Computing perspective projections

Project for CS507 - Computational Geometry
(Prof. Godfried T. Toussaint)
by
Jean-Philippe Tardif

Contents

1 Introduction

Computing a perspective projection of a 3-D model is getting its 2-D image on a particular plane by projecting each vertex of the model on that plane in the direction of the projection center. The most obvious example of such a projection is taking a picture of our surrounding world with a camera. It is almost trivial that such a projection removes a lot of information about the particular object. Naturally, inference over an image is sometimes possible, but there is always ambiguity even though a lot of information about the model is previously known. For instance, can you tell which face of the cube in figure 1 is nearest?

A question arises from that observation: what kind of projection maximizes the information we can get from a single image? Instead of answering that question, I will actually give examples of projection where a lot of information is lost. Consider the square in figure 2. Of course, it is impossible to tell if it is a square in space or a cube. If it is the latter, then we call this projection degenerate.

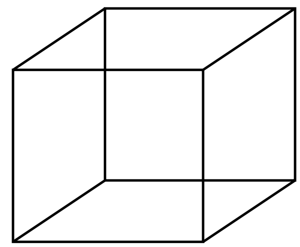

Figure 1: A wire-frame cube projected on the image plane. What is the front and the back?.

Figure 2: Is this the projection of a cube or a square?

A degenerate projection is a very generic term that defines certain projection that have a configuration of points or lines we would like to avoid. In this project, we face the problem of projection with distinct x-coordinates and non-parallel line in the projection. Gomez et al [1] have presented algorithms to compute projections without those degeneracies. They achieve $O(n\ log\ n)$. At the end of this tutorial, it should be obvious that avoiding degeneracies (especially the one treated here) put us closer to "computing projections that maximize the information", which is still an opened problem.

In this tutorial, we will use a very different approach to [1]. We will show how the algorithms work in the case of a pin-hole camera, the most popular camera model in computer vision [2]. Also, this formulation is closer to the implementation of the algorithm.

2 Definitions

In this section, we introduce basic concepts about projection from a camera.

Linear transformation: a formal definition can be found at MathWorld. Here, we will consider a linear transformation as a mapping from two spaces where lines become lines or points.

Pin-hole camera: This a model that maps points and lines in 3D to a 2D plane called the projection plane. Each one is projected in the direction of the projection center. From this definition, we observe that a line become a point only if and only if two points on the line and the projection center are collinear. Consider the vertices, P_i, $i \in [0, n-1]$, of a polyhedron. Given a camera C with projection center $c = (c_x, c_y, c_z)$, we define the projected point p_i on the camera plane as:

$$p_i = JP_i$$

where

$$J = K\ R\ [\ I \mid c\ ]_{3\times 4}$$

This model is illustrated in figure 3. Even though it is very general, I will only present what is necessary for the algorithms presented next. The matrix K contains the internal parameters of the camera. That is where we put the focal length f, which is the distance from the projection plane to the projection center. In the next algorithm, we will want the projection center $c = (0, 0, c_z)$ to get further away from an unchanged projection plane. In the camera model, this is equivalent as decreasing c_z and increasing f by the same value. Also, in the rest of the explanation, we will always make the assumption that the projection center is located on the negative Z-axis. This is because we want to ensure that it stays behind the projection plane located on $Z = 1$. Nevertheless, the algorithm will work for any camera configuration (perspective ones) because we can always change the coordinate system to get this configuration. This is related to R and c, the rotation and translation matrix of the model (figure 4).

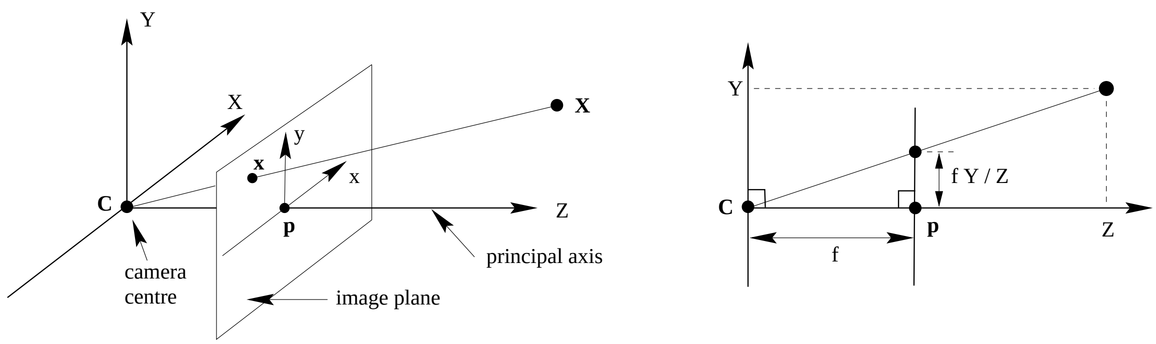

Figure 3: Pin-hole camera model [2].

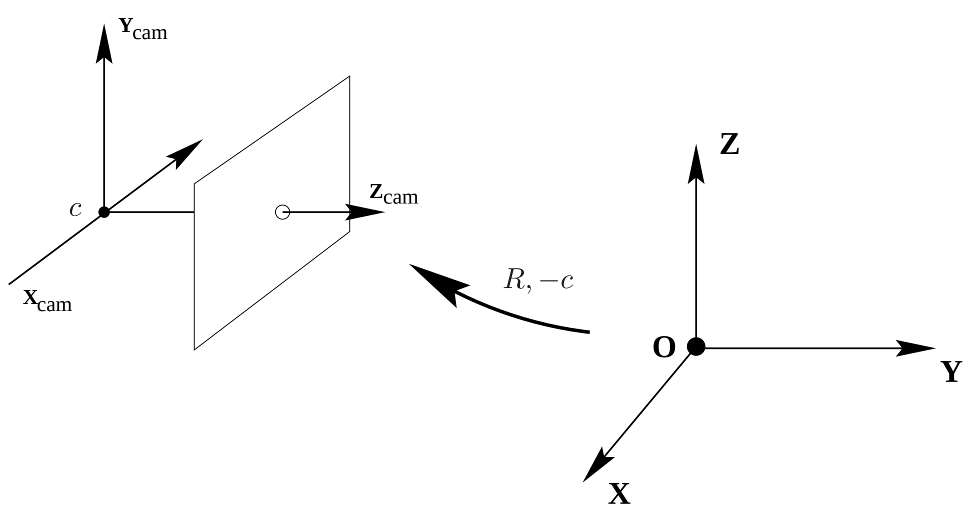

Figure 4: Pin-hole camera model: changing the coordinates system so the projection plane is at $Z = 1$ [2]. In this figure, the camera center is $(0, 0, 0)$, because $f = 1$.

Before beginning with the actual problem we are facing in this project, I will illustrate the effect of increasing the focal length f and decreasing c_z by the same value. What is the effect of this transformation? First, it keeps the projection plan at $Z = 1$. Consider left part of figure 2, this is the side view of the field of view of a camera when $f = c_z = 2$. In the right part of the figure, you get the effect of changing the parameters. First, there is a zoom effect. This is because the field of view decreases. Also, the perspective effect is reduced more and more. This is because when $f \to \infty$ and $c_z \to \infty$, the perspective projection becomes an orthographic projection. In the latter, all relative angle are preserve.

How can this be useful? Essentially, the overall aspect of the object changes. Points in the image move and line relative angles change too. Also, a very important result in projective geometry states that changing a linear transformation in a continuous manner induces a continuous transformation of the object in the new space. Thus, when changing c_z and f, the points and the lines in the images move continuously.

3 Projection with distinct x-coordinates

Definition: A camera C, such that all p_i have distinct x-coordinates in the image.

We can check for this condition with this simple algorithm (see also figure 6):

```
For each point p_i
    Project on X-axis

Sort the list           O(n log n)

Check for duplicates
```

3.1 Existence of such a projection

Let's consider each pair of vertices P_i, P_j, $j \neq i$ of the polyhedron. There are exactly n^2 such pairs. The projection center and each of those pairs of points i and j (assuming no-three-point co-linearities) yield n^2 planes in space intersecting the projection plane in a certain line l_{ij}. A forbidden projection J is one such that at least one line l_{ij} is parallel to the Y-axis of the image plane. Rotating the projection plane (changing R of the camera model) around the projection axis, each of those line becomes parallel to the Y-axis one after another. But since there are $O(n^2)$ forbidden positions and that each line have measure zero in space, there exists an infinite number of valid projection.

up to this point, we have considered fixed projections, with a fixed projection center rotating around the projection axis. In [1], proof is presented using intersecting planes. Then we consider a forbidden projection center c such that the projection of the points on the projection plane from c yields at least one non-distinct x-coordinates. We know that each pair of points (no two points with same y-coordinates) can define a plane that intersects the projection plane in a line parallel to the Y-axis of the image. This construction is possible by intersecting the line P_iP_j with the projection plane yielding a point p_{ij}. Using any other point t_{ij} with the same x-coordinate as p_{ij}, you build the plane $P_iP_jt_{ij}$, (see figure 8). Each

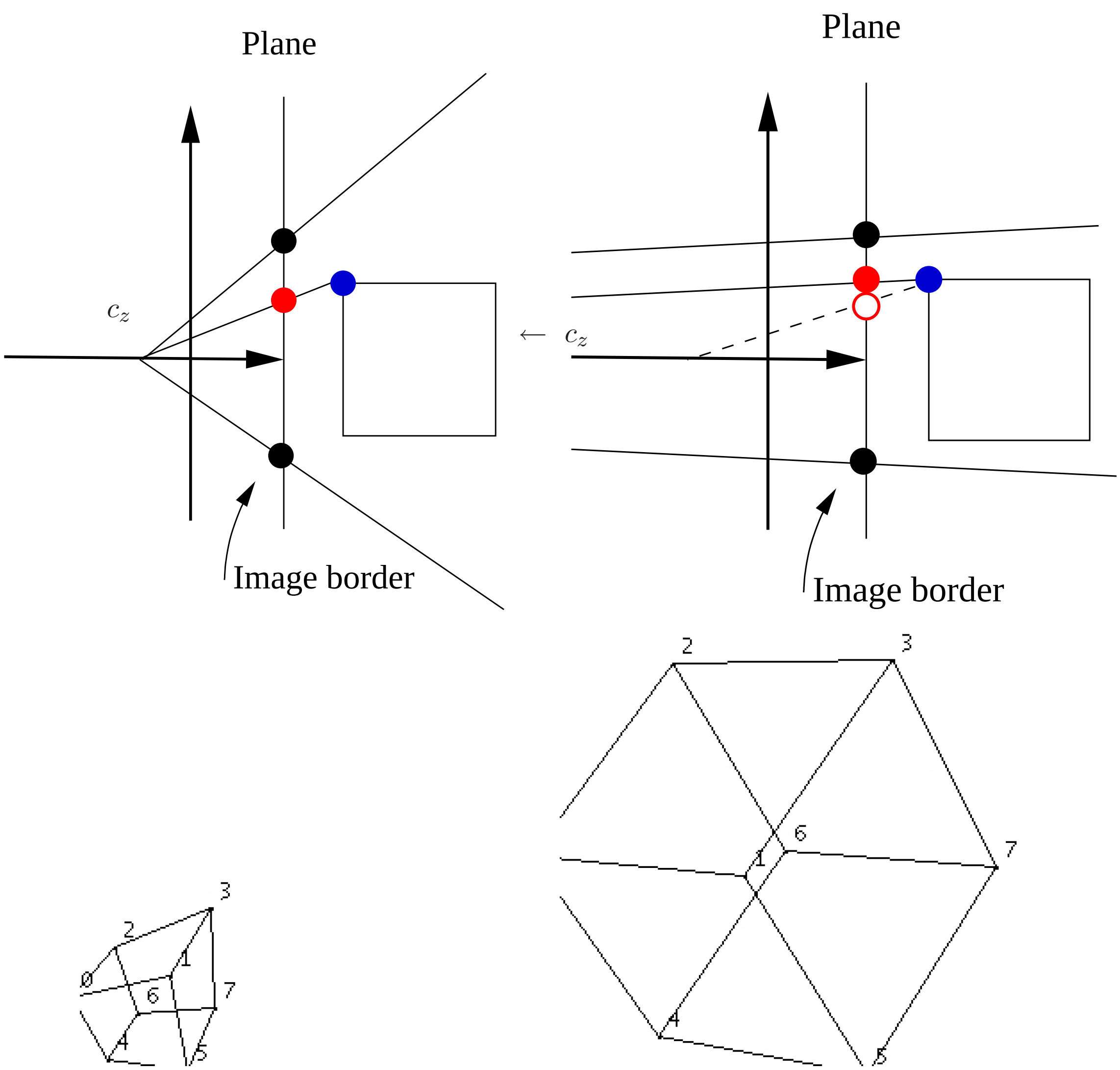

Figure 5: Effect of increasing the focal length. For the purist, this is not exactly consistent with what the camera model really does, but the effect is exactly the same.

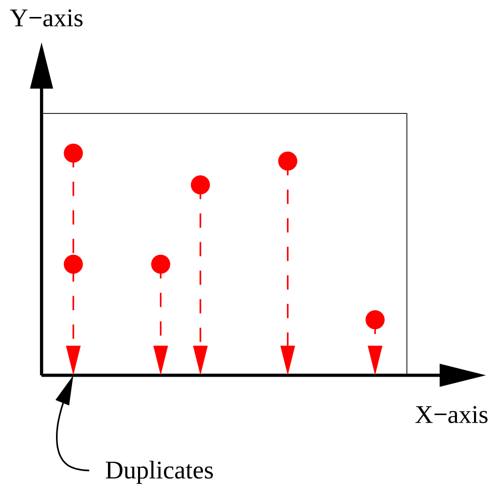

Figure 6: Projection of p_i on the x-axis.

of these planes defines forbidden projection points in space. Nevertheless, each plane has measure zero in space and we conclude once again that there is an infinite number of allowed projection.

Notice that proving this theorem is quite easy and there are many ways to do it. The interesting part is that every proof gives almost all the ingredient for computing a valid projection. In the next section, we will use that last proof.

3.2 Simple algorithm for computing a valid projection center

This is the algorithm presented in [1], modified to be compatible with the notation used here. Let's assume the point P_i and the projection J_0 with projection center c_0. Project each P_i using J_0. Check if all p_i have distinct x-coordinates. This task is achieved in $O(n\ log\ n)$ by sorting p_i relative to their x-coordinates, and then check for duplicates in the list. If all p_i have distinct x-coordinates, we are done. If not, then we know that at least two points have the same x-coordinates. Let's define c_1 the closest forbidden projection center such that $c_{1,z} < c_{0,z}$ (getting further away from the P_i). Projecting from c_1, there are at least two points p_{i_j} and $p_{i_{j+1}}$ whose x-coordinates are equal. Moving the projection center from c_0 to c_1 is a continuous linear transformation in the projection plane. Thus, the points p_{i_j} and $p_{i_{j+1}}$ were already consecutive in the first sorted list of points. Therefore, we only need to consider each pair of consecutive points in the sorted list of points, as possible pair of points defining c_1 (figure 9). To find c_1, we consider each one of those pair of points P_i, and build the plane parallel to the Y-axis passing through those points (figure 8). Then we intersect each one of those plane with the Z-axis. This is done in $O(n)$. c_1 is the closest intersection to c_0, so any point on the **open line** segment $\overline{c_0 c_1}$ is allowed. Overall, this is a $O(n\ log\ n)$ algorithm. This is better than considering every pair of point, which is $O(n^2)$.

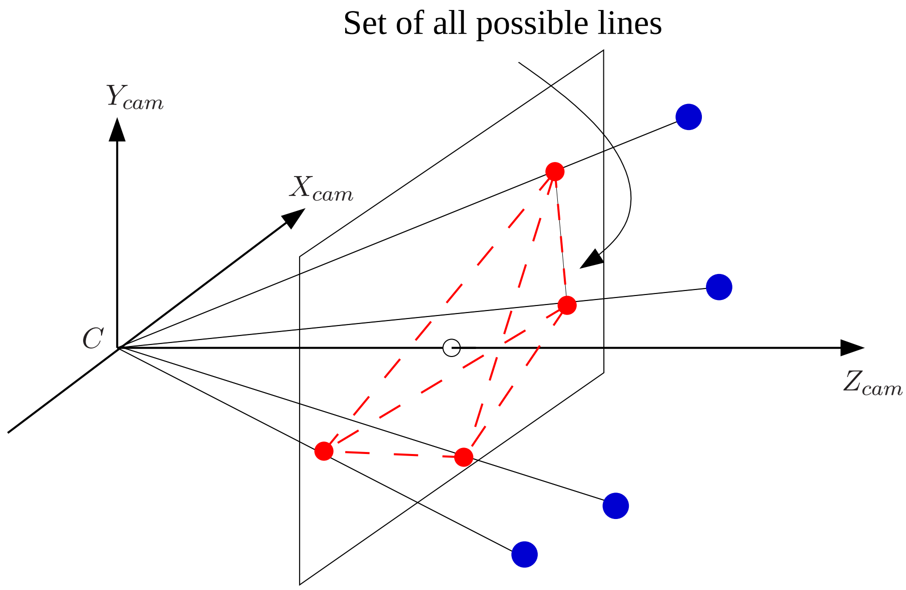

Figure 7: Construction of all possible lines from the projection of all the points. Rotating the camera around its Z-axis will yield one line at a time parallel to its Y-axis in the projection.

```
Current projection center c_0

if no two points with the same x-coordinate
    return

For all conscutive pairs {P_i, P_j}              O(n log n)
    Compute plane (see figure 8)
    Intersect with projection axis (Z-axis)

c_1: Closest to c_0
Open line segment c_0c_1 is valid
```

4 Non-parallel perspective projection

Definition: A camera such that the image does not contain any parallel lines.

More precisely, considering every line L_i built from two points $A_i = (x_{A_i}, y_{A_i}, z_{A_i})$ and $B_i = (x_{B_i}, y_{B_i}, z_{B_i})$, we would like no two projected line l_i parallel to l_j ($j \neq i$).

4.1 No two line parallel to the projection plane and parallel to each other

If such event occurs, then it is impossible to define a valid projection because both line will be parallel for any projection center. So the first step of the algorithm would be to detect if it is the case. This can be done in $O(n\ log\ n)$ with this procedure:

For every line $L_i = A_i, B_i$ assume that the Rt transformation have been applied. This

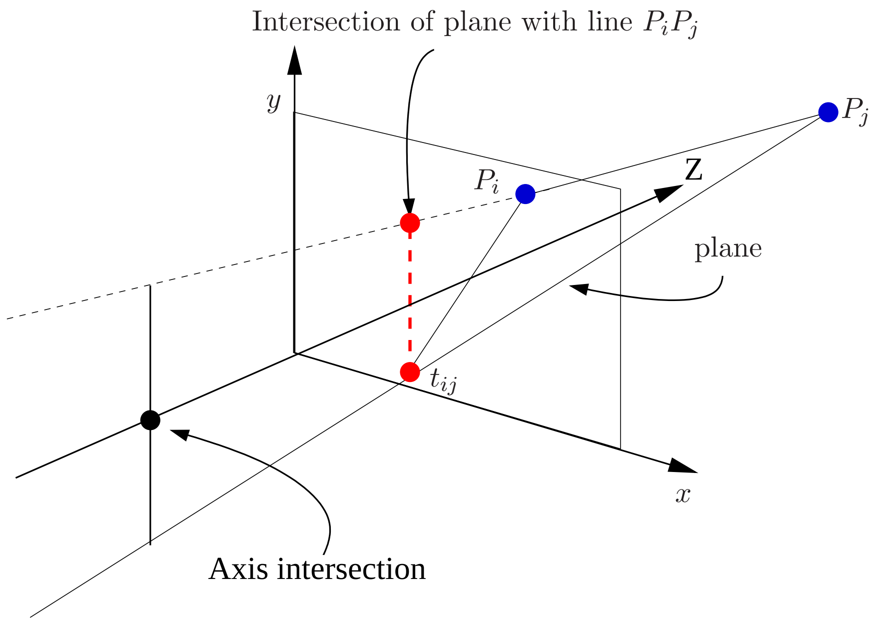

Figure 8: Constructing a plane in space of all forbidden projection center for two points

simplifies the rest of the algorithm because the projection plane is now at $Z = 1$. Check for points parallel to the projection plane with $z_{A_i} = z_{B_i}$. Each line in that set is intersected with the unit half-circle with positive x-coordinates and the intersection points are kept (figure 10:

$$\gamma_i = \frac{(\alpha_i, \beta_i)}{\sqrt{\alpha_i^2 + \beta_i^2}}$$

where

$$\alpha_i = |x_{A_i} - x_{B_i}|$$

and

$$\beta_i = \begin{cases} y_{B_i} - y_{A_i} & \text{if } \ x_{A_i} < x_{B_i} \\ |y_{B_i} - y_{A_i}| & \text{if } \ x_{A_i} = x_{B_i} \\ y_{A_i} - y_{A_i} & \text{otherwise} \end{cases} \tag{1}$$

Two lines l_i, l_j are parallel if $\gamma_i = \gamma_j$. Since the unit half-circle with positive x-coordinates is monotonic in Y, then we can sort w.r.t Y values of the intersections and check for duplicates. This can be achieved $O(n \ log \ n)$.

4.2 No two line parallel in the projection

To check this condition, we use an algorithm very similar to the one described in the previous section. We can define each line of the projection in the same way as previously, except that each line is defined with 2-D points. The only difference is that each line needs to be

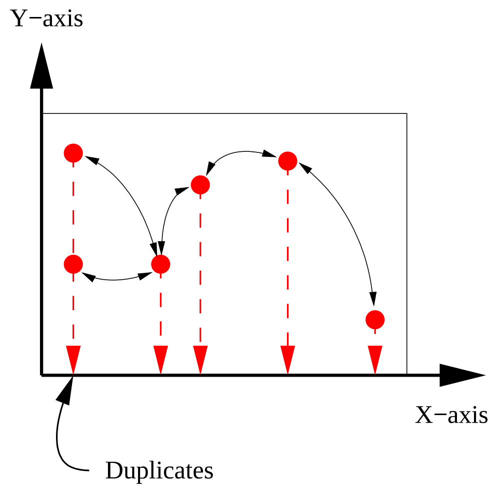

Figure 9: We only need to consider consecutive pairs in the sorted list of points
.

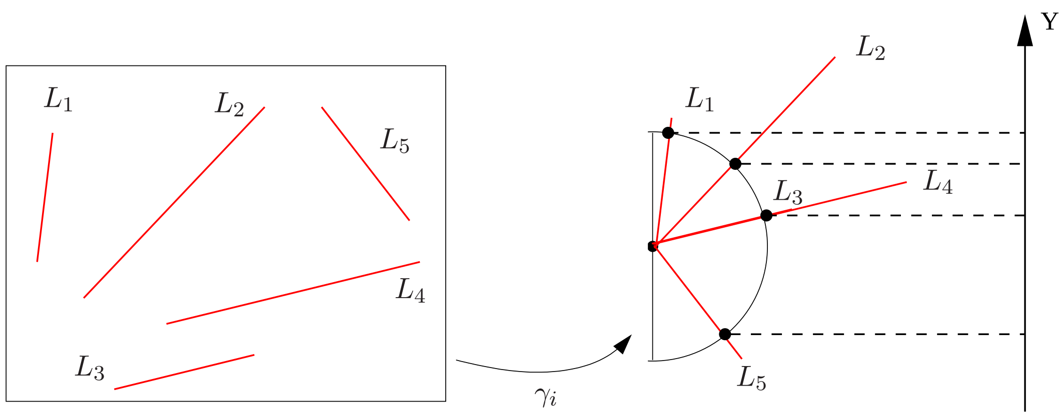

Figure 10: Construction of γ_i values and sorting.

considered by computing the Il value. Again by sorting, we can check if two lines are parallel in the image plane in $O(n\ log\ n)$.

Here is a summary of the algorithm:

for each line l_i intersect with unit half-circle with positive x-coordinates $\rightarrow \gamma_i$	
Sort intersections w.r.t Y-axis	$O(n\ log\ n)$
Check for duplicates	

4.3 Existence of such a projection

Every pair of segments defines forbidden projection centers. This happens when the slope of the two projected Line l_i and l_j are equal:

$$m_{l_i} = m_{l_j}$$

Like before, let's apply the transformation RT to each point in space so that the projection plane is $Z = 1$ and the projection center c is located somewhere on the Z-axis with $c_z <= 0$ and $c_x = c_y = 0$. In this particular configuration, the projection of each point on the projection plane has the form:

$$(\frac{x}{c_z + z}, \frac{y}{c_z + z})$$

where (x, y, z) is the point in space. Notice that the focal length is not in the equation. This is simply because the distance to the plane is not important, only the relative distance of the projection center and the object changes the parallelism of the lines. We can use this relation for the points defining the two lines in the slopes inequalities. Rearranging the equation yields this relation:

$$(\frac{x_{B_i}}{cz + z_{B_i}} - \frac{x_{A_i}}{c_z + z_{A_i}})(\frac{y_{B_j}}{c_z + z_{B_j}} - \frac{y_{A_j}}{c_z + z_{A_j}}) = (\frac{x_{B_j}}{c_z + z_{B_j}} - \frac{x_{A_j}}{c_z + z_{A_j}})(\frac{y_{B_i}}{c_z + z_{B_i}} - \frac{y_{A_i}}{c_z + z_{A_i}})$$

which is of the form (thanks to Mathematica) :

$$Ac_z^2 + Bc_z + C = 0$$

where,

$$A = -x_{A_j}y_{A_i} + x_{B_j}y_{A_i} + x_{A_i}y_{A_j} - x_{B_i}y_{A_j} + x_{A_j}y_{B_i} - x_{B_j}y_{B_i} - x_{A_i}y_{B_j} + x_{B_i}y_{B_j}$$

$$\begin{aligned} B = {} & x_{B_i}y_{A_j}z_{A_i} - x_{A_j}y_{B_i}z_{A_i} + x_{B_j}y_{B_i}z_{A_i} - x_{B_i}y_{B_j}z_{A_i} - x_{B_j}y_{A_i}z_{A_j} + \\ & x_{B_j}y_{B_i}z_{A_j} + x_{A_i}y_{B_j}z_{A_j} - x_{B_i}y_{B_j}z_{A_j} + x_{A_j}y_{A_i}z_{B_i} - x_{B_j}y_{A_i}z_{B_i} - \\ & x_{A_i}y_{A_j}z_{B_i} + x_{A_i}y_{B_j}z_{B_i} + x_{A_j}y_{A_i}z_{B_j} - x_{A_i}y_{A_j}z_{B_j} + x_{B_i}y_{A_j}z_{B_j} - x_{A_j}y_{B_i}z_{B_j} \end{aligned}$$

$$\begin{aligned} C = {} & -x_{B_j}y_{B_i}z_{A_i}z_{A_j} + x_{B_i}y_{B_j}z_{A_i}z_{A_j} + x_{B_j}y_{A_i}z_{A_j}z_{B_i} - x_{A_i}y_{B_j}z_{A_j}z_{B_i} - \\ & x_{B_i}y_{A_j}z_{A_i}z_{B_j} + x_{A_j}y_{B_i}z_{A_i}z_{B_j} - x_{A_j}y_{A_i}z_{B_i}z_{B_j} + x_{A_i}y_{A_j}z_{B_i}z_{B_j} \end{aligned}$$

which is a polynomial of degree two in c_z. Thus, solving this inequality yield at most two forbidden projection center on the Z-axis. Therefore, the Z-axis is not a line of forbidden projection center (more generally, the projection axis) and that conclude the proof. A very similar proof is presented in [1].

4.4 Simple algorithm for computing a valid projection center

Again, we present a slightly modified algorithm from the one presented in [1]. Let's assume a set of lines L_i and the projection J_0 with projection center c_0. Again, let's also assume that c_0 is located at the origin and all points P_i are in front of the projection plane at $Z = 1$ and have positive coordinates.

Project each L_i using J_0. With the algorithm presented before, check if no two line l_i are parallel. If so, c_0 is a valid projection center. Otherwise, let's define c_1 the closest forbidden projection center such that $c_{1,z} < c_{0,z}$ (getting further away from the L_i). Then there are at least two points L_{i_j} and $L_{i_{j+1}}$ whose projections are parallel. Moving the projection center from c_0 to c_1 is a continuous transformation, showing that the line L_{i_j} and $L_{i_{j+1}}$ were already consecutive in their γ_i value. Therefore, we only need to consider each pair of consecutive line in the sorted list of γ_i as possible pairs of points defining c_1. To find c_1, we consider each one of those pair of line and solve the polynomial equation shown before. This is done in $O(n)$. Then, we choose c_1 as the closest intersection to c_0. Any point on the open line segment $\overline{c_0 c_1}$ is allowed.

Here is a summary:

Project each L_i from c_0

if no two line l_i are parallel
    return

build sorted list of line ($O\ log\ n$)

for each consecutive pair
    Solve polynomial equation

c_1: closest from c_0

Any point on the open line $\overline{c_0 c_1}$ is allowed

5 Applet

A java applet illustrates how these two algorithms work. The important part of the interface are the canvases *Projection* and *Observation*. The first one is the actual view of the camera for which we want to detect degeneracies and compute valid views. The other one allows the user to see what is happening from another point of view. You can see the projection plane, the camera and the object at the same time. When an algorithm is applied, graphical information is also added to this view.

5.1 Control

The two 3-D canvas have very different control functionality. Let's summarize how they both work:

Projection: With the mouse, you can rotate the object around the X-axis and Y-axis of the world. Using the third button of the mouse, you can also rotate it around the Z-axis. By pressing any key, you can change the focal length of the camera. You will notice that the shape of the object changes. This is completely normal. As the focal length increases, the perspective projection approximates more and more an orthogonal projection. increasing the size of the focal length also creates a zooming effect [2]. You will notice that moving the object or changing the focal length of the camera has an immediate effect on the *Observation*.

Observation: In this view, you control a virtual camera located around the object and the projection plane. With the mouse, you can rotate this camera around the scene or tilt it using button3. By pressing any key, you can translate the camera in x and y.

With the buttons on top of the window, you can verify if there really is a degeneracy. If so, you can use the other button to compute a valid projection. When it has completed, you can check again that the projection is valid for this particular degeneracy.

6 Online version of this tutorial

A html version of this page, with the applet can be found at:
http://www.iro.umontreal.ca/ tardifj/CS504/

References

[1] F. Gomez, S. Ramaswami, and G. Toussaint. On removing non-degeneracy assumptions in computational geometry. In *3rd Italian Conference on Algorithms and Complexity*, 1997.

[2] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge, 2000.