Camera Models

Fall 2017

Daniel G. Aliaga
Department of Computer Science
Purdue University
Human Eye
Thin Lens System
Thin Lens Equation

\[
\frac{1}{d_0} + \frac{1}{d_i} = \frac{1}{f}
\]
“Pinhole” Camera Image
Digression:
Non-Pinhole Camera Models...

• Why restrict the camera to a pinhole camera model?
Multiperspctive Imaging

Hand-crafted

semi-automated...

to produce this...

[Roman-Vis04]
Multiperspectvie Imaging

[Seitz-CGA03]
Multiple COP Images

[Rademacher-SIG98]
Multiple COP Images

**Figure 5** Castle model. The red curve is the path the camera was swept on, and the arrows indicate the direction of motion. The blue triangles are the thin frusta of each camera. Every $64^{th}$ camera is shown.

**Figure 6** The resulting $1000 \times 500$ MCOP image. The first fourth of the image, on the left side, is from the camera sweeping over the roof. Note how the courtyard was sampled more finely, for added resolution.

**Figure 7** The projection surface (image plane) of the camera curve.

**Figure 8** Three views of the castle, reconstructed solely from the single MCOP image above. This dataset captures the complete exterior of the castle.

[Rademacher-SIG98]
Multiperspective Imaging for Cel Animation

Figure 1 A multiperspective panorama from Disney’s 1940 film *Pinocchio*. (Used with permission.)
Multiperspective Imaging for Cel Animation

[Wood-SIG97]
Multiperspective Imaging for Cel Animation

[Wood-SIG97]
Multiperspective Imaging for Cel Animation

[Wood-SIG97]
Occlusion-Resistant Cameras

[Aliaga-CGA07]
Occlusion Cameras

<table>
<thead>
<tr>
<th>Simulator Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1 Depth image (DI).</td>
</tr>
<tr>
<td>Figure 2 Images rendered from DI and OCRI, viewpoint 4&quot; left of reference viewpoint.</td>
</tr>
<tr>
<td>Figure 3 OCRI.</td>
</tr>
<tr>
<td>Figure 4 Images rendered from DI and OCRI. Wireframe shows spherical display volume.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Photographs of 3D Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5 3D images rendered from DI (left), OCRI (middle), and original geometric model (right), all photographed from reference view.</td>
</tr>
<tr>
<td>Figure 6 DI and OCRI 3D images from viewpoint translated 4&quot; left.</td>
</tr>
<tr>
<td>Figure 7 DI and OCRI 3D images from side view.</td>
</tr>
</tbody>
</table>

[Popescu-JDT06]
Graph Cameras

Portal-based graph camera image (top left and fragment right) and PPC image for comparison (bottom left)

Occluder-based graph camera image (top left), PPC image for comparison (bottom left), and ray visualizations (right)

Enhanced street-level navigation

[Popescu-SIGA09]
Let’s get back on track: Pinhole Camera Model...
What is perspective projection?

The perspective projection formula is:

\[ x = \frac{f X}{\bar{Z}} \]

\[ y = \frac{f Y}{\bar{Z}} \]
“Computer Graphics” Pinhole Camera Model

eye

image plane

object

f

d
What is perspective projection?

- **Eye/viewpoint**
- **Optical axis**
- **Image plane**

The equations for perspective projection are:

\[
\frac{y}{f} = \frac{Y}{Z} \quad \text{and} \quad \frac{x}{f} = \frac{X}{Z}
\]

This results in:

\[
y = f \frac{Y}{Z} \quad \text{and} \quad x = f \frac{X}{Z}
\]
Perspective Camera Parameters

• Intrinsic/Internal
  – Focal length \( f \)
  – Principal point (center) \( p_x, p_y \)
  – Pixel size \( s_x, s_y \)
  – (Distortion coefficients) \( k_1, \ldots \)

• Extrinsic/External
  – Rotation \( \phi, \varphi, \psi \)
  – Translation \( t_x, t_y, t_z \)
Perspective Camera Parameters

• Intrinsic/Internal
  – Focal length \( f \)
  – Principal point (center) (=middle of image)
  – Pixel size (=1, irrelevant)
  – (Distortion coefficients) (=0, assuming no bugs 😊)

• Extrinsic/External
  – Rotation \( \phi, \varphi, \psi \)
  – Translation \( t_x, t_y, t_z \)
Focal Length

Assume $c = 0$:

$$
\begin{pmatrix}
  x \\
  y
\end{pmatrix} = \begin{pmatrix}
  fX/Z \\
  fY/Z
\end{pmatrix}
$$

$$
\begin{pmatrix}
  fX \\
  fY \\
  Z
\end{pmatrix} = \begin{bmatrix}
  X \\
  Y \\
  Z
\end{bmatrix}
$$
Focal Length

Assume $c \neq 0$:

\[
\begin{pmatrix}
  x \\
  y
\end{pmatrix} = 
\begin{pmatrix}
  fX'/Z' \\
  fY'/Z'
\end{pmatrix}
\]

\[
\begin{pmatrix}
  fX' \\
  fY' \\
  Z'
\end{pmatrix} =
\begin{bmatrix}
  \frac{x}{y} & fX' & fY' \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
  X \\
  Y \\
  Z
\end{pmatrix}
\]
Focal Length

In general, where $M$ is some 4x4 matrix:

$$
\begin{pmatrix}
  x \\
y
\end{pmatrix} = \begin{pmatrix} fX' / Z' \\
fY' / Z' \end{pmatrix}
$$

$$
\begin{pmatrix}
fX' \\
fY' \\
Z'
\end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix} M_{4x4} \begin{pmatrix} X \\
Y \\
Z \\
1 \end{pmatrix}
$$
World to Camera Matrix $M$

\[ \tilde{x}_c = R(\tilde{X} - C) \]
\[ \tilde{x}_c = R\tilde{X} - RC \]

\[ \tilde{x}_c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \]

\[ R = R_x R_y R_z \]
3x3 rotation matrices

\[ t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}^T \]
translation vector
Perspective Projection Process

• Thus, given \( \tilde{X} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \)

...the perspective projection is

\[
\tilde{x}_p = PM \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \tilde{x}_{px} / \tilde{x}_{pz} \\ \tilde{x}_{py} / \tilde{x}_{pz} \end{bmatrix}
\]
Projection Matrices

- Basic Perspective Projection:
  \[
  \begin{bmatrix}
  f & 0 & 0 \\
  0 & f & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]

- Basic Orthographic:
  \[
  \begin{bmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
Projection Matrices

- Passthrough Z Perspective Projection:

\[
\begin{bmatrix}
    n & 0 & 0 & 0 \\
    0 & n & 0 & 0 \\
    0 & 0 & n+f & -nf \\
    0 & 0 & 1 & 0
\end{bmatrix}
\]

If \(Z = n\), then \(z = \left(n + f - \frac{nf}{n}\right) = n\)

If \(Z = f\), then \(z = \left(n + f - \frac{nf}{f}\right) = f\)
• To map [-1,+1] to viewing frustum (l,r,b,t,n,f):

\[
\begin{bmatrix}
    2n/(r-l) & 0 & 0 & -(r+l)/(r-l) \\
    0 & 2n/(t-b) & 0 & -(t+b)/(t-b) \\
    0 & 0 & 2n/(n-f) & -(n+f)/(n+f) \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]
OpenGL Equivalent

...  
glMatrixMode(GL_PROJECTION);  
...  
 gluPerspective(60, 1.0, 0.1, 1000.0);  
...  
 glMatrixMode(GL_MODELVIEW);  
...  
 glTranslatef(tx, ty, tz);  
glRotatef(rx, 1, 0, 0);  
glRotatef(ry, 0, 1, 0);  
glRotatef(rz, 0, 0, 1);  

/* or glLoadMatrixf(mat); */  
...