CSE 152: Computer Vision

Hao Su

Review of Neural Networks

Q1: Difference between "Neural Networks" and "Convolutional Neural Networks"

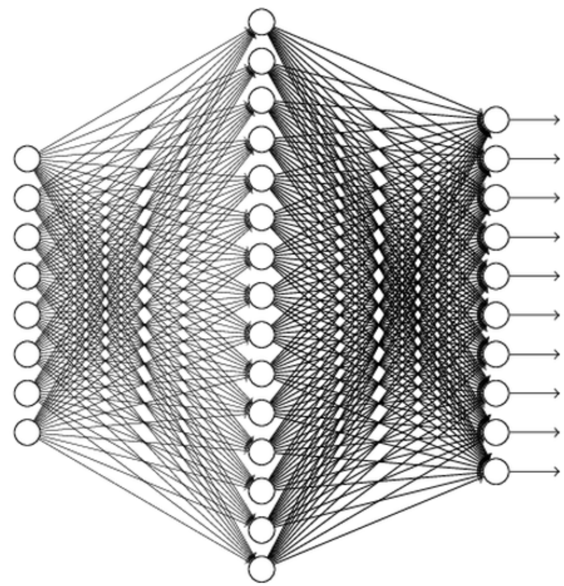
Neural Networks

- A universal function approximator by composing multiple layers
- Interleaved linear layers and non-linear layers (e.g., ReLU)

- A general concept

Fully-Connect Network

- Also known as “multilayer perceptron” (MLP)
- Every neuron in one layer is connected with every neuron in the next layer



- In our homework, we denote by mlp(n_1, n_2, …,n_k), where n_i is the number of neurons in the i-th layer.

In pytorch:

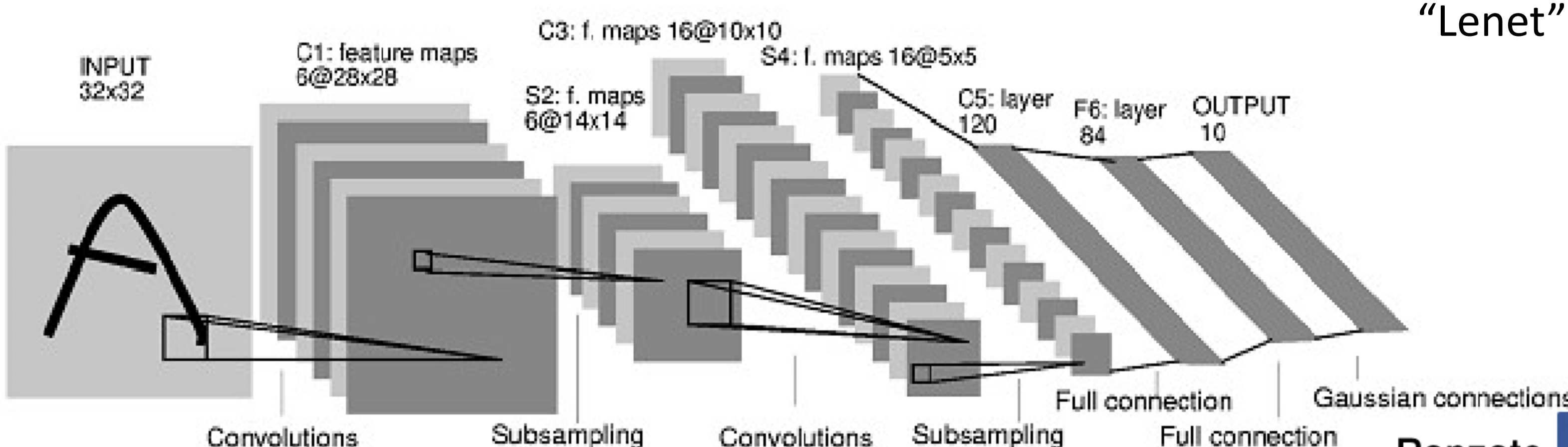
```
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.layers = nn.Sequential(
            nn.Linear(784, 100),
            nn.ReLU(),
            nn.Linear(100, 10)
        )

    def forward(self, x):
        # convert tensor (128, 1, 28, 28) --> (128, 1*28*28)
        x = x.view(x.size(0), -1)
        x = self.layers(x)
        return x
```

Convolutional Neural Network

- The next layer is obtained by applying a linear filter and some non-linear operation (e.g., max pooling, ReLU)

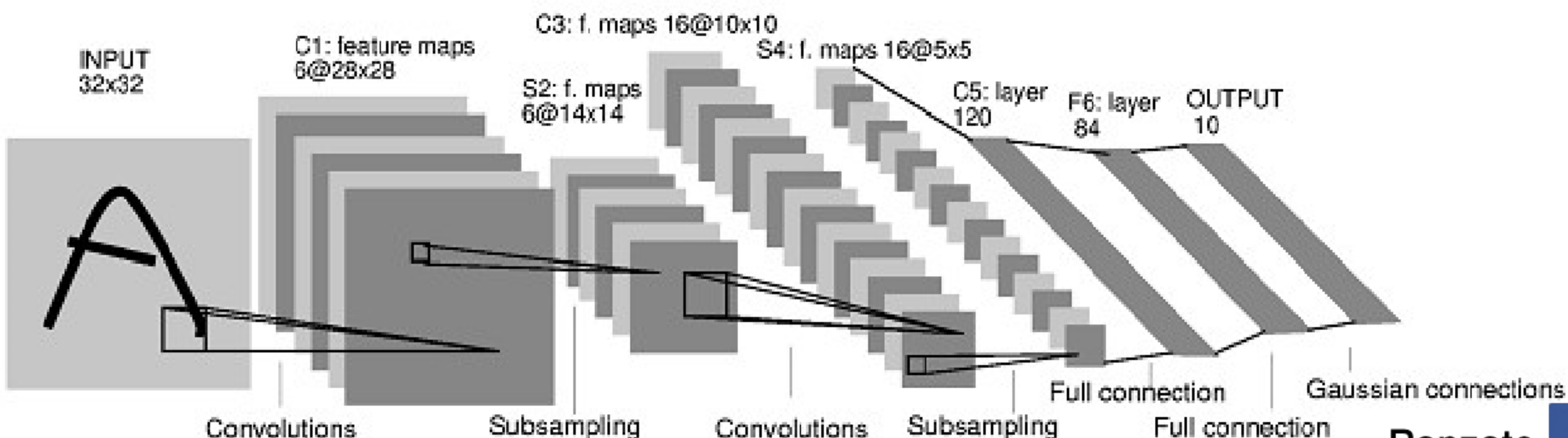
“Lenet”



Ranzato

Convolutional Neural Network

- where parameters are stored: in convolutional kernels and bias
- where data is stored: in feature maps



Ranzato

In pytorch

```python
class LeNet5(nn.Module):

    def __init__(self, n_classes):
        super(LeNet5, self).__init__()

        self.feature_extractor = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=2),
            nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1),
            nn.Tanh(),
            nn.AvgPool2d(kernel_size=2),
            nn.Conv2d(in_channels=16, out_channels=120, kernel_size=5, stride=1),
            nn.Tanh()
        )

        self.classifier = nn.Sequential(
            nn.Linear(in_features=120, out_features=84),
            nn.Tanh(),
            nn.Linear(in_features=84, out_features=n_classes),
        )


    def forward(self, x):
        x = self.feature_extractor(x)
        x = torch.flatten(x, 1)
        logits = self.classifier(x)
        probs = F.softmax(logits, dim=1)
        return logits, probs
```

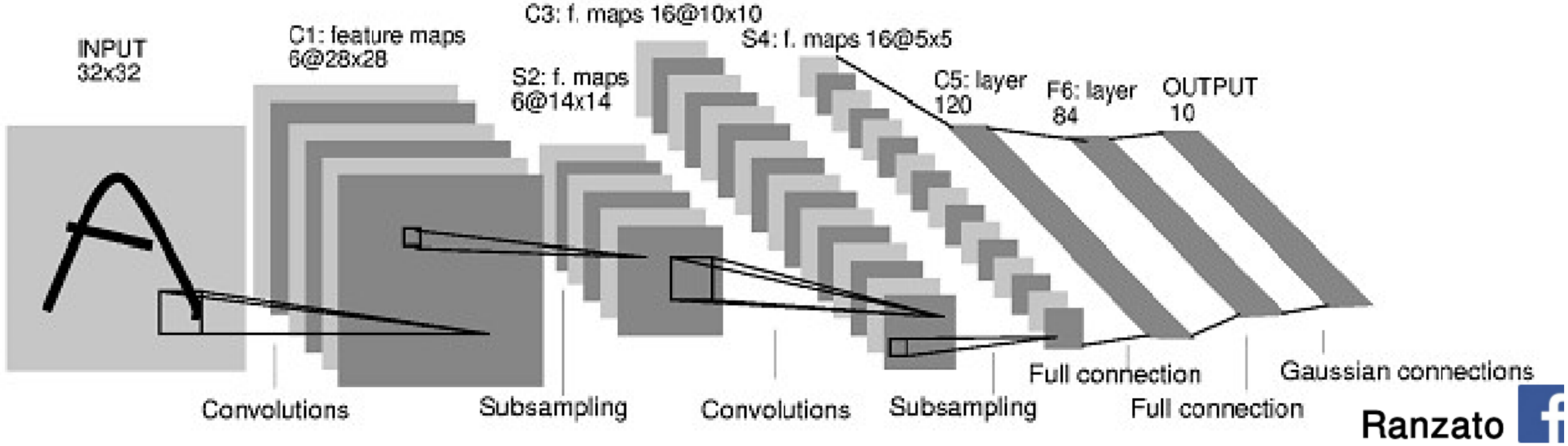
How to set hyper-parameters?

- Hyper-parameters: parameters not learned by the network but set by you.
- kernel size, stride, number of output channels, learning rate, optimizer, …
- Some tricks that might be useful: http://karpathy.github.io/2019/04/25/recipe/

How to set hyper-parameters?

- **Best practice: Start from classical networks, and adjust according to feedback.**
- Classical networks are distilled from experiments in many thousand papers that have burned many millions of dollars.
- LeNet, AlexNet, ResNet, DenseNet, PointNet, SparseConvNet, …

Q2: Back-propagation, Gradient descent, Connection with Networks



- Parameters: all to be learned by gradient descent: kernel weights, bias, any other unknowns in layers
- Denoted by θ when we formulate optimization problem by convention

minimize $L(\theta)$

- We use stochastic gradient descent to minimize the loss
- To compute gradient, we do "back-propagation":
  - A network is $y = f_n(f_{n-1}(\cdots f_1(x)\ldots))$, where f_i is the i-th layer, with parameters θ_i
  - To compute gradient by chain rule, we have

$$\frac{\partial f_n}{\partial f_{n-1}} \frac{\partial f_{n-1}}{\partial f_{n-2}} \cdots \frac{\partial f_i}{\partial \theta_i}$$

  - Product of matrices

Outline

- Vectors and Matrices
  - Basic matrix operations
  - Determinants, norms, trace
  - Special matrices
- Transformation Matrices
  - Homogeneous matrices
  - Translation
- Matrix inverse
- Matrix rank

Transformation

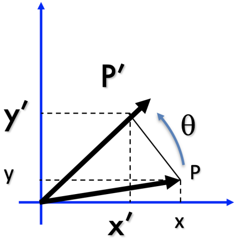
- Matrices can be used to transform vectors in useful ways, through multiplication: $x' = Ax$
- Simplest is scaling:

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

(Verify by yourself that the matrix multiplication works out this way)

Rotation (2D case)

Counter-clockwise rotation by an angle θ



$$x' = \cos\theta x - \sin\theta y$$
$$y' = \cos\theta y + \sin\theta x$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = RP$$

Transformation Matrices

- Multiple transformation matrices can be used to transform a point:

$$p' = R_2 R_1 S p$$

Transformation Matrices

- Multiple transformation matrices can be used to transform a point:

$$p' = R_2 R_1 S p$$

- The effect of this is to apply their transformations one after the other, from **right to left**

Transformation Matrices

- Multiple transformation matrices can be used to transform a point:

$$p' = R_2 R_1 S p$$

- The effect of this is to apply their transformations one after the other, from **right to left**
- In the example above, the result is

$$(R_2(R_1(Sp)))$$

Homogeneous System

- In general, a matrix multiplication lets us linearly combine components of a vector

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

  - This is sufficient for scale, rotate, skew transformations
  - But notice, we cannot add a constant! :(

Homogeneous System

- The (somewhat hacky) solution? Stick a "1" at the end of every vector:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

- Now we can rotate, scale, and skew like before, AND translate (note how the multiplication works out, above)
- This is called "homogeneous coordinates"

Homogeneous System

- In homogeneous coordinates, the multiplication works out so the rightmost column of the matrix is a vector that gets added

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

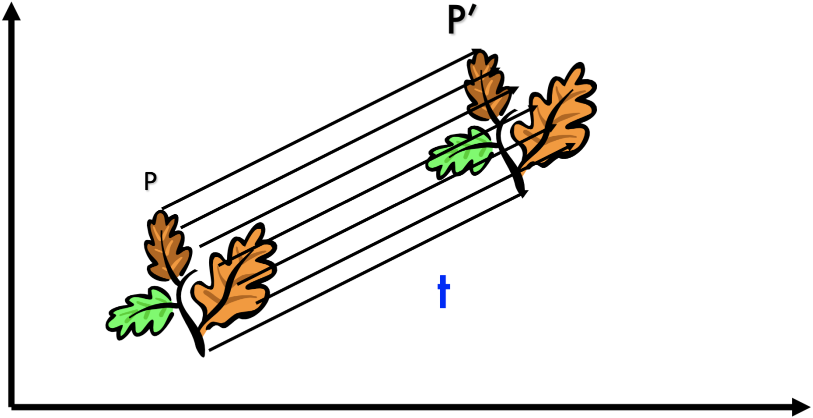
- Generally, a homogeneous transformation matrix will have a bottom row of [001], so that the result has a "1" at the bottom, too.

Homogeneous System

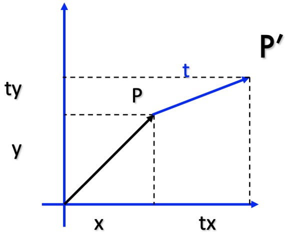
- One more thing we might want: to divide the result by something:
  - Matrix multiplication cannot actually divide
  - So, **by convention**, in homogeneous coordinates, we'll divide the result by its last coordinate after doing a matrix multiplication

$$\begin{bmatrix} x \\ y \\ 7 \end{bmatrix} \Rightarrow \begin{bmatrix} x/7 \\ y/7 \\ 1 \end{bmatrix}$$

2D Transformation using Homogeneous Coordinates



2D Transformation using Homogeneous Coordinates



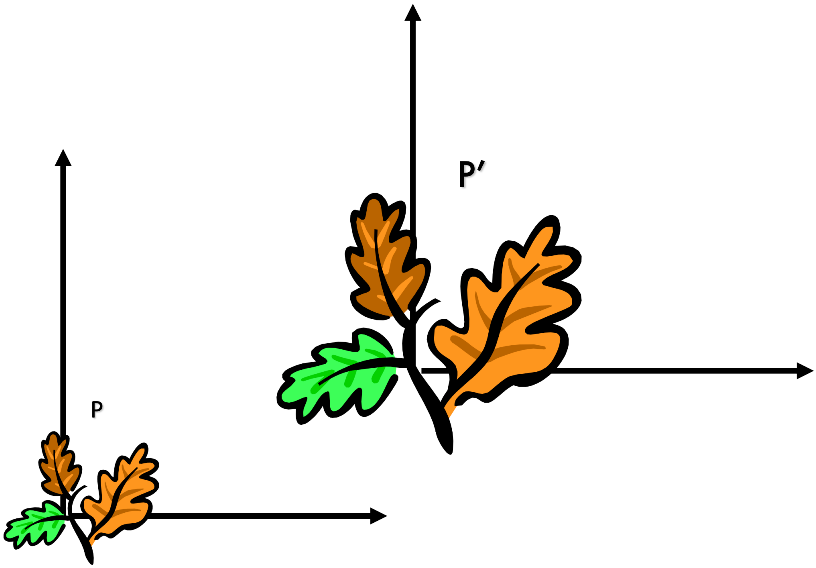
$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{t} = (t_x, t_y) \rightarrow (t_x, t_y, 1)$$

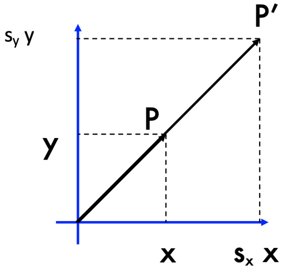
$$\mathbf{P}' \rightarrow \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \boxed{\begin{matrix} t_x \\ t_y \end{matrix}} \\ 0 & 1 & \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boxed{\begin{matrix} x \\ y \\ 1 \end{matrix}} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \cdot \mathbf{P} = \mathbf{T} \cdot \mathbf{P}$$

Scaling



Scaling Equation



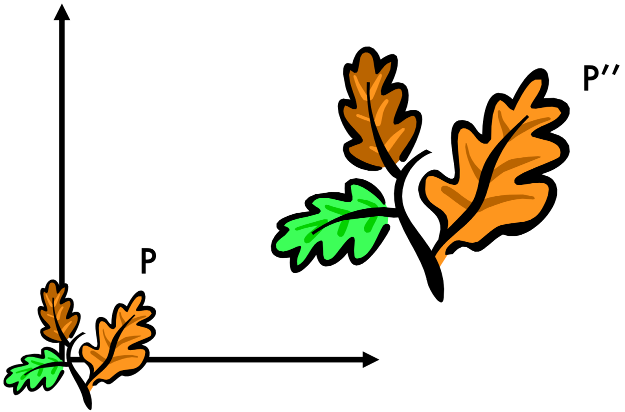
$$\mathbf{P} = (\mathrm{x}, \mathrm{y}) \rightarrow \mathbf{P}' = (\mathrm{s}_\mathrm{x}\mathrm{x}, \mathrm{s}_\mathrm{y}\mathrm{y})$$

$$\mathbf{P} = (x, y) \rightarrow (x, y, 1)$$

$$\mathbf{P}' = (s_x x, s_y y) \rightarrow (s_x x, s_y y, 1)$$

$$\mathbf{P}' \rightarrow \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{S}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{S}' & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \cdot \mathbf{P} = \mathbf{S} \cdot \mathbf{P}$$

Scaling & Translating



$$P'' = T \cdot P' = T \cdot (S \cdot P) = T \cdot S \cdot P$$

Scaling & Translating

$$P'' = T \cdot S \cdot P = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + t_x \\ s_y y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} S & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling+Rotation+Translation

$$P' = (T\ R\ S)\ P$$

$$P' = T \cdot R \cdot S \cdot P = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \boxed{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} RS & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

CSE 152: Computer Vision

Hao Su

Lecture 11: Camera Models

Credit: CS231a, Stanford, Silvio Savarese

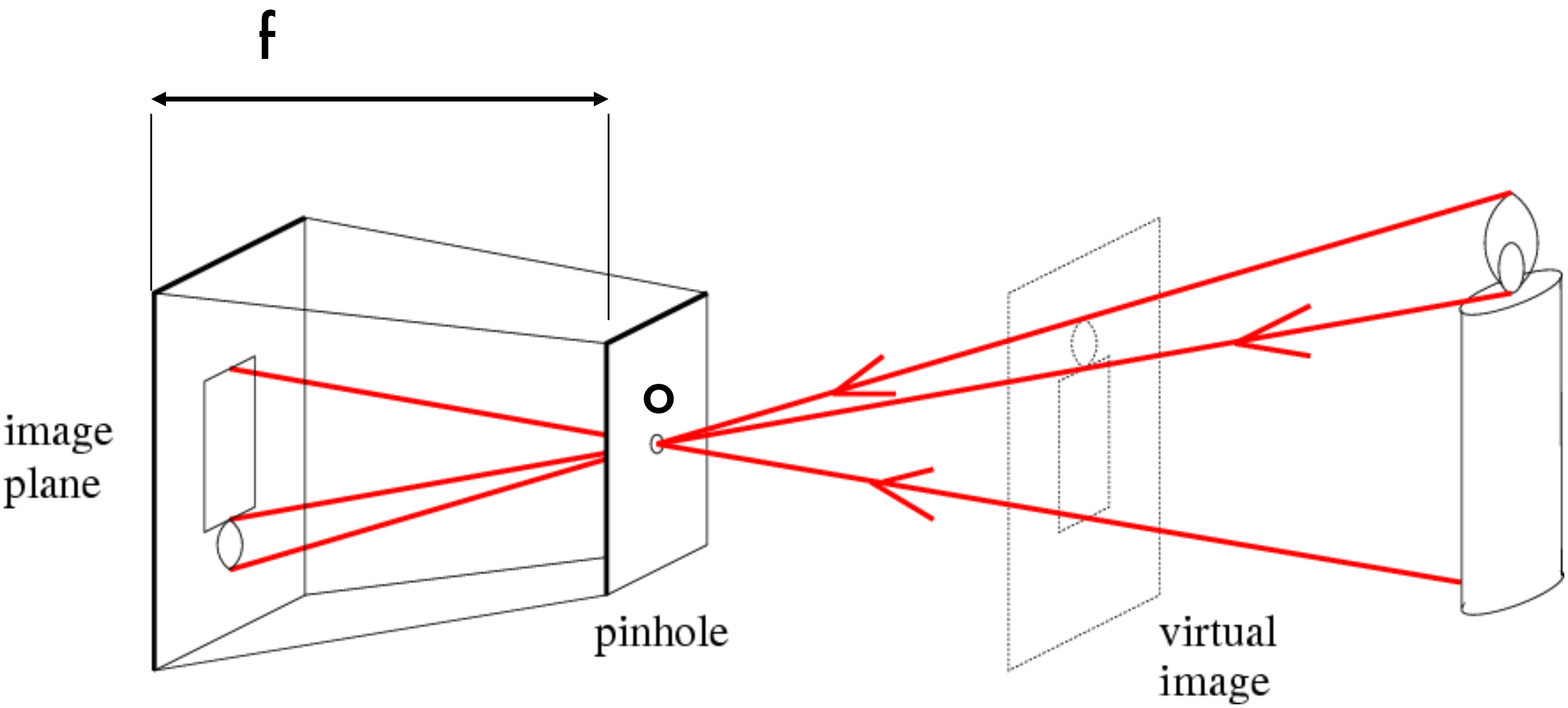
Agenda

- Pinhole cameras
- Cameras & lenses
- The geometry of pinhole cameras

Agenda

- **Pinhole cameras**
- Cameras & lenses
- The geometry of pinhole cameras

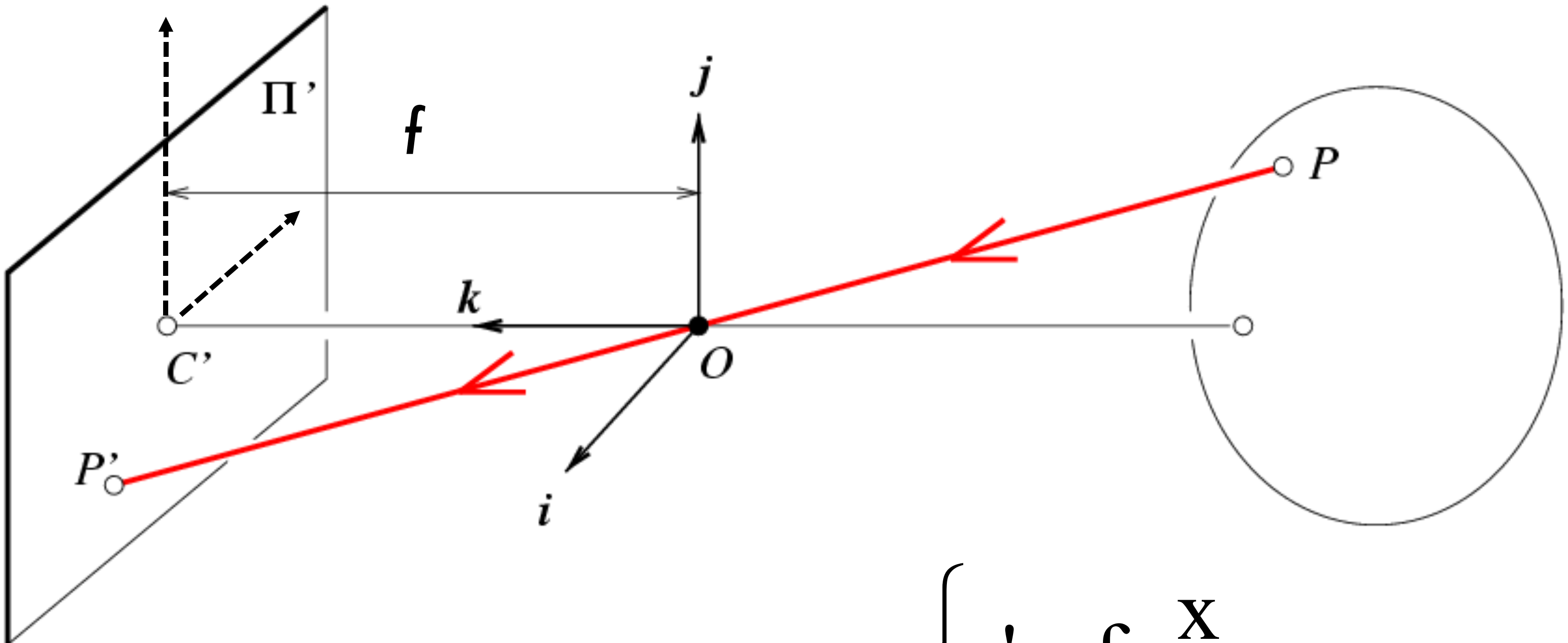
Pinhole camera



f = focal length

o = aperture = pinhole = center of the camera

Pinhole camera

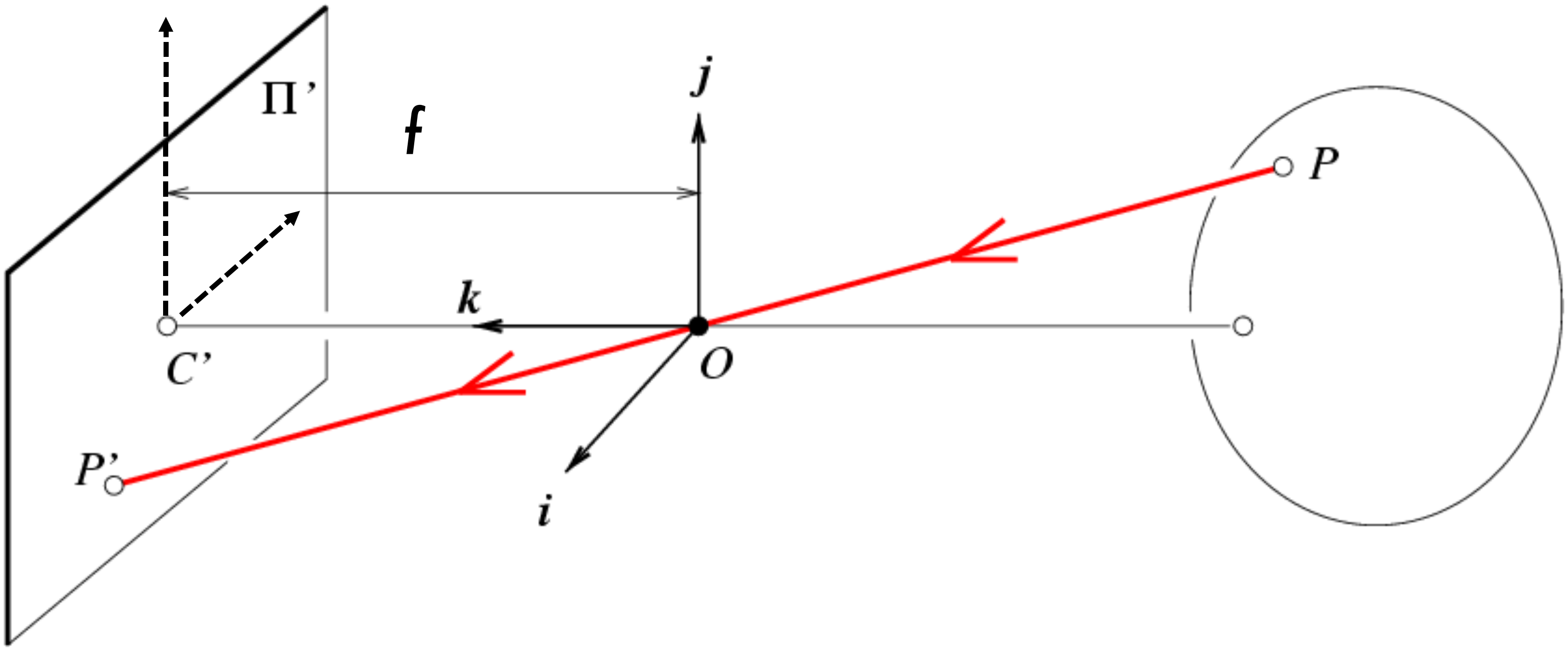


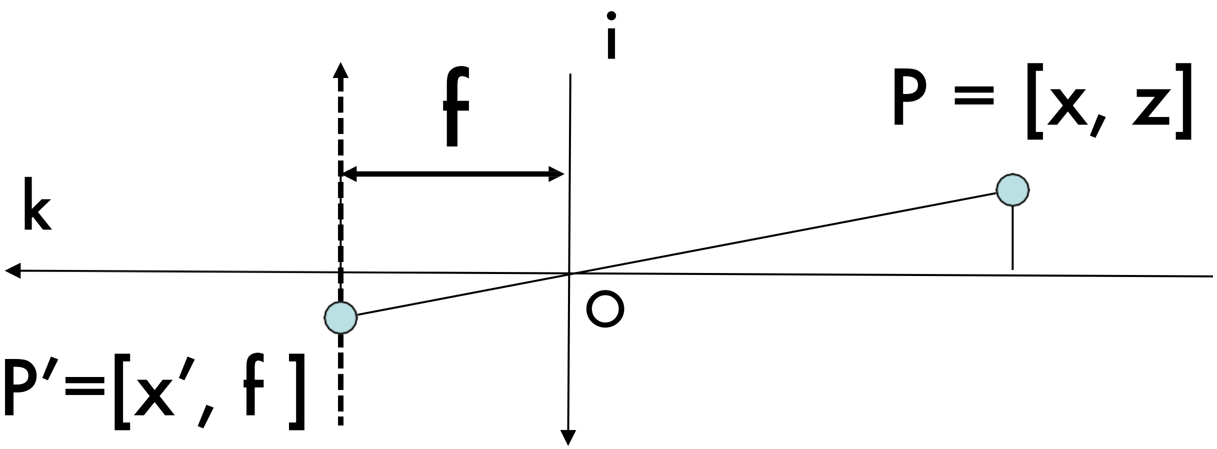
$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{cases} x' = f \dfrac{x}{z} \\ y' = f \dfrac{y}{z} \end{cases} \quad \text{[Eq. 1]}$$

Derived using similar triangles

Pinhole camera



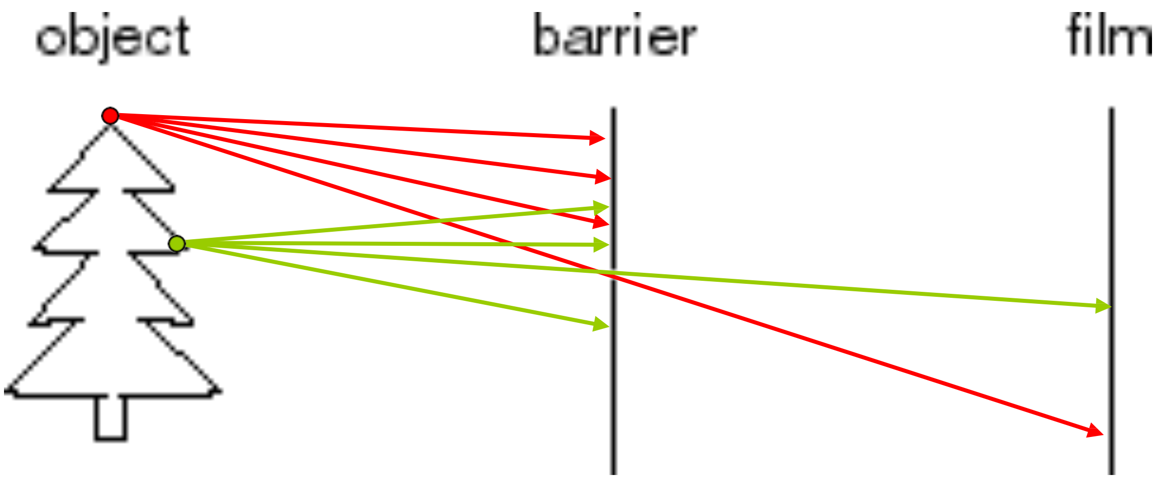


[Eq. 2]

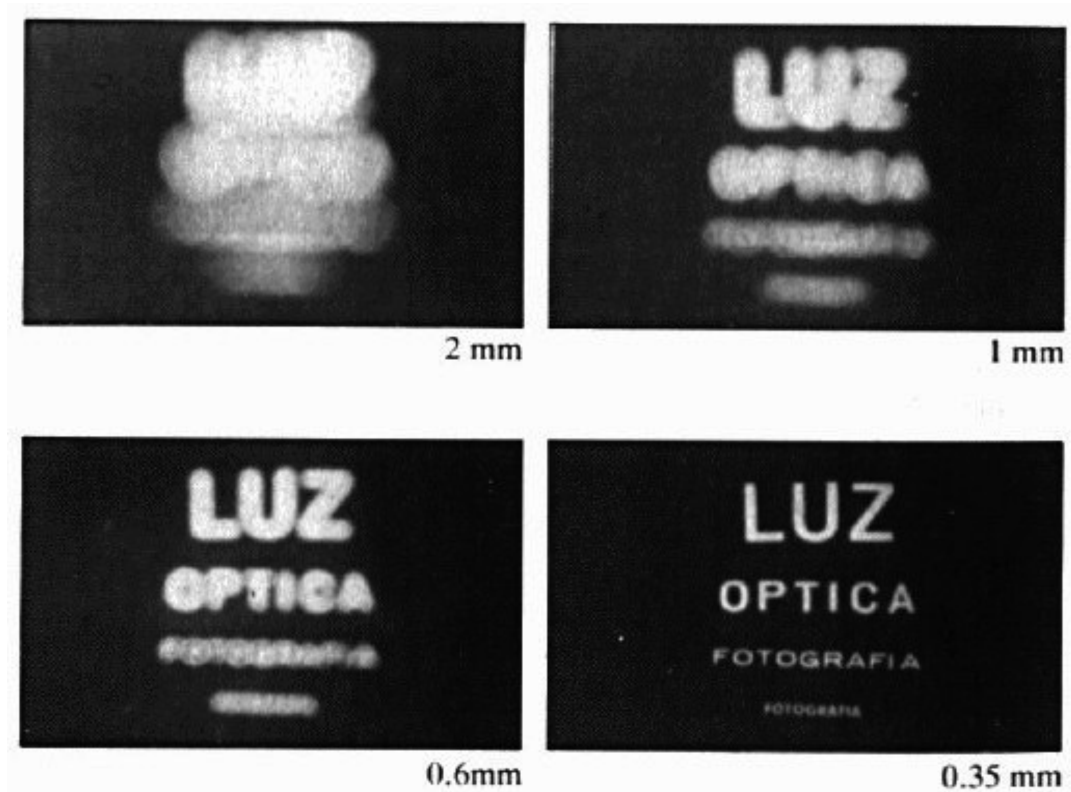
$$\frac{x'}{f} = \frac{x}{z}$$

Pinhole camera

Is the size of the aperture important?



Shrinking aperture size



-What happens if the aperture is too small?

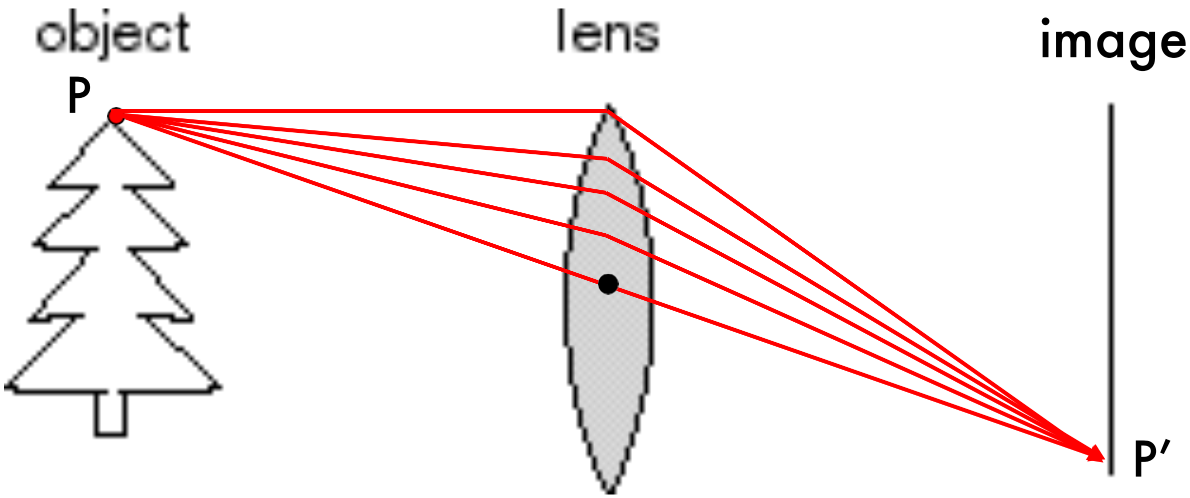
-Less light passes through

Adding lenses!

Agenda

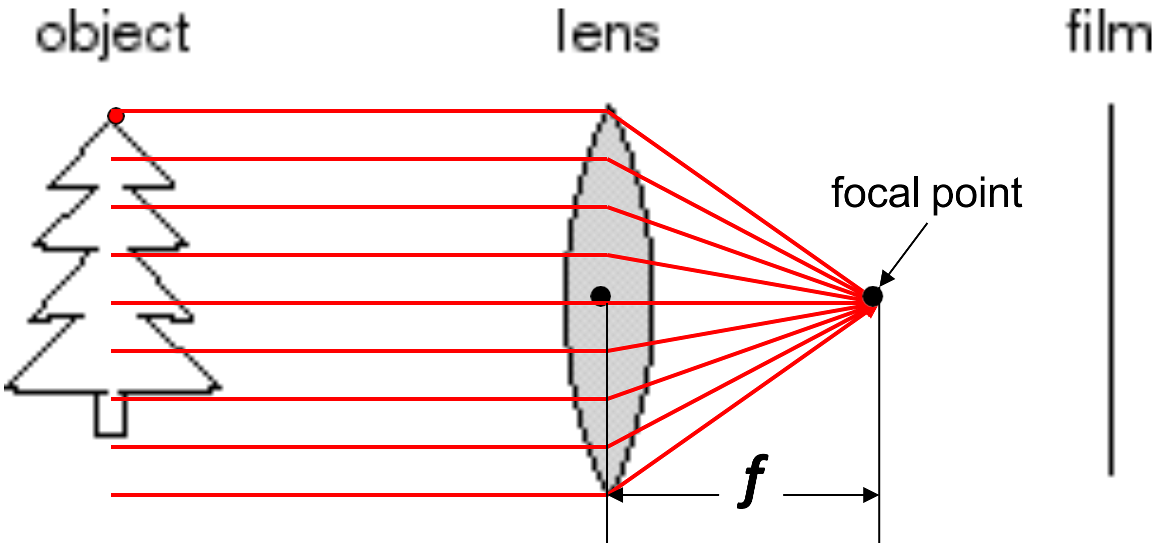
- Pinhole cameras
- **Cameras & lenses**
- The geometry of pinhole cameras

Cameras & Lenses



- A lens focuses light onto the film

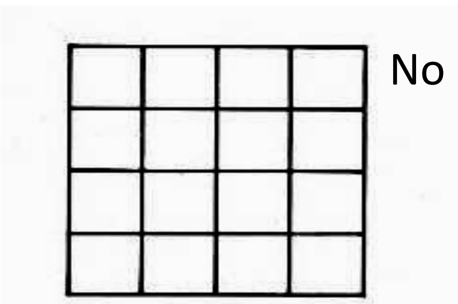
Cameras & Lenses



- A lens focuses light onto the film
  - All rays parallel to the optical (or principal) axis converge to one point (the *focal point*) on a plane located at the *focal length f* from the center of the lens.
  - Rays passing through the center are not deviated

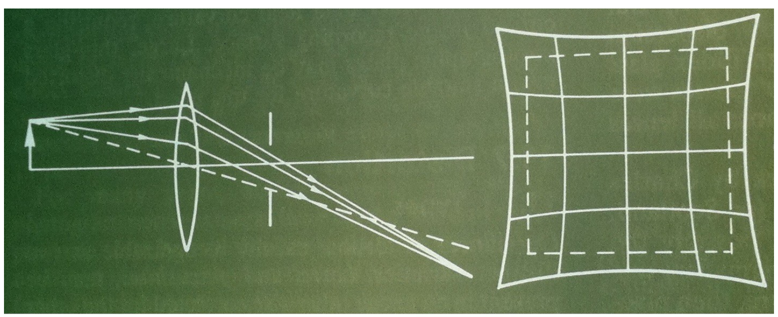
Issues with lenses: Radial Distortion

– Deviations are most noticeable for rays that pass through the edge of the lens



No distortion

Pin cushion



Barrel (fisheye lens)

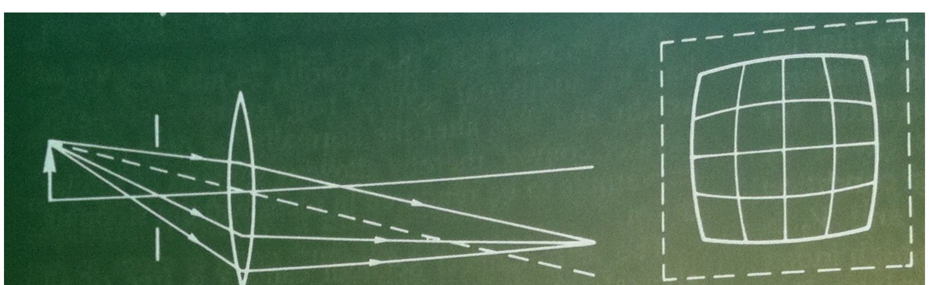
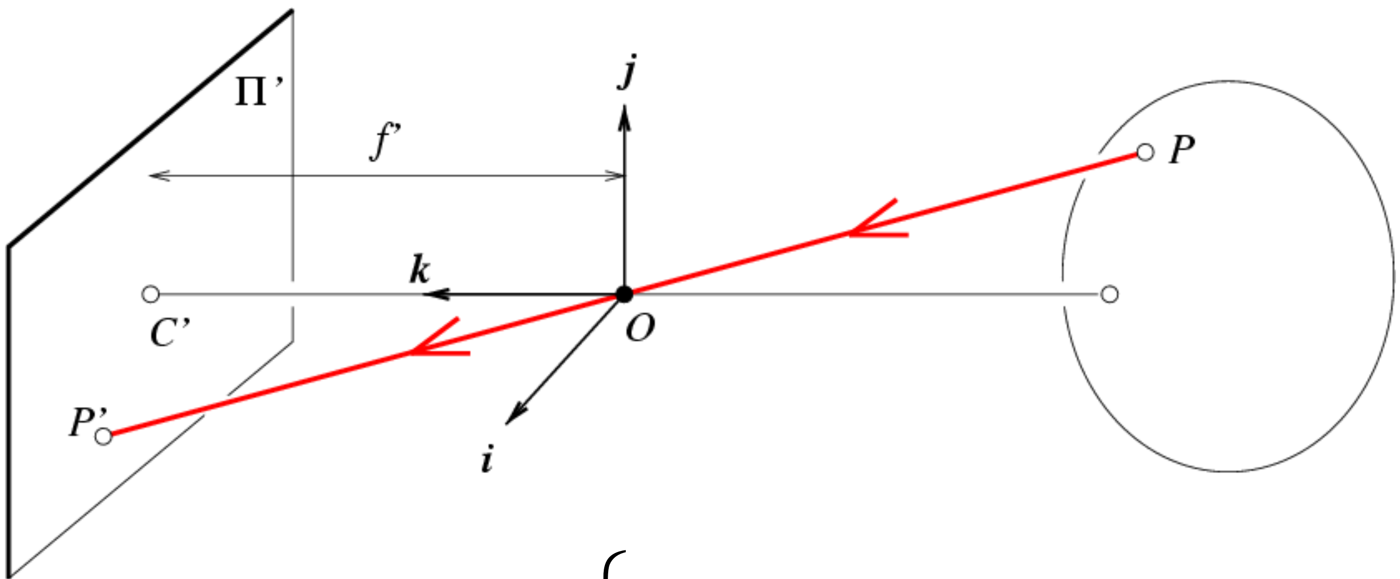


Image magnification decreases with distance from the optical axis

Agenda

- Pinhole cameras
- Cameras & lenses
- **The geometry of pinhole cameras**
  - Intrinsic
  - Extrinsic

Pinhole camera



$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{cases} x' = f \dfrac{x}{z} \\ y' = f \dfrac{y}{z} \end{cases}$$

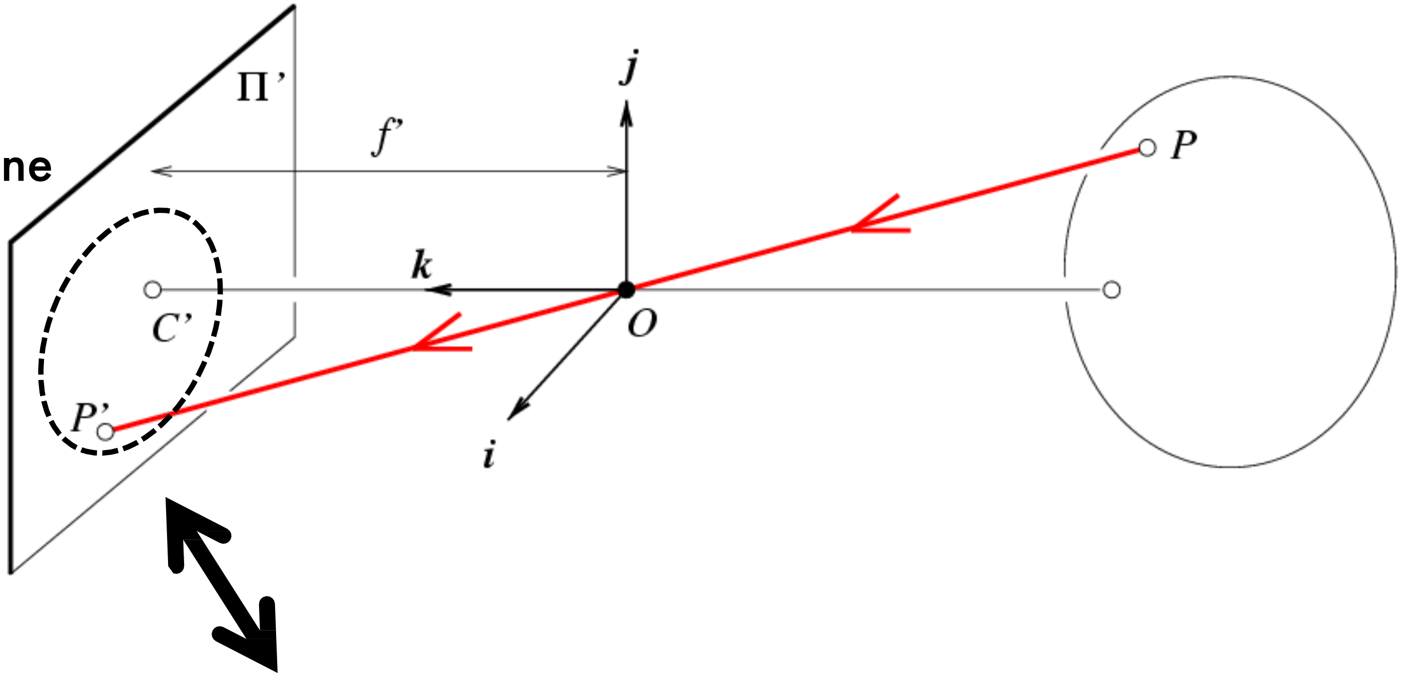
$$\Re^3 \rightarrow \Re^2$$

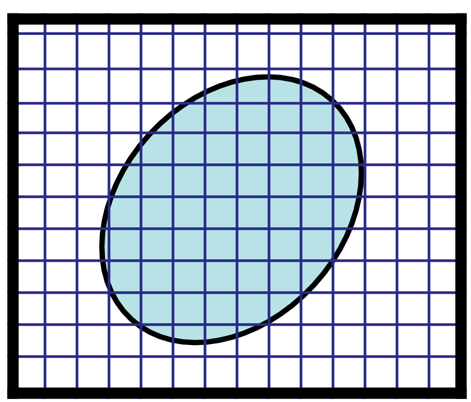
[Eq. 1]

f = focal length
o = center of the camera

From retina plane to images

Retina plane

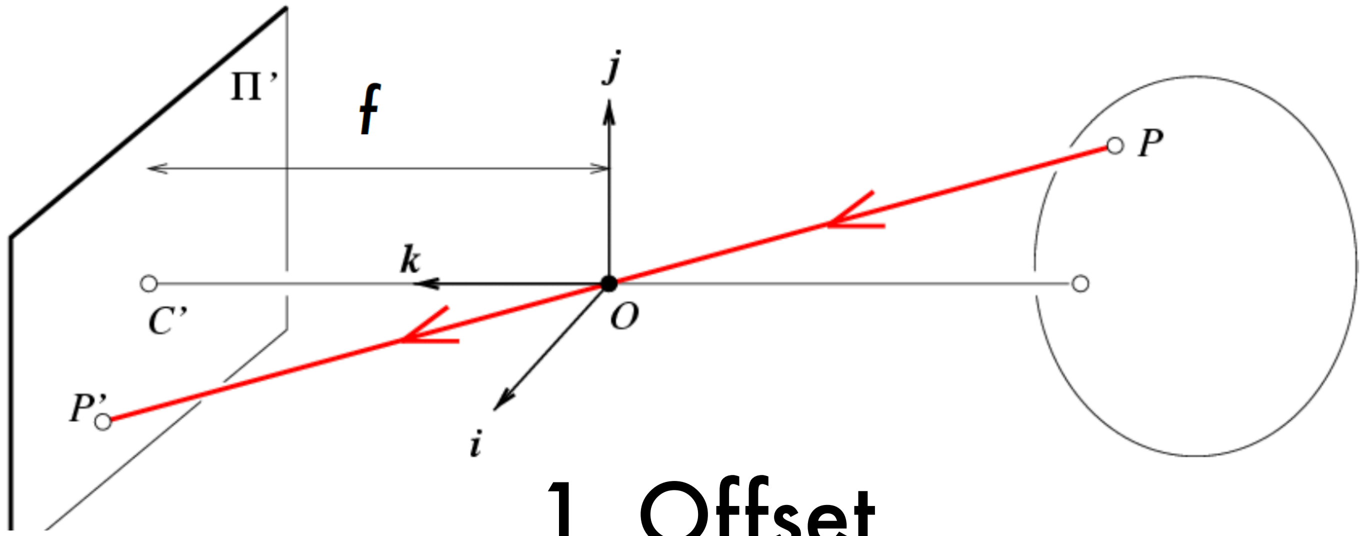




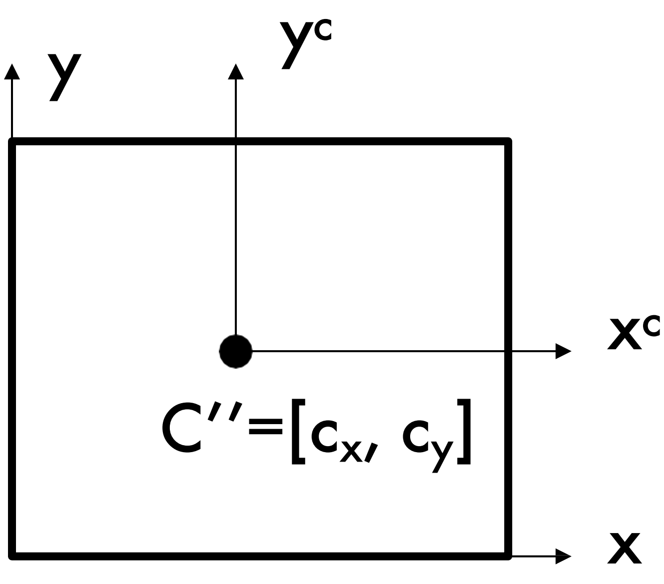
Digital image

Pixels, bottom-left coordinate systems

Coordinate systems



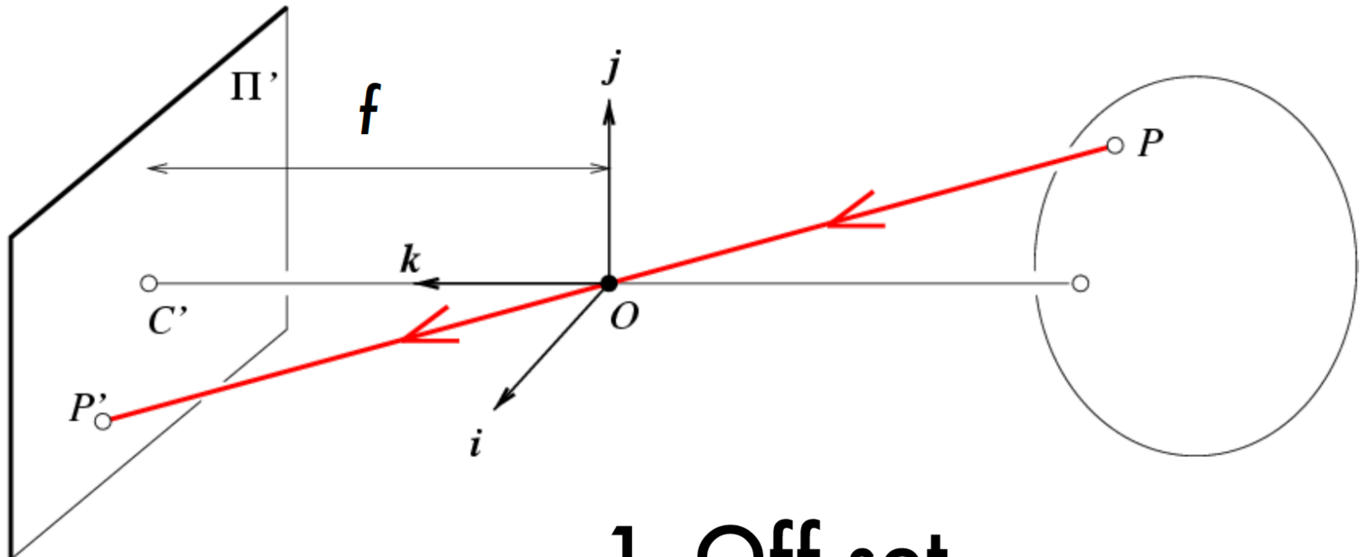
1. Offset



$$(x, y, z) \rightarrow (f\frac{x}{z} + c_x, f\frac{y}{z} + c_y)$$

[Eq. 5]

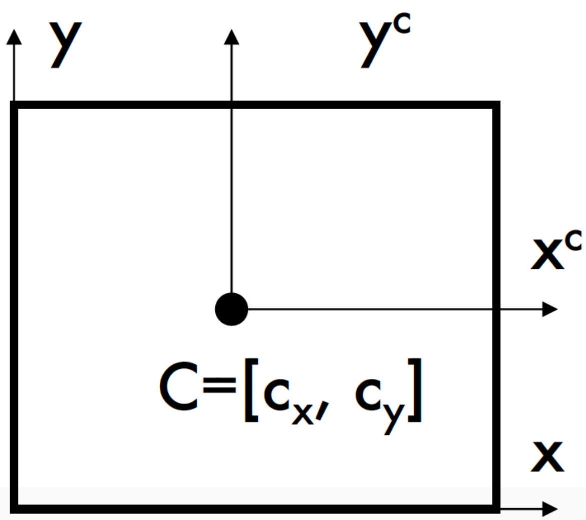
Converting to pixels



1. Off set
2. From metric to pixels

$$(x, y, z) \rightarrow (\underbrace{f\,k}_{\alpha}\frac{x}{z} + c_x, \underbrace{f\,l}_{\beta}\frac{y}{z} + c_y)$$

[Eq. 6]



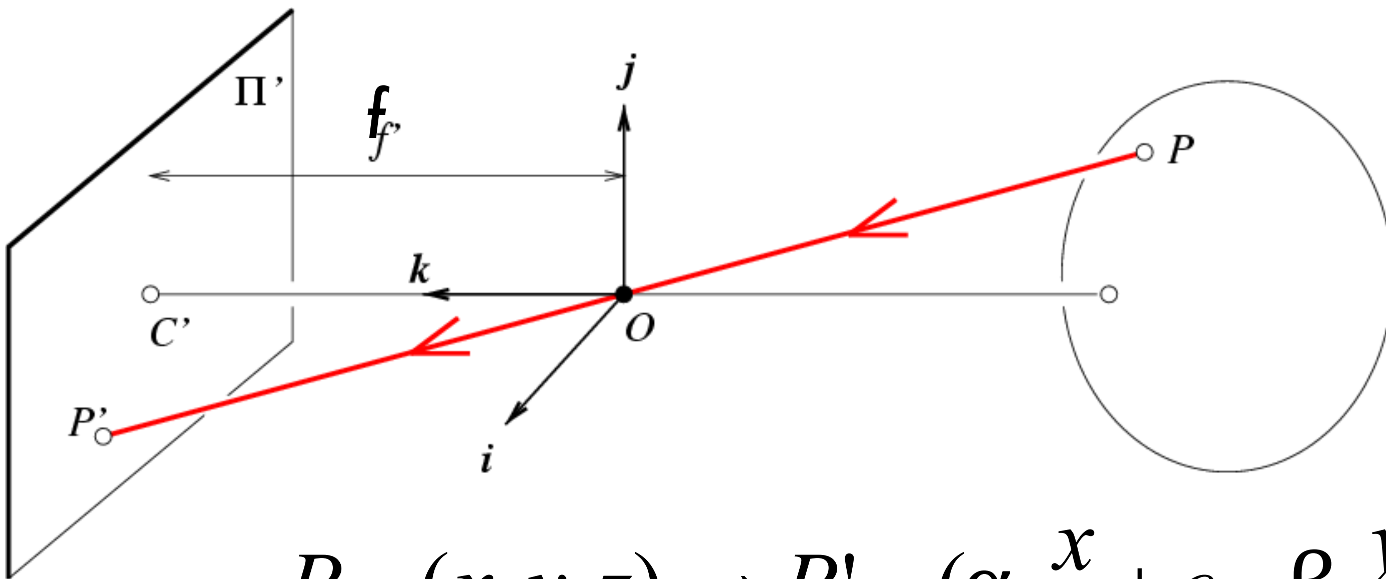
Units: k,l : pixel/m

f : m

Non-square pixels

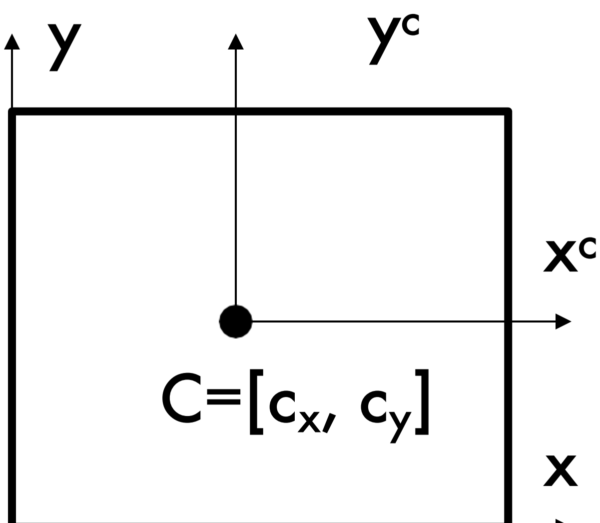
α, β : pixel

Is this projective transformation linear?



$$P = (x, y, z) \rightarrow P' = \left(\alpha \frac{x}{z} + c_x,\ \beta \frac{y}{z} + c_y\right)$$

[Eq. 7]



- Is this a linear transformation?

  No — division by z is nonlinear

- Can we express it in a matrix form?

Homogeneous coordinates

E→H

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

homogeneous image coordinates

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

homogeneous scene coordinates

- Converting back *from* homogeneous coordinates

H→E

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

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

Projective transformation in the homogenous coordinate system

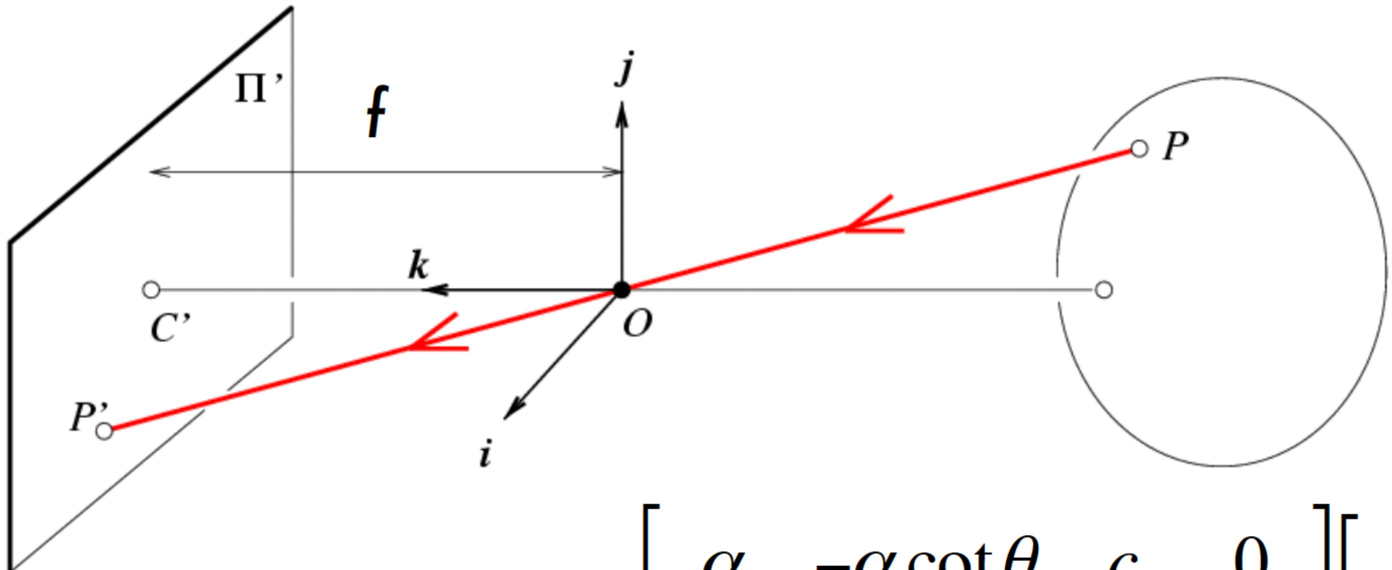
$$P_h' = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{P_h}$$

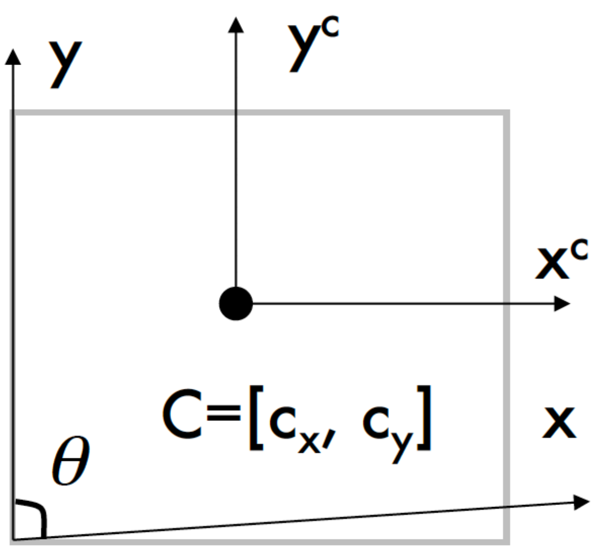
[Eq.8]

$$M = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\overbrace{P_h'}^{\text{Homogenous}} \rightarrow \overbrace{P' = (\alpha \frac{x}{z} + c_x, \beta \frac{y}{z} + c_y)}^{\text{Euclidian}}$$

Camera Skewness

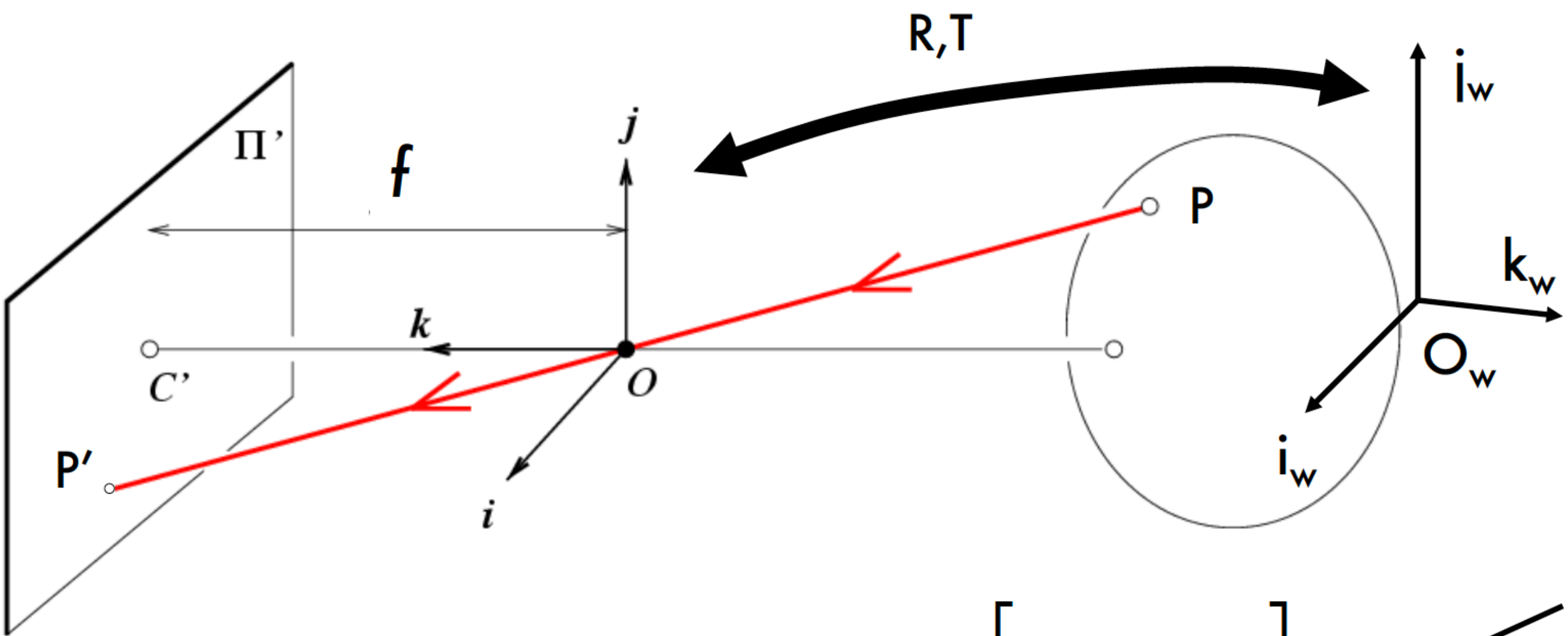




$$P' = \begin{bmatrix} \alpha & -\alpha\cot\theta & c_x & 0 \\ 0 & \dfrac{\beta}{\sin\theta} & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

How many degree does K have?
5 degrees of freedom!

World reference system



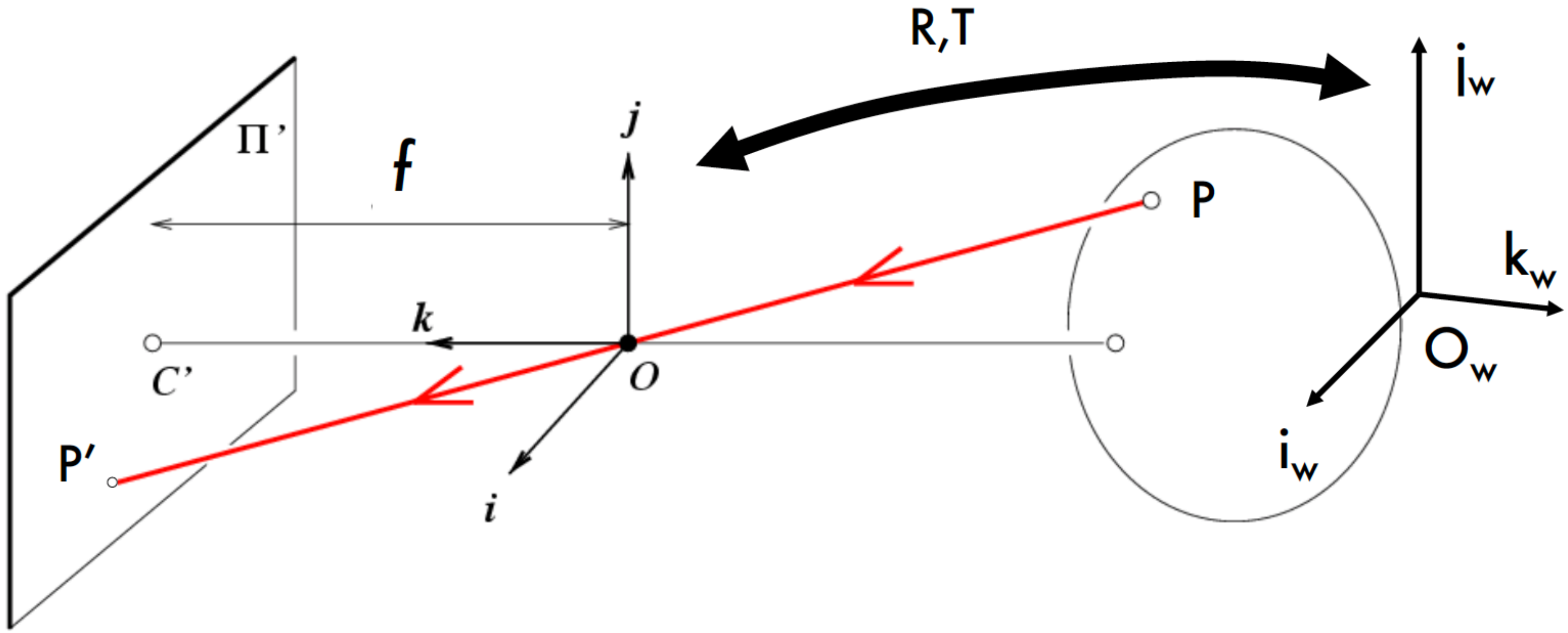
In 4D homogeneous coordinates: $P = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}_{4\times 4} P_w$ $\quad P_w = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$

intrinsic **parameters** extrinsic **parameters**

$$P' = K\begin{bmatrix} I & 0 \end{bmatrix} P = K\begin{bmatrix} I & 0 \end{bmatrix}\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}_{4\times 4} P_w = \underbrace{\boxed{K\begin{bmatrix} R & T \end{bmatrix}}}_{M} P_w$$

[Eq.11]

The projective transformation



$$P'_{3\times1} = M_{3x4}\,P_w = K_{3\times3}\begin{bmatrix} R & T \end{bmatrix}_{3\times4} P_{w\,4\times1}$$

How many degrees of freedom?

5 + 3 + 3 =11!

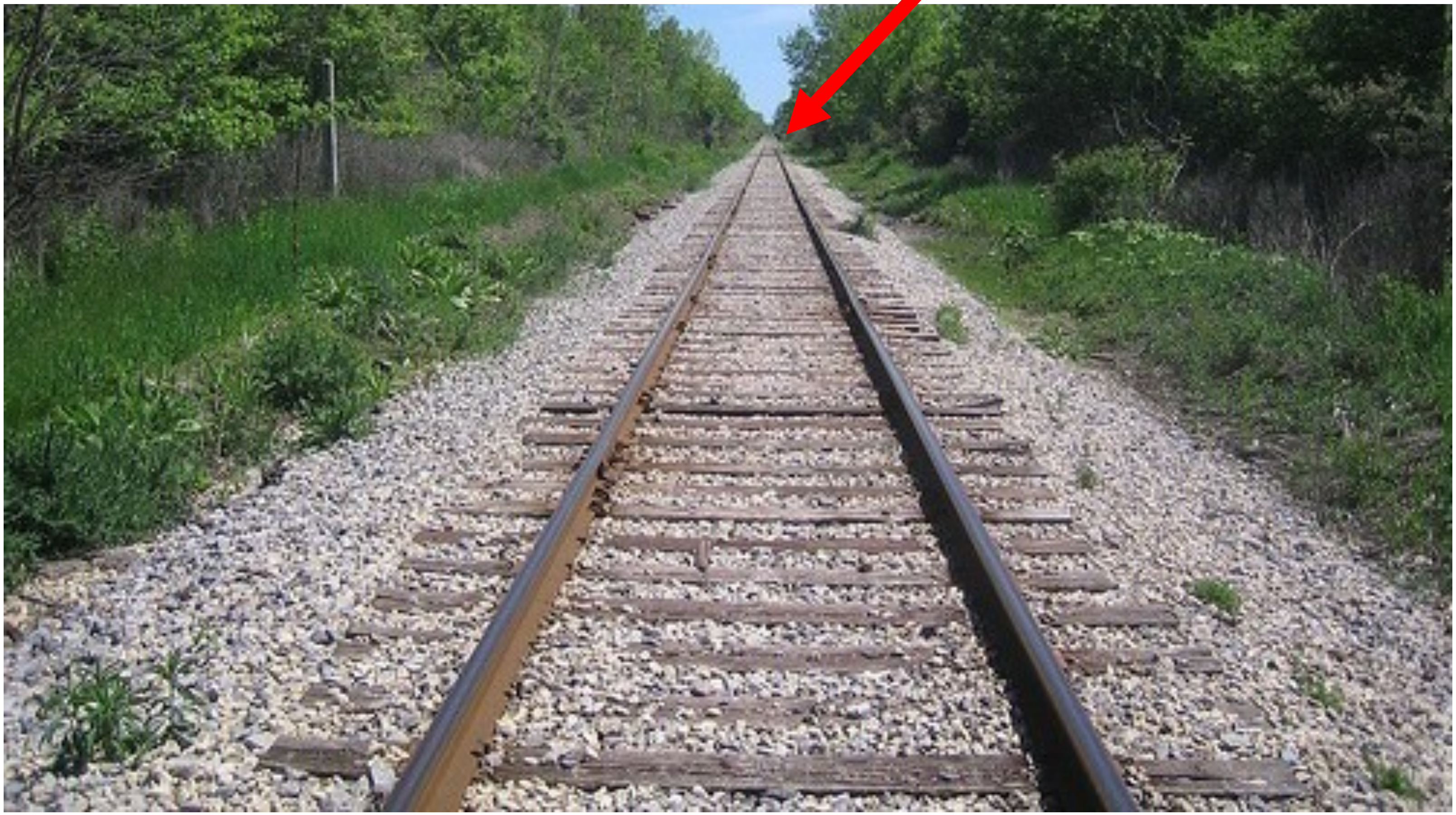
Properties of projective transformations

- Points project to points
- line project to lines, rays or degenerate into points
- Distant objects look smaller

Properties of Projection

- Angles are not preserved
- Parallel lines meet (except for horizontal lines)

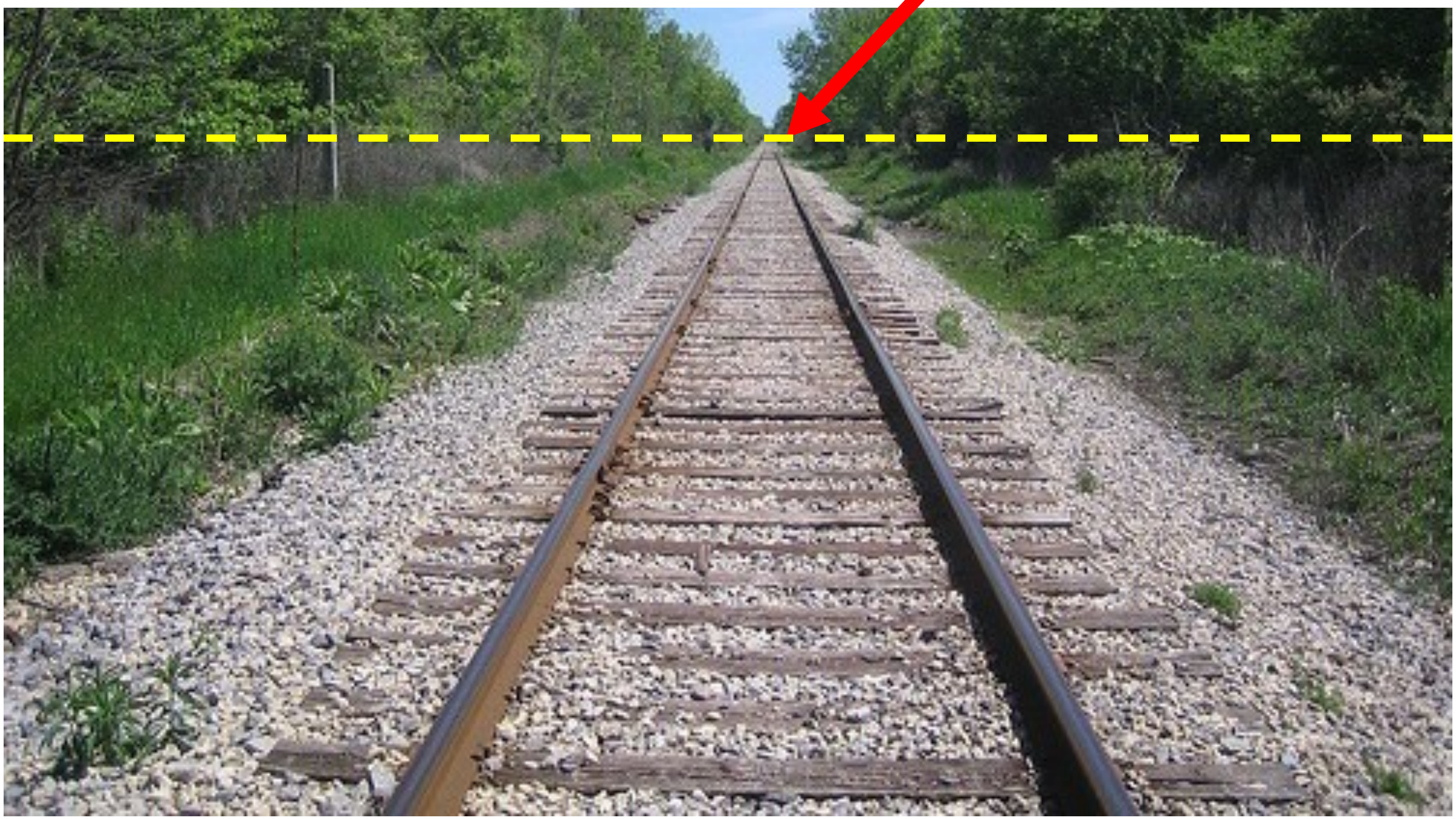
Parallel lines in the world intersect in the image at a "vanishing point"



Horizon line (vanishing line)

- Angles are not preserved
- Parallel lines meet (except for horizontal lines)

Parallel lines in the world intersect in the image at a "vanishing point"



Horizon line (vanishing line)