Pinhole camera

and homogeneous coordinates

Morten R. Hannemose, mohan@dtu.dk

February 2, 2024

02504 Computer vision course lectures,
DTU Compute, Kgs. Lyngby 2800, Denmark

Learning objectives

After this lecture you should be able to:

- explain homogeneous coordinates
- convert to and from homogeneous coordinates
- perform relevant coordinate transformations
- explain the pinhole camera model

Presentation topics

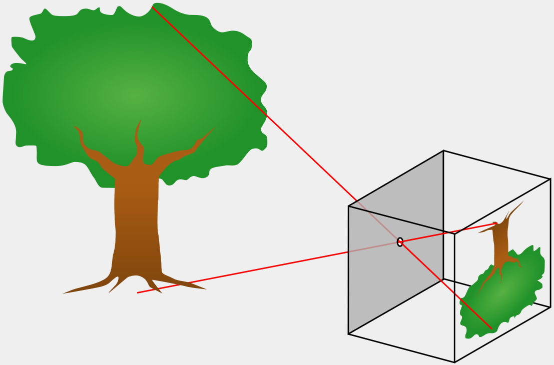
Pinhole camera

What is a “good” camera model?

. . . in terms of accuracy vs. ease-of-use?

1. As accurate as possible
2. As easy to use as possible
3. somewhere between 1 and 2

Pinhole camera



Light travels in straight lines.

The projected image appears upside down.

Each point in an image corresponds to a direction from the camera

Real life example

Can I get two volunteers?

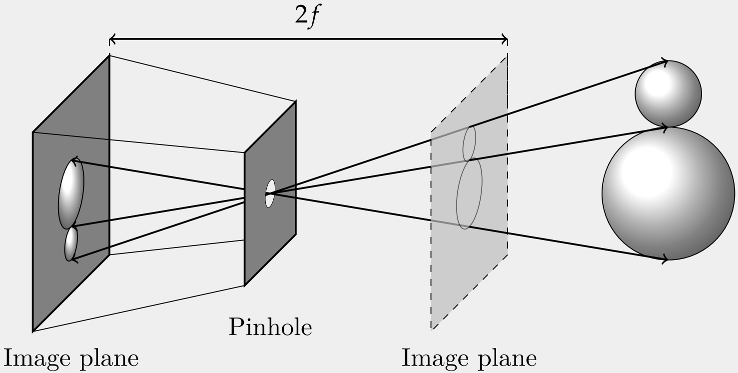
Real life example

Can I get two volunteers?

- A point seen in a single camera must be along a specific line in the other camera.
- Seeing the same point in two cameras is enough to find the point in 3D.

Perspective transformations

Pinhole camera again

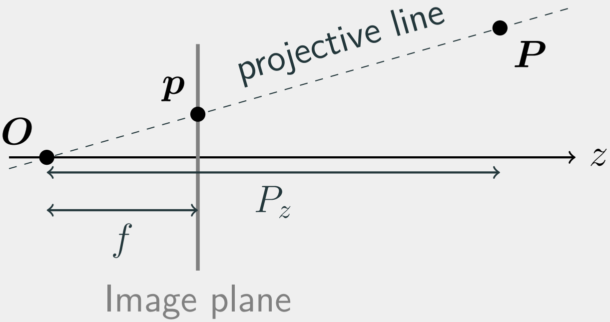


When modelling we place the "image plane" in front of the camera.
The distance from image plane to camera is the focal length f.

Perspective projection

$$\boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix},$$

$$\boldsymbol{p} = \frac{f}{P_z} \begin{bmatrix} P_x \\ P_y \end{bmatrix}$$

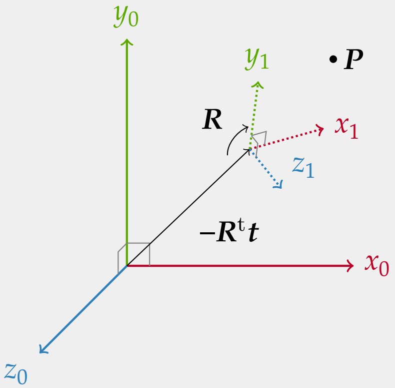


Rigid transformations

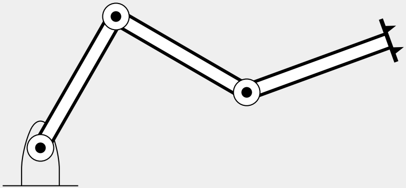
Rigid transformations: rotations and translations

Rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$

$$\boldsymbol{P}_1 = \boldsymbol{R}\boldsymbol{P}_0 + \boldsymbol{t}$$



Robot arm transformations



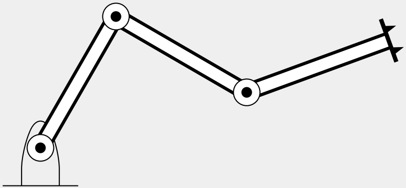
$$\boldsymbol{P}_1 = \boldsymbol{R}_1 \boldsymbol{P}_0 + \boldsymbol{t}_1$$

$$\boldsymbol{P}_2 = \boldsymbol{R}_2 \boldsymbol{P}_1 + \boldsymbol{t}_2$$

$$\boldsymbol{P}_3 = \boldsymbol{R}_3 \boldsymbol{P}_2 + \boldsymbol{t}_3$$

$$\boldsymbol{P}_4 = \boldsymbol{R}_4 \boldsymbol{P}_3 + \boldsymbol{t}_4$$

Robot arm transformations



$$P_4 = R_4(R_3(R_2(R_1 P_0 + t_1) + t_2) + t_3) + t_4$$
$$P_4 = R_4 R_3 R_2 R_1 P_0 + R_4 R_3 R_2 t_1 + R_4 t_3 + R_4 R_3 t_2 + t_4$$

Similar to the math we need to transform from the coordinate system of one camera to another.

Homogeneous coordinates

Homogeneous coordinates

Here is a point in 3D:

$$\boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

Homogeneous coordinates

Here is a point in 3D:

$$\boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix}$$

What if we made it more complicated and used four numbers?

$$\boldsymbol{P_h} = \begin{bmatrix} sP_x \\ sP_y \\ sP_z \\ s \end{bmatrix}$$

Uhm, okay?

So this means that

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \\ 6 \\ 2 \end{bmatrix}$$

are the same point in homogeneous coordinates

Euclidean transformations again

Rotation $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}$, with columns $\boldsymbol{r}_i$, and translation $\boldsymbol{t}$

$$\boldsymbol{P}_1 = \boldsymbol{R}\boldsymbol{P}_0 + \boldsymbol{t}$$

Euclidean transformations again

Rotation $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3\end{bmatrix}$, with columns $\boldsymbol{r}_i$, and translation $\boldsymbol{t}$

$$\boldsymbol{P}_1 = \boldsymbol{R}\boldsymbol{P}_0 + \boldsymbol{t} = \boldsymbol{r}_1 P_x + \boldsymbol{r}_2 P_y + \boldsymbol{r}_3 P_z + \boldsymbol{t}$$

Euclidean transformations again

Rotation $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3\end{bmatrix}$, with columns $\boldsymbol{r}_i$, and translation $\boldsymbol{t}$

$$\boldsymbol{P}_1 = \boldsymbol{R}\boldsymbol{P}_0 + \boldsymbol{t} = \boldsymbol{r}_1 P_x + \boldsymbol{r}_2 P_y + \boldsymbol{r}_3 P_z + \boldsymbol{t}$$

$$\boldsymbol{P}_1 = \begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t}\end{bmatrix} \qquad \begin{bmatrix}P_x \\ P_y \\ P_z \\ 1\end{bmatrix}$$

Euclidean transformations again

Rotation $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 \end{bmatrix}$, with columns $\boldsymbol{r}_i$, and translation $\boldsymbol{t}$

$$\boldsymbol{P}_1 = \boldsymbol{R}\boldsymbol{P}_0 + \boldsymbol{t} = \boldsymbol{r}_1 P_x + \boldsymbol{r}_2 P_y + \boldsymbol{r}_3 P_z + \boldsymbol{t}$$

$$\boldsymbol{P}_1 = \underbrace{\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{bmatrix}}_{\text{Transform: } \tilde{\boldsymbol{T}}} \underbrace{\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}}_{\text{Homogeneous: } \boldsymbol{P}_{0h}} = \tilde{\boldsymbol{T}}\boldsymbol{P}_{0h}$$

Homogeneous euclidean transformations

Fully homogeneous euclidean transformations become

$$\boldsymbol{P}_{1h} = \boldsymbol{T}\boldsymbol{P}_{0h}$$

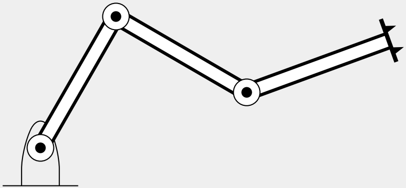
$$\begin{bmatrix} \boldsymbol{P}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_0 \\ 1 \end{bmatrix}$$

Homogeneous euclidean transformations

The homogeneous transformation $\boldsymbol{T}$ takes on the 4×4 form

$$\boldsymbol{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}$$

Robot arm and homogeneous transformations



$$\boldsymbol{Q}_{4h} = \boldsymbol{T}_4\boldsymbol{T}_3\boldsymbol{T}_2\boldsymbol{T}_1\boldsymbol{Q}_{0h}$$

Not just easier; It is faster! and let's us do a lot of mathemagic.

We can represent a rigid transformation both as a 3×4 and as a 4×4 matrix.

The *in*homogeneous p corresponds to the homogeneous p_h by

$$\boldsymbol{p}_h = \begin{bmatrix} s\boldsymbol{p} \\ s \end{bmatrix}$$

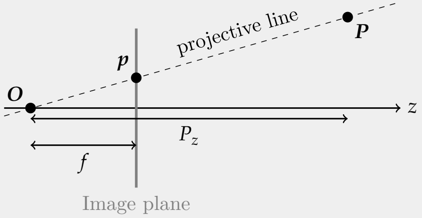
Questions? Short break

The projective transformation

Assume $f = 1$:

$$p_x = \frac{P_x}{P_z}, \quad p_y = \frac{P_y}{P_z}$$

$$\boldsymbol{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} s p_x \\ s p_y \\ s \end{bmatrix} = \boldsymbol{p}_h$$



Projective transformation is like assuming point in 3D is a 2D homogeneous point.

There are many different notations for homogeneous coordinates

$$\boldsymbol{q} = \begin{bmatrix} s\boldsymbol{p} \\ s \end{bmatrix}$$

Lines in homogenous coordinates

Consider the line given implicitly by

$$0 = ax + by + c$$

We can write this in homogeneous coordinates

$$\begin{aligned} &= \begin{bmatrix} a \\ b \\ c \end{bmatrix}^\mathsf{T} \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} \\ &= \boldsymbol{l}^\mathsf{T} \boldsymbol{p}_h \end{aligned}$$

Lines in homogenous coordinates

If $a^2 + b^2 = 1$ and the scale of the homogeneous point is 1 then,

$$\begin{aligned} d &= \begin{bmatrix} a \\ b \\ c \end{bmatrix}^\top \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ &= \boldsymbol{l}^\top \boldsymbol{p}_h \end{aligned}$$

d is the signed distance from the point to the line.

The homogeneous coordinate system — summary

The additional imaginary scale s, or alternatively dimension w

$$\boldsymbol{u} = s \begin{bmatrix} \boldsymbol{v} \\ 1 \end{bmatrix} = \begin{bmatrix} s\boldsymbol{v} \\ s \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}' \\ w \end{bmatrix}$$

- Dimensionality is $N + 1$
- Points have a scale $s \neq 0$
- Directions have $w = 0$
- Many-to-one correspondence: $\boldsymbol{u} \in \mathbb{R}^{N+1}$ and $\boldsymbol{v} \in \mathbb{R}^N$

Getting *in*homogeneous coordinates back

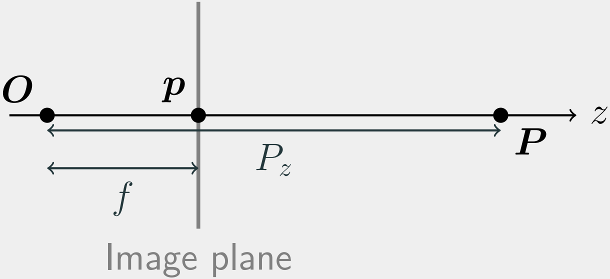
$$\boldsymbol{v} = \Pi\left(\boldsymbol{u}\right) = \Pi\left(\begin{bmatrix}\boldsymbol{v}' \\ s\end{bmatrix}\right) = \boldsymbol{v}'/s$$

Trivial inverse

Pinhole camera model

Principal point

Let $\boldsymbol{P}$ be a point exactly in the viewing direction of the camera.



$\boldsymbol{p}$ is called the principal point.

Where is $\boldsymbol{p}$ in the image on the right?

Principal point

Where is $(0, 0)$ in the image?

- After projective transformation

Principal point

Where is (0, 0) in the image?

- After projective transformation
  - At the principal point

Principal point

Where is (0, 0) in the image?

- After projective transformation
  - At the principal point
- Pixel coordinates
  - Upper left corner

We introduce δ_x and δ_y to translate (0, 0) from the principal point to the upper left corner.

It is typically around half of the resolution of the camera.

Principal point

Projection is now

$$p_x = \frac{f}{P_z} P_x + \delta_x$$
$$p_y = \frac{f}{P_z} P_y + \delta_y$$

Can we write all of this using homogeneous coordinates?

Principal point

Yes we can!

$$\boldsymbol{p}_h = \underbrace{\begin{bmatrix} f & 0 & \delta_x \\ 0 & f & \delta_y \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{K}} \boldsymbol{P}$$

And it even looks nice! This is called the camera matrix.

It contains intrinsic camera parameters.

Extrinsics

- So far we have assumed the camera is at the origin (0, 0, 0)
- Does not generalize well to multiple cameras
- Solution:
  - Introduce a canonical "world" coordinate system
  - Transform points from world to camera before projecting

Extrinsics

- We parametrize the this transformation with a
  - $\boldsymbol{R}$ (rotation)
  - $\boldsymbol{t}$ (translation)
- These are the extrinsics
- Transforming to the reference frame of the camera:

$$\begin{aligned} \boldsymbol{P}_{cam} &= \boldsymbol{R}\boldsymbol{P} + \boldsymbol{t} \\ &= \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ 1 \end{bmatrix} \end{aligned}$$

Projection matrix

Projecting a single point in 3D to the camera

$$\boldsymbol{p}_h = \boldsymbol{K}\boldsymbol{P}_{cam}$$

Projection matrix

Projecting a single point in 3D to the camera

$$
\begin{aligned}
\boldsymbol{p}_h &= \boldsymbol{K}\boldsymbol{P}_{cam} \\
&= \underbrace{\boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}}_{\mathcal{P}}\boldsymbol{P}_h
\end{aligned}
$$

Projection matrix

Projecting a single point in 3D to the camera

$$
\begin{aligned}
\boldsymbol{p}_h =& \boldsymbol{K}\boldsymbol{P}_{cam} \\
=& \underbrace{\boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}}_{\mathcal{P}}\boldsymbol{P}_h =
\end{aligned}
$$

$$
\begin{bmatrix} sp_x \\ sp_y \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & \delta_x \\ 0 & f & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}
$$

Wrapping up

- The translation is not the position of the camera
- What happens if the last coordinate of $\boldsymbol{P}_h$ is not 1?

Wrapping up

- The translation is not the position of the camera
- What happens if the last coordinate of $\boldsymbol{P}_h$ is not 1?
  - We get a scaled version of $\boldsymbol{P}_{cam}$ but, it is along the same line, so it projects to the same point.

Projection matrix:

$$\boldsymbol{q} = \mathcal{P}\boldsymbol{P}_h = \boldsymbol{K}\begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\end{bmatrix}\boldsymbol{P}_h$$

The matrix $\mathcal{P}$ is known as the projection matrix

(don't call it the camera matrix)

Exercise information

- Use Python interactively
  - Jupyter notebook
  - VS Code
  - Spyder
  - etc.
- Makes it easier to debug

Exercise information: Storing points on the computer

- Storing multiple one-dimensional vectors happens frequently
- Matrices are ideal for this
- We always operate on column vectors, so these matrices should be $3 \times n$ for many 3D vector (for example)
- Matrix multiplication lets you project many points at once
  - `ph = P.dot(Ph)` or even shorter
  - `ph = P@Ph`

Comment about exercises

- NumPy is your friend!
- If you need for-loops, you're probably not doing it the easy way.
  - No exercises today need for-loops (except the provided function)
- Converting from homogeneous coordinates to regular
  - `p = ph[:-1]/ph[-1]`
- Ask the TAs 🙂

Learning objectives

After this lecture you should be able to:

- explain homogeneous coordinates
- convert to and from homogeneous coordinates
- perform relevant coordinate transformations
- explain the pinhole camera model

Exercise time!