Advanced Topics Multimedia Video (5LSH0), Module 02

3D Geometry, Camera Model & Projection Matrix

Egor Bondarev, Peter H.N. de With
(e.bondarev@tue.nl)

Credit for some slides to James Hays, Derek Hoiem, Alexei Efros, Steve Seitz, and David Forsyth

Multiview 3D video / Outline

- **Camera geometry**
  - Pinhole camera model
  - Projective geometry
  - Projection matrix,
  - Intrinsic/extrinsic camera parameters
- **Camera calibration**

3D world projected on 2D photo views

3D world projected on 2D photo views

3D world projected on 2D photo views
5

But from another view…
6

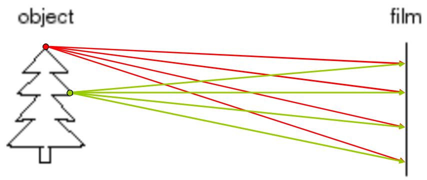
So, this hour we learn
7

Mapping between image and world coordinates

- Pinhole camera model
- Projective geometry
  - Vanishing points and lines
- Projection matrix

How can we create something that captures the scenery?
8

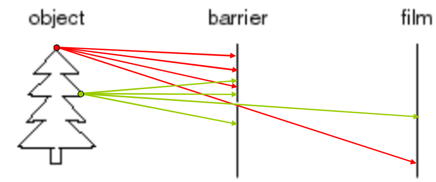
Designing camera



– Idea 1: put a photon's sensor in front of an object
– Do we get a good image?

Slide source: Seitz

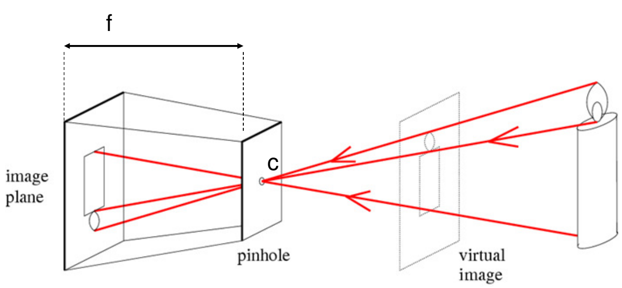
Designing camera - Pinhole camera



– Idea 2: add a barrier with opening to block off most of the rays
  – This reduces blurring
  – The opening known as the **aperture**

Slide source: Seitz

Pinhole camera Model

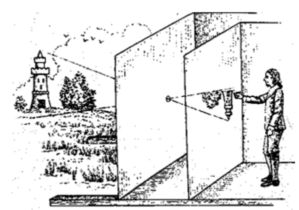


f = focal length
c = center of the camera

Figure from Forsyth

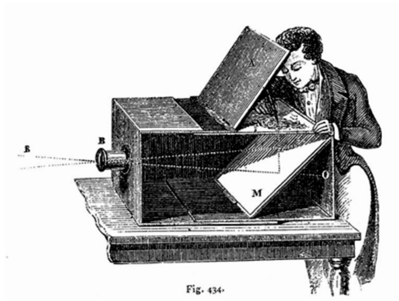
Camera obscura: the pre-camera

* **Known from ancient periods in China and Greece (e.g. Mo-Ti, China, 470BC to 390BC)**



Camera obscura at Lisbon Castle

Camera Obscura used for Tracing



Lens Based Camera Obscura, 1568

First Photograph

Oldest surviving photograph

Joseph Niepce, 1826

Photograph of the first photograph

Stored at UT Austin

Pinhole vs Lens Camera

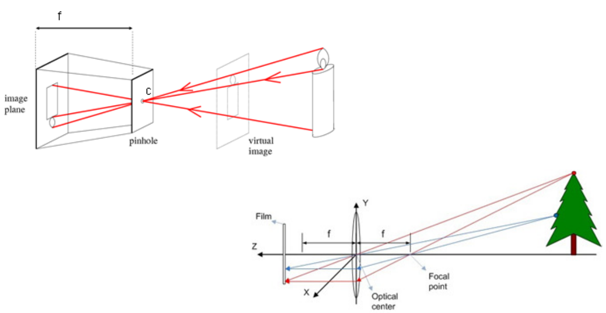
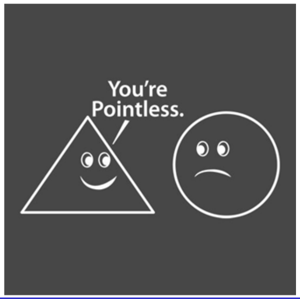


Figure from Forsyth

Multiview 3D video / Outline

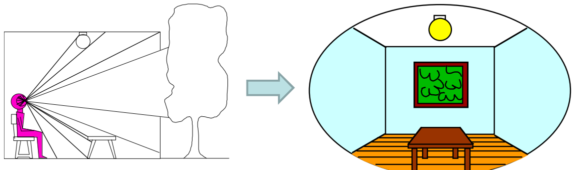
- **Camera geometry**
  - Pinhole camera model
  - Projective geometry
  - Projection matrix,
  - Intrinsic/extrinsic camera parameters
- **Camera calibration**

Projective Geometry



Projection:
Dimensionality Reduction Machine (3D to 2D)

3D world *2D image*



Point of observation

Figures © Stephen E. Palmer

Slide source: Seitz

Projection can be tricky…

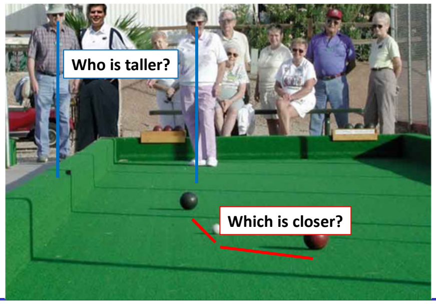
Slide source: Seitz

Projection can be tricky…

Projective Geometry

What is lost?

* Length



Projective Geometry

What is lost?

* Length

Length is not preserved

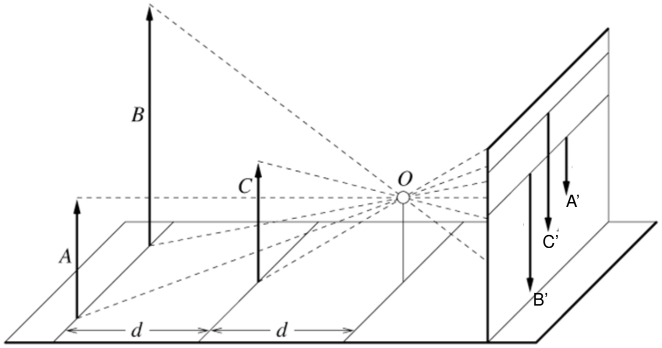
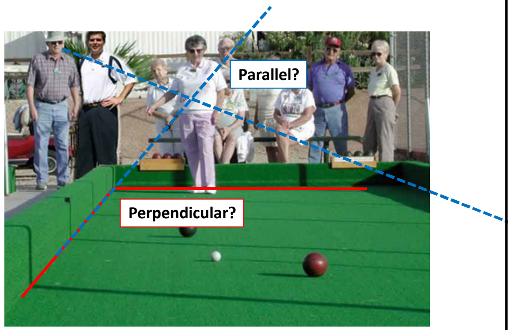


Figure by David Forsyth

Projective Geometry

What is lost?

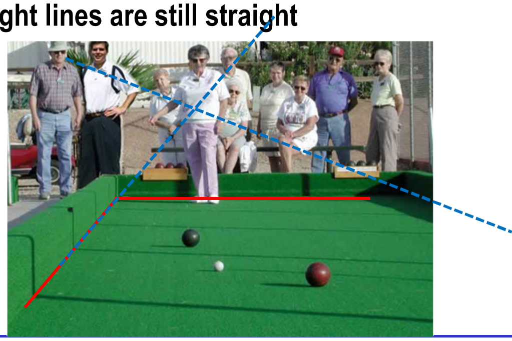
* Length
* Angles



Projective Geometry

What is preserved?

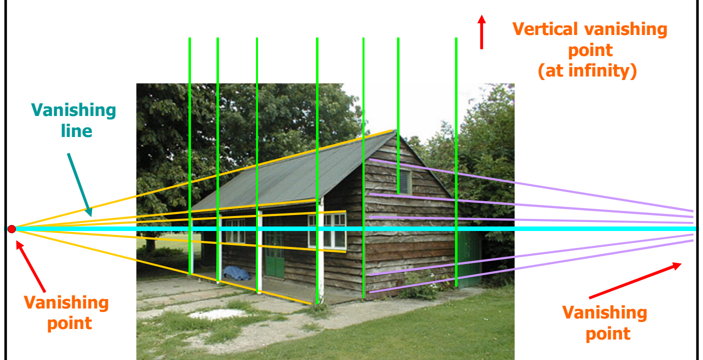
* Straight lines are still straight



Vanishing points and lines

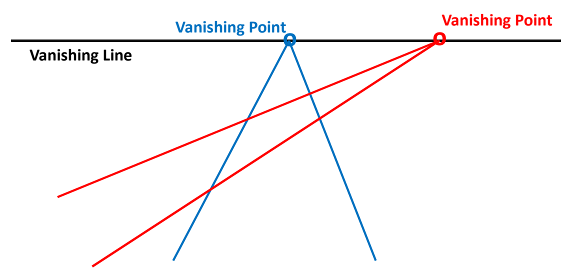
Two parallel rails intersect in the image plane at the vanishing point

Vanishing points and lines

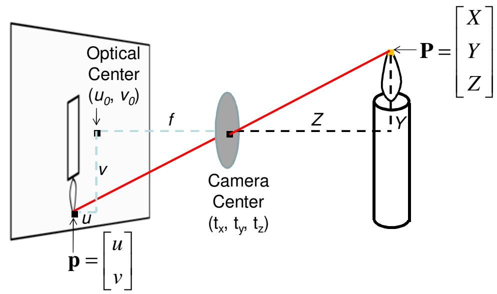


Slide from Efros, Photo from Criminisi

Vanishing points and lines

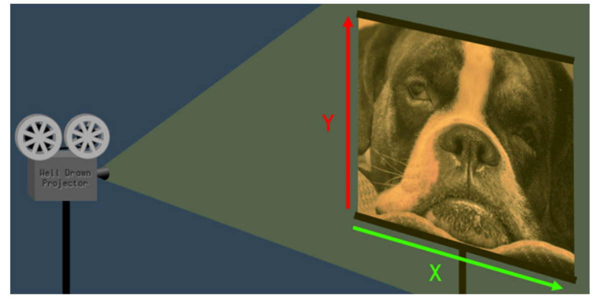


Projective Geometry: world coordinates → image coordinates



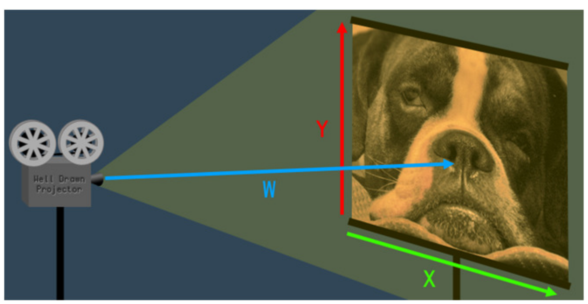
Projective Geometry Intro

- Imagine a projector that is projecting a 2D image onto a screen
- It's easy to identify the X and Y dimensions of the projected image



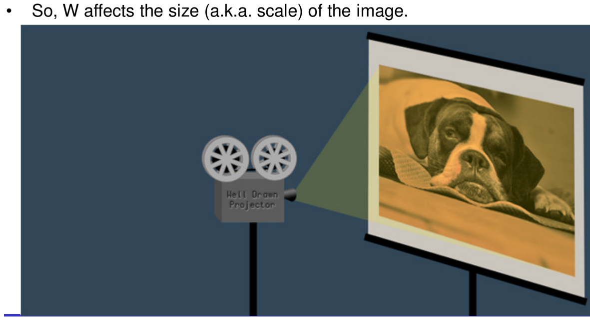
Projective Geometry Intro

- You can see the W dimension too
- The W dimension is the distance from the projector to the screen.



Projective Geometry Intro

- what does W do?
- imagine what happens with X and Y, if you decrease W
- the whole 2D image becomes smaller – X and Y decreases
- So, W affects the size (a.k.a. scale) of the image.



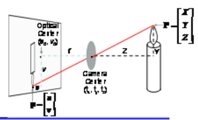
Homogeneous coordinates

- In projections on our image sensor, we do not know distances to objects
- So, we need to write scale-invariant coordinates (independent on distance)
  - Use homogeneous coordinates
  - By adding one more parameter

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates



Homogeneous coordinates

Converting to *homogeneous* coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous image coordinates — homogeneous 3D scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Homogeneous coordinates

Invariant to scaling

$$k \begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates — Cartesian Coordinates

Point in Cartesian is ray in Homogeneous

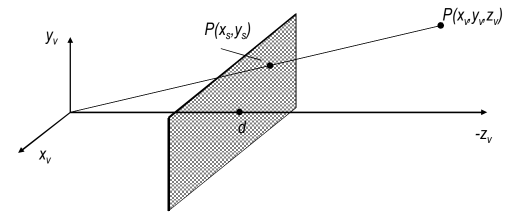
Multiview 3D video / Outline

- **Camera geometry**
  - Pinhole camera model
  - Projective geometry
  - Projection matrix,
  - Intrinsic/extrinsic camera parameters
- **Camera calibration**

Camera Projection Matrix

Basic Perspective Projection

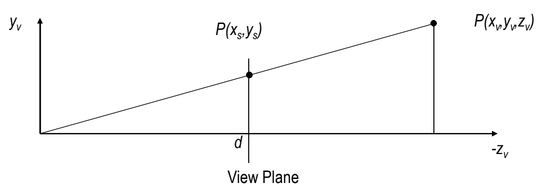
- **If you know *P(x$_v$,y$_v$,z$_v$)* and *d*, what is *P(x$_s$,y$_s$)*?**
  – Where does a point in view space end up on the screen?



Basic Perspective Projection

- **Similar triangles give:**

$$\frac{x_s}{d} = \frac{x_v}{z_v} \qquad \frac{y_s}{d} = \frac{y_v}{z_v}$$



Simple Perspective Transformation

- **Using homogeneous coordinates we can write:**
  – Our big advantage of homogeneous coordinates:
    - Scale (distance) invariant

$$\begin{bmatrix} x_s \\ y_s \\ d \end{bmatrix} \equiv \begin{bmatrix} x_v \\ y_v \\ z_v \\ z_v/d \end{bmatrix} \qquad \mathbf{P}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \mathbf{P}_v$$

41

Moving to Camera Matrix

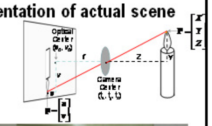
- That was a simple geometric case
- With actual camera, we have a set of parameters, which introduce function *F*

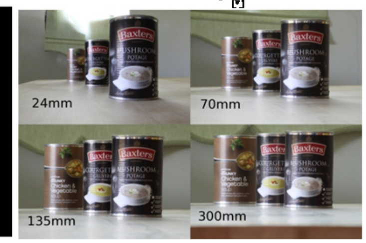
$$\mathbf{P}_s \overset{F}{=} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \mathbf{P}_v$$

42

Moving to Camera Matrix

- Camera (imager sensor and lenses) may change representation of actual scene
  - Varying focal length

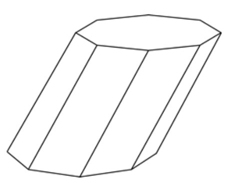




43

Moving to Camera Matrix

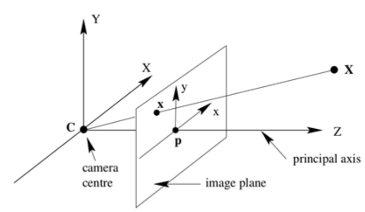
- Camera (imager sensor and lenses) may change representation of actual scene
  - Varying focal length
  - Skew (non-rectangular pixels)
  - Radial distortions

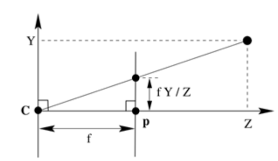


44

Moving to Camera Matrix

- Camera (imager sensor and lenses) may change representation of actual scene
  - Varying focal length
  - Skew (non-rectangular pixels)
  - Radial distortions
  - Principal Point offset

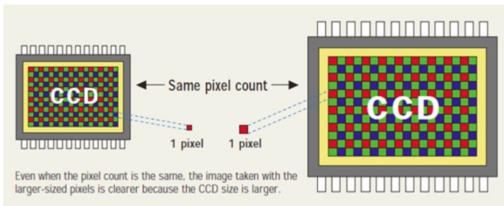




Moving to Camera Matrix

45

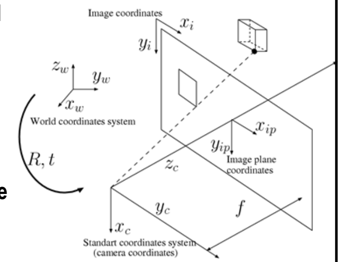
- Camera (imager sensor and lenses) may change representation of actual scene
  - Varying focal length
  - Skew (non-rectangular pixels)
  - Radial distortions
  - Principal Point offset
  - Pixel size (aspect ratio)



Camera geometry / Concept – (1)

46

- **To understand the 3D structure of objects/scene, a relation between point coordinates in the 3D world to pixels position is required.**
- **We have 2 coordinates systems: image and world.**
- **Goal: map points in 3D space (the world coordinate system) to the image plane (the image coordinates system).**



Camera geometry / Concept – (2)

47

- **Link the world and image coordinates by a set of parameters known as intrinsic and extrinsic parameters.**
- ***Intrinsic parameters:***
  - *focal length,*
  - width and height of the pixel on the sensor,
  - position of the principal point (origin of the image coordinates system).
- ***Extrinsic parameters:***
  - *camera position,*
  - *camera orientation.*

Pinhole Camera Projection Matrix

48

- **To take the above camera nuances in account:**

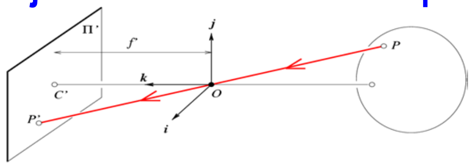
$$\mathbf{P}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \mathbf{P}_v$$

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

x: Image Coordinates: (u,v,1), also P_s
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1), also P_v

49

Projection Matrix with Assumptions



Intrinsic Assumptions
- Square pixels
- Optical center at (0,0)
- No skew

Known focal length *f*

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \underbrace{\begin{bmatrix} f&0&0\\0&f&0\\0&0&1\end{bmatrix}}_{K}\begin{bmatrix}0\\0\\0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

Slide Credit: Saverese

50

Remove assumption: known optical center

Intrinsic Assumptions
- Square pixels
- No skew

Known optical center *u, v*

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix} f&0&u_0&0\\0&f&v_0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

51

Remove assumption: square pixels

Intrinsic Assumptions
- No skew

Pixel aspect ratio is *alfa / betta*

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix} \alpha&0&u_0&0\\0&\beta&v_0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

52

Remove assumption: non-skewed pixels

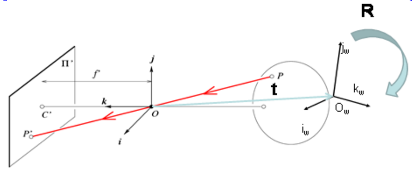
Intrinsic Assumptions
- No

Skew is not 0, but *s*

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix} \alpha&s&u_0&0\\0&\beta&v_0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

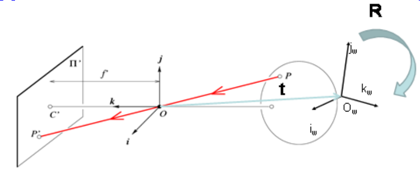
Note: different books use different notation for parameters

Oriented and Translated Camera



* *Extrinsic* parameters define orientation and location of the camera in the world coordinate system

Oriented and Translated Camera



Extrinsic Assumptions
- No rotation, camera axes aligned with World Coordinate System (WCS)
- Principal point is located in the center of WCS

$$\mathbf{x} = \mathbf{K}\boxed{[\mathbf{I} \quad \mathbf{t}]}\mathbf{X} \Rightarrow s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \boxed{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Allow camera translation

Extrinsic Assumptions
- No rotation

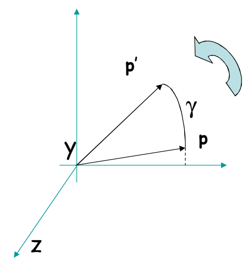
Known camera offset from origin of WCS – *tx, ty, tz*

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{t}]\mathbf{X} \Rightarrow w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \boxed{t_x} \\ 0 & 1 & 0 & \boxed{t_y} \\ 0 & 0 & 1 & \boxed{t_z} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Slide Credit: Saverese

3D Rotation of Points

Rotation around the coordinate axes, counter-clockwise:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Slide Credit: Savarese

3D Rotation of Points

Rotation around the coordinate axes

$$
\begin{aligned}
&R_Z R_Y R_X \\
&= \begin{pmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos B & 0 & \sin B \\ 0 & 1 & 0 \\ -\sin B & 0 & \cos B \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{pmatrix} \\
&= \begin{pmatrix} \cos C \cos B & -\sin C & \cos C \sin B \\ \sin C \cos B & \cos C & \sin C \sin B \\ -\sin B & 0 & \cos B \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{pmatrix} \\
&= \begin{pmatrix} \cos C \cos B & -\sin C \cos A + \cos C \sin B \sin A & \sin C \sin A + \cos C \sin B \cos A \\ \sin C \cos B & \cos C \cos A + \sin C \sin B \sin A & -\cos C \sin A + \sin C \sin B \cos A \\ -\sin B & \cos B \sin A & \cos B \cos A \end{pmatrix}
\end{aligned}
$$

Slide Credit: Savarese

3D Rotation of Points

Rotation around the coordinate axes

$$
R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}
$$

$$
R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}
$$

$$
R_z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
\begin{vmatrix} x' \\ y' \\ z' \end{vmatrix} = R_\gamma R_\beta R_\propto \begin{vmatrix} x \\ y \\ z \end{vmatrix}
$$

$$
\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.
$$

TU/e PdW-SZ-EB / 2017 Fac. EE SPS-VCA
Adv. Topics MMedia Video / **5LSH0 /** **Mod 02 3D intro & multiview video** VCA

Allow also camera rotation

$$
\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}
$$

$$
w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

TU/e PdW-SZ-EB / 2017 Fac. EE SPS-VCA
Adv. Topics MMedia Video / **5LSH0 /** **Mod 02 3D intro & multiview video** VCA

Degrees of freedom in Camera Matrix

$$
\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}
$$

5 6

$$
w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

TU/e PdW-SZ-EB / 2017 Fac. EE SPS-VCA
Adv. Topics MMedia Video / **5LSH0 /** **Mod 02 3D intro & multiview video** VCA

Camera Matrix

We can finally map a 3D point to the image!

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Unfortunately, not yet.
We need to find these parameters for each camera, first

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Multiview 3D video / Outline

* Camera geometry
  - Pinhole camera model
  - Projective geometry
  - Projection matrix,
  - Intrinsic/extrinsic camera parameters
* **Camera calibration**

Projective camera / Summary – (2)

* Combining the **camera matrix** (intrinsic parameters) and the **rotation/translation matrix** (extrinsic parameters) we obtain the camera calibration matrix C.

pixel coordinates → $p_{ip} = \underbrace{\mathrm{K}}_{3\times 4}\underbrace{\mathrm{E}}_{4\times 4} p_w = \underbrace{\mathrm{C}}_{3\times 4} p_w$ ← world coordinates

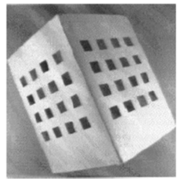
intrinsic parameters

extrinsic parameters

camera calibration matrix

* Projection matrix 3x4 has 11 degrees of freedom (scaling invariance).

Camera calibration



* **Goal: estimating coefficients of the camera calibration matrix.**
  - Once the camera calibration matrix parameters are known, the camera is "calibrated".
* **Simple calibration algorithm**
  - It is assumed that the world coordinates of 3D points are **known** with their corresponding 2D pixel coordinates.
  - Points are usually arranged in a special pattern for easy calibration.

Linear method for estimating matrix C – (1)

* **World-point coordinates and image-pixel positions are linked with the camera calibration matrix**

$$\begin{pmatrix} \lambda x_{ip} \\ \lambda y_{ip} \\ \lambda \end{pmatrix} = \underbrace{\begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}}_{projection\ matrix\ C} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix}$$

* P **is the 3x4 projection matrix, can be written as** $C = K[R|t]$
* **Algorithm consists of two steps:**
  - 1. Compute matrix C with a set of known 3D positions and their respective positions in the image.
  - 2. Extrinsic and intrinsic parameters are estimated from C.

Linear method for estimating matrix C – (2)

* **Use world point coordinates** p_w **and their corresponding pixel coordinates** p_{ip} **in the image to determine** C.
  - Each correspondence generates two equations

$$x_{ip} = \frac{x_w p_{11} + y_w p_{12} + z_w p_{13} + p_{14}}{x_w p_{31} + y_w p_{32} + z_w p_{33} + p_{34}}$$

$$y_{ip} = \frac{x_w p_{11} + y_w p_{12} + z_w p_{13} + p_{14}}{x_w p_{31} + y_w p_{32} + w z_w p_{33} + p_{34}}$$

which can be written as

$$x_{ip}(x_w p_{31} + y_w p_{32} + z_w p_{33} + p_{34}) = x_w p_{11} + y_w p_{12} + z_w p_{13} + p_{14}$$
$$y_{ip}(x_w p_{31} + y_w p_{32} + w z_w p_{33} + p_{34}) = x_w p_{11} + y_w p_{12} + z_w p_{13} + p_{14}$$

Linear method for estimating matrix C – (3)

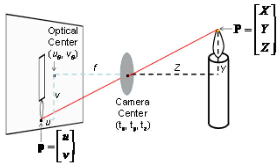
* **Stack equations into one equation system:**

$$\begin{pmatrix} x_w & y_w z_w & 1 & 0 & 0 & 0 & 0 & -x_{ip}x_w & -x_{ip}y_w & -z_{ip}z_w & -x_{ip} \\ x_w & y_w z_w & 1 & 0 & 0 & 0 & 0 & -x_{ip}x_w & -x_{ip}y_w & -z_{ip}z_w & -x_{ip} \\ & & & & & & \vdots & & & & \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ \vdots \\ p_{34} \end{pmatrix} = 0$$

* **The equation has 12 unknown parameters: at least 6 correspondence points are required.**
* **Typically, more points are used.**
  - Equation system gets over-constrained.
  - Equation is then solved using a least squares minimization.

69

Looking Back



Pinhole camera model

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous 3D scene coordinates

Camera projection matrix

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

70

Questions