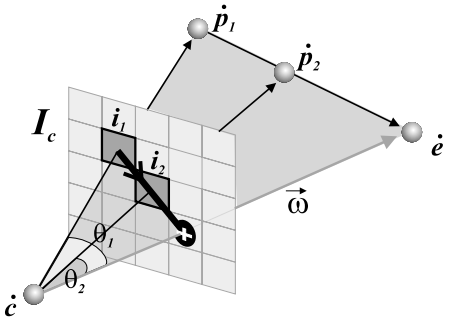
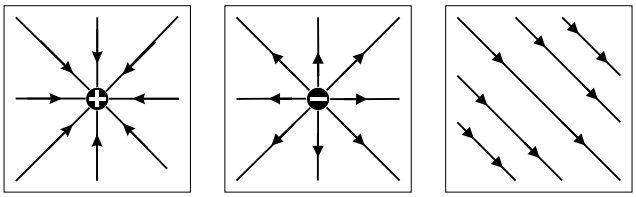


Figure 4: The drawing order between two pixels that lie on the same epipolar line.

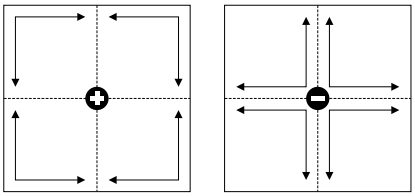
Only three main categories of drawing order exist. If the positive epipolar ray intersects with the projection manifold, a converging pattern (Figure 5(a)) will be obtained. On the other hand, if the negative epipolar ray intersects, a diverging pattern is resulted (Figure 5(b)). If the epipolar rays parallel to the projection manifold, the epipoles can be regarded as located infinitely far away and the epipolar lines will be all in parallel (Figure 5(c)).

McMillan derived two drawing orders of pixels from the patterns in Figure 5. They are shown in Figure 6. Following these drawing orders, the visibility can be correctly resolved. No depth-buffering is required. Pixels on different epipolar lines can be drawn in arbitrary order.



(a) (b) (c)

Figure 5: The drawing patterns



(a) (b)

Figure 6: Two drawing orders derived from the patterns of epipolar lines.

3 About the author

Tien-Tsin Wong, Department of Computer Science & Engineering,
The Chinese University of Hong Kong, Shatin, Hong Kong.
ttwong@acm.org
http://www.cse.cuhk.edu.hk/∼ttwong/

References

[1] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '93)*, pages 279–288, 1993.

[2] Chi-Wing Fu, Tien-Tsin Wong, and Pheng-Ann Heng. Triangle-based view interpolation without depth-buffering. *Journal of Graphics Tools*, 3(4):13–31, 1998.

[3] Stephane Laveau and Olivier Faugeras. 3-D scene representation as a collection of images. In *Proceedings of the Twelfth International Conference on Pattern Recognition (ICPR '94)*, pages 689–691, Jerusalem, Israel, October 1994.

[4] Leonard McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, 1997.

[5] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, pages 39–46, August 1995.

[6] K. Prazdny. On the information in optical flows. *Computer Vision, Graphics and Image Processing*, 22(9):239–259, 1983.