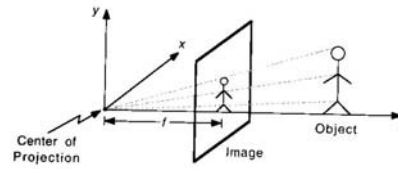


## PERSPECTIVE CAMERA MODEL

Formalizing the pinhole camera model:

- Properties of the perspective projection
- Linear formulation
- Calibration
- Distortions

## PERSPECTIVE PROJECTION

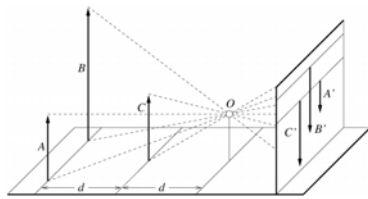


$$u = f \frac{X}{Z}$$

$$v = f \frac{Y}{Z}$$

Pinhole geometry without image reversal

## PROJECTION IS NON LINEAR



$$u = f \frac{X}{Z}$$

$$v = f \frac{Y}{Z}$$

→ Reformulate it as a linear operation.

## HOMOGENEOUS COORDINATES

Homogeneous representation of 2D point:

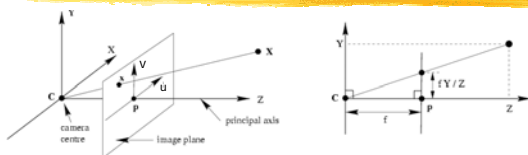
$$\mathbf{x} = (x_1, x_2, x_3) \text{ represents } \left( \frac{x_1}{x_3}, \frac{x_2}{x_3} \right)$$

Homogeneous representation of 3D point:

$$\mathbf{x} = (x_1, x_2, x_3, x_4) \text{ represents } \left( \frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4} \right)$$

→ Projections become linear transformations.

## SIMPLE PROJECTION MATRIX



$$u = f \frac{X}{Z}$$

$$v = f \frac{Y}{Z}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \text{ with } u = \frac{x}{z}, v = \frac{y}{z}$$

$$= \begin{bmatrix} f & 0 & 0 & 1 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

## INTRINSIC AND EXTRINSIC PARAMETERS

1. Camera may not be at the origin, looking down the z-axis  
→ Extrinsic parameters
2. One unit in camera coordinates may not be the same as one unit in world coordinates  
→ Intrinsic parameters

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

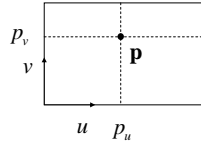
## LINEAR CAMERA MODEL

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{intrinsic parameters} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \text{Transformation} \\ \text{representing} \\ \text{extrinsic parameters} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \mathbf{K} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{Rt} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where  $\mathbf{K}$  is a 3x3 matrix and  $\mathbf{Rt}$  a 4x4 matrix.

## PRINCIPAL POINT



$$u' = u + p_u = fX/Z + p_u$$

$$v' = v + p_v = fY/Z + p_v$$

$$\mathbf{K} = \begin{bmatrix} f & p_u \\ & f & p_v \\ & & & 1 \end{bmatrix}$$

## INHOMOGENEOUS SCALING

$$u' = u + p_u = \alpha_u X/Z + p_u$$

$$v' = v + p_v = \alpha_v Y/Z + p_v$$

$$\mathbf{K} = \begin{bmatrix} \alpha_u & & p_u \\ & \alpha_v & p_v \\ & & & 1 \end{bmatrix}$$

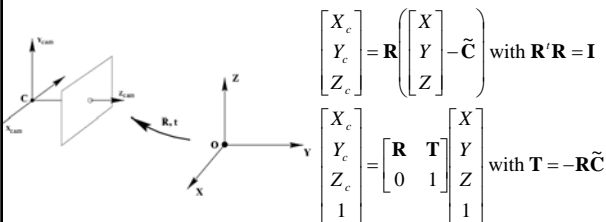
The pixels are not necessarily square.

## AXIS SKEW

$$\mathbf{K} = \begin{bmatrix} \alpha_u & s & p_u \\ & \alpha_v & p_v \\ & & & 1 \end{bmatrix}$$

Encodes the non-orthogonality of the  $u$  and  $v$  directions.

## ROTATION / TRANSLATION



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \tilde{\mathbf{C}} \text{ with } \mathbf{R}^T \mathbf{R} = \mathbf{I}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \text{ with } \mathbf{T} = -\mathbf{R}\tilde{\mathbf{C}}$$

→ Rotations and translations also expressed in terms of matrix multiplications in projective space.

## FULL PROJECTION MATRIX

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_u & s & p_u \\ 0 & \alpha_v & p_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \text{ with } \mathbf{T} = -\mathbf{R}\tilde{\mathbf{C}} \text{ and } \mathbf{R}^T \mathbf{R} = \mathbf{I}$$

$$= \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \mid -\tilde{\mathbf{R}}\tilde{\mathbf{C}}]\mathbf{X} = \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\tilde{\mathbf{C}}]\mathbf{X} = [\mathbf{M} \mid \mathbf{m}]\mathbf{X} = \mathbf{P}\mathbf{X}$$

## CAMERA PARAMETERS

### Internal Parameters:

- Horizontal and vertical scaling (2)
- Principal points (2)
- Skew of the axis (1)

### External Parameters:

- Rotations (3)
- Translations (3)

→ **11 free parameters.**

## CALIBRATION GRID



## LEAST-SQUARES MINIMIZATION

For  $1 \leq i \leq n$

$$P_u(\mathbf{X}_i) = u_i + \varepsilon_{u,i}$$

$$P_v(\mathbf{X}_i) = v_i + \varepsilon_{v,i}$$

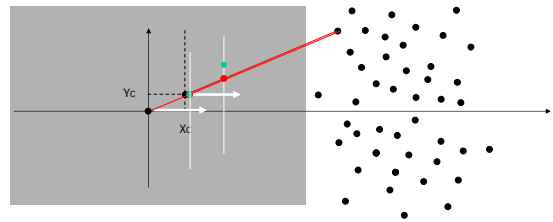
Minimize

$$\sum_{i=1}^n \varepsilon_{u,i}^2 + \varepsilon_{v,i}^2 = \sum_{i=1}^n (P_u(\mathbf{X}_i) - u_i)^2 + (P_v(\mathbf{X}_i) - v_i)^2$$

with respect to the unknown calibration parameters.

## LINE CAMERA

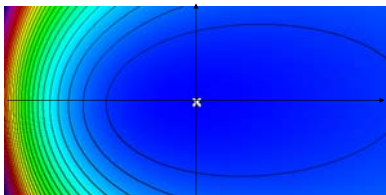
- 1D camera under 2D translation, parameterized by  $(X_c, Y_c)$ , the camera center coordinates.
- 100 points taken at random in  $[400;1000] \times [-500; +500]$
- True camera position at  $(0, 0)$



## ENERGY LANDSCAPE

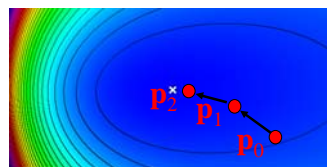
White cross: True camera position;

Black cross: Global minimum of the objective function.



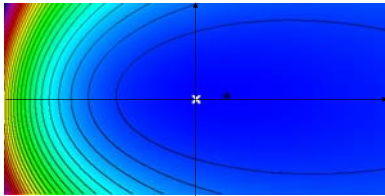
→ The global minimum of the objective function is close to the true camera pose.

## NUMERICAL OPTIMIZATION



## NOISY CORRESPONDENCES

White cross: True camera position;  
Black cross: Global minimum of the objective function.



→ The global minimum of the object function is now further from the true camera pose.

## DECOMPOSITION

Finding the camera center:

$$\mathbf{0} = \mathbf{P}\mathbf{C} = [\mathbf{M} | \mathbf{m}] \begin{bmatrix} \mathbf{C} \\ 1 \end{bmatrix} \Rightarrow \mathbf{M}\tilde{\mathbf{C}} = -\mathbf{m}$$

Finding the camera orientation and internal parameters:

$\mathbf{M} = \mathbf{K}\mathbf{R}$  where  $\mathbf{K}$  is triangular superior and  $\mathbf{R}$  is orthogonal

→  $\mathbf{K}$  and  $\mathbf{R}$  can be computed by RQ decomposition

## OPEN GL



Two scenes rendered with a shading language developed at Stanford.

Cross platform 3D graphics library that takes advantage of specialized graphics hardware.

Polygons are affected by the current color, transformation, drawing mode, etc.

## GAMES



## 4x4 HOMOGENEOUS MATRICES

$$\begin{bmatrix} x \\ y \\ z \\ h \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & 1 \\ a & b & c & d \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Display point if and only if  $h_{\min} \leq h \leq h_{\max}$

→ Hardware acceleration to quickly perform multiplications of 4 vectors and 4x4 matrices

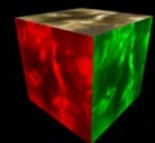
## MODEL VIEW MATRIX

- Transforms the viewpoint and objects within the scene.

- Example:

```
glMatrixMode(GL_MODELVIEW); // set the current matrix
glLoadIdentity();          // load the identity matrix
glTranslatef(10.5, 0, 0);   // translate 10.5 units along x-axis
glRotatef(45, 0, 0, 1);    // rotate 45 degrees CCW around z-axis
DrawCube();                 // cube is defined centered around
                             // origin
```

- Where will this end up?
- Answer:
  - On the x-axis, rotated 45 degrees CCW.



## PROJECTION MATRIX

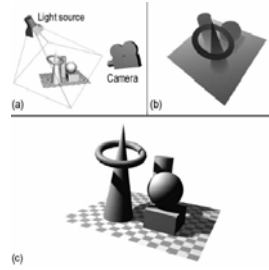
Sets up a perspective projection.

- A few available options:
  - // sets up a user defined viewing frustum
  - `glFrustum (...);`
  - // Calculates viewing frustum for you, given field-of-view in degrees, aspect ratio, and near and far clipping planes.
  - `gluPerspective (fovy, aspect, near, far);`
  - // Creates an orthographic projection. Useful for 2D rendering.
  - `glOrtho (...);`

Example:

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(64, windowWidth, windowHeight, 4, 4096);
```

## Z BUFFER



- Occlusion handling
- Creating shadows
- Voxelization
- .....

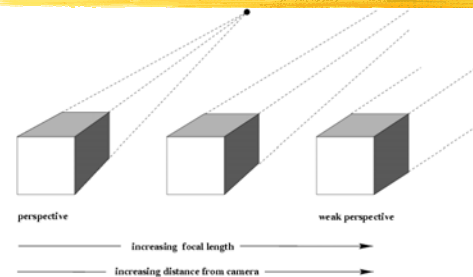
The Magic of the Z-Buffer: A survey, T. Theoaris et al. 2001.

## CAMERA AT INFINITY



As focal length and distance to camera increase, the image remains the same size but perspective effects diminish.

## WEAK PERSPECTIVE



## AFFINE CAMERAS

$$P = \begin{bmatrix} m_{11} & m_{12} & m_{13} & t_1 \\ m_{21} & m_{22} & m_{23} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The projection is linear in Euclidean space

- Parallel world lines project to parallel image lines
- The optical center is a point at infinity

## MOVING BACK THE OPTICAL CENTER

Original projection :

$$P_0 = K[R' - R\tilde{C}] = K \begin{bmatrix} \mathbf{r}'_1 & -\mathbf{r}'_1\tilde{C} \\ \mathbf{r}'_2 & -\mathbf{r}'_2\tilde{C} \\ \mathbf{r}'_3 & -\mathbf{r}'_3\tilde{C} \end{bmatrix}$$

The optical center moves back by  $t\mathbf{r}_3$  :

$$P_t = K \begin{bmatrix} \mathbf{r}'_1 & -\mathbf{r}'_1(\tilde{C} - t\mathbf{r}_3) \\ \mathbf{r}'_2 & -\mathbf{r}'_2(\tilde{C} - t\mathbf{r}_3) \\ \mathbf{r}'_3 & -\mathbf{r}'_3(\tilde{C} - t\mathbf{r}_3) \end{bmatrix} = K \begin{bmatrix} \mathbf{r}'_1 & -\mathbf{r}'_1\tilde{C} \\ \mathbf{r}'_2 & -\mathbf{r}'_2\tilde{C} \\ \mathbf{r}'_3 & d_t \end{bmatrix}$$

## ZOOMING IN

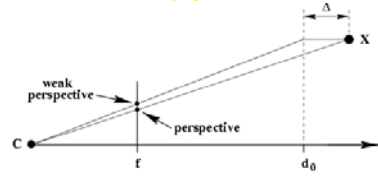
Zooming in to conserve the image size :

$$\mathbf{P}_t = \mathbf{K} \begin{bmatrix} d_t/d_0 & & & \\ & d_t/d_0 & & \\ & & 1 & \\ & & & \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^t & -\mathbf{r}_1^t \tilde{\mathbf{C}} \\ \mathbf{r}_2^t & -\mathbf{r}_2^t \tilde{\mathbf{C}} \\ \mathbf{r}_3^t & d_t \end{bmatrix} = d_t/d_0 \mathbf{K} \begin{bmatrix} \mathbf{r}_1^t & -\mathbf{r}_1^t \tilde{\mathbf{C}} \\ \mathbf{r}_2^t & -\mathbf{r}_2^t \tilde{\mathbf{C}} \\ \mathbf{r}_3^t d_0/d_t & d_0 \end{bmatrix}$$

In the limit :

$$\mathbf{P}_\infty = \lim_{t \rightarrow \infty} \mathbf{P}_t = \mathbf{K} \begin{bmatrix} \mathbf{r}_1^t & -\mathbf{r}_1^t \tilde{\mathbf{C}} \\ \mathbf{r}_2^t & -\mathbf{r}_2^t \tilde{\mathbf{C}} \\ 0 & d_0 \end{bmatrix} = \begin{bmatrix} \mathbf{M} & \mathbf{T} \\ 0 & 1 \end{bmatrix}$$

## QUALITY OF THE APPROXIMATION



Valid approximation if

- The range of depths is small compared to the average depth.
- The field of view is limited

Usually true when using a long focal length

## RADIAL DISTORTION



Short focal length → Large distortion.  
Long focal length → Small distortion.

## UNDISTORTING



$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(r) \begin{pmatrix} x \\ y \end{pmatrix}$$

$$r = \sqrt{x^2 + y^2}$$

$$\hat{x} = x_c + L(r)(x - x_c)$$

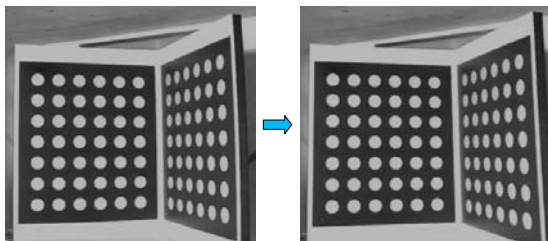
$$\hat{y} = y_c + L(r)(y - y_c)$$

with :

$$r^2 = (x - x_c)^2 + (y - y_c)^2$$

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \dots$$

## MAKING STRAIGHT LINES STRAIGHT



## UNDISTORTING



## PROJECTIVE CAMERAS

---

Once the image is undistorted, the camera projection can be formulated as a projective transform.

→ The pinhole camera model applies.