Virtual Humans – Winter 23/24

Lecture 2_1 – Image Formation

Prof. Dr.-Ing. Gerard Pons-Moll
University of Tübingen / MPI-Informatics
Overview

• Part1. Image formation – Introduction
• Part2. Image formation – Projection Matrix
Computer Vision goal: Predict 3D human mesh from a single image

Input (2D)  Output (3D)
Computer Graphics goal: Render photorealistic avatars

Input (2D)  Output (3D)
For both problems we need to understand first image formation models.
The laws of perspective

✦ common assumptions
1. Light leaving an object travels in straight lines.
2. These lines converge to a point at the eye.

✦ natural perspective (Euclid, 3rd c. B.C.)
3a. More distant objects subtend smaller visual angles.
The laws of perspective

\[ \theta_2 > \theta_1 \]

- natural perspective  (Euclid, 3rd c. B.C.)
  3a. More distant objects subtend smaller visual angles.
Roman wall paintings

from Villa Publius Fannius Synistor, Boscoreale, Pompeii (c. 40 B.C.)

Still life with peaches, from Herculaneum (before 79 A.D.)
The laws of perspective

✦ common assumptions
  1. Light leaving an object travels in straight lines.
  2. These lines converge to a point at the eye.

✦ natural perspective  (Euclid, 3rd c. B.C.)
  3a. More distant objects subtend smaller visual angles.

✦ linear perspective  (Filippo Brunelleschi, 1413)
  3b. A perspective image is formed by the intersection of these lines with a “picture plane” (the canvas).
The laws of perspective

\[ \frac{y_2}{y_1} = \frac{\theta_2}{\theta_1} \]

\[ \theta_2 > \theta_1 \]

\[ y_2 > y_1 \]

\* natural perspective  \quad (Euclid, 3rd c. B.C.)

3a. More distant objects subtend smaller visual angles.

\* linear perspective  \quad (Filippo Brunelleschi, 1413)

3b. A perspective image is formed by the intersection of these lines with a “picture plane” (the canvas).
the division by \( z \) means that the size of an object in a photograph is inversely proportional to its distance from the camera.
Vanishing points

Q. How many vanishing points can there be in a perspective drawing?

1-point

2-point

3-point

(D’Amelio)
Q. Should the distant ends of a long facade be drawn smaller than its center in a perspective drawing?

Pause the video and think
Q. Should the distant ends of a long facade be drawn smaller than its center in a perspective drawing?

Should it look like this picture?
Q. Should the distant ends of a long facade be drawn smaller than its center in a perspective drawing?

- no, in linear perspective straight lines remain straight
- lines parallel to the picture plane do not converge
- they appear smaller when you view the drawing, due to natural perspective (angles subtended at eye)
Q. Why does this perspective drawing look distorted? 

(Dubery)
Q. Why does this perspective drawing look distorted?

- it’s not distorted; it’s a correct linear perspective
- you’re viewing the drawing from too far away
Recap

- **natural perspective**
  - visual angle subtended by a feature in the world
- **linear perspective**
  - intersections of lines of sight with a picture plane
  - the correct way to make a drawing on a flat surface
- **vanishing points**
  - one per direction of line in the scene
  - lines parallel to the picture plane do not converge

Questions?
Single lens reflex camera (SLR)

Nikon F4 (film camera)
Why not use sensors without optics?
Pinhole camera
(a.k.a. camera obscura)

- linear perspective with viewpoint at pinhole
- tilting the picture plane changes the number and location of vanishing points
Equivalence of Dürer’s glass and *camera obscura* (contents of whiteboard)

- both devices compute 2D planar geometric projections, i.e. projections along straight lines through a point and onto a plane
  - the images differ only in scale (and a reflection around the origin)
Effect of pinhole size

Photograph made with small pinhole

Photograph made with larger pinhole

(London)
Effect of pinhole size

Photograph made with small pinhole

Photograph made with larger pinhole

(London)
Replacing the pinhole with a lens
Replacing the pinhole with a lens

- A photographic camera produces the same 2D planar geometric projection as a camera obscura:
  - A lens replaces the pinhole, and film or a digital sensor becomes the picture plane.
  - Rotating the camera (and lens) around the lens’s center adds or removes vanishing points.
Geometrical optics

- parallel rays converge to a point located at focal length $f$ from lens

- rays going through center of lens are not deviated
  - hence same perspective as pinhole
Gauss’s ray tracing construction

- rays coming from points on a plane parallel to the lens are focused to points on another plane parallel to the lens
Changing the focus distance

- to focus on objects at different distances, move sensor relative to lens
- in a handheld camera, one actually moves the lens, not the sensor

by convention, the “focus distance” is on the object side of the lens
From Gauss’s ray construction to the Gaussian lens formula

- positive $s_i$ is rightward, positive $s_o$ is leftward
- positive $y$ is upward
From Gauss’s ray construction to the Gaussian lens formula

$$\frac{|y_i|}{y_o} = \frac{s_i}{s_o}$$
From Gauss’s ray construction to the Gaussian lens formula

\[
\frac{|y_i|}{y_o} = \frac{s_i}{s_o} \quad \text{and} \quad \frac{|y_i|}{y_o} = \frac{s_i - f}{f}
\]

\[
\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f}
\]
Changing the focus distance

- To focus on objects at different distances, move sensor relative to lens.
- In a handheld camera, one actually moves the lens, not the sensor.

By convention, the “focus distance” is on the object side of the lens.

\[
\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f}
\]
Pinhole camera model

Image height

Object height

Depth

Lens camera model

From Gauss’s ray construction to the Gaussian lens formula

\[
\frac{y}{y_o} = \frac{s_i}{s_o} \quad \text{and} \quad \frac{y}{y_o} = \frac{s_i - f}{f} \\
\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f}
\]
Lens camera vs Pinhole camera

• For the lens camera model, we derived
  \[ \frac{1}{z} + \frac{1}{s_i} = \frac{1}{f} \]

• As the depth goes to infinity \( z \to \infty \) we obtain \( s_i \to f \)
  That is, lens with focal length \( f \) is equivalent to pinhole at distance \( f \)
Changing the focal length

- if the sensor size is constant, the field of view becomes smaller

\[ FOV = 2 \arctan \left( \frac{h}{2f} \right) \]
Changing the sensor size

- if the sensor size is smaller, the field of view is smaller too

- smaller sensors either have fewer pixels, or smaller pixels, which are noisier
Changing the focal length versus changing the viewpoint

- changing the focal length lets us move back from a subject, while maintaining its size on the image
- but moving back changes perspective relationships

wide-angle

telephoto and moved back
Changing the focal length versus changing the viewpoint

- moving forward while shortening the focal length lets you keep objects at one depth the same size
- in cinematography, this is called the dolly-zoom, or “Vertigo effect”, after Alfred Hitchcock’s movie
Effect of focal length on portraits

- standard “portrait lens” is 85mm

wide angle  standard  telephoto
Recap

- pinhole cameras compute correct linear perspectives
  - but dark
- lenses gather more light
  - but only one plane of scene is in focus
  - distance from lens to this plane is called the \textit{focus distance}
  - change what’s in focus by moving the sensor or lens
- \textit{focal length} determines field of view
  - from wide angle to telephoto
  - depends on sensor size

Questions?
Let’s formalize mathematically...

The goal now is to express the image formation process using a single *Projection Matrix*
Object of Interest in World Coordinate System (U,V,W)
Imaging Geometry

Camera Coordinate System (X, Y, Z).

- Z is optic axis
- Image plane located f units out along optic axis
- f is called focal length
Forward Projection onto image plane.
3D \((X,Y,Z)\) projected to 2D \((x,y)\)
Our image gets digitized into pixel coordinates \((u,v)\)
Imaging Geometry

- Camera Coordinates
- Image (film) Coordinates
- World Coordinates

Pixel Coordinates
We want a mathematical model to describe how 3D World points get projected into 2D Pixel coordinates.

Our goal: describe this sequence of transformations by a big matrix equation!
Note, much of vision concerns trying to derive backward projection equations to recover 3D scene structure from images (via stereo or motion)

But first, we have to understand forward projection…
Forward Projection

World Coords
U
V
W

Camera Coords
X
Y
Z

Film Coords
x
y

Pixel Coords
u
v

3D-to-2D Projection
• perspective projection

We will start here in the middle, since we’ve already talked about this when discussing stereo.
Basic Perspective Projection

Scene Point

$p = (X,Y,Z)$

Image Point

$p = (x,y,f)$

Perspective Projection Eqns

\[ x = f \frac{X}{Z} \]
\[ y = f \frac{Y}{Z} \]
Basic Perspective Projection

Scene Point

\[ p = (X,Y,Z) \]

Image Point

\[ p = (x,y,f) \]

Perspective Projection Eqns

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

derived via similar triangles rule
Basic Perspective Projection

Scene Point

\[ P = (X,Y,Z) \]

Image Point

\[ p = (x,y,f) \]

Perspective Projection Eqns

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

\[ y = f \frac{Y}{Z} \]

derived via similar triangles rule
Basic Perspective Projection

Scene Point $P = (X,Y,Z)$

Image Point $p = (x,y,f)$

Perspective Projection Eqns

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

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

So how do we represent this as a matrix equation? We need to introduce homogeneous coordinates.
Homogeneous Coordinates

Represent a 2D point \((x,y)\) by a 3D point \((x',y',z')\) by adding a “fictitious” third coordinate.

By convention, we specify that given \((x',y',z')\) we can recover the 2D point \((x,y)\) as

\[
\begin{align*}
x &= \frac{x'}{z'} \\
y &= \frac{y'}{z'}
\end{align*}
\]

Note: \((x,y) = (x,y,1) = (2x, 2y, 2) = (k x, ky, k)\) for any nonzero \(k\) (can be negative as well as positive)
Perspective Matrix Equation
(in Camera Coordinates)

\[ x = f \frac{X}{Z} \]
\[ y = f \frac{Y}{Z} \]
\[ \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \]
Forward Projection

Rigid Transformation (rotation+translation) between world and camera coordinate systems
World to Camera Transformation

Rotate to align axes

Translate by $-C$ (align origins)

$P_C = R (P_W - C)$
Matrix Form, Homogeneous Coords

\[
P_C = R \left( P_W - C \right)
\]

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
= 
\begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & -c_x \\
0 & 1 & 0 & -c_y \\
0 & 0 & 1 & -c_z \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]
Example: Simple Stereo System

Left camera located at world origin (0,0,0) and camera axes aligned with world coord axes.
Simple Stereo, Left Camera

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
= 
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

camera axes aligned with world axes

located at world position (0,0,0)
Simple Stereo, Right Camera

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & -T_x \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

camera axes aligned with world axes

located at world position \((T_x,0,0)\)
Simple Stereo Projection Equations

Left camera

\[
\begin{bmatrix}
  x_l \\
  y_l \\
  1
\end{bmatrix} \sim \begin{bmatrix}
  f & 0 & 0 & 0 \\
  0 & f & 0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

\[ x_l = f \frac{X}{Z} \quad y_l = f \frac{Y}{Z} \]

Right camera

\[
\begin{bmatrix}
  x_r \\
  y_r \\
  1
\end{bmatrix} \sim \begin{bmatrix}
  f & 0 & 0 & 0 \\
  0 & f & 0 & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix} \begin{bmatrix}
  1 & 0 & 0 & -T_x \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

\[ x_r = f \frac{X - T_x}{Z} \quad y_r = f \frac{Y}{Z} \]
Bob’s sure-fire way(s) to figure out the rotation

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} =
\begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & -c_x \\
0 & 1 & 0 & \omega_x \\
0 & 0 & 1 & \omega_y \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

\[P_C = R P_W\]

This equation says how vectors in the world coordinate system (including the coordinate axes) get transformed into the camera coordinate system.
Figuring out Rotations

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} =
\begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

\[P_C = R P_W\]

what if world x axis (1,0,0) corresponds to camera axis (a,b,c)?

\[
\begin{pmatrix}
a \\
b \\
c \\
1
\end{pmatrix} =
\begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
1
\end{pmatrix}
= \begin{pmatrix}
a & r_{12} & r_{13} & 0 \\
a & r_{22} & r_{23} & 0 \\
a & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 \\
0 \\
0 \\
1
\end{pmatrix}
\]

we can immediately write down the first column of R!
Figuring out Rotations

Alternative approach: sometimes it is easier to specify what camera X, Y, or Z axis is in world coordinates. Then do rearrange the equation as follows.

\[
P_C = R P_W \rightarrow R^{-1}P_C = P_W \rightarrow R^T P_C = P_W
\]

\[
\begin{pmatrix}
    r_{11} & r_{21} & r_{31} & 0 \\
    r_{12} & r_{22} & r_{32} & 0 \\
    r_{13} & r_{23} & r_{33} & 0 \\
    0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    X \\
    Y \\
    Z \\
    1
\end{pmatrix} =
\begin{pmatrix}
    U \\
    V \\
    W \\
    1
\end{pmatrix}
\]
Figuring out Rotations

and likewise with world Y axis and world Z axis...

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
= 
\begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

- world X axis (1,0,0) in camera coords
- world Y axis (0,1,0) in camera coords
- world Z axis (0,0,1) in camera coords
Figuring out Rotations

and likewise with camera Y axis and camera Z axis...

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
= \begin{pmatrix}
r_{11} & r_{12} & r_{13} & 0 \\
r_{21} & r_{22} & r_{23} & 0 \\
r_{31} & r_{32} & r_{33} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
U \\
V \\
W \\
1
\end{pmatrix}
\]

axis is world coords

same axis in camera coords

camera X axis (1,0,0) in world coords

camera Y axis (0,1,0) in world coords

camera Z axis (0,0,1) in world coords
Note: External Parameters also often written as $R,T$

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

$$R \left( P_W - C \right)$$

$$= R P_W - R C$$

$$= R P_W + T$$
We now know how to transform 3D world coordinate points into camera coords, and then do perspective project to get 2D points in the film plane.

Next time: pixel coordinates
Intrinsic Camera Parameters

World Coords \rightarrow Camera Coords \rightarrow Film Coords \rightarrow Pixel Coords

U → X → x → u
V → Y → y → v
W → Z

Affine Transformation
Intrinsic parameters

• Describes coordinate transformation between film coordinates (projected image) and pixel array
• Film cameras: scanning/digitization
• CCD cameras: grid of photosensors

still in T&V section 2.4
Intrinsic parameters (offsets)

film plane
(projected image)

(pixel array)

\[ u = f \frac{X}{Z} + o_x \]
\[ v = f \frac{Y}{Z} + o_y \]

\( o_x \) and \( o_y \) called image center or principle point
Intrinsic parameters

sometimes one or more coordinate axes are flipped (e.g. T&V section 2.4)

film plane

\[ u = -f \frac{X}{Z} + o_x \]

\[ v = -f \frac{Y}{Z} + o_y \]

pixel array
Intrinsic parameters (scales)

Sampling determines how many rows/cols in the image.

- **Film**
  - Scanning resolution
  - C cols x R rows

- **CCD**
  - Analog
  - Resample
  - C cols x R rows
Effective Scales: $s_x$ and $s_y$

$$u = \frac{1}{s_x} f \frac{X}{Z} + o_x \quad v = \frac{1}{s_y} f \frac{Y}{Z} + o_y$$

Note, since we have different scale factors in x and y, we don’t necessarily have square pixels!

Aspect ratio is $s_y / s_x$
Adding the intrinsic parameters into the perspective projection matrix:

\[
\begin{bmatrix}
  x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
  f / s_x & 0 & o_x & 0 \\
  0 & f / s_y & o_y & 0 \\
  0 & 0 & 1 & 0
\end{bmatrix}\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

To verify:

\[
u = \frac{x'}{z'}, \quad v = \frac{y'}{z'}
\]

\[
u = \frac{1}{s_x} f \frac{X}{Z} + o_x, \quad v = \frac{1}{s_y} f \frac{Y}{Z} + o_y
\]
Note:

Sometimes, the image and the camera coordinate systems have opposite orientations: [the book does it this way]

\[
f \frac{X}{Z} = \begin{bmatrix} u - o_x \end{bmatrix} s_x
\]

\[
f \frac{Y}{Z} = \begin{bmatrix} v - o_y \end{bmatrix} s_y
\]

\[
\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -f/s_x & 0 & +o_x & 0 \\ 0 & -f/s_y & +o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\]
Note 2

In general, I like to think of the conversion as a separate 2D affine transformation from film coords \((x,y)\) to pixel coordinates \((u,v)\):

\[
\begin{pmatrix}
u' \\
v' \\
w'
\end{pmatrix} = \begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix} \begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]

\[
u = M_{\text{int}} \mathbf{P}_C = M_{\text{aff}} M_{\text{proj}} \mathbf{P}_C
\]
Summary: Forward Projection

World Coords → Camera Coords → Film Coords → Pixel Coords

\[ \begin{bmatrix} U \\ V \\ W \end{bmatrix} \xrightarrow{M_{\text{ext}}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \xrightarrow{M_{\text{proj}}} \begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{M_{\text{aff}}} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ \begin{bmatrix} U \\ V \\ W \end{bmatrix} \xrightarrow{M_{\text{ext}}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \xrightarrow{M_{\text{int}}} \begin{bmatrix} u \\ v \end{bmatrix} \]

\[ \begin{bmatrix} U \\ V \\ W \end{bmatrix} \xrightarrow{M} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} \]
Projection Models

Perspective

Increasing Focal Length / Distance from Camera

Weak Perspective / Orthographic

Increasing Focal Length / Distance from Camera
Orthographic projection of a 3D point $\mathbf{x}_c \in \mathbb{R}^3$ to pixel coordinates $\mathbf{x}_s \in \mathbb{R}^2$:

- The x and y axes of the camera and image coordinate systems are shared.
- Light rays are parallel to the z-coordinate of the camera coordinate system.
- During projection, the z-coordinate is dropped, x and y remain the same.
- Remark: the y coordinate is not shown here for clarity, but behaves similarly.
Scaled Orthographic Projection

In practice, world coordinates (which may measure dimensions in meters) must be scaled to fit onto an image sensor (measuring in pixels) ⇒ *scaled orthography*:

\[
x_s = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \end{bmatrix} x_c \iff \bar{x}_s = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{x}_c
\]

Remark: The unit for \( s \) is px/m or px/mm to convert metric 3D points into pixels.

Under orthography, structure and motion can be estimated simultaneously using factorization methods (e.g., via singular value decomposition).
Slide Credits

• Marc Levoy – Digital photography course at Stanford
• Robert Collins – Computer Vision at Penn. State University
• Andreas Geiger – Computer Vision at University of Tübingen