Textureless Layers

CMU-RI-TR-04-17

Qifa Ke, Simon Baker, and Takeo Kanade

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Abstract

Layers are one of the most well studied ways of representing a 3D scene. Although man-made scenes often contain constant intensity planar regions (textureless layers), it is almost always assumed that there is enough texture in each layer to compute the motion of the layer using image alignment and to base layer assignment on pixel-wise differencing. Since (the interior of) any textureless region is consistent with the motion of any layer, most existing algorithms assign constant intensity regions to the dominant layer, or to a random nearby layer. The one source of information that could be used to resolve the inherent ambiguity, namely the lines separating the constant intensity regions, is instead often treated as an outlier. In this paper we study the question of what can and cannot be computed in a 3D world consisting of a set of constant intensity planar regions (textureless layers). We derive an algorithm to determine when the shape of the scene is unique, when it is inherently ambiguous, and if so, what the set of possible scene shapes is.

Keywords: Layers, constant intensity regions, inherent ambiguities, Origami Theory.

1 Introduction

Layers are one of the most well studied ways of representing a 3D scene [Bergen *et al.*, 1992, Wang and Adelson, 1993, Hsu *et al.*, 1994, Darrell and Pentland, 1995, Sawhney and Ayer, 1996, Weiss and Adelson, 1996]. In such papers, it is usually assumed that there is enough texture in each layer to compute a homography or affine warp for that layer using image alignment [Lucas and Kanade, 1981, Bergen *et al.*, 1992]. In man-made scenes, however, there are often many textureless regions (layers.) Walls and ceilings are usually painted a single color. The tops of tables, and the sides of cabinets, are also usually textureless.

(The interior of) any textureless region is consistent with the motion of any layer. Most algorithms therefore tend to group textureless regions with the dominant layer, or with a large nearby layer. A good example is contained in Figure 13 in Sawhney and Ayer's seminal paper [Sawhney and Ayer, 1996]. This figure contains a person bouncing a table-tennis ball on their bat. Also visible is a large background plane. In the bottom of the figure part of the table-tennis table can also be seen. Conceptually, the scene consists of three layers, the dominant background layer, the layer of the table-tennis table, and the layer of the bat, ball, and arm. The dominant layer estimation algorithm in [Sawhney and Ayer, 1996], however, groups the (mostly) textureless table-tennis table with the dominant background layer. In the 2D video coding domain of [Sawhney and Ayer, 1996], this interpretation of the scene is valid. The textureless table-tennis table is consistent with the motion of the background plane. As a 3D interpretation of the scene it is not correct. The background and the table-tennis table lie in distinct 3D planes, one horizontal, the other vertical.

In this example the only visual information that could be used to determine that there are two layers rather than one is the line that separates them; i.e. the edge of the table-tennis table. When image alignment is used to compute the motion of layers, and pixel-wise differencing the consistency of pixels with layers, the information in the lines between constant intensity regions is given little very weight. As a result, the line between the table-tennis table and the background is treated as an outlier in [Sawhney and Ayer, 1996] rather than an important source of information.

In this paper we abstract the above problem and study the question of what can and cannot be computed in a 3D world consisting of a set of constant intensity (textureless) polygonal, planar regions (layers). In particular, we ask when the 3D shape of such as scene (the layer plane equations) can be recovered uniquely, and when the 3D shape is inherently ambiguous. Note that our ideal world of constant intensity, polygonal planar regions is similar to the "Origami World" of [Kanade, 1980]. Our analysis, however, is an analysis of what can be computed from multiple images using 3D vision, rather than a 2D analysis of junction labeling.

We assume that the constant intensity regions have been segmented, their polygonal boundaries have been detected, and the correspondences between the lines bounding the regions, and the regions themselves, have been computed. We derive an algorithm to compute the 3D shape of the scene, and the assignment of the lines to the layers. These two tasks correspond to the traditional layers tasks of layer motion estimation and pixel assignment to layers. In general, there are multiple solutions for the shape estimate and line assignment that are consistent with the input images. In such cases, our algorithm outputs every possible consistent interpretation of the images.

2 Problem Scenario

We assume that there is a single camera moving through a static scene. Equivalently, the scene could be moving rigidly, there could be multiple cameras, or a non-rigidly moving scene could be imaged simultaneously by multiple (stereo) cameras. How to extend the algorithm described in this paper to scenes where each layer is moving independently, but rigidly, is left as future work.

We assume that the scene under consideration consists of a collection of polygonal, constant intensity (and color), planar regions. The scene is imaged by a moving camera that captures the m images $I^1, I^2, \ldots, I^m$. See Figure 1 for an illustration. We assume the input images have been processed, the constant intensity regions have been segmented, 2D lines have been fit to the polygonal boundaries of the constant intensity regions, and the correspondences between the regions and the correspondences between the lines bordering the regions have been determined.

Denote the constant intensity regions in the j^{th} image $r_1^j, r_2^j, \ldots, r_n^j$. Knowing the correspon-

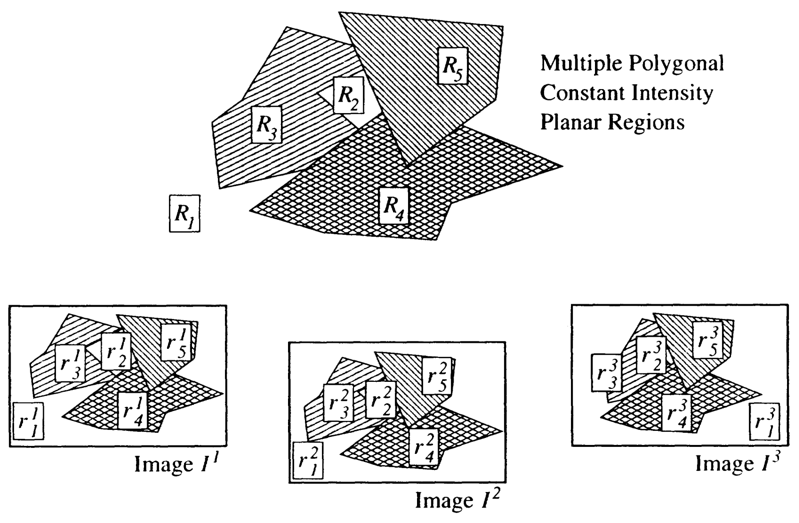


Figure 1: Problem Scenario. We assume that the scene under consideration consists of a collection of polygonal, constant intensity (and color), planar regions $R_1, R_2, \ldots, R_n$. The scene is imaged by a moving camera that captures the images $I^1, I^2, \ldots, I^m$.

dence between the regions means that r_i^j and r_i^k correspond to the same planar region in the scene for all j and k. Denote this 3D region R_i. Denote the i^{th} line in the j^{th} image $l_i^j = (\mathbf{f}_i^j, \mathbf{s}_i^j)$ where $\mathbf{f}_i^j$ is a column vector containing the 2D image coordinates of the first vertex, and $\mathbf{s}_i^j$ is a column vector containing the 2D image coordinates of the second vertex. Knowing the correspondence between the detected lines means that l_i^j and l_i^k are projections of the same 3D line in the scene for all j and k. Denote this 3D line L_i. Note, however, that because of occlusions, the points, $\mathbf{f}_i^j$ and $\mathbf{f}_i^k$, and the points, $\mathbf{s}_i^j$ and $\mathbf{s}_i^k$, do not necessarily correspond to the same 3D points in the scene for $j \neq k$. Also note that we assume that the lines are broken at all junctions, including t-junctions.

Each line borders two regions. While the regions and lines are being extracted, the two 3D regions bordering each 3D line are recorded in the following data-structure:

$$\text{Bordering}(L_i) \;=\; \{R_j, R_k\} \tag{1}$$

i.e. the 3D regions bordering the 3D line L_i are R_j and R_k.

Implicit in the above is the assumption that the motion of the camera is small enough so that the topology of the lines and regions does not change across the input images; i.e. no lines or regions appear or disappear in $I^1, I^2, \ldots, I^m$. We also assume that the depth ordering of the regions remains the same across the images. This is hardly an additional assumption since the ordering would only change if the camera moved from "in front of" to "behind" some of the planar regions, in which case the topology would likely change.

3 3D Line and Camera Matrix Recovery

Given enough corresponding 2D lines it is possible to reconstruct the 3D projective structure of the lines and the motion of the camera [Faugeras *et al.*, 1987, Weng *et al.*, 1993, Hartley, 1994, Taylor and Kriegman, 1995]. In particular, if 6 lines (in general position) are visible in 3 images it is possible to recover the 3D projective line equations and camera matrices [Faugeras *et al.*, 1987, Taylor and Kriegman, 1995]. If 13 lines are visible, there is even a linear algorithm to recover the scene structure and motion [Weng *et al.*, 1993, Hartley, 1994].

We assume that the 3D lines and the motion of the camera have been recovered, either using one of the above algorithms or some other technique. Denote the projective camera matrix for image I^j by $\mathbf{P}^j$. Denote the i^{th} 3D line, the one that corresponds to the 2D lines l_i^j, by $L_i = (\mathbf{F}_i, \mathbf{S}_i)$ where $\mathbf{F}_i$ and $\mathbf{S}_i$ are (column vectors containing) the 3D projective coordinates of two points on the line. Note that there is no correspondence between the 3D points $\mathbf{F}_i$ and $\mathbf{S}_i$ used to denote the 3D line and the 2D points $\mathbf{f}_i^j$ and $\mathbf{s}_i^j$.

4 Layer Estimation

The traditional approach to layer estimation consists of two tasks: (1) assign the pixels to the layers, (2) estimate the motion (affine warp or homography) [Bergen *et al.*, 1992] or plane equation [Baker *et al.*, 1998] of each layer. These two tasks are coupled. To estimate the motion of the layers, we need to know which pixels belong to each layer. To assign the pixels to the layers, we need know the motion of the layers. Layer estimation can be formulated in the Expectation-Maximization (EM) framework and is usually performed by iterating the two tasks [Sawhney and Ayer, 1996].

With textureless layers there are two corresponding tasks. In the first we assign 3D lines to layers (instead of assigning pixels to layers.) In the second task we compute the motion or plane equation of each layer. In the static scene scenario of this paper the situation is constrained enough to compute the plane equation of each layer. (If the scene is moving non-rigidly, the situation is less constrained and instead we would need to compute the motion of each layer as a homography.) Before we discuss each of the two tasks in turn, we first introduce some terminology for: (1) the assignment of lines to regions and (2) the layer plane equations.

4.1 Layer Assignment

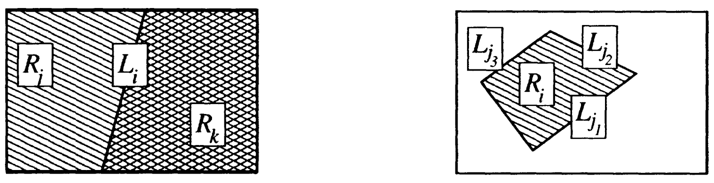
Suppose that Bordering$(L_i) = \{R_j, R_k\}$, as in Figure 2(a). There are three possible physical causes of the line L_i: (1) the region R_j is in front of and occluding the region R_k in which case R_j contains L_i but R_k does not, (2) the region R_k is in front of and occluding the region R_j in which case R_k contains L_i but R_j does not, and (3) the two regions R_j and R_k meet at and both contain the line L_i. There are therefore three ways to assign the 3D line L_i:

$$\text{Assign}(L_i) = \begin{cases} \{R_j\} & \text{if only } R_j \text{ contains } L_i \\ \{R_k\} & \text{if only } R_k \text{ contains } L_i \\ \{R_j, R_k\} & \text{if } R_j \text{ \& } R_k \text{ meet at } L_i. \end{cases} \tag{2}$$

In Section 4.3 we describe how Assign(L_i) can be computed once the plane equations are known.

4.2 Layer Plane Equations

Ideally we would like to estimate a plane equation for each region R_i. Unfortunately this is not always possible. To compute a plane equation we need at least two lines assigned to R_i. If less than 2 lines are assigned to R_i we cannot uniquely compute the plane equation. If one line is assigned to R_i we can only constrain the plane of R_i by that line. If no lines are assigned to R_i the plane is



(a) Line Assignment (b) Plane Equation Estimation

Figure 2: (a) Line assignment. There are three possibilities. (1) R_j is occluding R_k in which case L_i is assigned R_j. (2) R_k is occluding R_j in which case L_i is assigned R_k. (3) R_j and R_k meet at the line L_i in which case L_i is assigned both R_j and R_k. (b) Plane equation estimation. If two or more lines are assigned to R_i the plane equation of R_i can be estimated. See Section 4.4 for the details.

essentially unconstrained. Let π_i denote the plane equation of R_i where:

$$\pi_i = \begin{cases} (\mathbf{n}_i, d_i)^{\mathrm{T}} & \text{if two lines assigned to } R_i \\ L_j & \text{if one line } L_j \text{ assigned to } R_i \\ \emptyset & \text{if 0 lines assigned to } R_i\,. \end{cases} \tag{3}$$

In this definition $\mathbf{n}_i$ is a (column) vector normal to the plane and d_i is the distance to the plane (both defined up to scale); i.e. the fully constrained plane equation is defined by:

$$(x\; y\; z\; 1)\begin{pmatrix} \mathbf{n}_i \\ d_i \end{pmatrix} = 0. \tag{4}$$

In Section 4.4 we describe how π_i can be computed once the layer assignment is known.

4.3 Assigning Lines to Layers

Suppose that Bordering$(L_i) = \{R_j, R_k\}$, as in Figure 2(a). As in the traditional layers formulation, if the plane equations of the two layers R_j and R_k are known, it is possible to estimate Assign(L_i). If L_i lies in the plane of R_j, then $R_j \in$ Assign(L_i), and similarly for R_k. If L_i doesn't lie in either plane, there is an inconsistency. The estimate of one of the plane equations must be wrong. See Section 4.5 for more details. If π_j is fully defined and equals $(\mathbf{n}_j, d_j)^{\mathrm{T}}$, the layer assignment $\text{Assign}(L_i)$ can be computed:

$$\text{if} \begin{pmatrix} \mathbf{F}_i^{\mathrm{T}} & 1 \\ \mathbf{S}_i^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{n}_j \\ d_j \end{pmatrix} = \mathbf{0} \text{ then } R_j \in \text{Assign}(L_i) \tag{5}$$

where $L_i = (\mathbf{F}_i^{\mathrm{T}}, \mathbf{S}_i^{\mathrm{T}})$. If the plane $\boldsymbol{\pi}_j$ is just defined by one line and equals L_{j_i}, the assignment can be computed:

$$\text{if } L_i = L_{j_i} \text{ then } R_j \in \text{Assign}(L_i). \tag{6}$$

If the plane $\boldsymbol{\pi}_j$ is unconstrained and equals $\emptyset$, the assignment can be computed:

$$\text{if } \boldsymbol{\pi}_j = \emptyset \text{ then } R_j \notin \text{Assign}(L_i). \tag{7}$$

The equivalent of the rules in Equations (5)–(7) can also be applied for R_k to compute whether $R_k \in \text{Assign}(L_i)$.

4.4 Computing the Layer Plane Equations

Consider the region R_i in Figure 2(b) and the set of lines $L_{j_1}, \ldots, L_{j_p}$ assigned to R_i:

$$R_i \in \text{Assign}(L_{j_k}) \text{ for } k = 1, \ldots p. \tag{8}$$

The plane equation $\boldsymbol{\pi}_i$ of R_i can then be computed. First it is checked whether $p = 0$. If $p = 0$ then $\boldsymbol{\pi}_i = \emptyset$. Second it is checked if all of the lines L_{j_i} are the same line; i.e. co-linear. If all of the lines L_{j_i} are the same then $\boldsymbol{\pi}_i = L_{j_1}$. Finally, if there are more than two distinct lines, $\boldsymbol{\pi}_i = (\mathbf{n}_i, d_i)^{\mathrm{T}}$ is computed as follows. The p lines $L_{j_1}, \ldots, L_{j_p}$ are defined by the $2 \times p$ points, $\mathbf{F}_{j_1}, \mathbf{S}_{j_1}, \ldots, \mathbf{F}_{j_p}$,

$\mathbf{S}_{j_p}$. Since all of these points must lie in $\boldsymbol{\pi}_i$ it follows that:

$$\begin{pmatrix} \mathbf{F}_{j_1}^{\mathrm{T}} & 1 \\ \mathbf{S}_{j_1}^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \mathbf{F}_{j_p}^{\mathrm{T}} & 1 \\ \mathbf{S}_{j_p}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{n}_i \\ d_i \end{pmatrix} = \mathbf{A}_i \boldsymbol{\pi}_i = \mathbf{0}. \tag{9}$$

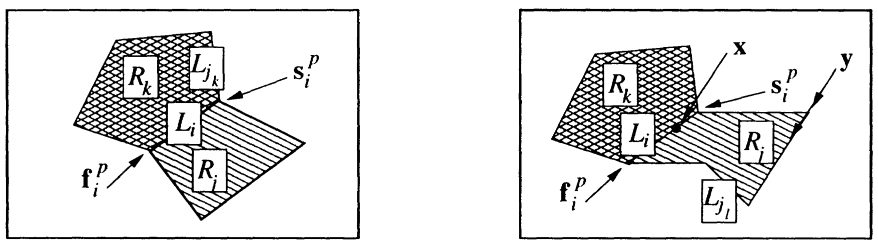
The plane equation $\boldsymbol{\pi}_i$ can therefore be computed by performing a Singular Value Decomposition (SVD) on $\mathbf{A}_i$. If there is no solution to Equation (9), the lines L_{j_k} are not co-planar. The assignment of lines to regions is therefore (locally) inconsistent. See Section 4.5 for more details.

4.5 Layer Consistency

Layer estimation is usually posed as optimizing a notion of layer consistency. Suppose the assignment of pixels to layers and the motion (or plane equations) of the layers are given. The layer consistency is a function that specifies how good a solution this is. With textured layers, the layer consistency is usually a measure of how well the layers regenerate the input images. With textureless layers, layer consistency is boolean valued. Either the layers are consistent with the input images or they are not. Suppose that $\text{Assign}(L_i)$ has been computed for each line L_i and the plane equation $\boldsymbol{\pi}_i$ for each region R_i. There are then two components to the layer consistency: (1) local consistency, i.e. for each region, is the plane equation consistent with the assignment of lines to that region, and (2) depth ordering, i.e. is the depth ordering implied by the plane equations consistent with the occlusion ordering implied by the assignment of lines to regions.

4.5.1 Local Consistency

Local consistency means that the plane equation for each region R_i is consistent with the 3D line equations of the all of the lines L_j that are assigned to R_i, $R_i \in \text{Assign}(L_j)$. We also check that $\text{Assign}(L_j)$ is well defined. Local consistency consists of checking the following conditions:



(a) Depth Ordering 'Two Lines" (b) Depth Ordering 'One Line"

Figure 3: Depth ordering consistency checks for planes defi ned by (a) two lines and (b) one line.

- For all j, $\text{Assign}(L_j) \neq \emptyset$.
- For all j, $\text{Assign}(L_j) \subseteq \text{Bordering}(L_j)$.
- For all i, j, $R_i \in \text{Assign}(L_j)$, $L_j = (\mathbf{F}_j^{\mathrm{T}}, \mathbf{S}_j^{\mathrm{T}})$:
  - If $\pi_i = (\mathbf{n}_i, d_i)^{\mathrm{T}}$ is defined by two lines then:

$$\begin{pmatrix} \mathbf{F}_j^{\mathrm{T}} & 1 \\ \mathbf{S}_j^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{n}_i \\ d_i \end{pmatrix} = \mathbf{0}. \tag{10}$$

  - If $\pi_i = L_{j_i}$ is defined by 1 line then $L_{j_i} = L_j$.
  - If $\pi_i = \emptyset$ is defined by 0 lines then inconsistent.

4.5.2 Depth Ordering

If the depth ordering is correct, then for every line L_i assigned to R_j where $\text{Bordering}(L_i) = \{R_j, R_k\}$, the plane of R_j must be "in front of" the plane of R_k along the line L_i. In this definition, "in front of" means closer to or at the same distance from the camera center of projection. Because checking this condition for planes just defined by one line is tricky, it is easiest to check this condition for each region R_j in turn (rather than each line L_i). Depending on whether R_j is defined by 2, 1, or 0 lines, there is a different condition to be checked. We now describe each of these conditions in turn for the region R_j with plane equation π_j.

Planes Defined by Two Lines

Suppose that the plane equation $\boldsymbol{\pi}_j = (\mathbf{n}_j, d_j)^{\mathrm{T}}$ of R_j is defined by two lines. We then consider each line L_i for which $R_j \in \text{Bordering}(L_i)$. If $R_k \in \text{Bordering}(L_i)$ is the other region that borders L_i the situation is as in Figure 3(a). The two points $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$ are the end points of the 2D line l_i^p in image I^p that corresponds to the 3D line L_i. Although there are three possibilities for $\text{Assign}(L_i)$, we only need to consider:

1. $R_j \in \text{Assign}(L_i)$: We need to check that the plane of R_j lies in front of the plane of R_k along L_i.

2. $R_k \in \text{Assign}(L_i)$: We need to check that the plane of R_k lies in front of the plane of R_j along L_i.

The case that both $R_j, R_k \in \text{Assign}(L_i)$ is then taken care of by checking both conditions. The way in which these conditions are checked depends on how many lines $\boldsymbol{\pi}_k$, the plane equation of R_j, is defined by:

1. If $\boldsymbol{\pi}_k = (\mathbf{n}_k, d_k)^{\mathrm{T}}$ is defined by two lines, the depth ordering of the planes is checked by checking the depth ordering along the rays defined by the two points $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$. See Appendix A for a description of how to do this.

2. Suppose $\boldsymbol{\pi}_k = L_{j_k}$ is defined by one line. If $\mathbf{f}_i^p$ lies on $l_{j_k}^p$ the projection of L_{j_k} into I^p we check the depth ordering of the plane $\boldsymbol{\pi}_j$ and the line L_{j_k} along the ray defined by $\mathbf{f}_i^p$. Similarly if $\mathbf{s}_i^p$ lies on $l_{j_k}^p$ we check the depth ordering of the plane $\boldsymbol{\pi}_j$ and the line L_{j_k} along the ray defined by $\mathbf{s}_i^p$. If L_{j_k} equals L_i then both of these conditions are satisfied and so the depth must be checked for both $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$. Again, see Appendix A for a description of how to check the depth along a given ray. Figure 3(a) illustrates the case that only $\mathbf{s}_i^p$ lies on the projection of L_{j_k} into I^p.

3. If $\boldsymbol{\pi}_k = \emptyset$ is defined by 0 lines, then assuming local consistency has been checked, it must be the case that $R_j \in \text{Assign}(L_i)$ and $R_k \notin \text{Assign}(L_i)$. There is then nothing to do to check

that R_j lies in front of R_k.

Planes Defined by One Line

Suppose that the plane equation $\pi_j = L_{j_l}$ of R_j is defined by one line. We then consider each line L_i for which $R_j \in \text{Bordering}(L_i)$. If $R_k \in \text{Bordering}(L_i)$ is the other region that borders L_i the situation is as in Figure 3(b). The 2D lines l_i^p and $l_{j_l}^p$ are then intersected to give the pixel $\mathbf{x}$. If $\mathbf{x}$ lies between the two end points $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$ of the 2D line l_i^p the depth ordering check for "planes defined by 2 lines" is performed for the point $\mathbf{x}$ rather than for $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$. The depth of $\mathbf{x}$ for R_j is computed by intersecting with the line L_{j_l}. See Appendix A for the details. Otherwise the depth ordering check is exactly as above for "planes defined by two lines." If $L_i = L_{j_l}$ the depth ordering check should be made for both $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$. If $\mathbf{x}$ exists, it usually, although not always, equals either $\mathbf{f}_i^p$ or $\mathbf{s}_i^p$. Figure 3(b) illustrates the case that $\mathbf{x}$ does not equal either of the end points of l_i^p.

The depth ordering check described above checks that every point on the line L_{j_l} is correctly ordered with respect to the neighboring planes. Since the plane $\pi_j = L_{j_l}$ is just defined by this one line, this is the main thing that we need to check about R_j. It is not the only thing, however. The plane π_j could be any plane "rotated" about the line L_{j_l}. Although we do not know the rotation of this plane, we need to check that there is a "rotation" that is consistent with the depth ordering implied by the layer assignment.

In particular, consider Figure 3(b) where the plane equation of R_j is defined by the one line $\pi_j = L_{j_l}$. Consider the point $\mathbf{s}_i^p$. Since $L_i \neq L_{j_l}$ then $\text{Assign}(L_i) = \{R_k\}$. We therefore know that the region R_k is in front of R_j at the point $\mathbf{s}_i^p$. This puts a constraint on the rotation of the plane of R_j about $\pi_j = L_{j_l}$. We determine whether all of these constraints can be simultaneously satisfied as follows.

Consider any fixed vertex $\mathbf{y}$ of R_j that does not lie on the 2D line $l_{j_l}^p$ corresponding to L_{j_l}. We then consider every line L_i that borders R_j that does not equal L_{j_l}, as in Figure 3(b). We therefore know that $\text{Assign}(L_i) = \{R_k\}$ where R_k is the other region bordering L_i. We then consider the two points $\mathbf{s}_i^p$ and $\mathbf{f}_i^p$. Since $\text{Assign}(L_i) = \{R_k\}$ we can compute the 3D location of the points on

R_k that project to these two points. For each point in turn we compute the plane through the line L_{j_l} and the point on R_k corresponding to $\mathbf{s}_i^p$ or $\mathbf{f}_i^p$. We then compute the depth of the intersection of this plane with the ray through $\mathbf{y}$. See Appendix A for a description of how. If $\mathbf{f}_i^p$ (or $\mathbf{s}_i^p$) is on the same side of L_{j_l} as $\mathbf{y}$ this distance is a lower bound on the distance of the point $\mathbf{y}$ (which implicitly constrains the "rotation"). Similarly, if $\mathbf{f}_i^p$ (or $\mathbf{s}_i^p$) is on the other side of L_{j_l} from $\mathbf{y}$ this distance is an upper bound on the distance of the point $\mathbf{y}$. If all of these constraints on the depth of $\mathbf{y}$ (over $\mathbf{f}_i^p$ and $\mathbf{s}_i^p$ for each $L_i \neq L_{j_l}$) cannot be simultaneously satisfied then there is a depth ordering inconsistency.

Note that if R_j is convex then all of the constraints on the depth of the point $\mathbf{y}$ are in the same direction and so there is no way that the "valid rotation" depth ordering consistency check can produce an inconsistency.

Planes Defined by 0 Lines

If the plane equation $\pi_j = \emptyset$ of R_j is defined by 0 lines there is nothing to check for the depth ordering. Since the plane is defined by 0 lines, all of the lines L_i that border R_j are assigned to other planes. We therefore just need to check if it possible that this plane could be behind all of the other planes. This condition can always be satisfied.

4.6 Finding Consistent Interpretations

In the traditional layers formulation the goal is to find the assignment of pixels to layers and motions (or plane equations) of the layers that optimizes the layer consistency measure. With our boolean valued consistency measure (see Section 4.5), we pose the textureless layers problem as finding the set of consistent assignments of lines to layers (see Section 4.1) and plane equations (see Section 4.2). In the traditional formulation the layer consistency measure is optimized using some form of gradient descent such as the EM algorithm [Sawhney and Ayer, 1996]. In the textureless case, the optimization is a combinatorial optimization. In both cases, we need to decide how to initialize the algorithm. We could either initialize the assignment or the plane equations. For textureless layers, this leads to two algorithms:

Algorithm 1

For every possible way to initialize Assign(L_i) for every line L_i; i.e. for the three alternatives in Equation (2):

1. Compute π_j for every region R_j using Section 4.4. For each region, check that there is an exact solution of Equation (9). If there is no exact solution, there is no consistent solution so begin the next initialization of Assign(L_i). In this way the local consistency is checked for each region as we compute its plane equation. Hence, there is no need to check it separately.
2. Check the depth ordering for each region R_j. If any R_j fails, begin the next initialization of Assign(L_i).
3. Output the set of consistent assignments and the corresponding computed plane equations.

Algorithm 2

For every possible way to initialize the plane equations π_j for every region R_j:

1. Compute the line assignment Assign(L_i) for every line L_i using the procedure in Section 4.3. In the process check that L_i is assigned to at least one of the two regions in Bordering(L_i). If not, there is no consistent solution so begin the next initialization of π_j. In this way the local consistency is checked for each line as we compute the line assignment. There is therefore no need to check local consistency separately.
2. Check the depth ordering for each region R_j. If any R_j fails, begin with the next initialization of π_j.
3. Output the set of consistent plane equations and the corresponding computed line assignments.

The set of possible initial plane equations can be generated in the following way. Given region R_j consider every line L_{j_1} that borders R_j. Each of these defines one possible plane equation $\pi_j = L_{j_i}$ defined by one line. Secondly, consider every pair of lines L_{j_1} and L_{j_2} that border R_j. These lines are checked to see whether they are coplanar (using Section 4.4). If they are coplanar, the plane equation defined by those two lines is computed using Section 4.4 and added to the list of plane equation candidates. Finally, the plane defined by 0 lines $\pi_j = \emptyset$ is added to the candidates.

4.6.1 Choosing the Algorithm to Use

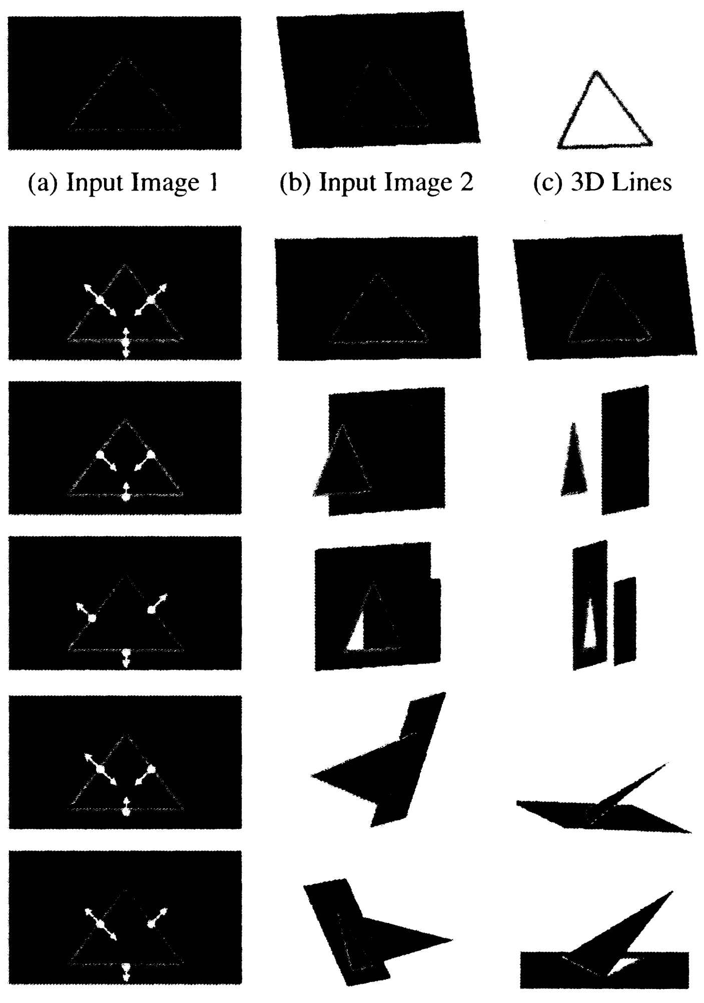
To decide whether to use Algorithm 1 or 2, we count the number of initializations that will be required and choose the smaller of the two. For Algorithm 1 the number will be:

$$3^{\text{no lines}} \tag{11}$$

For Algorithm 2 the number of initializations will be:

$$\prod_{j=1}^{n} \text{cand}(R_j) \tag{12}$$

where $\text{cand}(R_j)$ is the number of candidate plane equations for region R_j. Although theoretically $\text{cand}(R_j)$ can be as large as $\frac{1}{2}(l^2 + l + 2)$ where l is the number of lines bordering R_j, in practice is it usually closer to the minimum possible which is $l+1$. Comparing the values in Equation (11) and Equation (12) we can decide which algorithm to use. Almost always it is better to use Algorithm 2 because there are usually far more lines than regions; i.e. no lines $\gg n$.



(a) Input Image 1 (b) Input Image 2 (c) 3D Lines

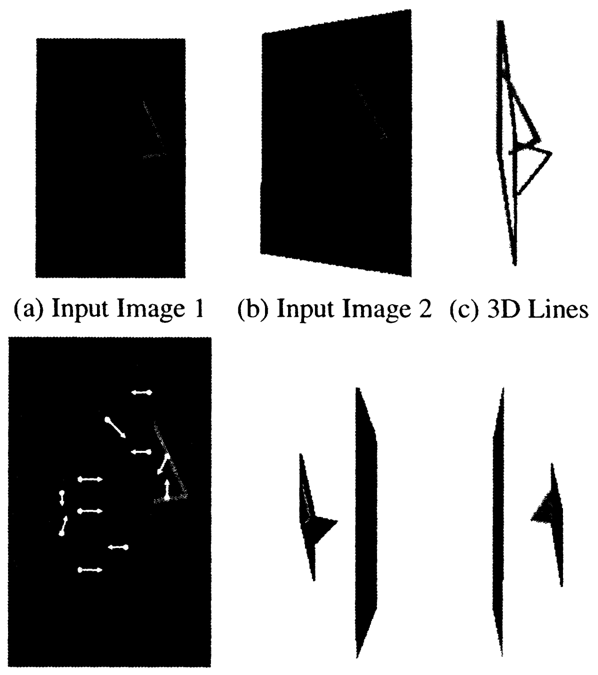
(d) Layer Assignment (e) Rendering 1 (f) Rendering 2

Figure 4: An example (a–c) with multiple globally consistent solutions (d–f). The 9 solutions can be grouped into 5 types, each displayed in a separate row. (d) The layer assignment. (e–f) Renderings of the planes. See `http://www.ri.cmu.edu/projects/project_528.html` for fly-by movies.

5 Experimental Results

5.1 Illustrative Examples

We first applied our algorithm to a variety of synthetic inputs to illustrate the various possible scenarios. The most common case, illustrated in Figure 4, is that there are multiple globally consistent solutions. Two input images and the 3D lines are shown in Figures 4(a), (b), and (c). The scene consists of a single triangle in 3D space, with two textureless regions, one in-



(a) Input Image 1 (b) Input Image 2 (c) 3D Lines

(d) Layer Assignment (e) Rendering 1 (f) Rendering 2

Figure 5: An example (a–c) with a single globally consistent solution (d–f).

side and one outside the triangle. Our algorithm finds 9 globally consistent solutions. The solutions can be grouped into 5 types. One solution of each type is shown in rows 2–6 of Figure 4. In the first column of each row we illustrate the layer assignment by drawing arrows on one of the input images to show which region(s) each line is assigned to. In the other 2 columns we present renderings of the computed plane equations from 2 different viewpoints. See `http://www.ri.cmu.edu/projects/project_528.html` for fly-by movies. In the first type of solution (row 2, 1 solution) there is a single plane with a triangular region "painted" on it. In the second type of solution (row 3, 1 solution) the triangle is a plane "floating" in front of a background plane (which is unconstrained.) In the third type of solution (row 4, 1 solution) the triangle is a "hole" in a plane (with an unconstrained background plane.) In the fourth type of solution (row 5, 3 solutions) the triangle is plane in front of a background plane which is joined to the triangle along one edge. In the fifth type of solution (row 6, 3 solutions) the triangle is "hole" in front of a background plane which is joined to the "hole" along one edge.

An example with a single globally consistent solution is shown in Figure 5. (Note that there

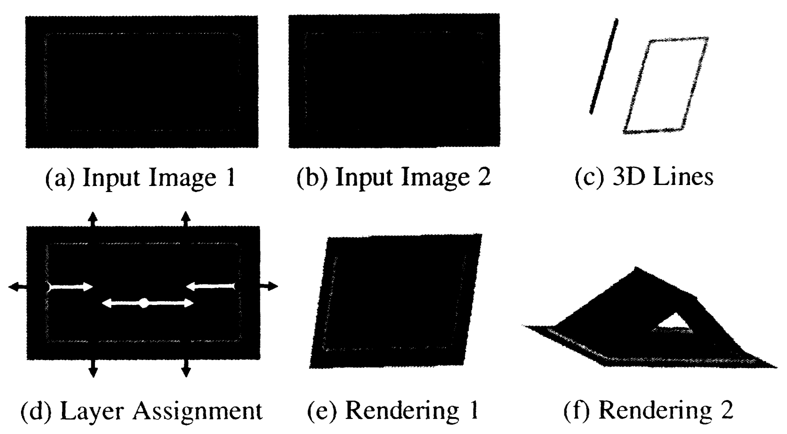


Figure 6: An example (a–c) with 24 locally consistent solutions, none of which are globally consistent solutions. (d–f) One of the locally consistent solutions.

are actually 24 locally consistent solutions.) This case is rare. There are almost always multiple solutions. The scene consists of 3 layers, a quadrilateral and 2 triangles. Two input images and the 3D lines are shown in Figures 5(a), (b), and (c), the assignment of lines to layers in Figure 5(d), and 2 renderings of the recovered planes in Figures 5(e) and (f). See the project webpage `http://www.ri.cmu.edu/projects/project_528.html` for a fly-by movie.

An example with 24 locally consistent solutions but no globally consistent solution is shown in Figure 6. The scene consists of 3 regions, 2 neighboring rectangular regions "inside" a background region. The 3D lines consist of a rectangle with a line "floating" in front of it. One of the 24 locally consistent solutions is shown in Figures 6(d–f). This solution consists of a 2 plane "wedge" on top of the background plane. This "solution" is not consistent with the input images because the geometry of the solution in Figures 6(e–f) implies that the input images should contain the edges of the wedge. However, the 3D lines in Figure 6(c) only contain the floating edge and the edges of the background rectangle. Note that if the floating edge were "behind" the rectangle in Figure 6(c) there would be a globally consistent solution. See the project webpage `http://www.ri.cmu.edu/projects/project_528.html` for a fly-by movie.

An example with no locally consistent solutions is shown in Figure 7. Although the input images consist of a single triangle just like in Figure 4, no pair of 3D lines are co-planar. It is therefore impossible to assign the 3 lines to the 2 regions and then compute a valid plane equation

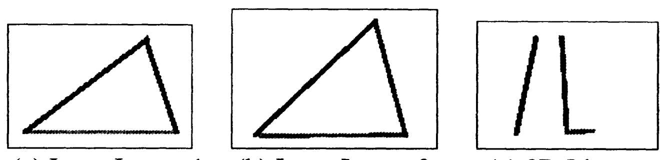

(a) Input Image 1 (b) Input Image 2 (c) 3D Lines

Figure 7: An example with no locally consistent solutions. The inputs (a–b) consist of a single triangle like in Figure 4. No pair of 3D lines (c) are co-planar, however, and so there is no solution.

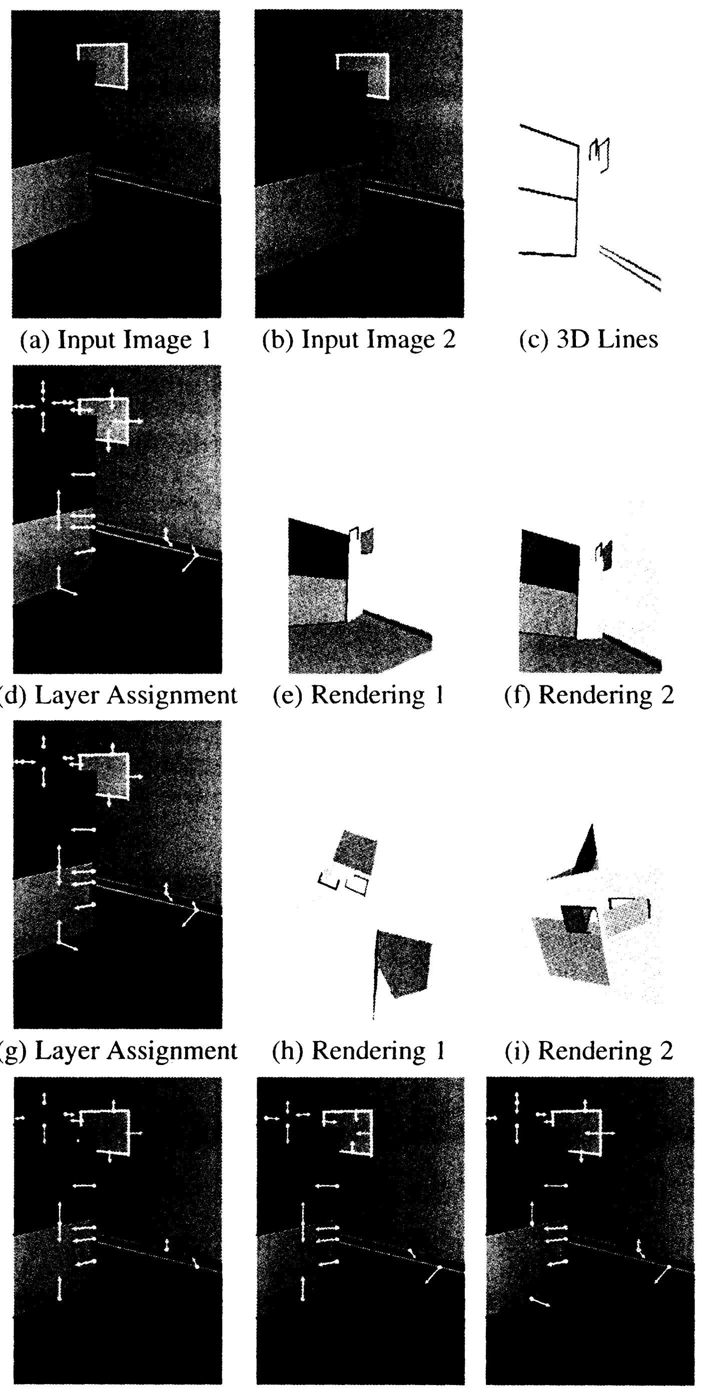
for both regions. See also `http://www.ri.cmu.edu/projects/project_528.html`.

5.2 Results on Real Images

We also applied our algorithm to a set of images of a real scene of a "corner walkway". Two input images, with the 2D lines overlayed on them, are included in Figures 8(a) and (b). The scene is typical of many encountered in 3D reconstruction and robot navigation tasks. Because the scene is largely man-made, there is little, if any, texture in any of the regions. The recovered 3D lines are shown in Figure 8(c). Our algorithm finds 448 globally consistent solutions. The solution that is "most plausible" to us as humans is shown in Figures 8(d–f). The line assignment is shown in Figure 8(d). Figures 8(e) and (f) contain 2 renderings of the reconstructed 3D planes. See `http://www.ri.cmu.edu/projects/project_528.html` for a fly-by movie.

Figures 8(g–i) show another solution. In this solution, the "door" is ajar; i.e. it is a plane defined by the one line where the door is connected to the wall. This solution is also a valid solution (assuming we cannot see the small "crack" at the top of the door if it is ajar.) The degree to which the door is ajar is not uniquely computed by our algorithm (although there are constraints on it.) We just know that the door is connected to the wall along the appropriate line. Also in this solution, the "white-board" has becomes a hole (an opening into the room behind.) Although less likely in practice, walls can have holes in the them and so this solution is a valid 3D interpretation.

Figures 8(j), (k) and (l) show the line assignments for three more solutions. In Figure 8(j), the "floor" is a unconstrained plane (i.e., a hole). This is unlikely in practice, however, this reasoning is based on the high-level knowledge that there is normally a ground plane. In Figure 8(k), the "wall" is an unconstrained plane. This leaves the "white-board" floating in mid air. This interpretation is



(a) Input Image 1 (b) Input Image 2 (c) 3D Lines

(d) Layer Assignment (e) Rendering 1 (f) Rendering 2

(g) Layer Assignment (h) Rendering 1 (i) Rendering 2

(j) Another Solution (k) Another Solution (l) Another Solution

Figure 8: A real example (a–c) with 448 globally consistent solutions. (d) The layer assignment of the "most plausible" solution. (e–f) Renderings of the planes. (g–i) Another solution. (j–l) The layer assignment for three more solutions. See also `http://www.ri.cmu.edu/projects/project_528.html`.

also unlikely, however, this reasoning is based on the knowledge that objects rarely float in mid air. As with all the solutions, the layers are a valid 3D interpretation of the images.

The results in Figure 8 are typical in the sense that there are generally a large number of globally consistent solutions. The main cause is the large number of solutions with planes constrained by 1 line, which can explode combinatorially. Of the 448 solutions, 434 contain regions with one or more plane equations constrained by a single line. Of the remaining 14 solutions, 13 solutions contain layers with unconstrained plane equations. The final solution, where every layer is defined by two or more lines, is the one shown in Figure 8(d–f), i.e. the "most plausible" one.

The development of heuristics (such as choosing the solution with the largest number of plane equations defined by 2 or more lines, or the solution with the largest number of edge assignments) and the use of prior knowledge (such as the knowledge that there is normally a ground plane and that planes cannot float in mid air) is left as future work.

6 Conclusion

We have investigated the task of computing a layered representation of a scene when the scene consists of a collection of constant intensity regions. Assuming the scene is static, or moving rigidly, we have presented an algorithm to compute every set of layer plane equations and layer assignments that are consistent with the inputs. Our world of "Textureless Layers" is very similar to the "Origami World" of [Kanade, 1980]. Our algorithm is similar in that: (1) the only source of information is a set of lines, and (2) it is a (brute force) combinatorial search for a consistent solution. On the other hand, our algorithm considers the consistency of a set of 3D planes with multiple input images rather than looking for a consistent 2D junction labeling in a single image.

This paper is a first attempt to study "Textureless Layers". Questions for further study include: (1) how to extend our algorithm to non-rigid scenes where each layer is moving independently, (2) how to combine our algorithm with traditional layers algorithms when the scene consists of both textured and textureless regions, (3) how to extend our algorithm to cope with topology changes in the input images, (4) whether it is possible to combine our algorithm with [Kanade, 1980], and

(5) how to choose the "most plausible" solution.

References

[Baker *et al.*, 1998] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 434–441, 1998.

[Bergen *et al.*, 1992] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the European Conference on Computer Vision*, pages 237–252, 1992.

[Darrell and Pentland, 1995] T. Darrell and A.P. Pentland. Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):474–487, 1995.

[Faugeras *et al.*, 1987] O.D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from point and line matches. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 25–33, June 1987.

[Hartley, 1994] R. Hartley. Projective reconstruction from line correspondences. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 903–907, 1994.

[Hsu *et al.*, 1994] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *Proceedings of the International Conference on Pattern Recognition*, pages 743–746, 1994.

[Kanade, 1980] T Kanade. A theory of origami world. *Artificial Intelligence*, 13:279–311, 1980.

[Lucas and Kanade, 1981] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of IJCAI*, pages 674–679, 1981.

[Sawhney and Ayer, 1996] H. Sawhney and S. Ayer. Compact representations of videos through dominant and multiple motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.

[Taylor and Kriegman, 1995] C.J. Taylor and D.J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, 1995.

[Wang and Adelson, 1993] J. Wang and E.H. Adelson. Layered representation for motion analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–366, 1993.

[Weiss and Adelson, 1996] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 321–326, 1996.

[Weng *et al.*, 1993] J. Weng, N. Ahuja, and T. Huang. Optimal motion and structure estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):864–884, 1993.

A Checking Depth Ordering

At several points in Section 4.5.2 we assumed that we could check the depth ordering of planes and lines along rays in space defined by pixels in the input images. Suppose that we are working with the image I^p captured with a camera with projection matrix $\mathbf{P}^p$. The ray through the pixel $(u, v)^{\mathrm{T}}$ in image I^p is defined by:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \equiv (\mathbf{P}^p)^* \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + \lambda \mathbf{p}^p \tag{13}$$

where $(\mathbf{P}^p)^* = (\mathbf{P}^p)^{\mathrm{T}}[\mathbf{P}^p(\mathbf{P}^p)^{\mathrm{T}}]^{-1}$ is the pseudo-inverse of the camera matrix $\mathbf{P}^p$, $\mathbf{p}^p$ any vector in the null space of $\mathbf{P}^p$ ($\mathbf{P}^p\mathbf{p}^p = \mathbf{0}$), λ is an unknown scalar, and $\equiv$ denotes equality up to scale. In Section 4.5.2 $(u, v)^{\mathrm{T}}$ is set to be one of the end points of one of the 2D lines $l_i^p = (\mathbf{f}_i^p, \mathbf{s}_i^p)$.

As λ varies in Equation (13), the point $(x, y, z, 1)^{\mathrm{T}}$ traces out a line in the 3D projective space. In order to compute depth ordering we need to know whether λ increases or decreases into the scene. This can be determined by: (1) Estimate the value of λ that corresponds to the image plane by setting the bottom (4th) row of the right hand side of Equation (13) to zero and solving. Denote the result of this λ_0. (2) Intersect any ray with any of the planes to estimate another value of λ. See below for how to do this. Denote the result of this λ_1. If $\lambda_1 > \lambda_0$ then λ increases into the scene.

To complete the description of how to compute the depth ordering along the ray defined by the pixel $(u, v)^{\mathrm{T}}$ in image I^p, all we need to do is describe how the value of λ in Equation (13) can be computed for planes (and lines.) The value of λ that corresponds to the intersection of the ray through $(u, v)^{\mathrm{T}}$ with the plane $\boldsymbol{\pi}_j = (\mathbf{n}_j, d_j)^{\mathrm{T}}$ is the solution of:

$$\left[(\mathbf{P}^p)^* \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + \lambda \mathbf{p}^p \right]^{\mathrm{T}} \boldsymbol{\pi}_j \; = \; 0. \tag{14}$$

The value of λ that corresponds to the intersection of the ray through $(u, v)^{\mathrm{T}}$ with the line $L_j = (\mathbf{F}_j, \mathbf{S}_j)$ (or the plane $\boldsymbol{\pi}_j = L_j$ defined by one line) is the solution of:

$$(\mathbf{P}^p)^* \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + \lambda \mathbf{p}^p \; = \; \gamma \mathbf{F}_j + \mu \mathbf{S}_j \tag{15}$$

for λ (throwing away the results for γ, and μ). Assuming that the pixel $(u, v)^{\mathrm{T}}$ lies on the projection of the 3D line L_j into the image I^p (which is always the case in Section 4.5.2) this over-constrained system of equations (4 equations and 3 unknowns) must have an exact solution.