计算机视觉

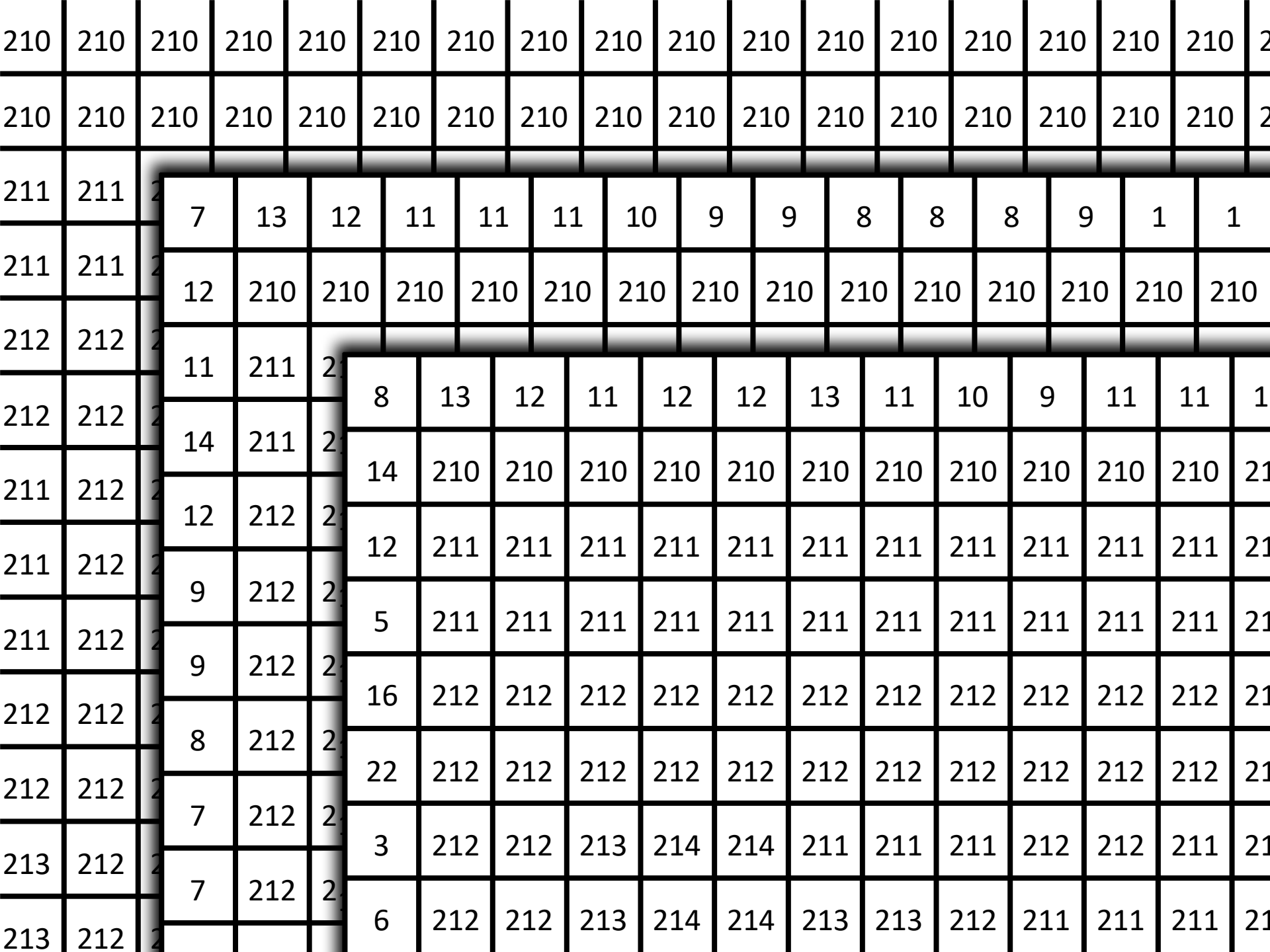
图像形成 上

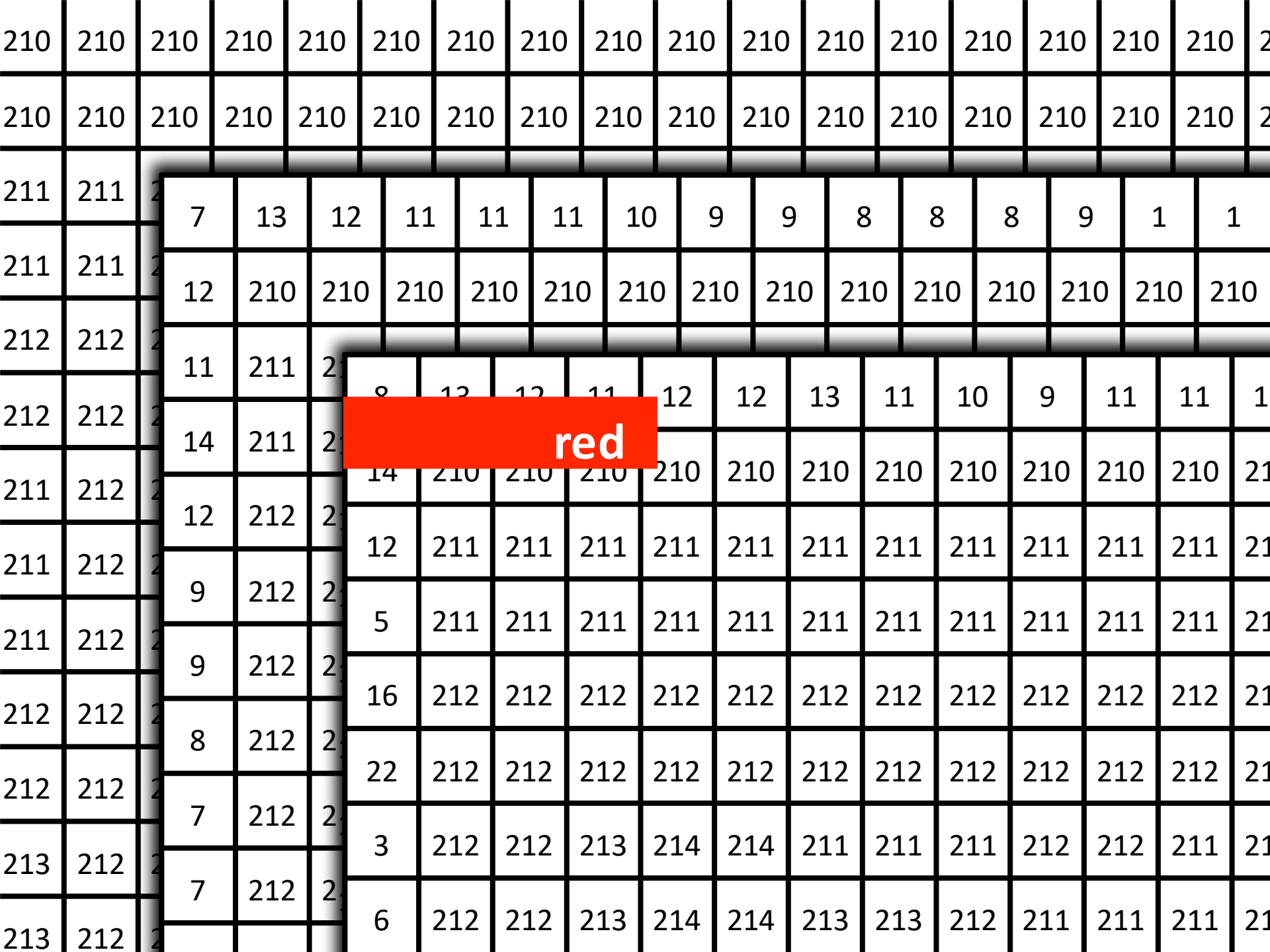
中国传媒大学
COMMUNICATION UNIVERSITY OF CHINA

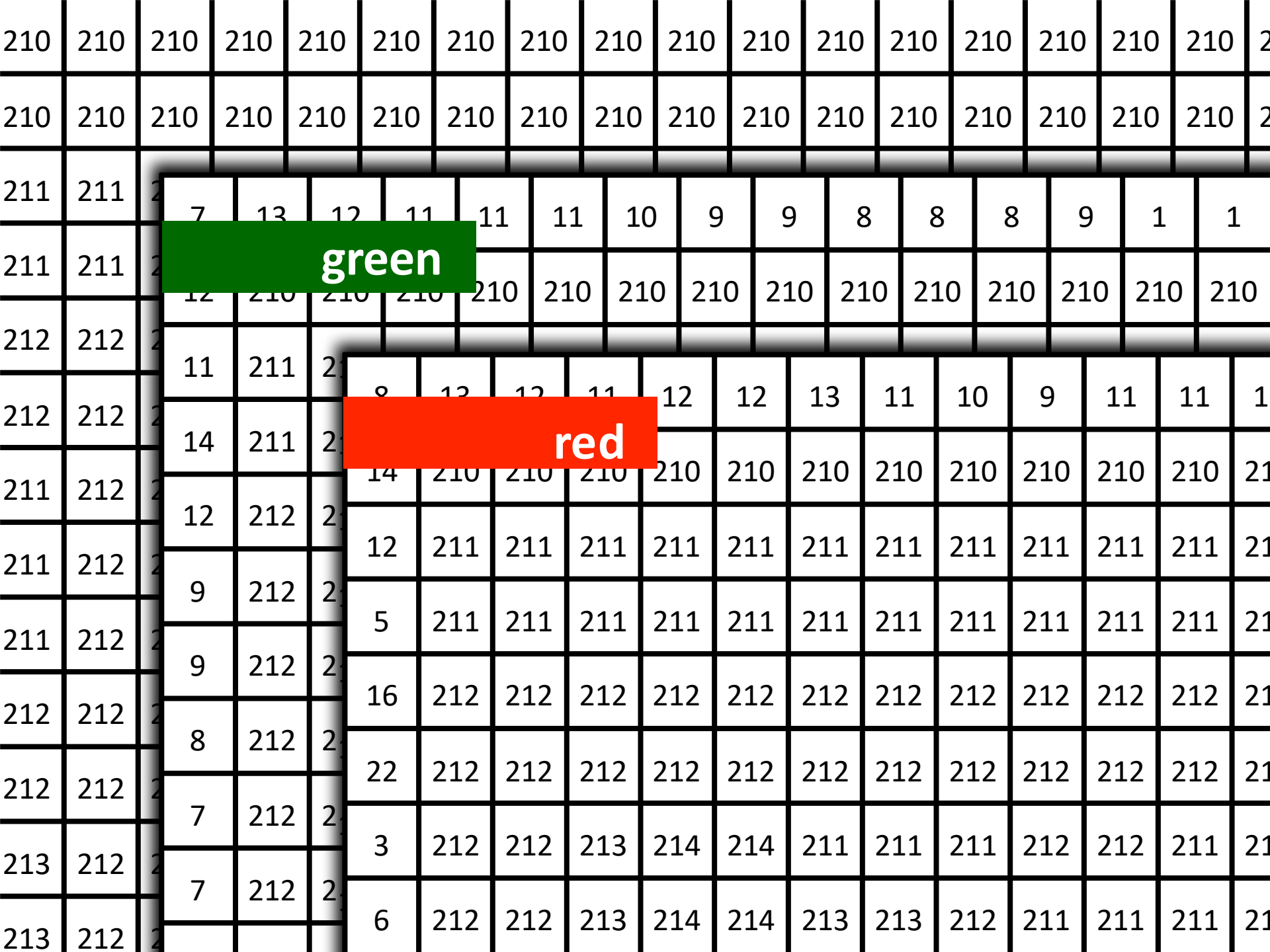
什么是图像？

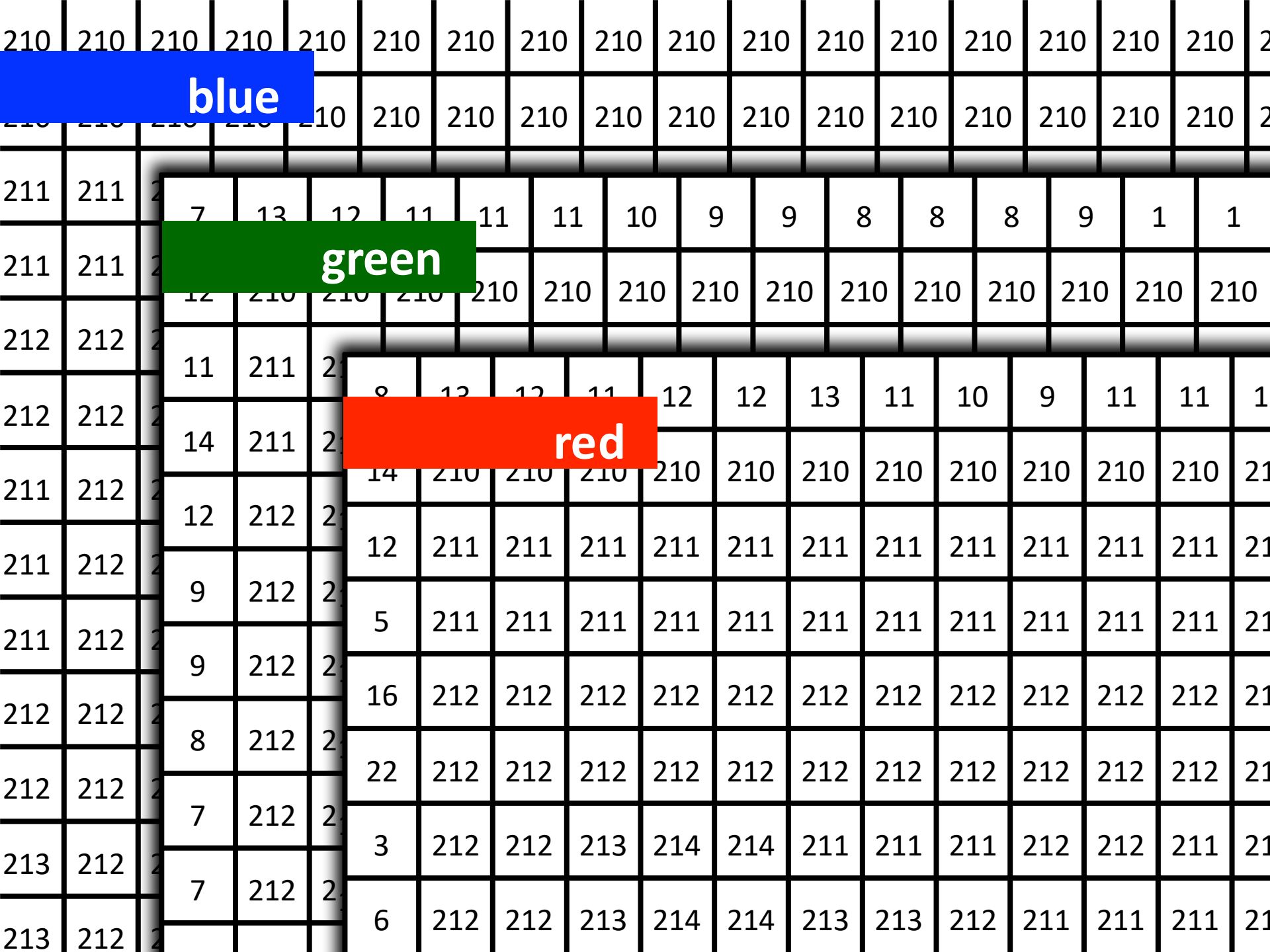

OFF
ON
AF-ON
RAW
JPEG
START
STOP
MENU
INFO.
SET
LOCK
Canon


OFF
ON
RAW
JPEG
MENU
INFO.
START
STOP
AF-ON
SET
LOCK
Canon


210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 2
210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 210 2
211 211 2
211 211 2
212 212 2
212 212 2
211 212 2
211 212 2
211 212 2
212 212 2
212 212 2
213 212 2
213 212 2
7 13 12 11 11 11 10 9 9 8 8 8 9 1 1
12 210 210 210 210 210 210 210 210 210 210 210 210 210 210
11 211 2
14 211 2
12 212 2
9 212 2
9 212 2
8 212 2
7 212 2
7 212 2
8 13 12 11 12 12 13 11 10 9 11 11 1
14 210 210 210 210 210 210 210 210 210 210 210 21
12 211 211 211 211 211 211 211 211 211 211 211 21
5 211 211 211 211 211 211 211 211 211 211 211 21
16 212 212 212 212 212 212 212 212 212 212 212 21
22 212 212 212 212 212 212 212 212 212 212 212 21
3 212 212 213 214 214 211 211 211 212 212 211 21
6 212 212 213 214 214 213 213 212 211 211 211 21


red


green
red


blue
green
red


blue
green
red
图像
≠
理解


约瑟夫·尼塞福尔·涅普斯于1826年

坦普尔大街街景，路易·达盖尔于1838年

微信朋友圈每天上传图片10亿张

针孔相机

光敏记录面

物体

针孔相机

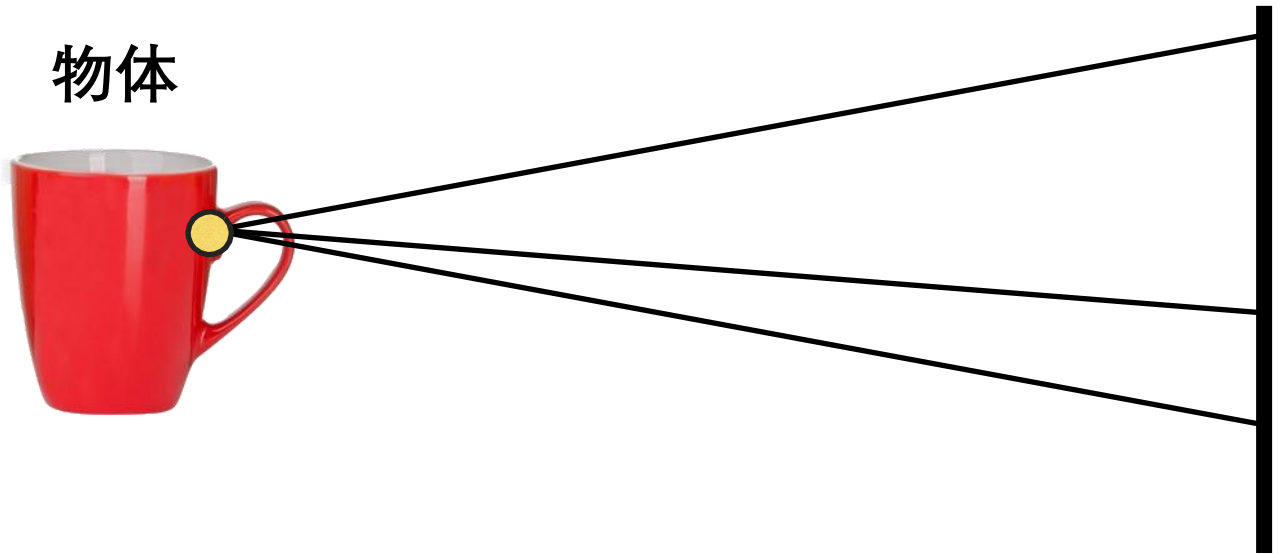
光敏记录面

物体

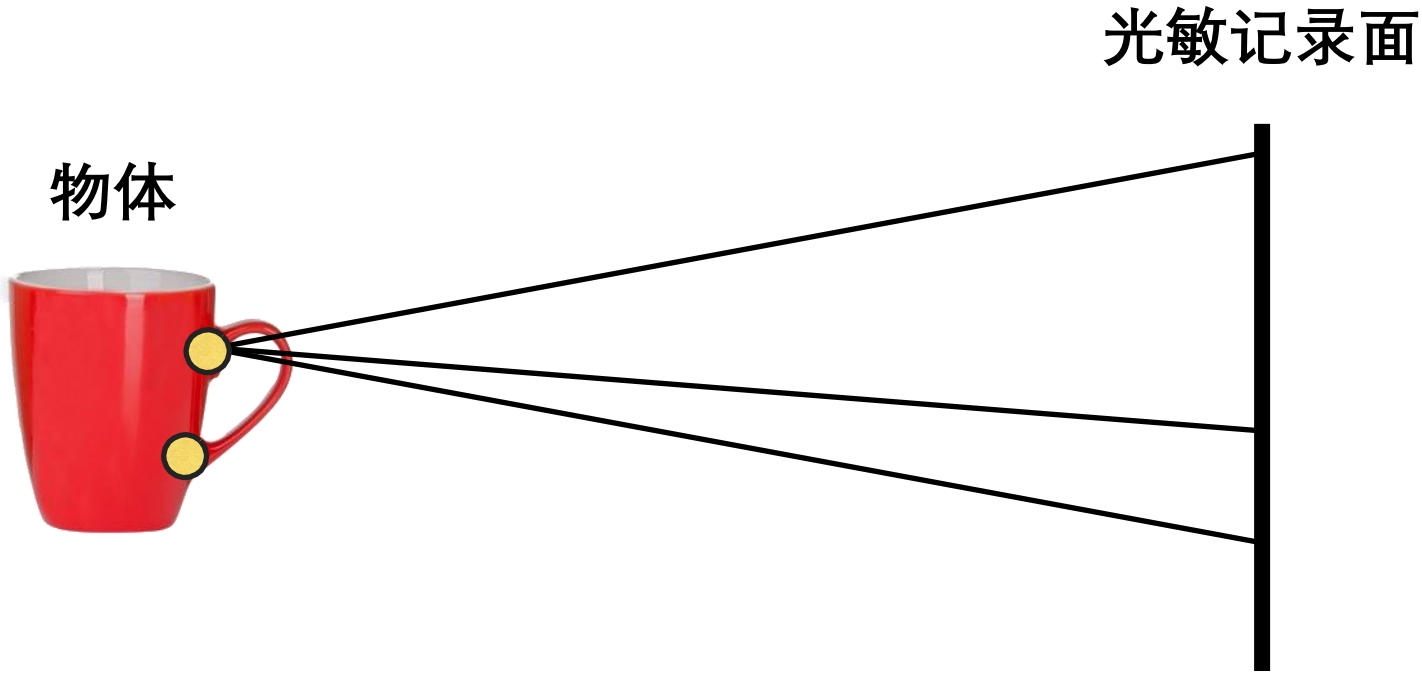
针孔相机

光敏记录面

物体



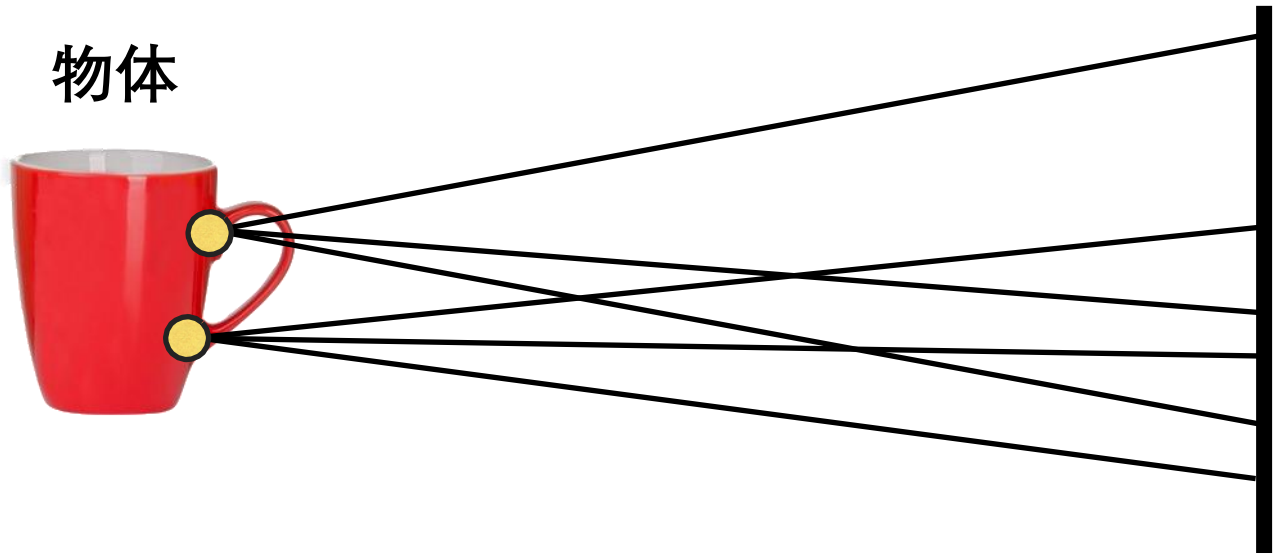
针孔相机



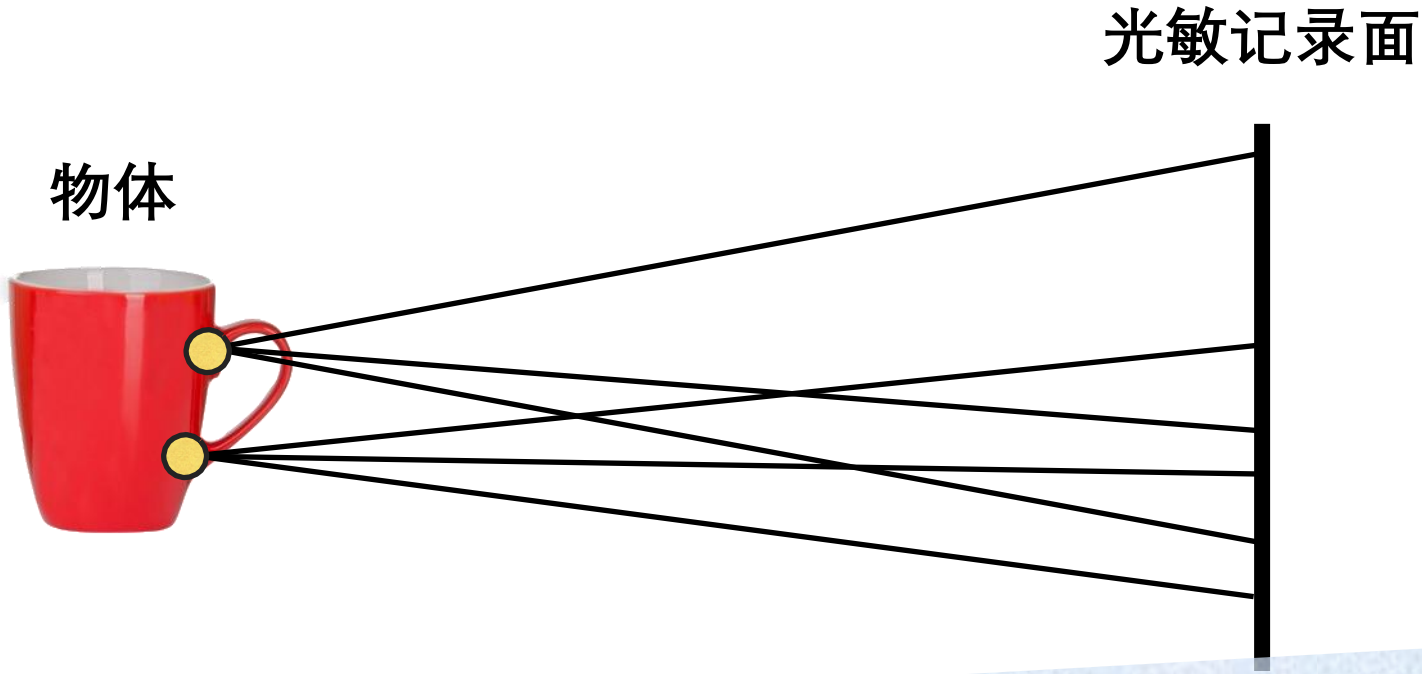
针孔相机

光敏记录面

物体



针孔相机

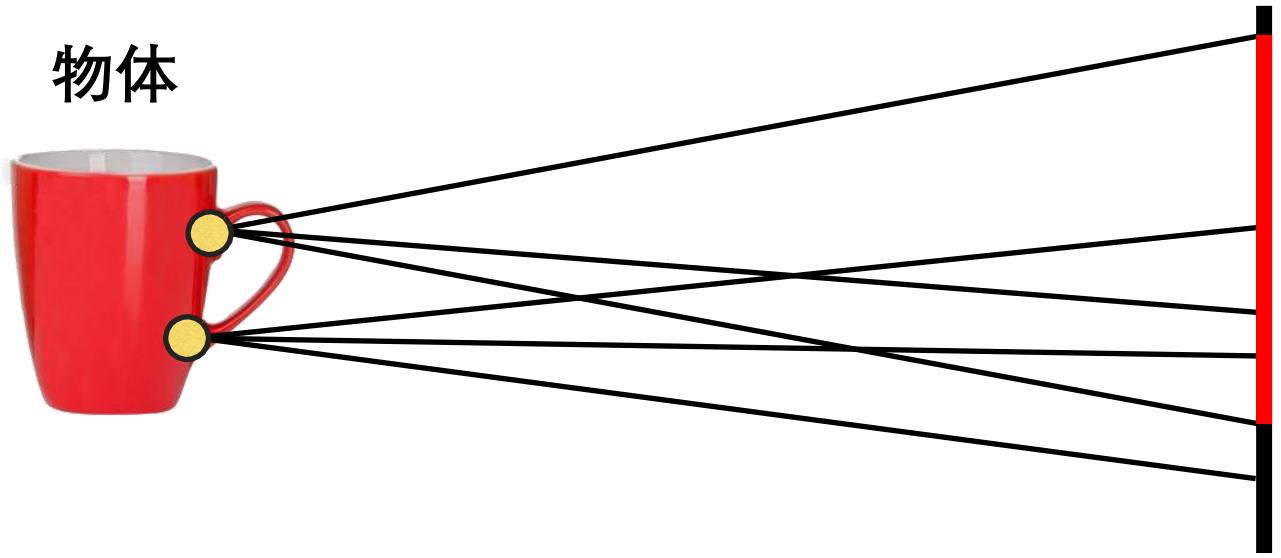


我们能得到一个合理的图像吗？

针孔相机

光敏记录面

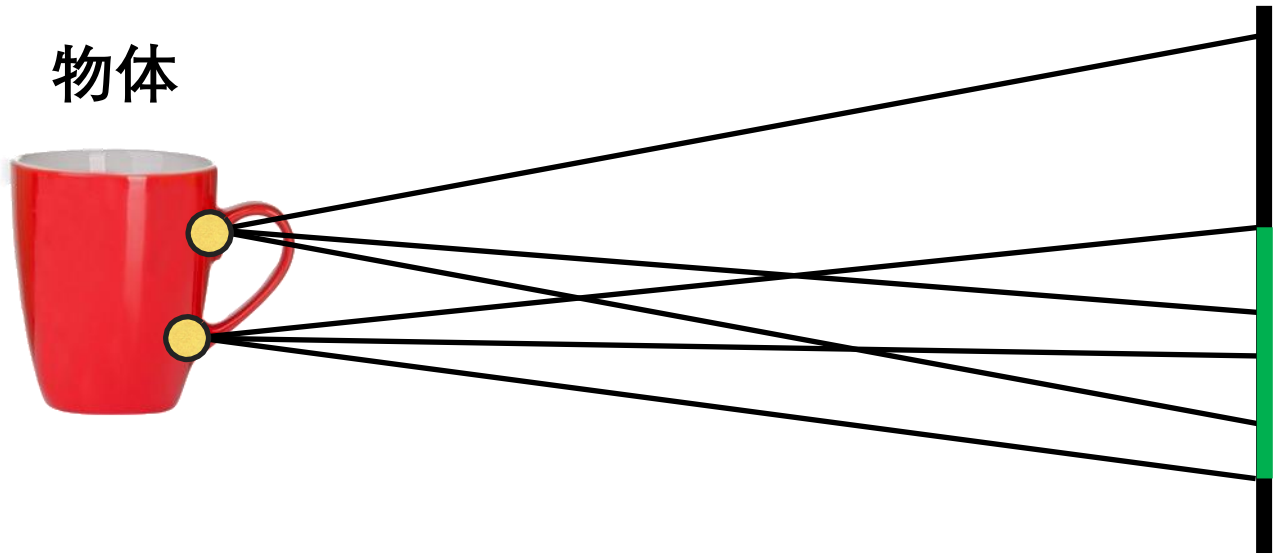
物体



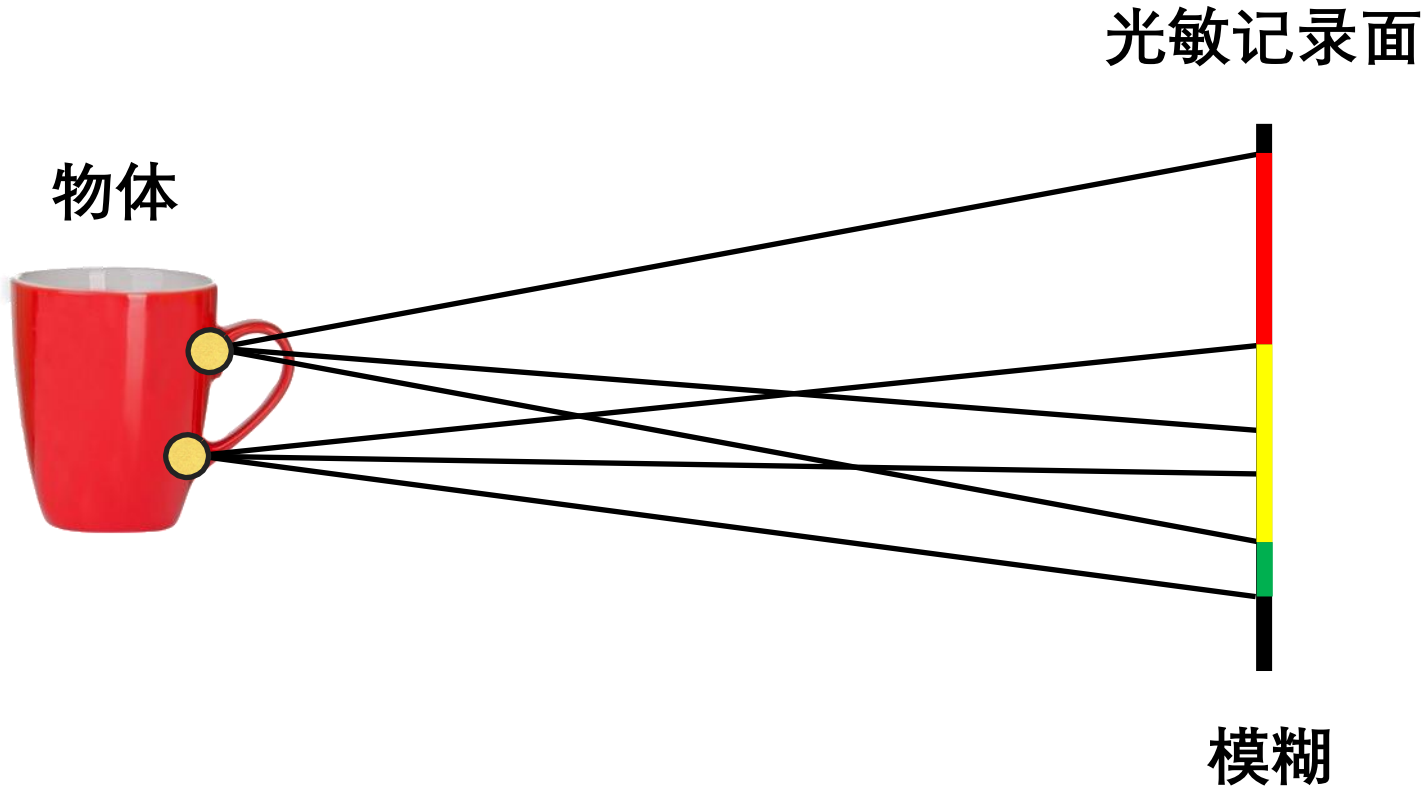
针孔相机

光敏记录面

物体



针孔相机



针孔相机

光敏记录面

物体

针孔相机

光敏记录面

物体

屏障

开口称为孔径

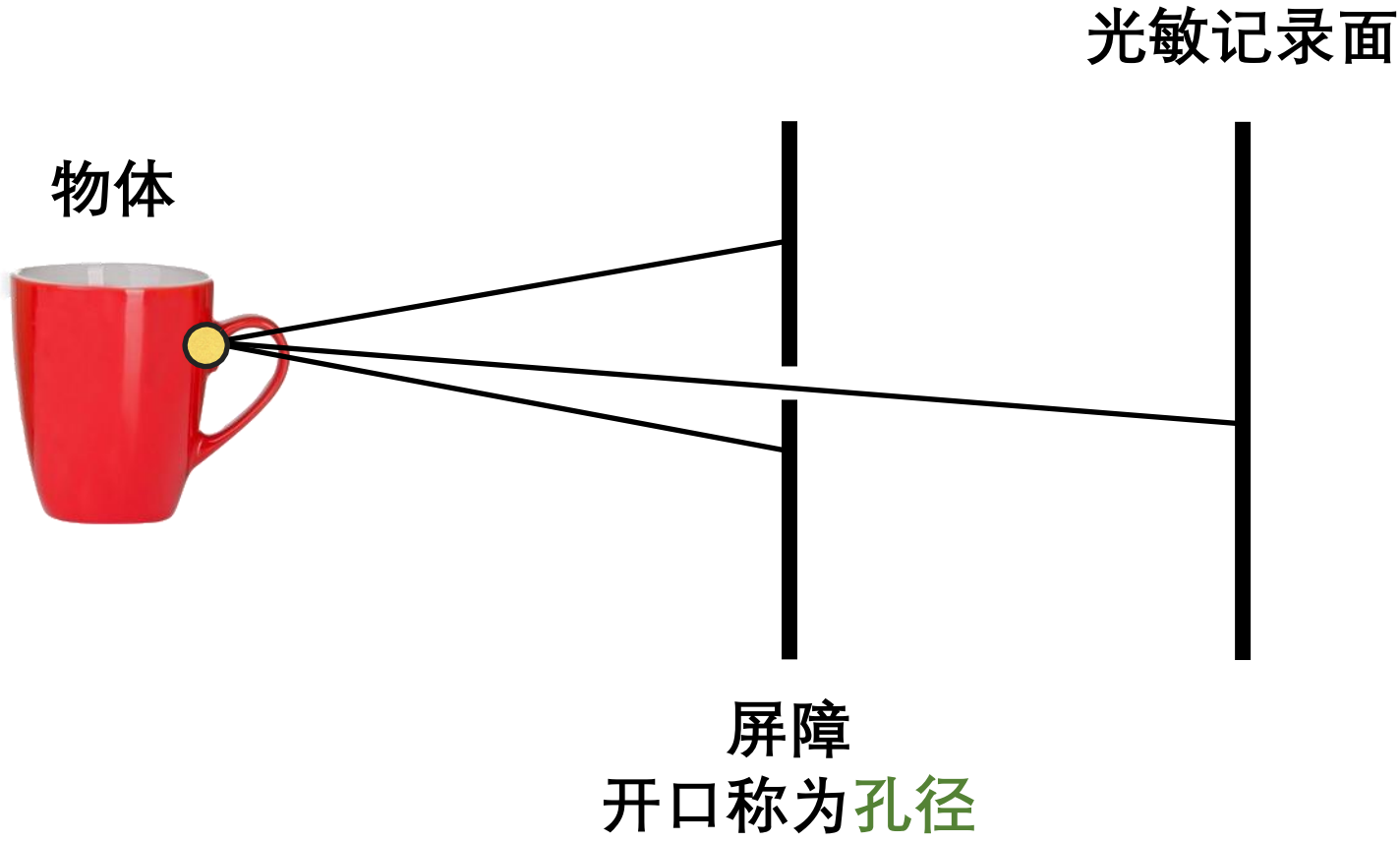
针孔相机

光敏记录面

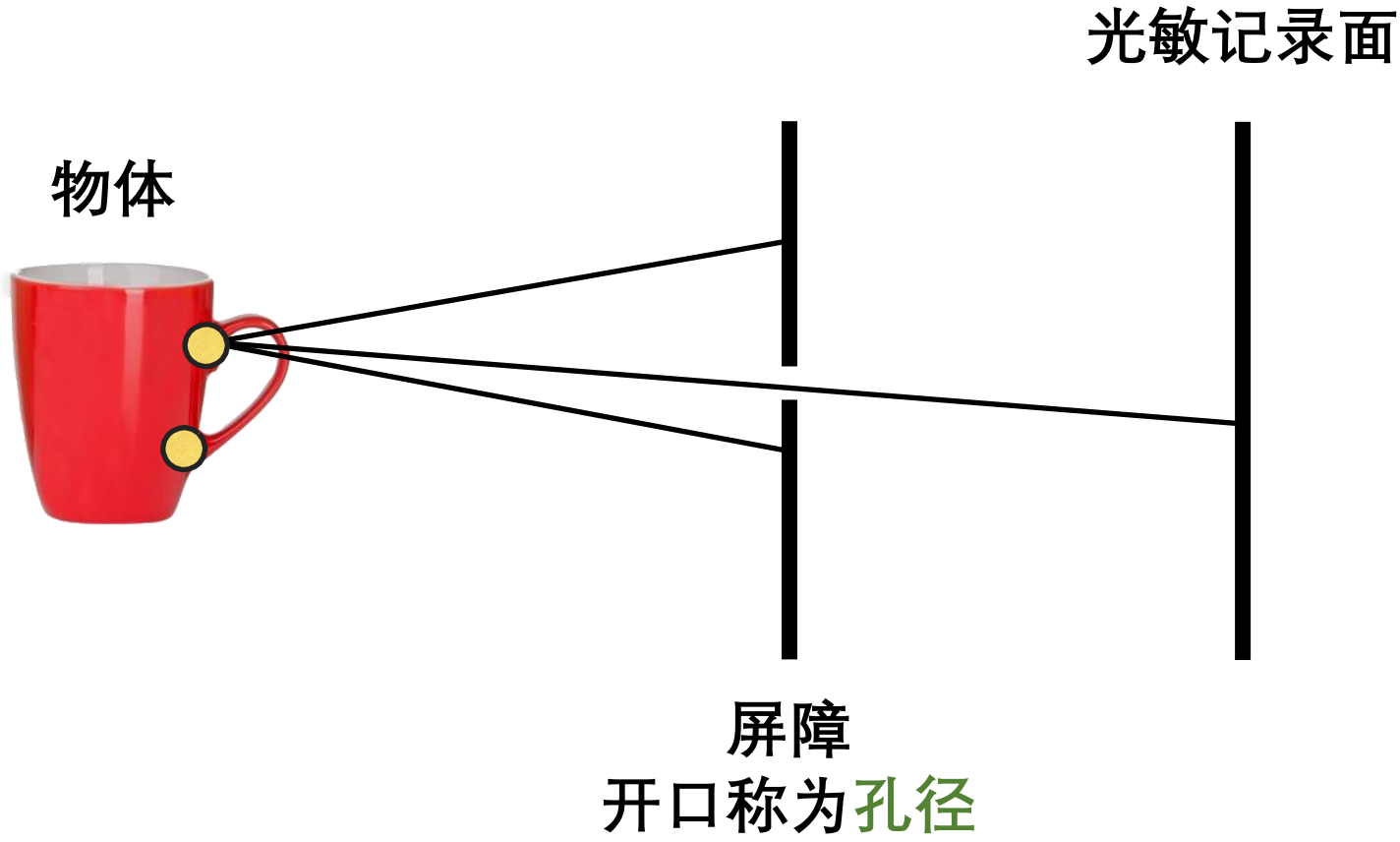
物体

屏障
开口称为孔径

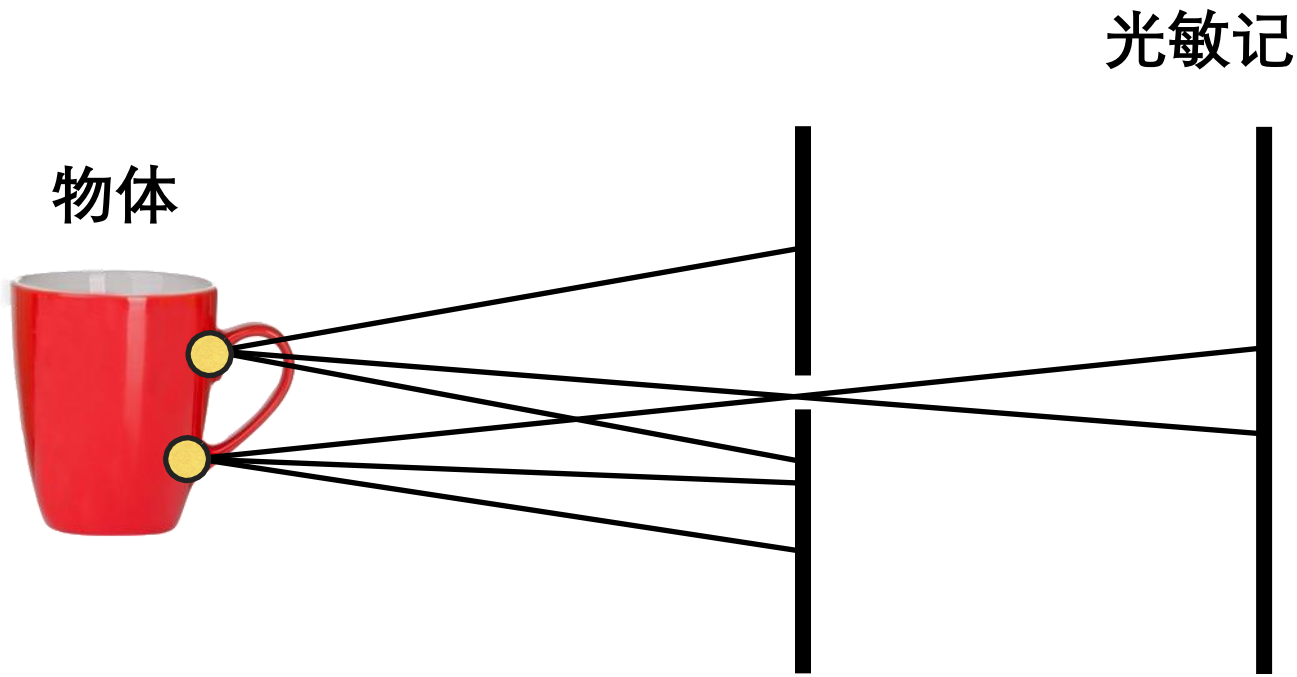
针孔相机



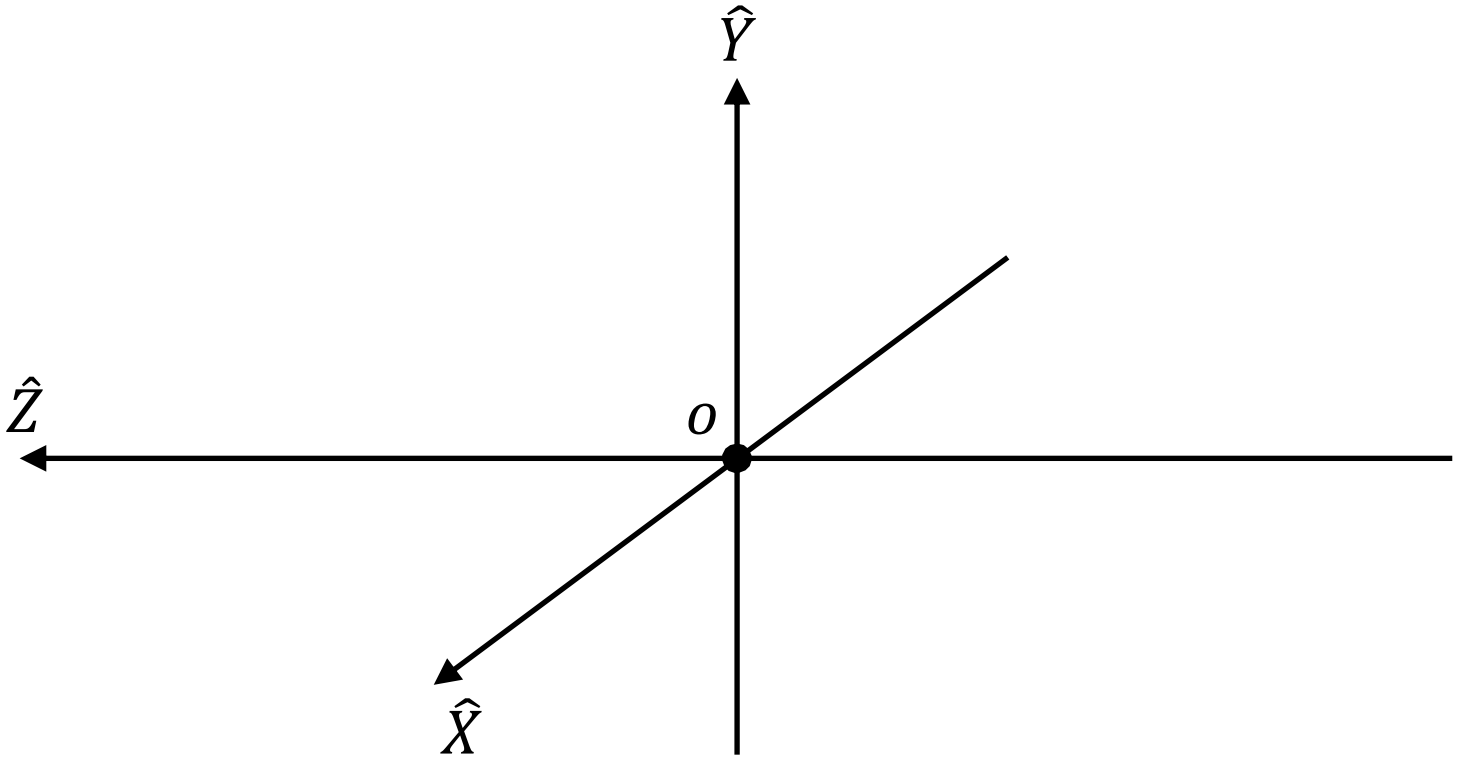
针孔相机



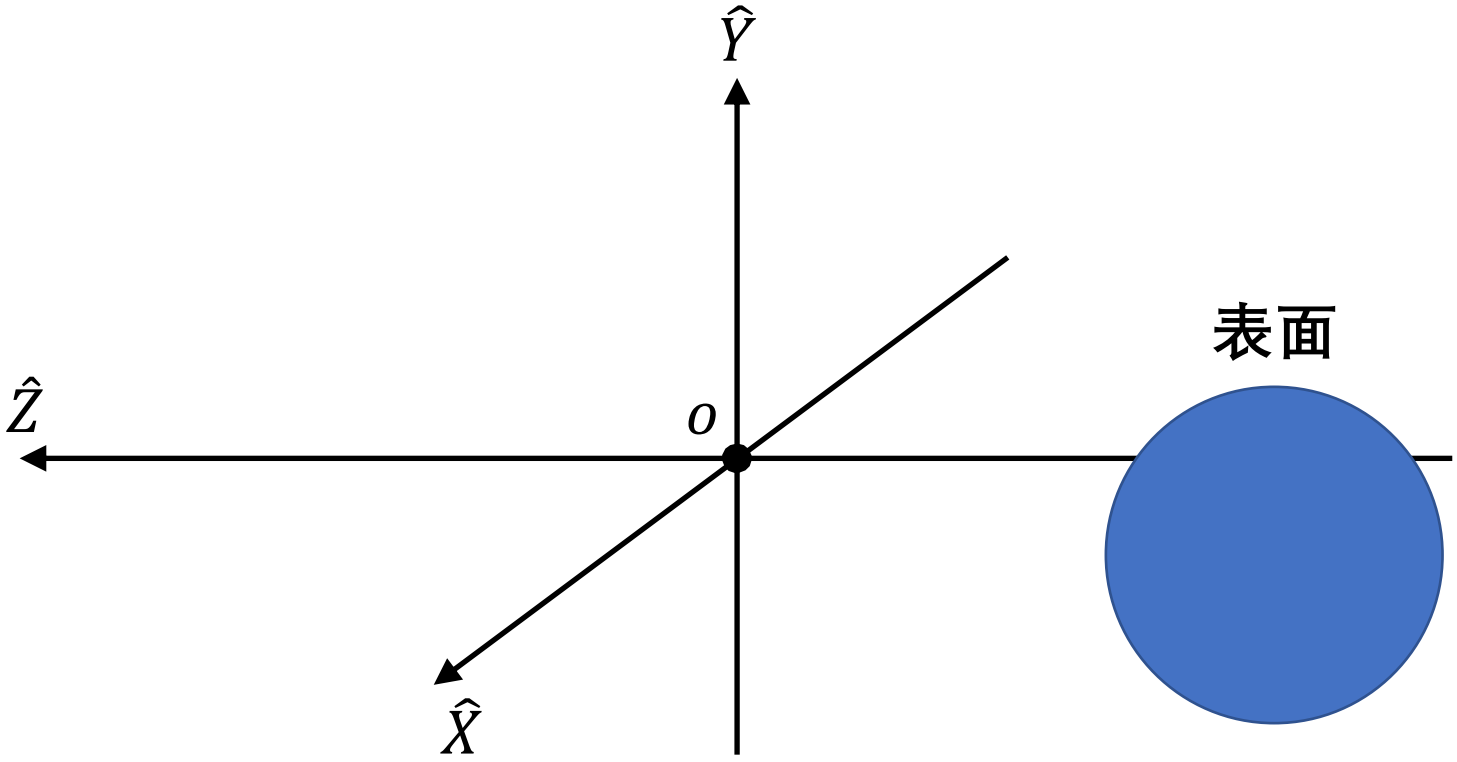
针孔相机



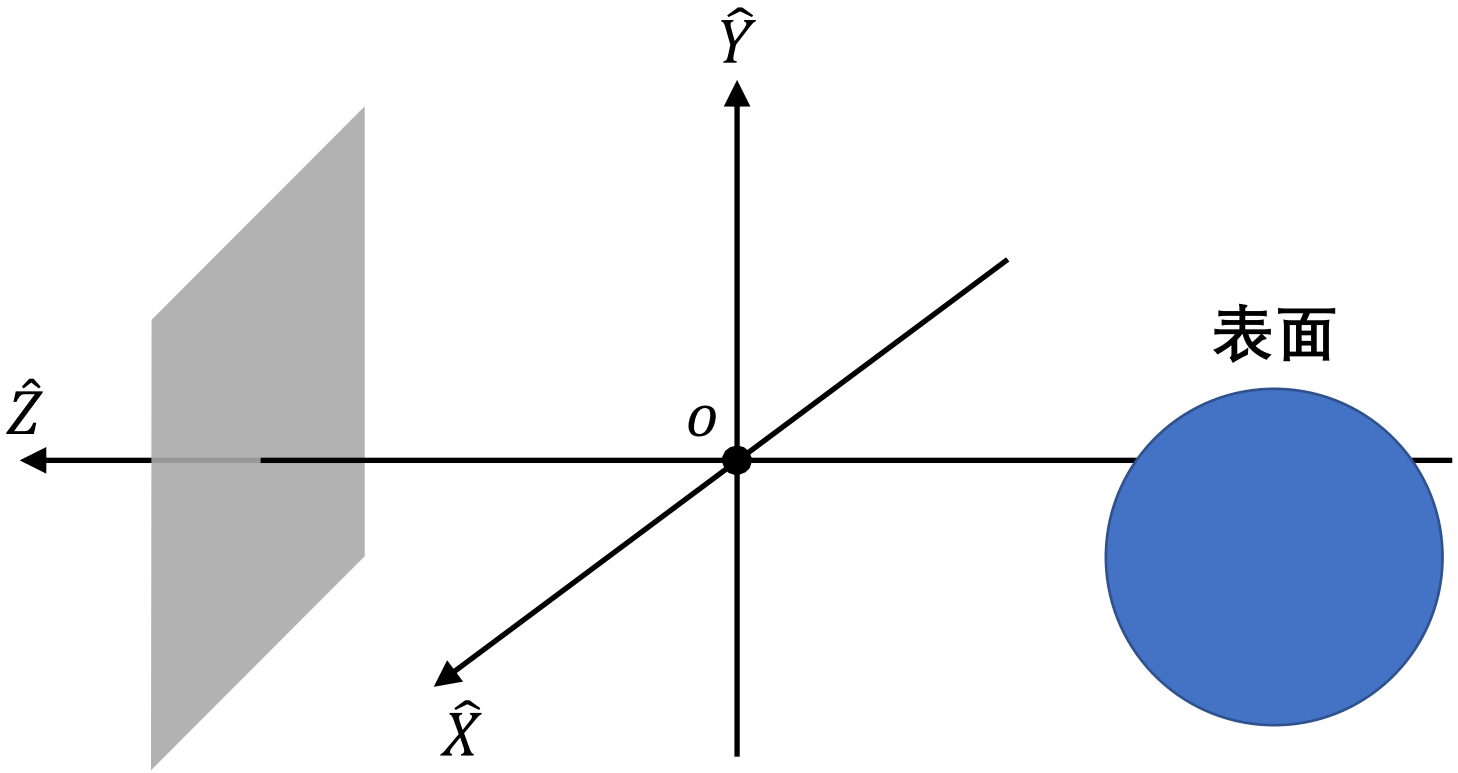
图像形成

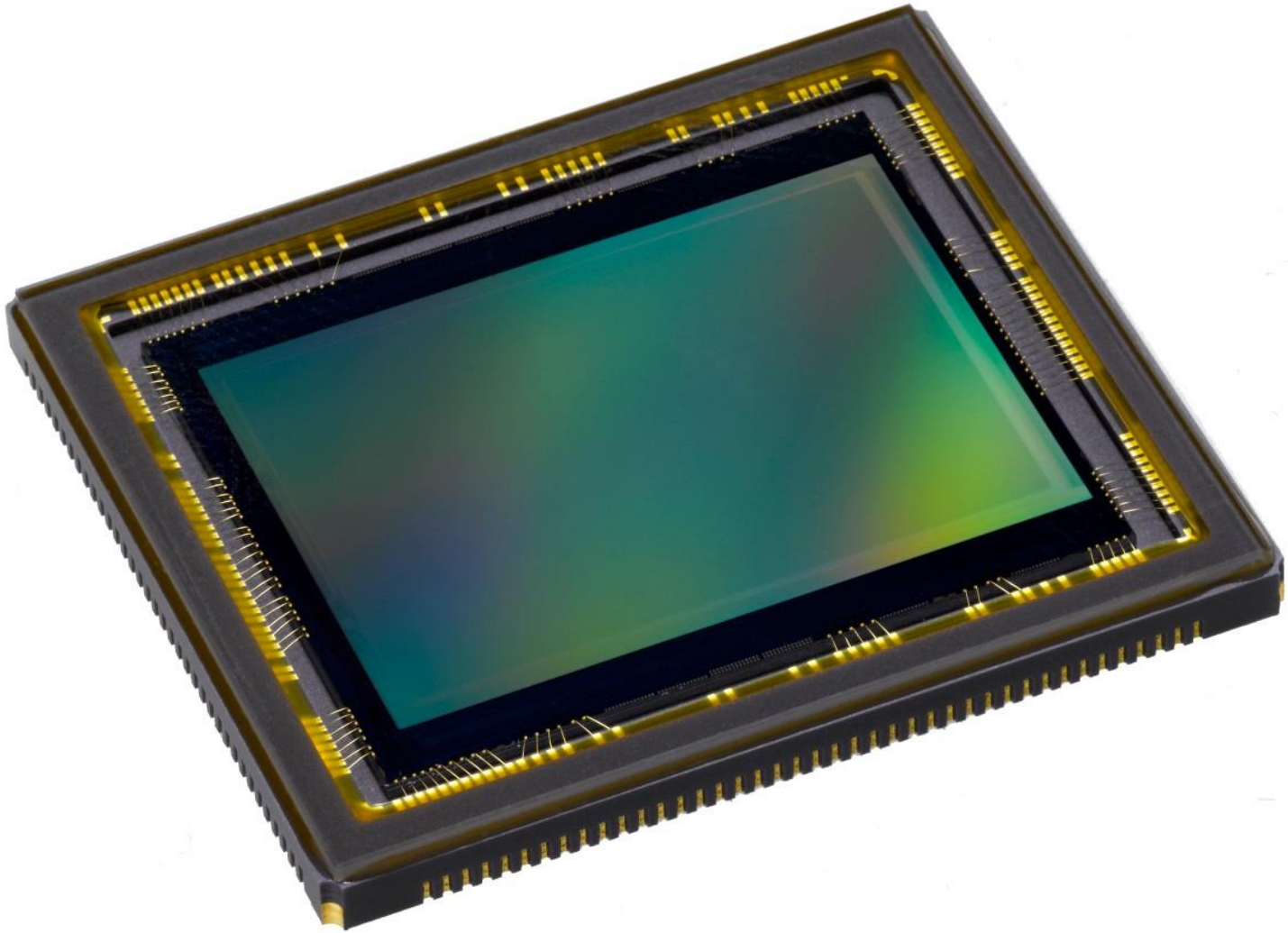


图像形成



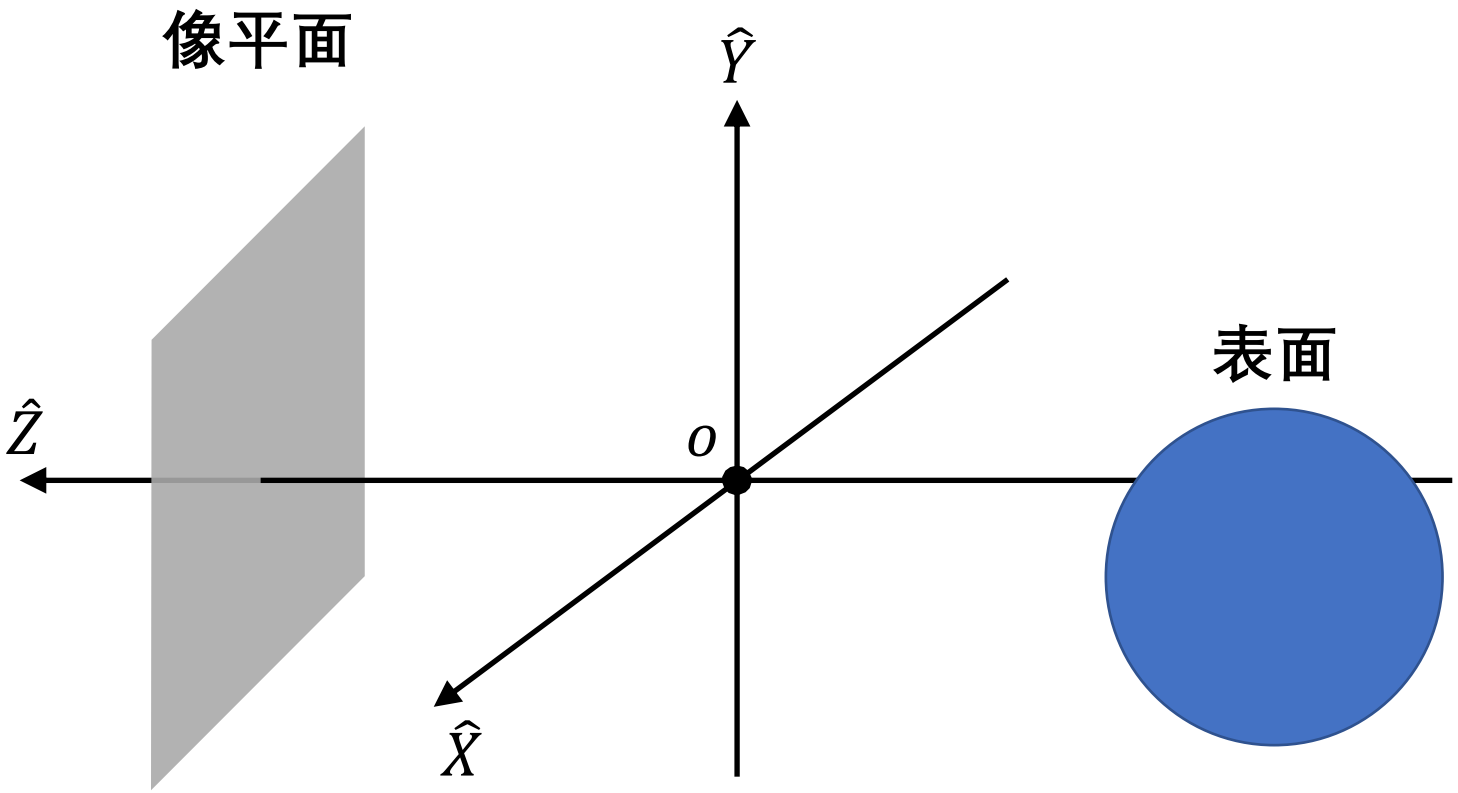
图像形成



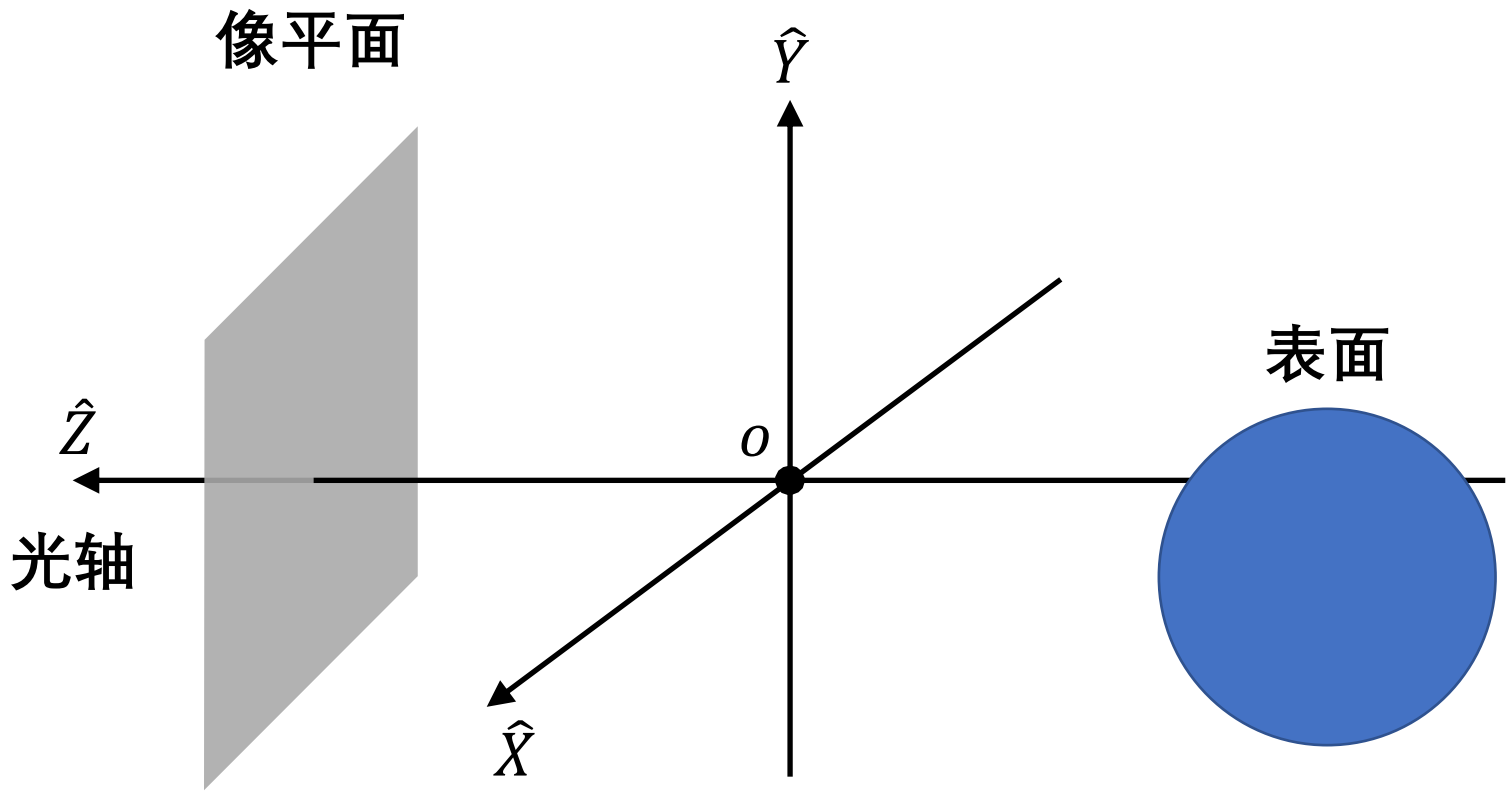


电荷耦合器件（CCD）

图像形成

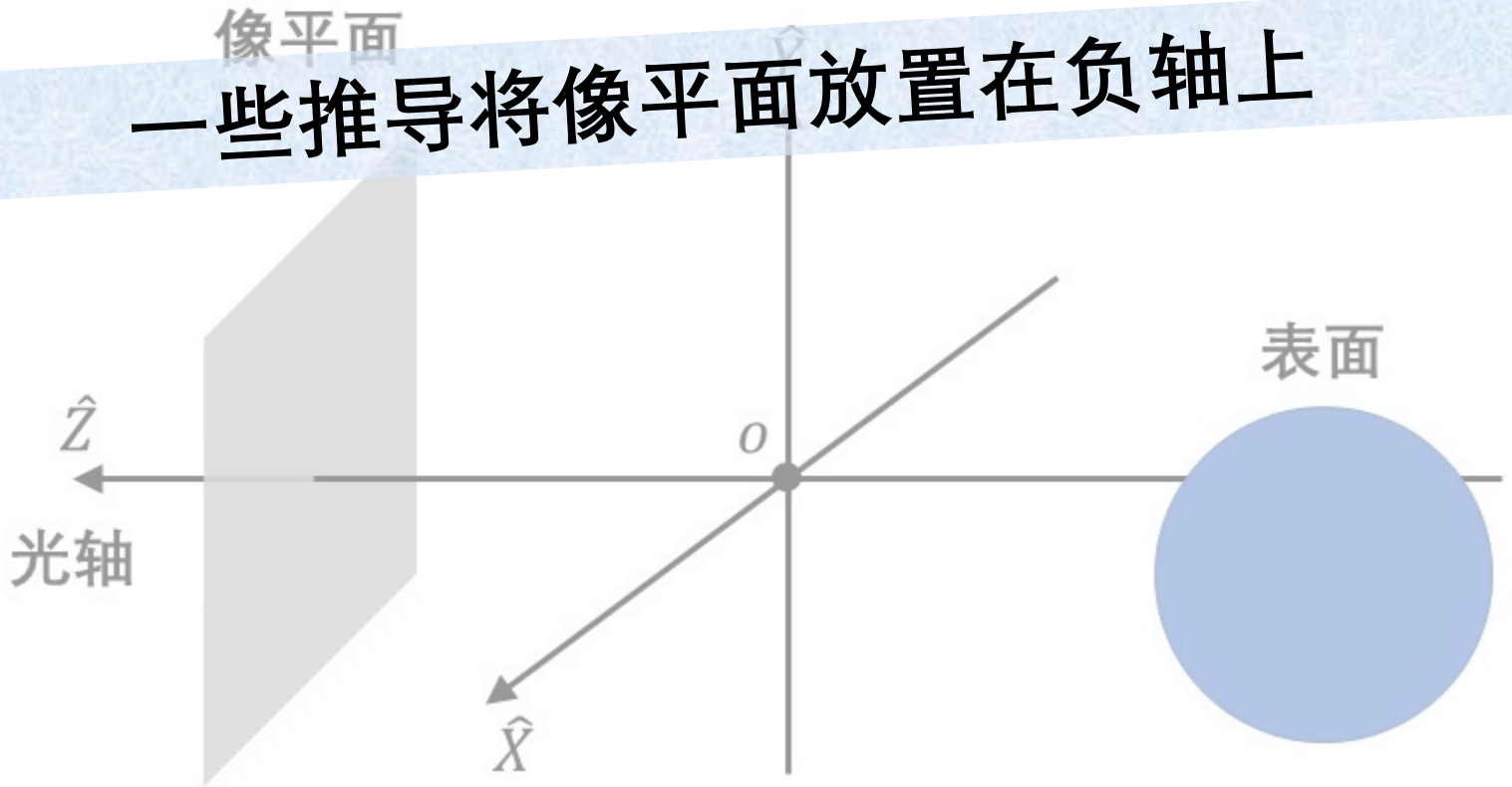


图像形成

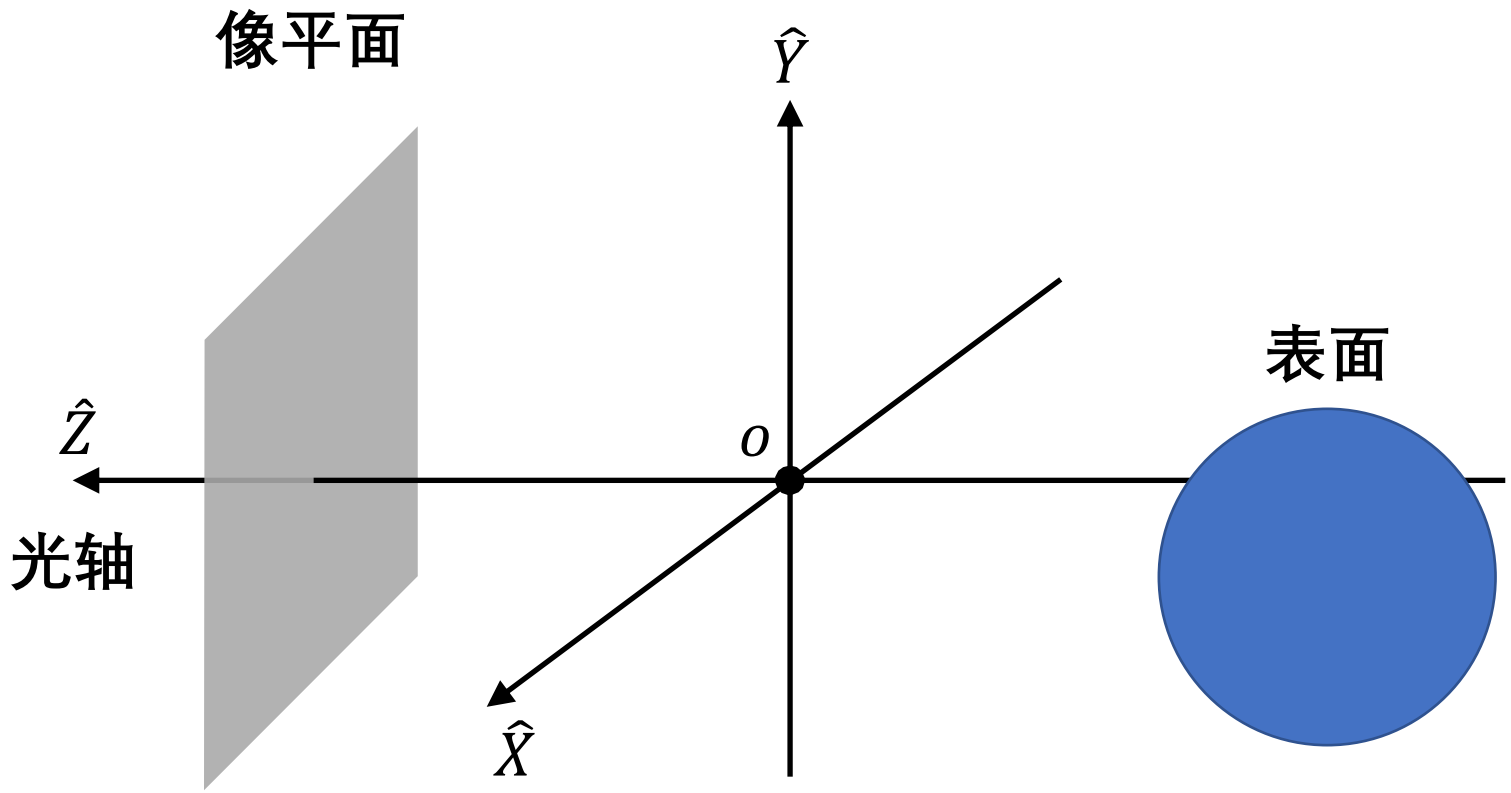


图像形成

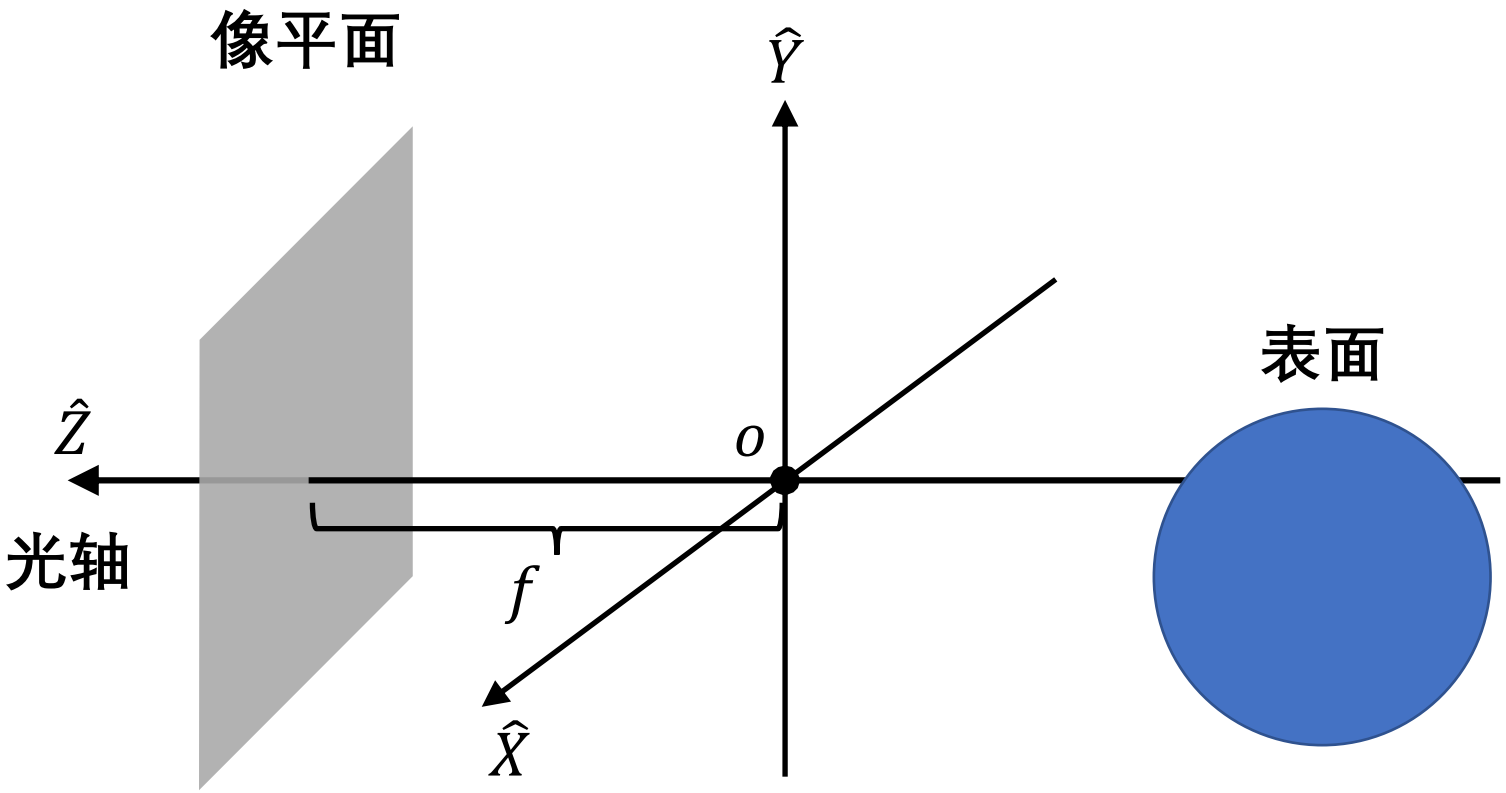
一些推导将像平面放置在负轴上



图像形成



图像形成




vimeo
200 mm Lens

ZOOM LENS EF 24-105mm 1:4 L IS USM
24-105mm
24 28 35 50 70 105
IMAGE STABILIZER
8000 56 93
AF-ONE

200mm

视场

12°

50mm

200mm

视场

12°

46°

24mm

50mm

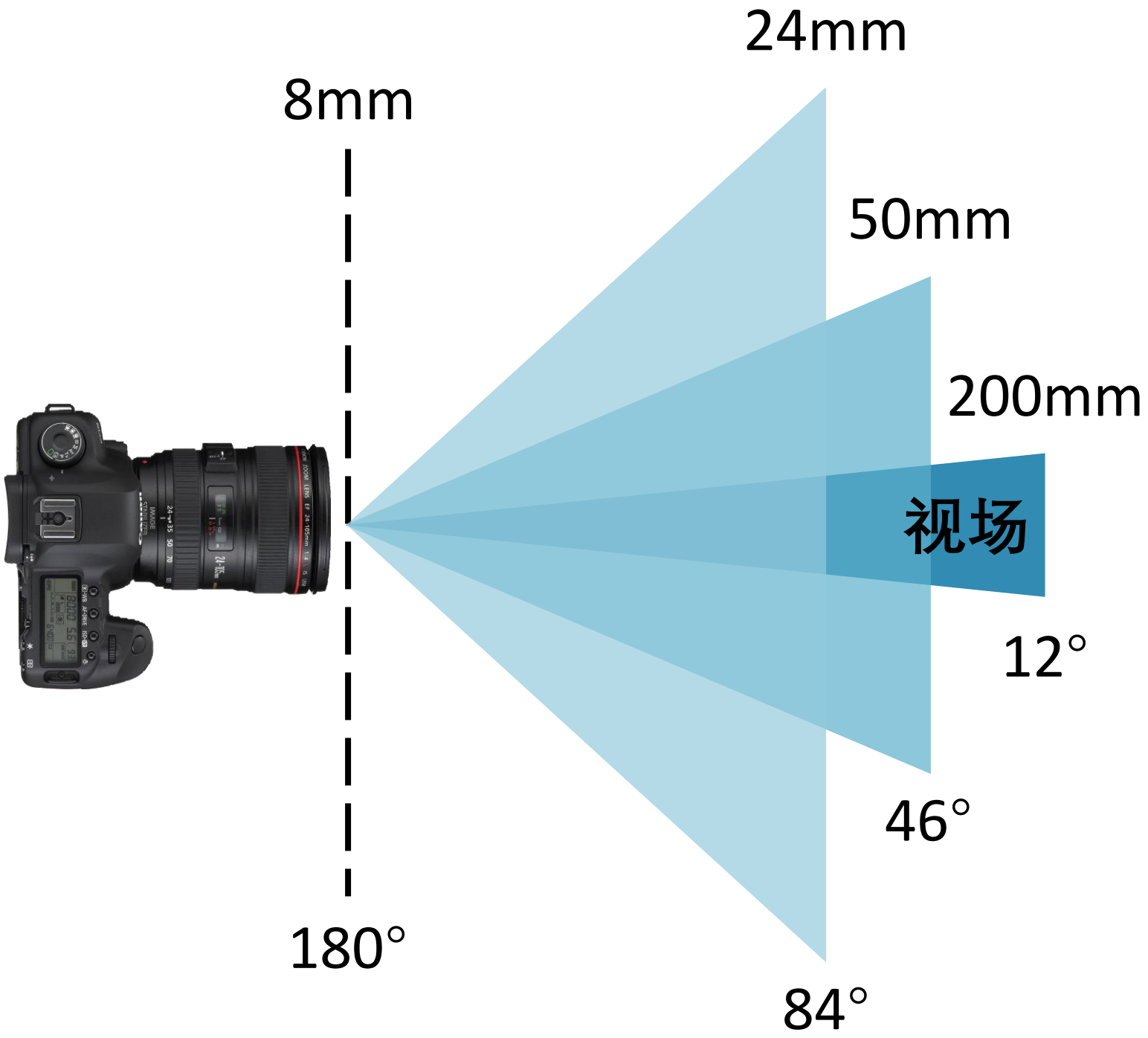
200mm

视场

12°

46°

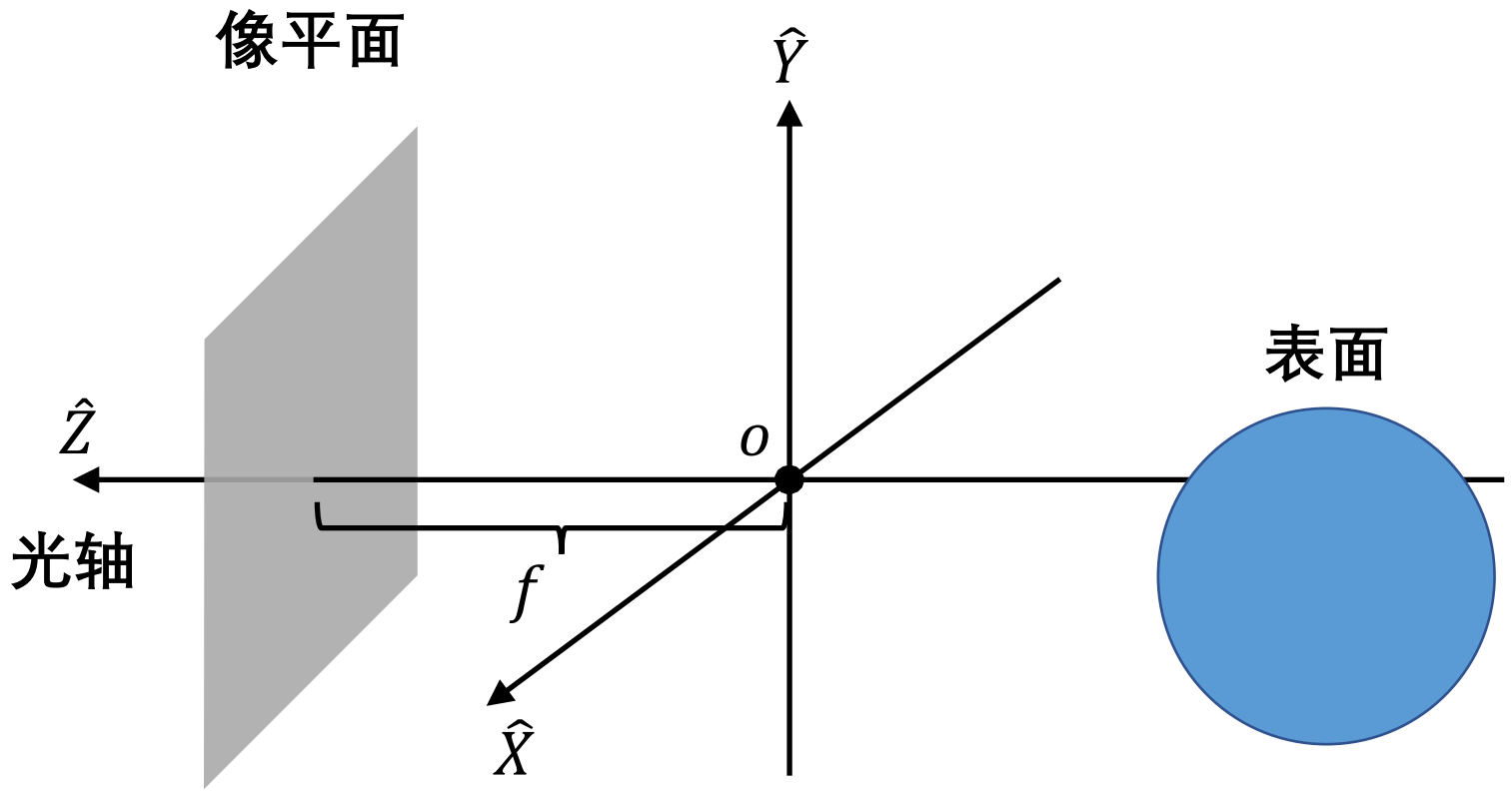
84°

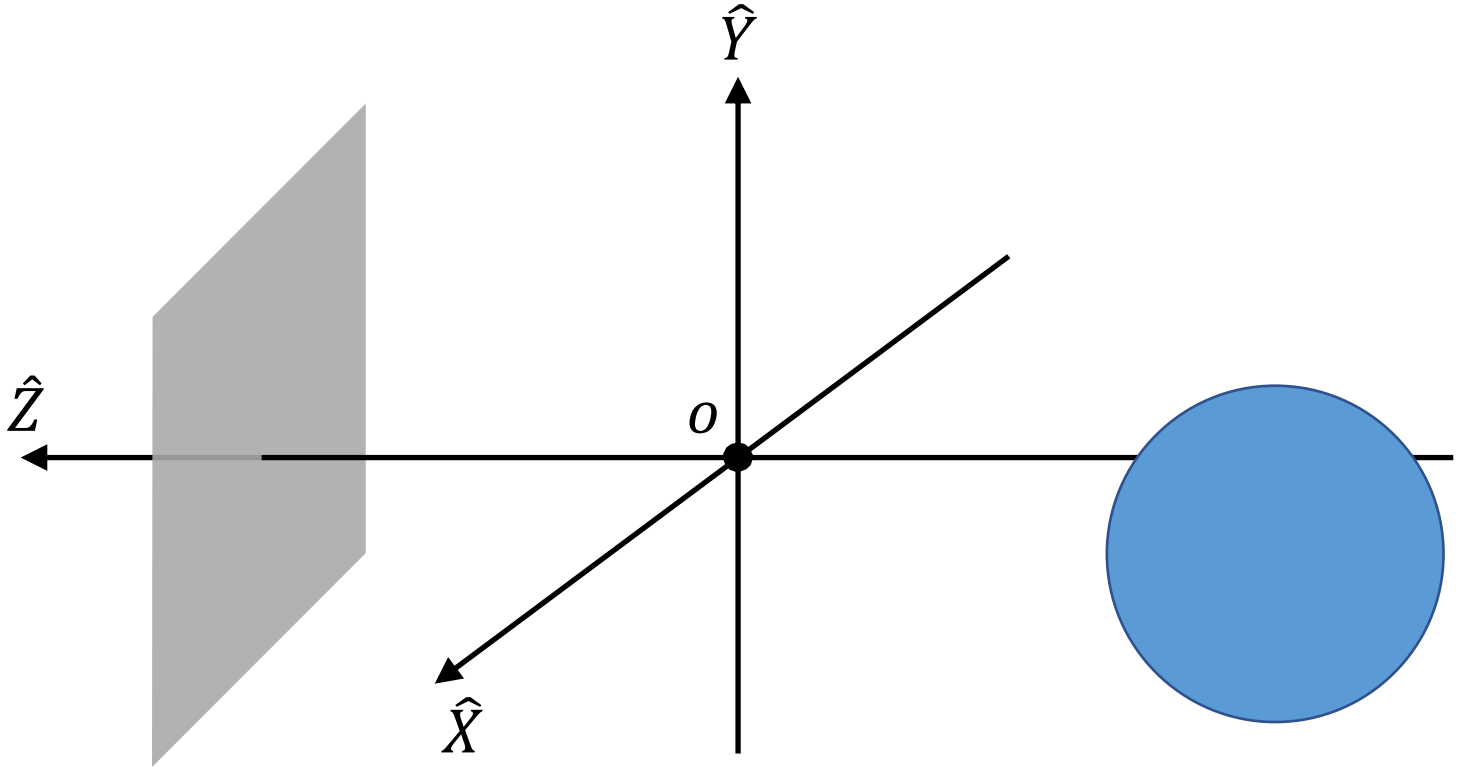

24mm
8mm
50mm
200mm
视场
12°
46°
180°
84°

“A camera adds ten pounds”

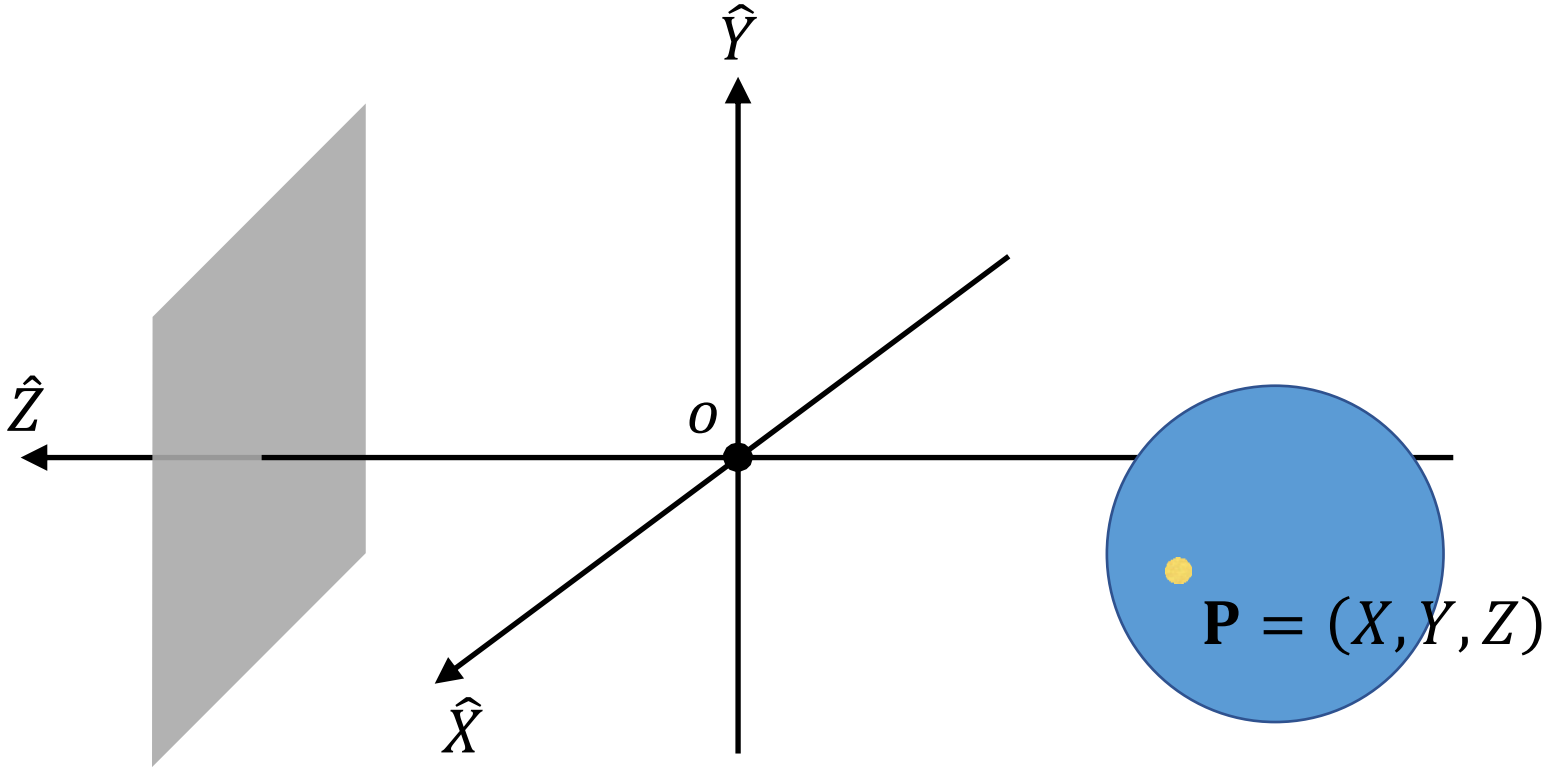

28mm

图像形成

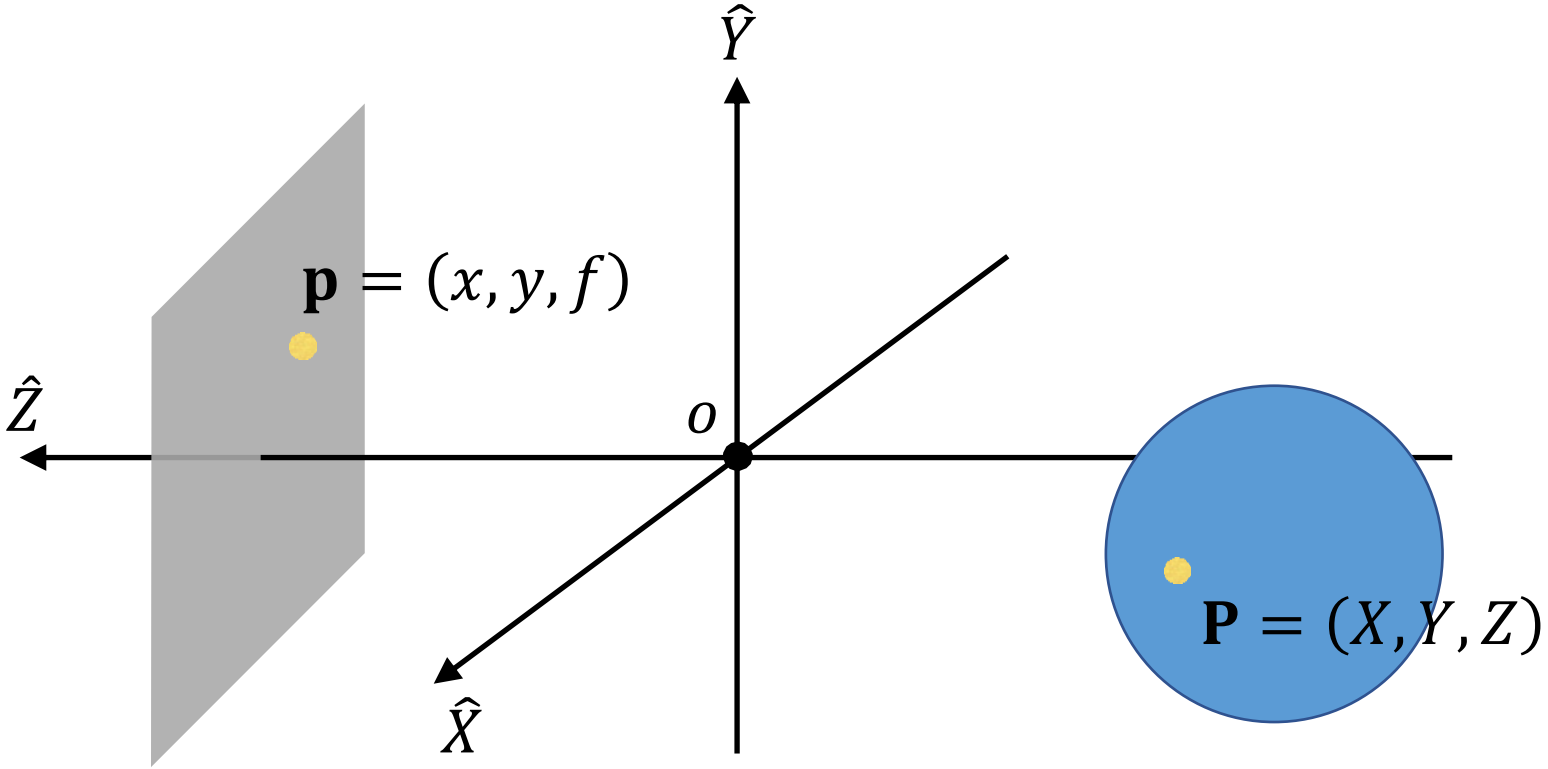




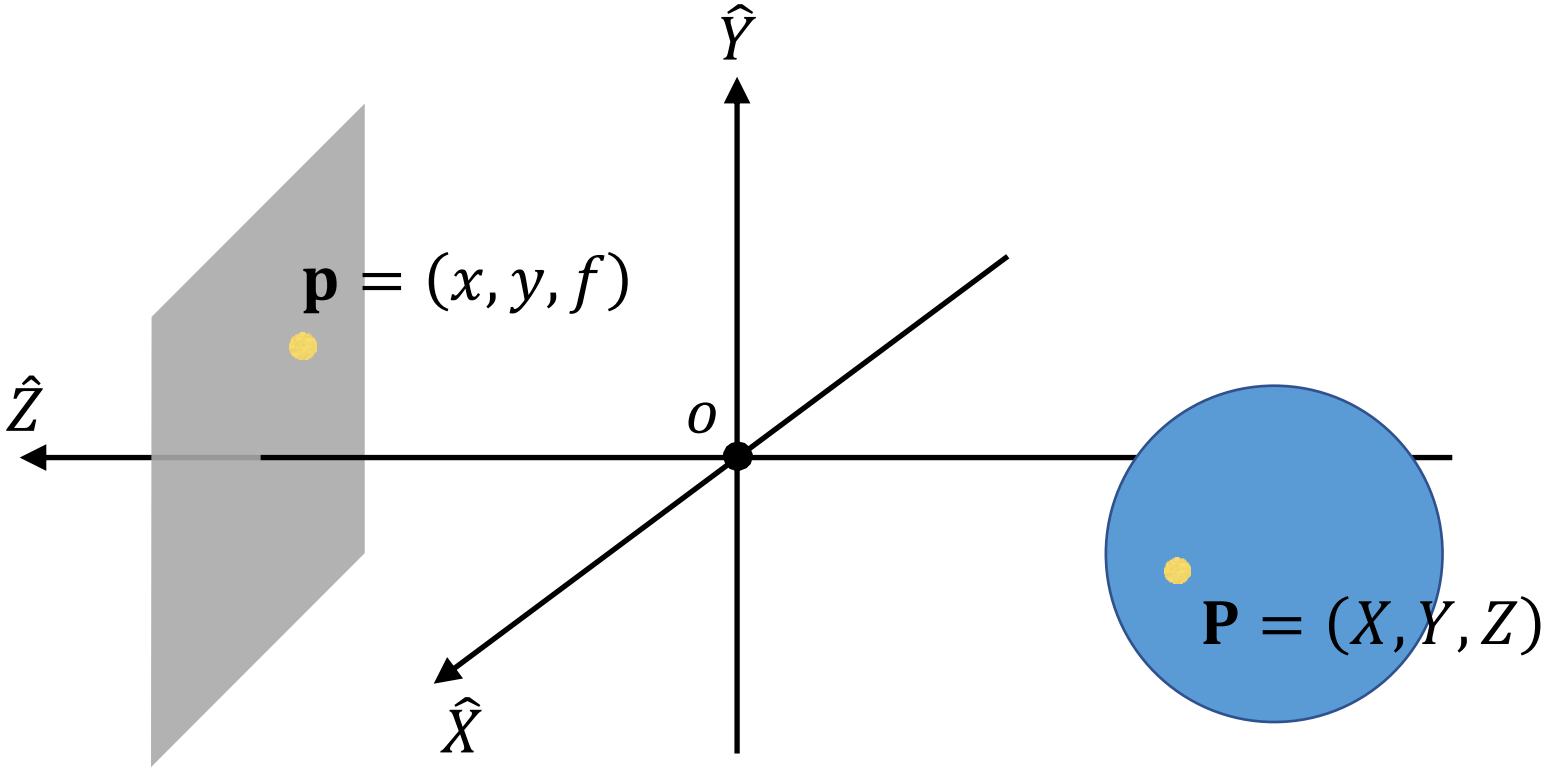
目标：将3D场景中点的位置与它的2D图像关联起来



目标：将3D场景中点的位置与它的2D图像关联起来

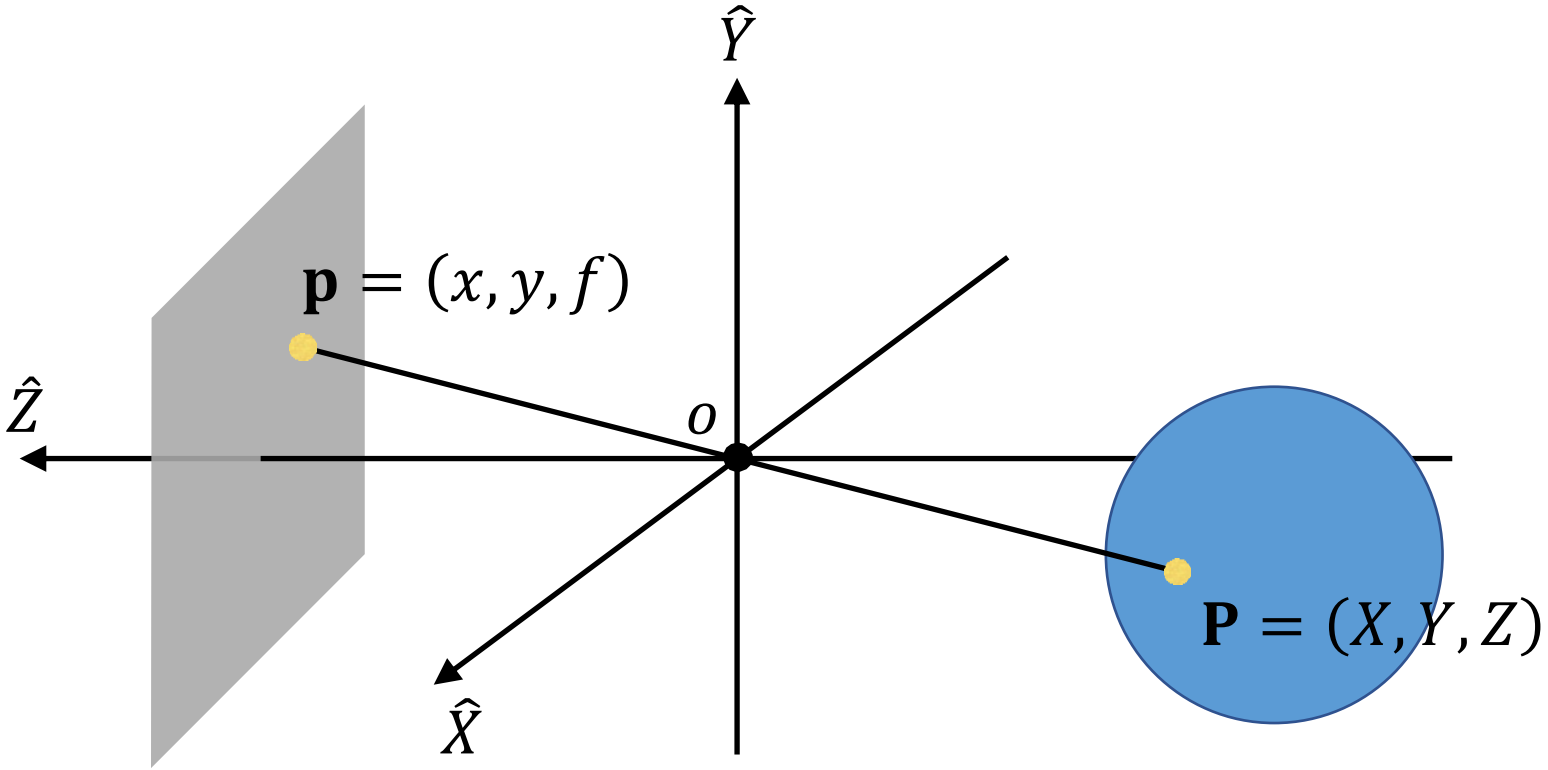


目标：将3D场景中点的位置与它的2D图像关联起来



目标： 将3D场景中点的位置与它的2D图像关联起来

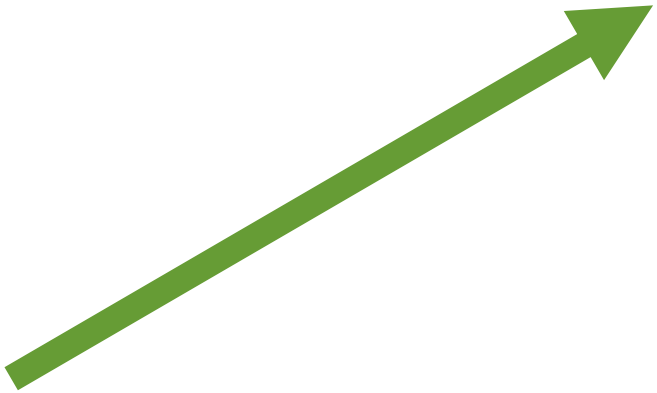
假设： P与p共线



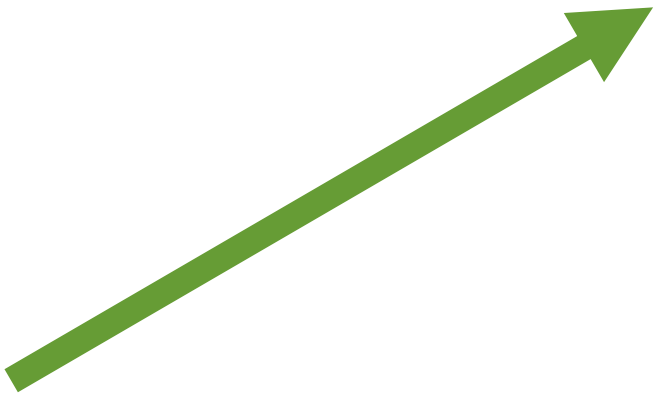
目标： 将3D场景中点的位置与它的2D图像关联起来

假设： P与p共线

回顾：线性代数

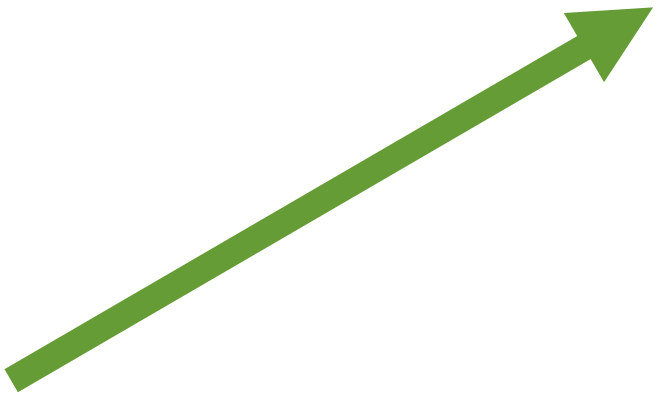


向量



大小

向量



向量

大小

方向

$$\mathbf{x} \in \mathbb{R}^n$$

$\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$$

定义：向量x的欧几里得范数或长度如下所示

定义：向量x的欧几里得范数或长度如下所示

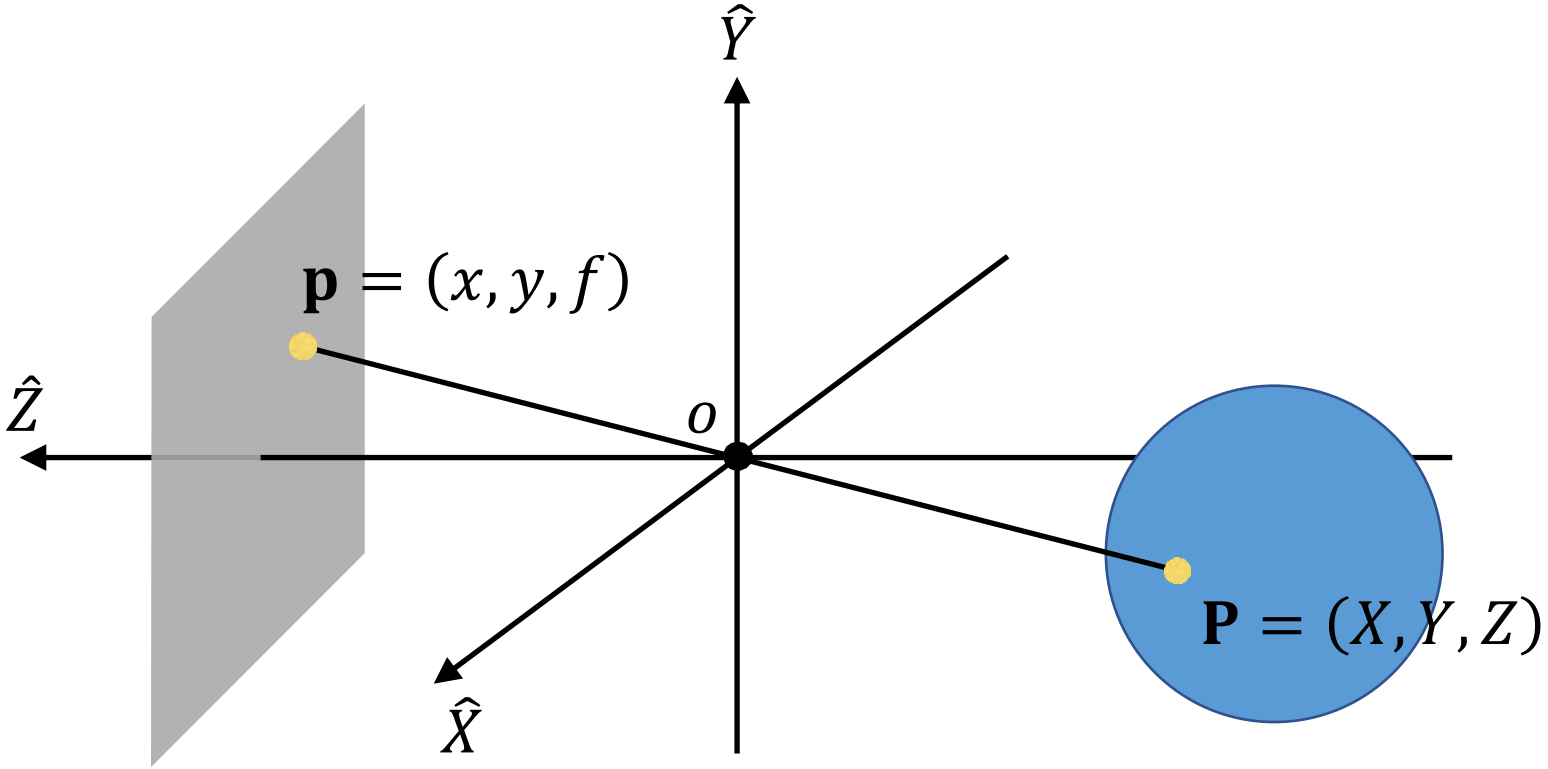
$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

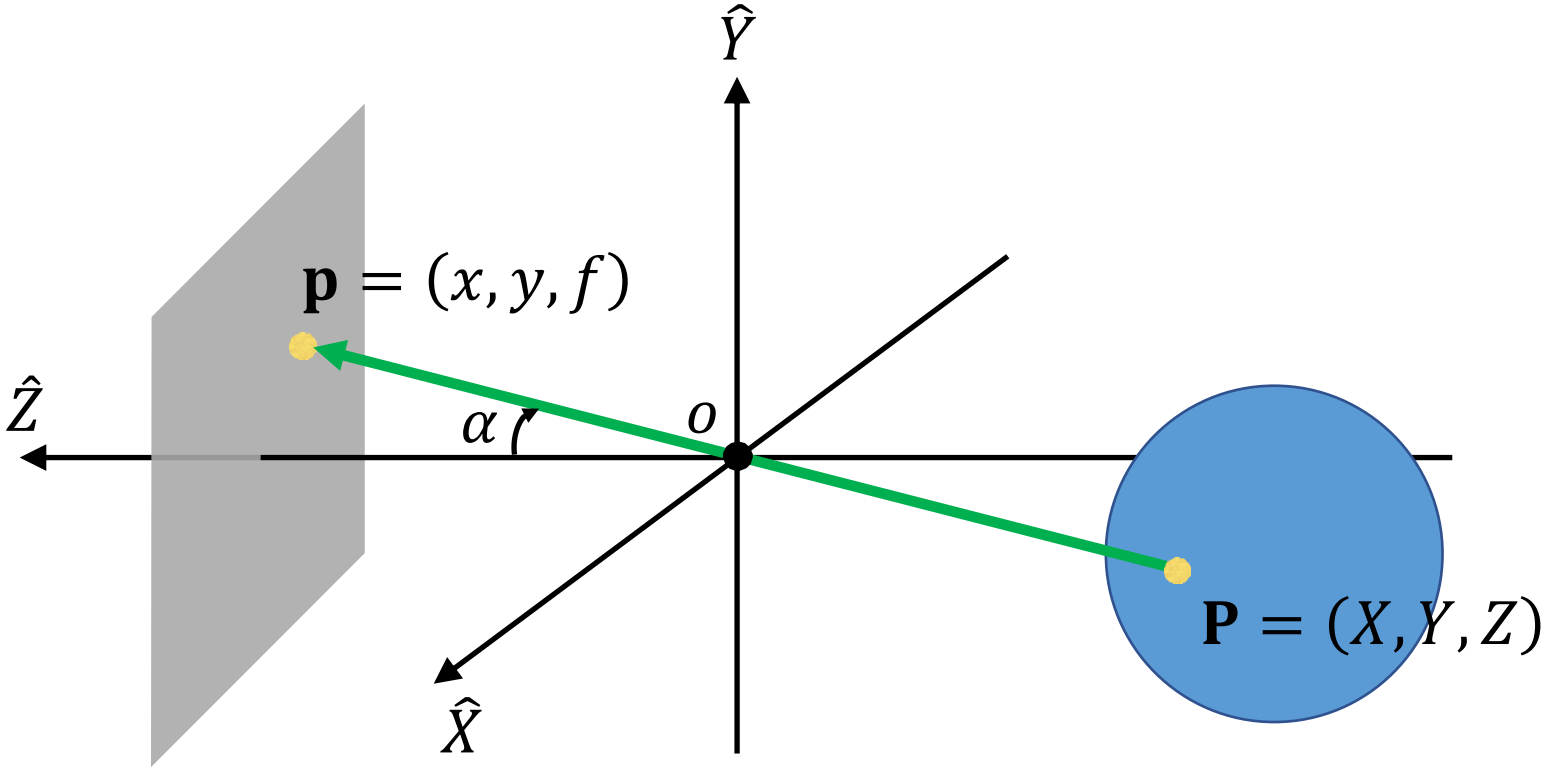
定义：向量x的欧几里得范数或长度如下所示

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

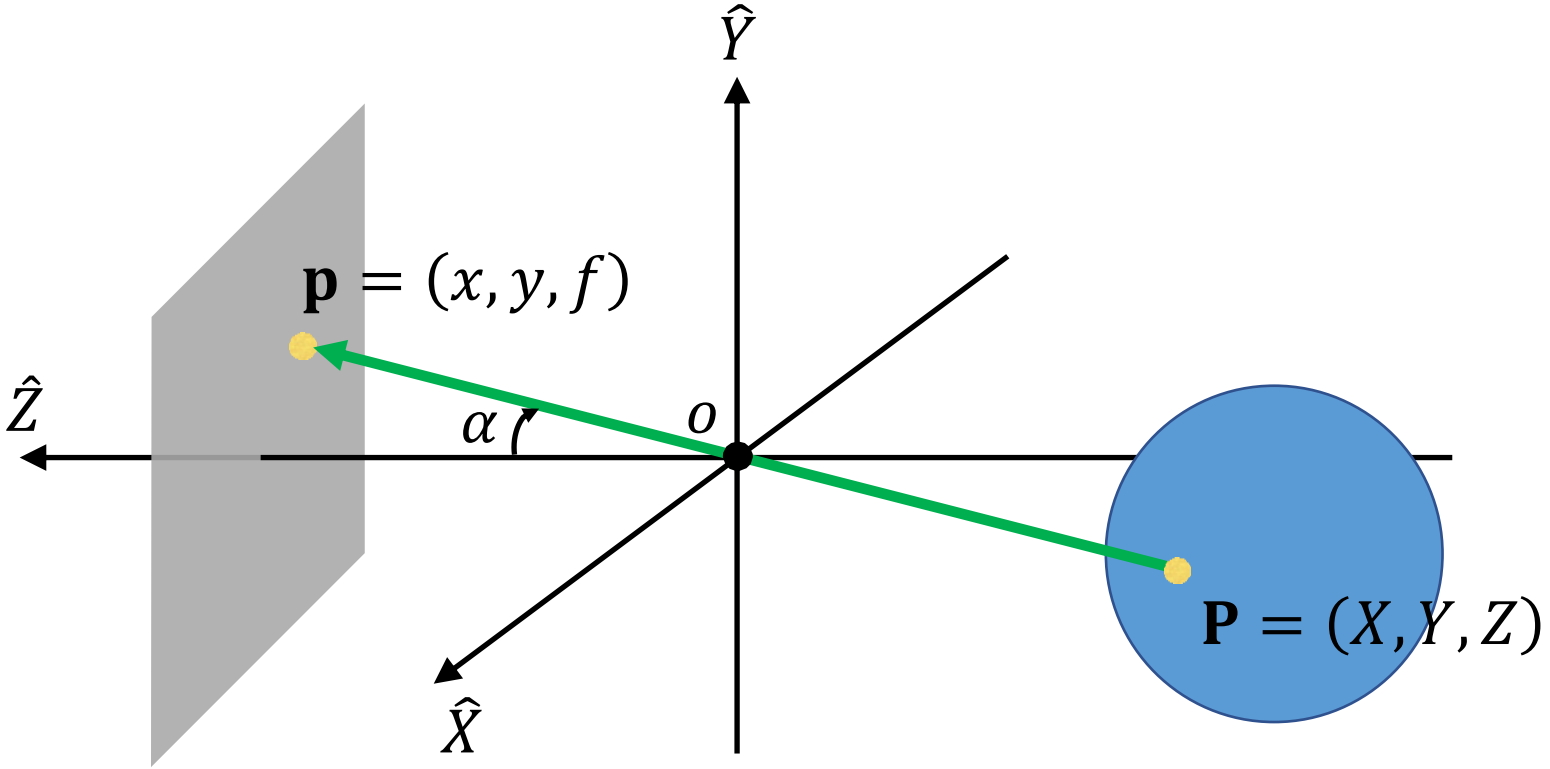
单位向量是长度为1的向量

回顾：线性代数


$\hat{Y}$
$\mathbf{p} = (x, y, f)$
$\hat{Z}$
o
$\mathbf{P} = (X, Y, Z)$
$\hat{X}$

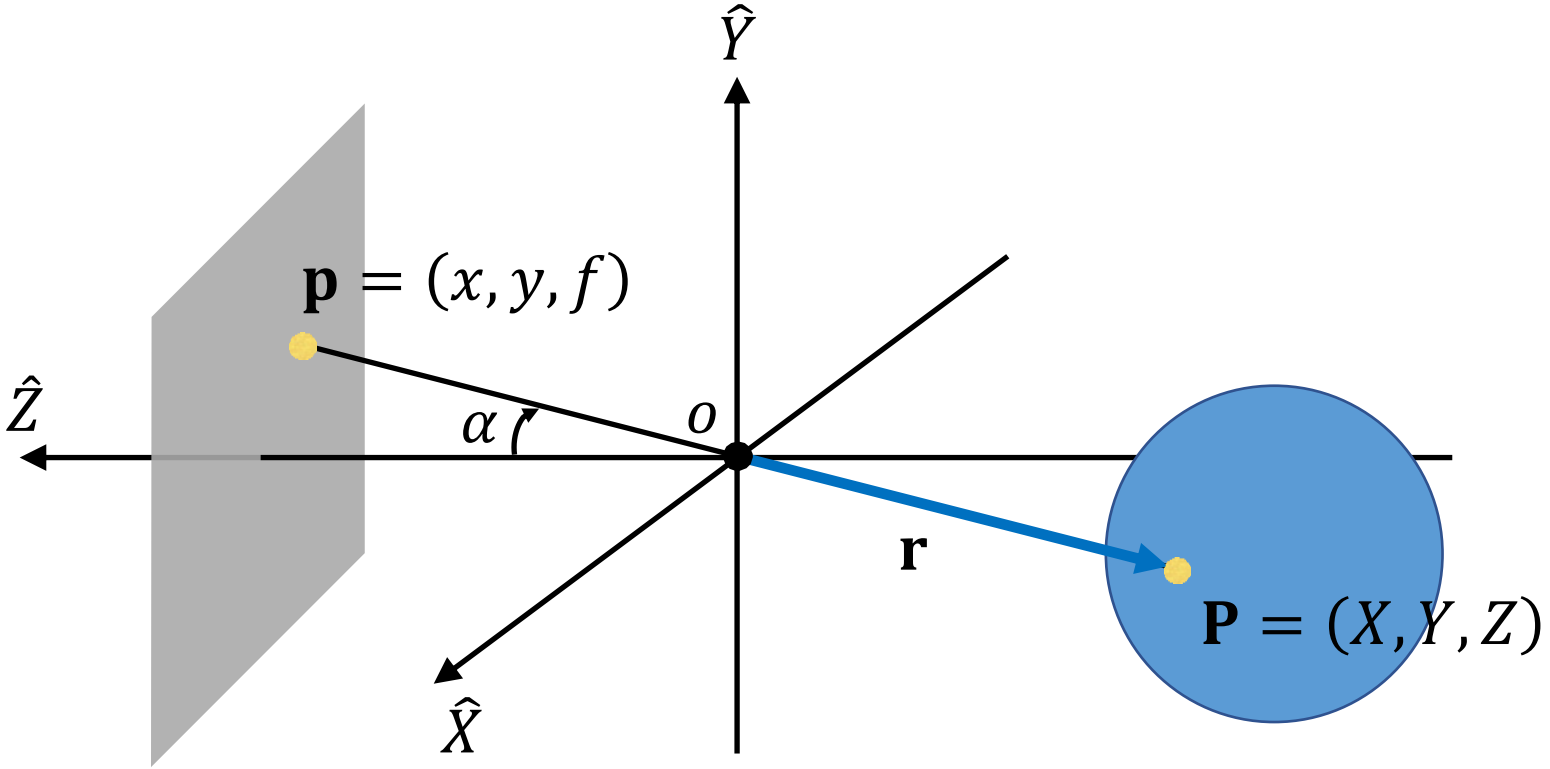


令射线Pp与光轴的夹角为α



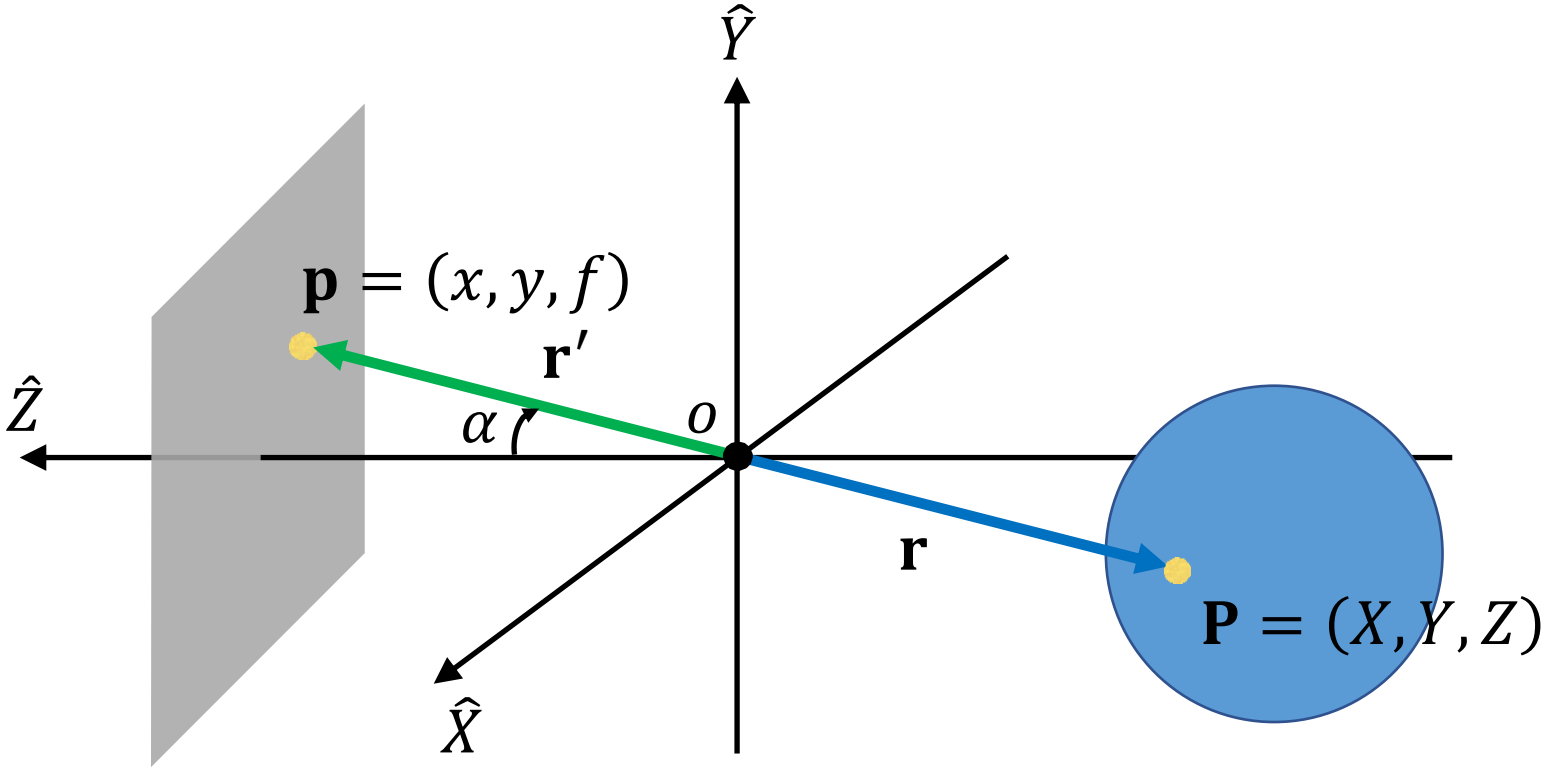
令射线Pp与光轴的夹角为α

该射线可以分成两部分



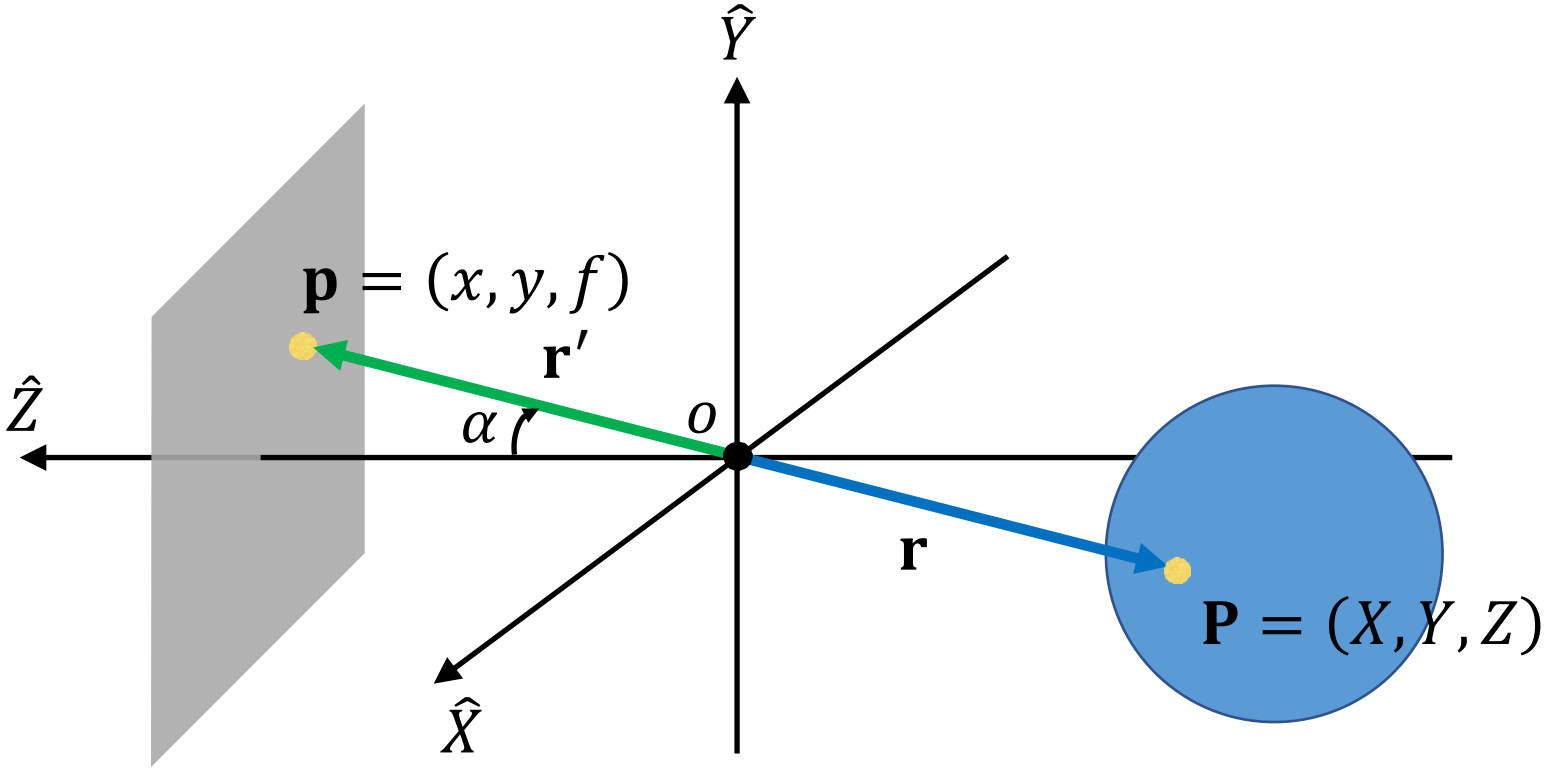
令射线Pp与光轴的夹角为α

该射线可以分成两部分

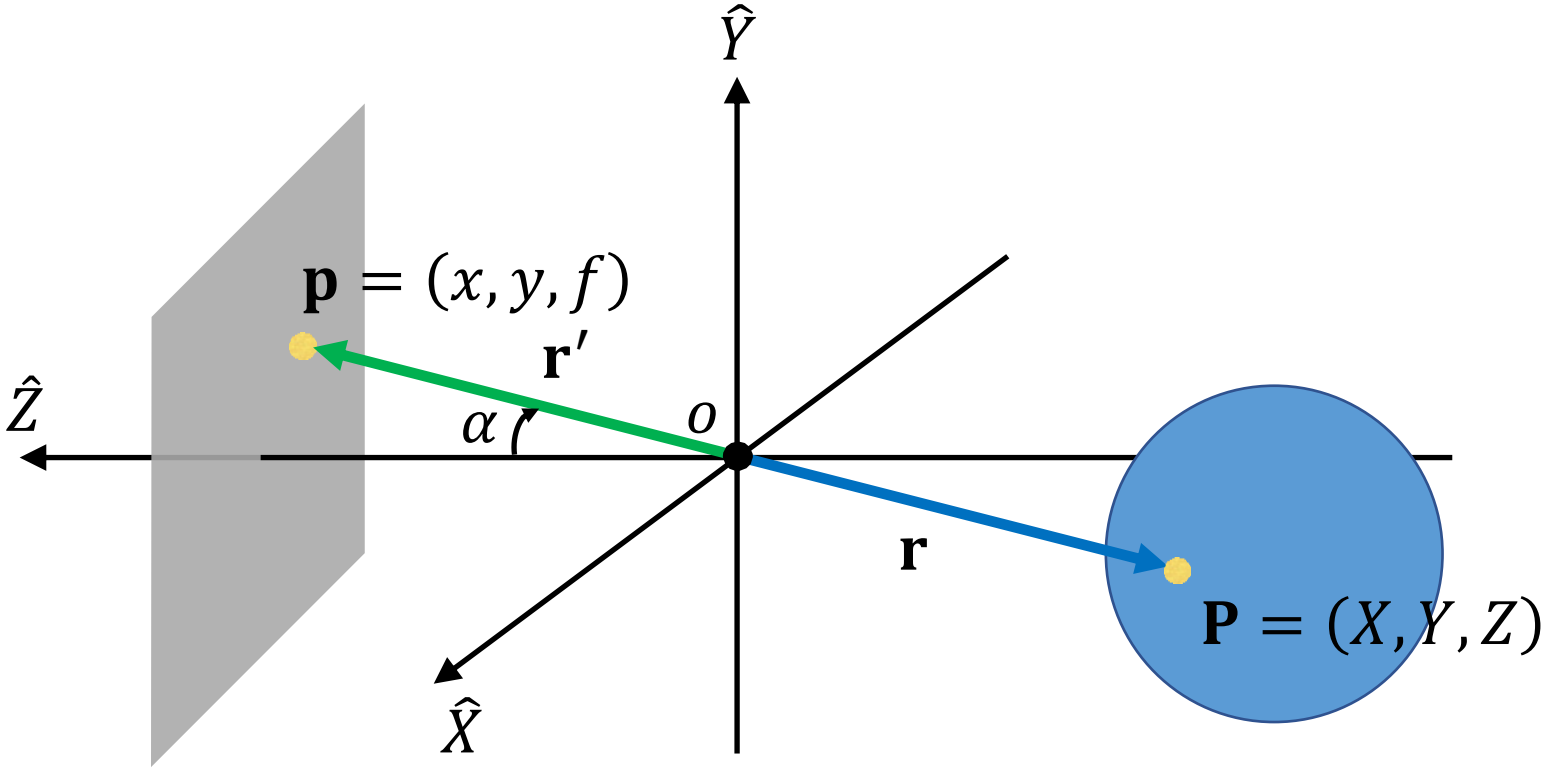


令射线Pp与光轴的夹角为α

该射线可以分成两部分

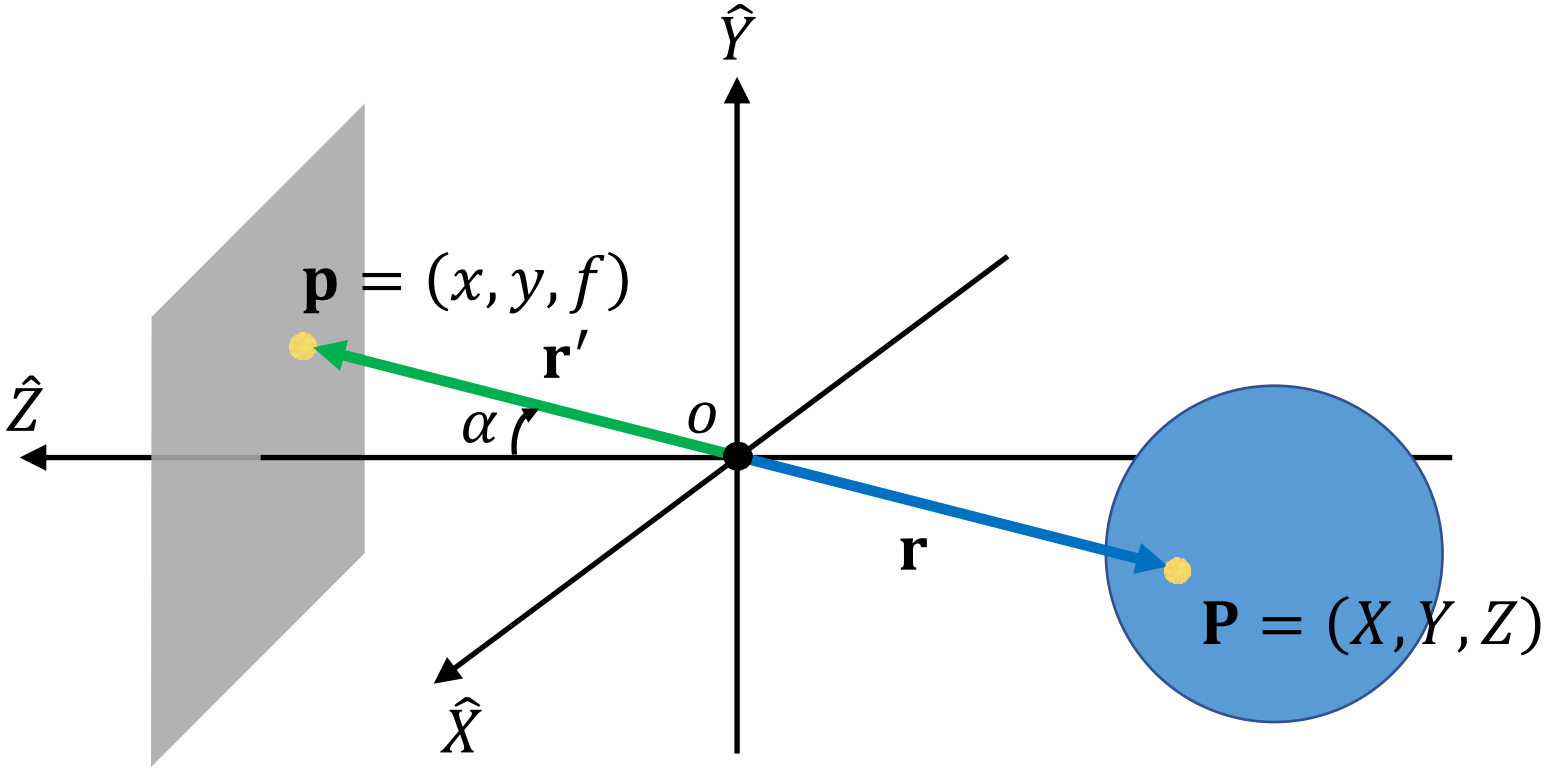

$\hat{Y}$
$\mathbf{p} = (x, y, f)$
$\mathbf{r}'$
$\hat{Z}$
α
o
$\mathbf{r}$
$\mathbf{P} = (X, Y, Z)$
$\hat{X}$

r和r′是什么关系呢?



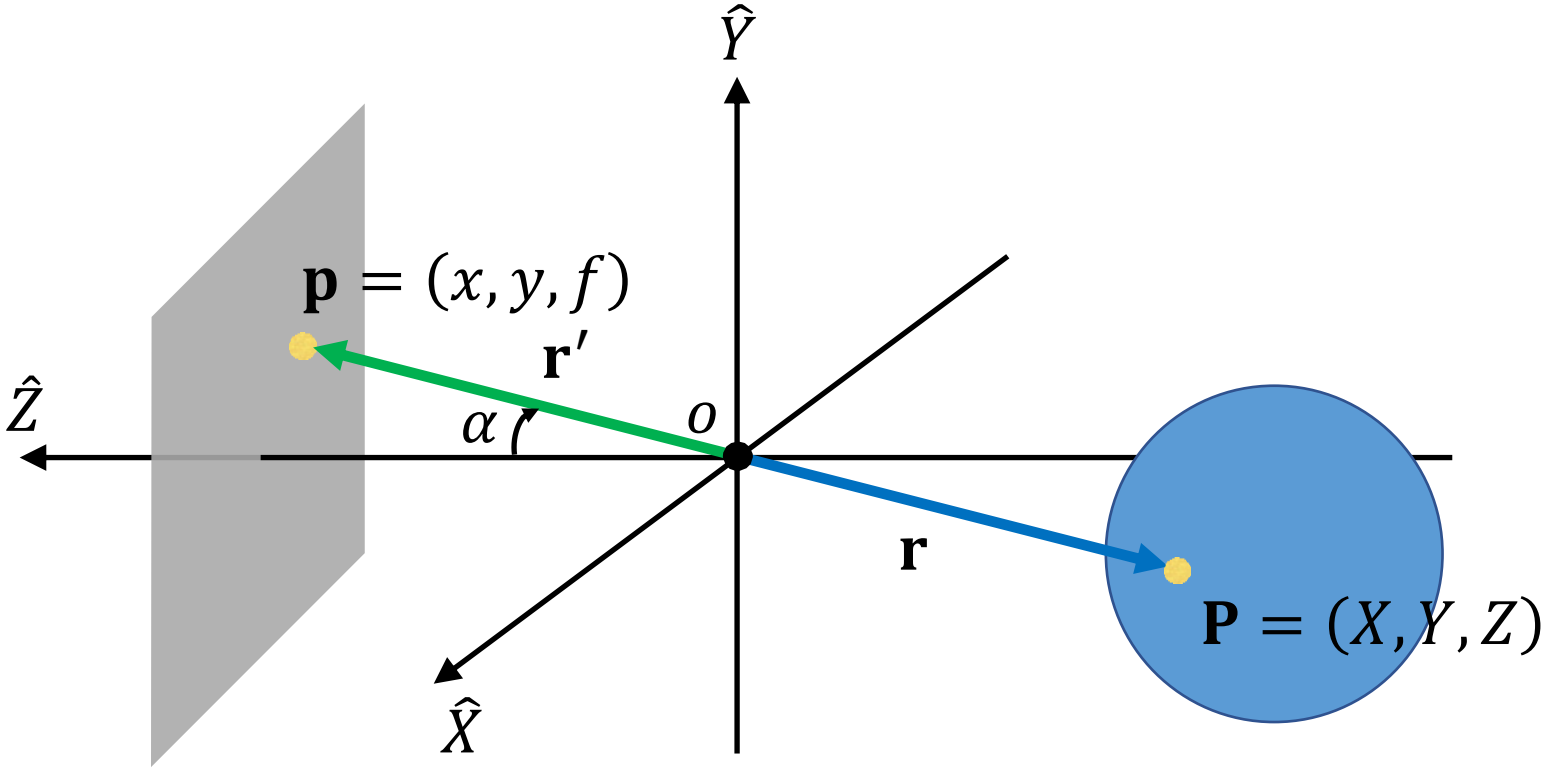
r和r′是什么关系呢?

这俩向量是共线的，相差一个负比例因子

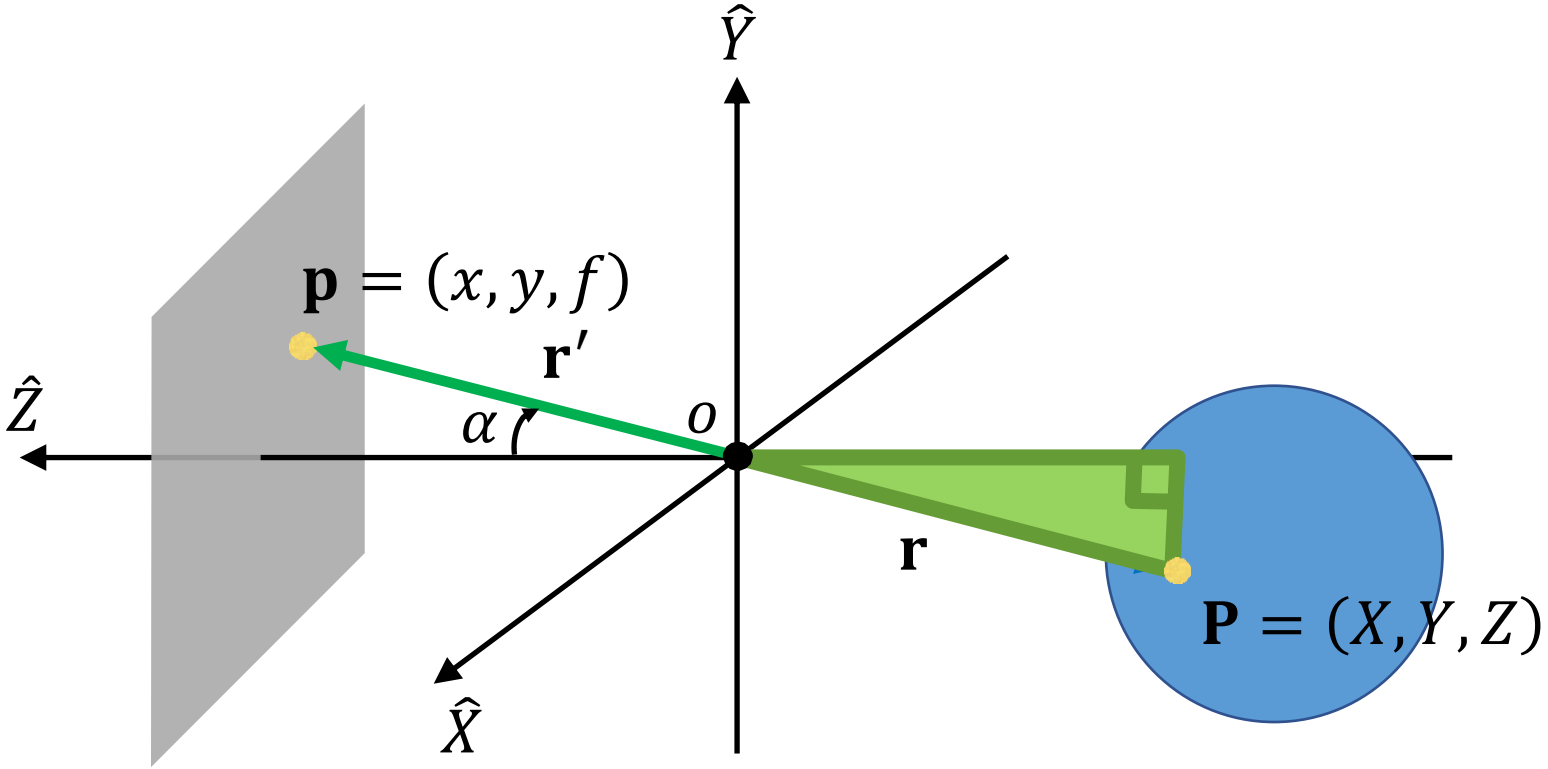


r和r′是什么关系呢?

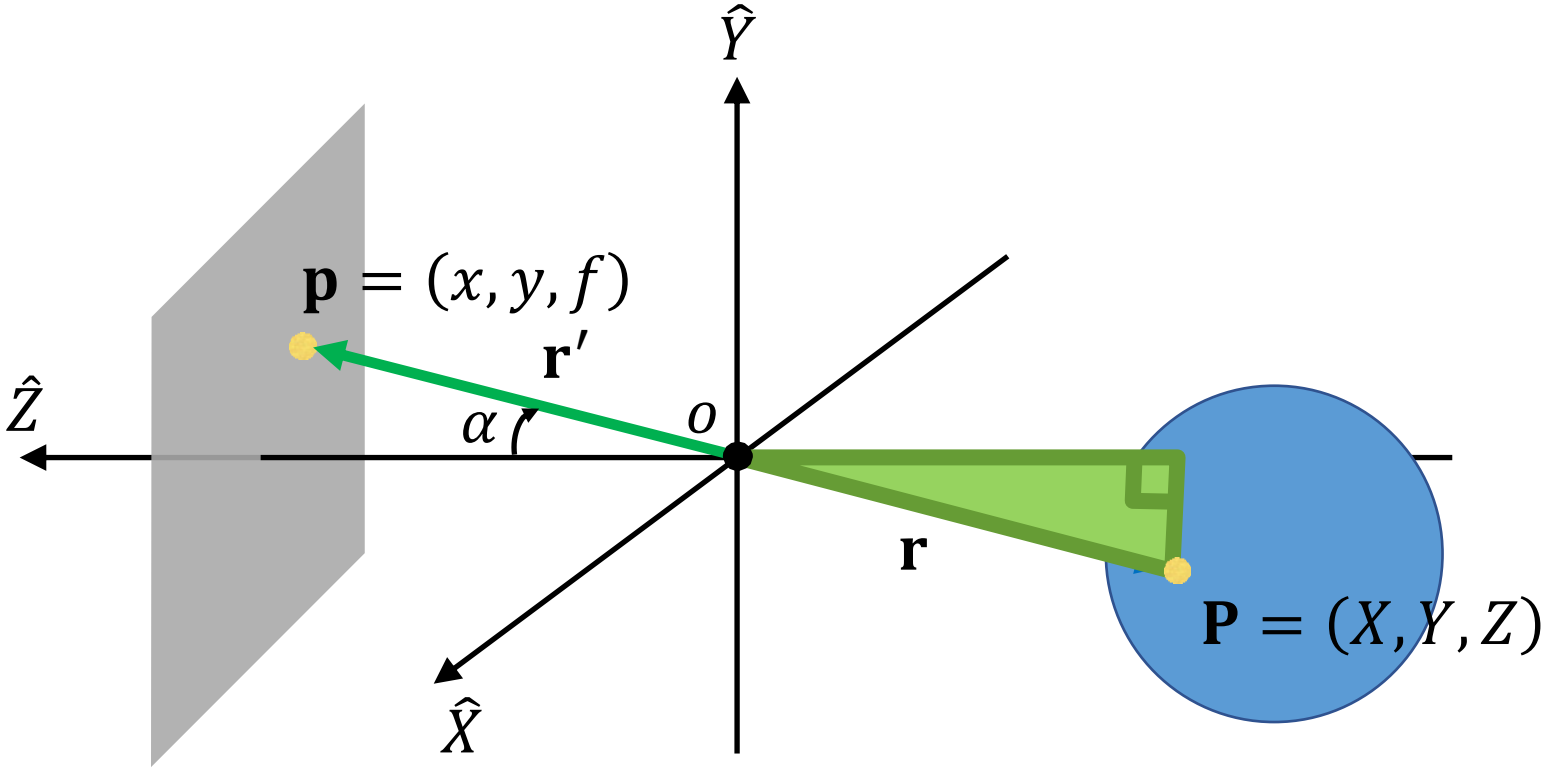
$$\frac{\mathbf{r}'}{\|\mathbf{r}'\|} = -\frac{\mathbf{r}}{\|\mathbf{r}\|}$$


$\hat{Y}$
$\mathbf{p} = (x, y, f)$
$\mathbf{r}'$
$\hat{Z}$
α
o
$\mathbf{r}$
$\mathbf{P} = (X, Y, Z)$
$\hat{X}$

r的长度?

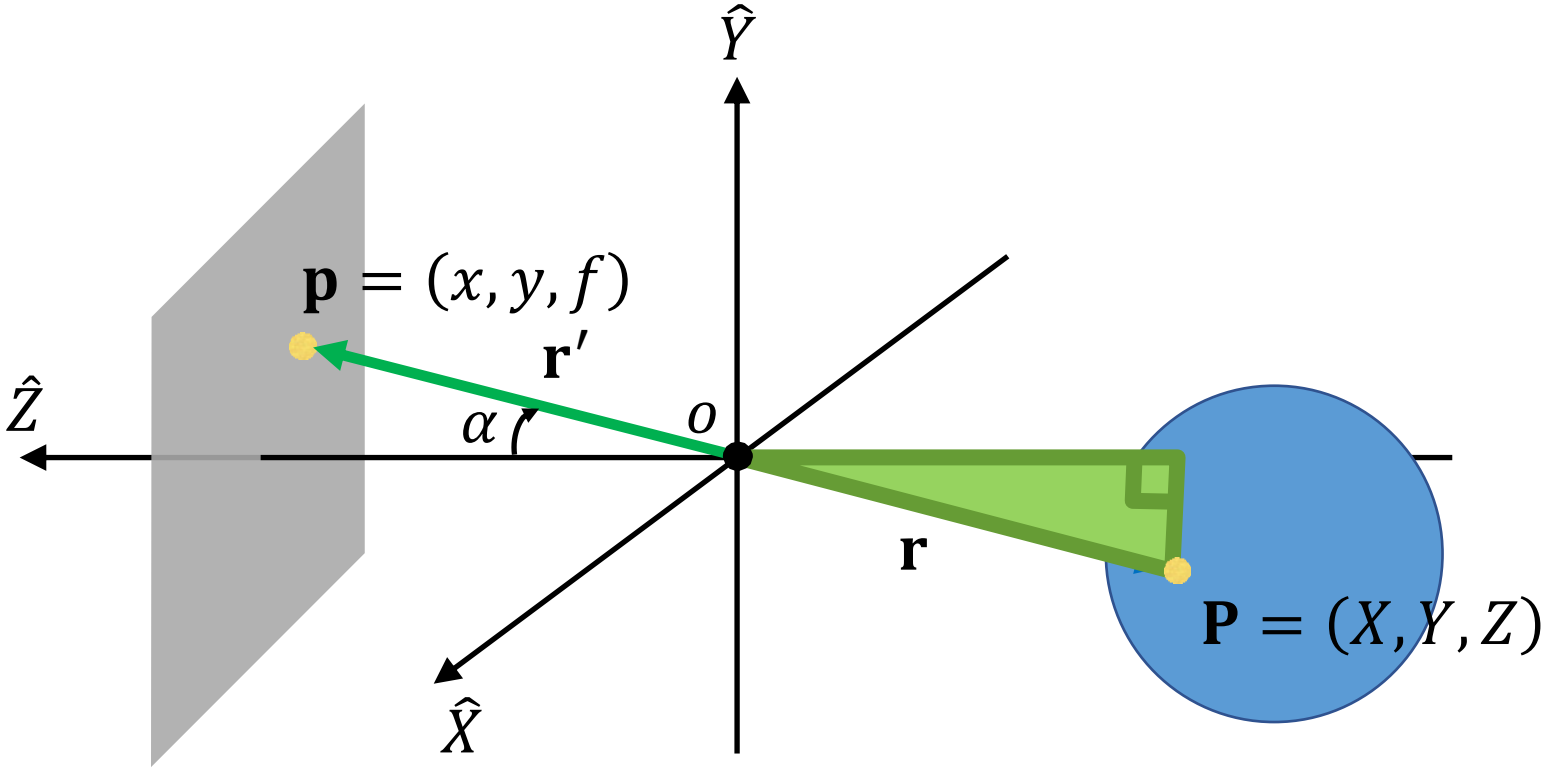

$\hat{Y}$
$\mathbf{p} = (x, y, f)$
$\mathbf{r}'$
$\hat{Z}$
α
o
$\mathbf{r}$
$\mathbf{P} = (X, Y, Z)$
$\hat{X}$

r的长度?



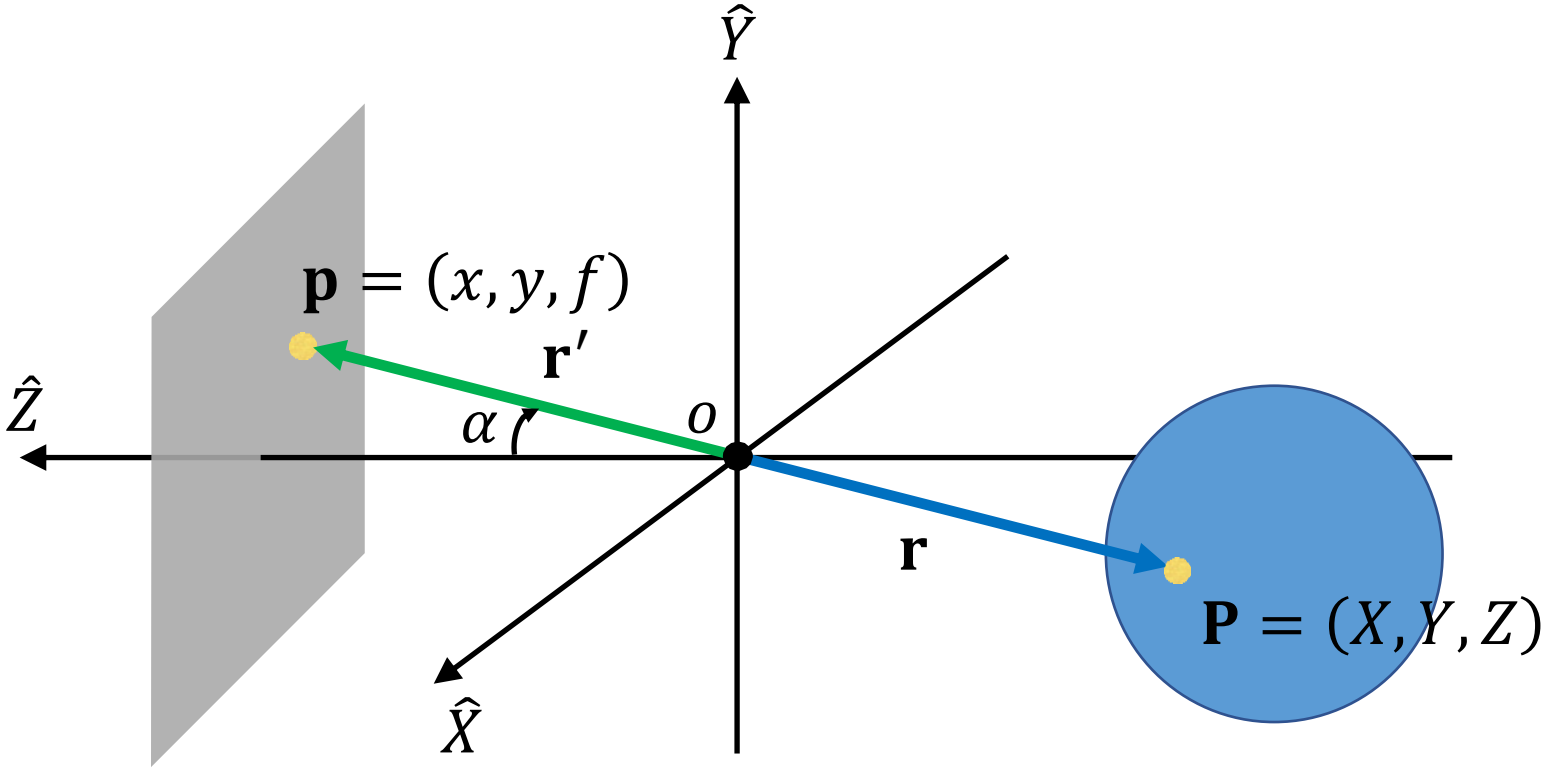
r的长度? $\cos\alpha = -\frac{Z}{\|\mathbf{r}\|}$

$$\|\mathbf{r}\| = -\frac{Z}{\cos\alpha}$$



r的长度? $\cos\alpha = -\frac{Z}{\|\mathbf{r}\|}$

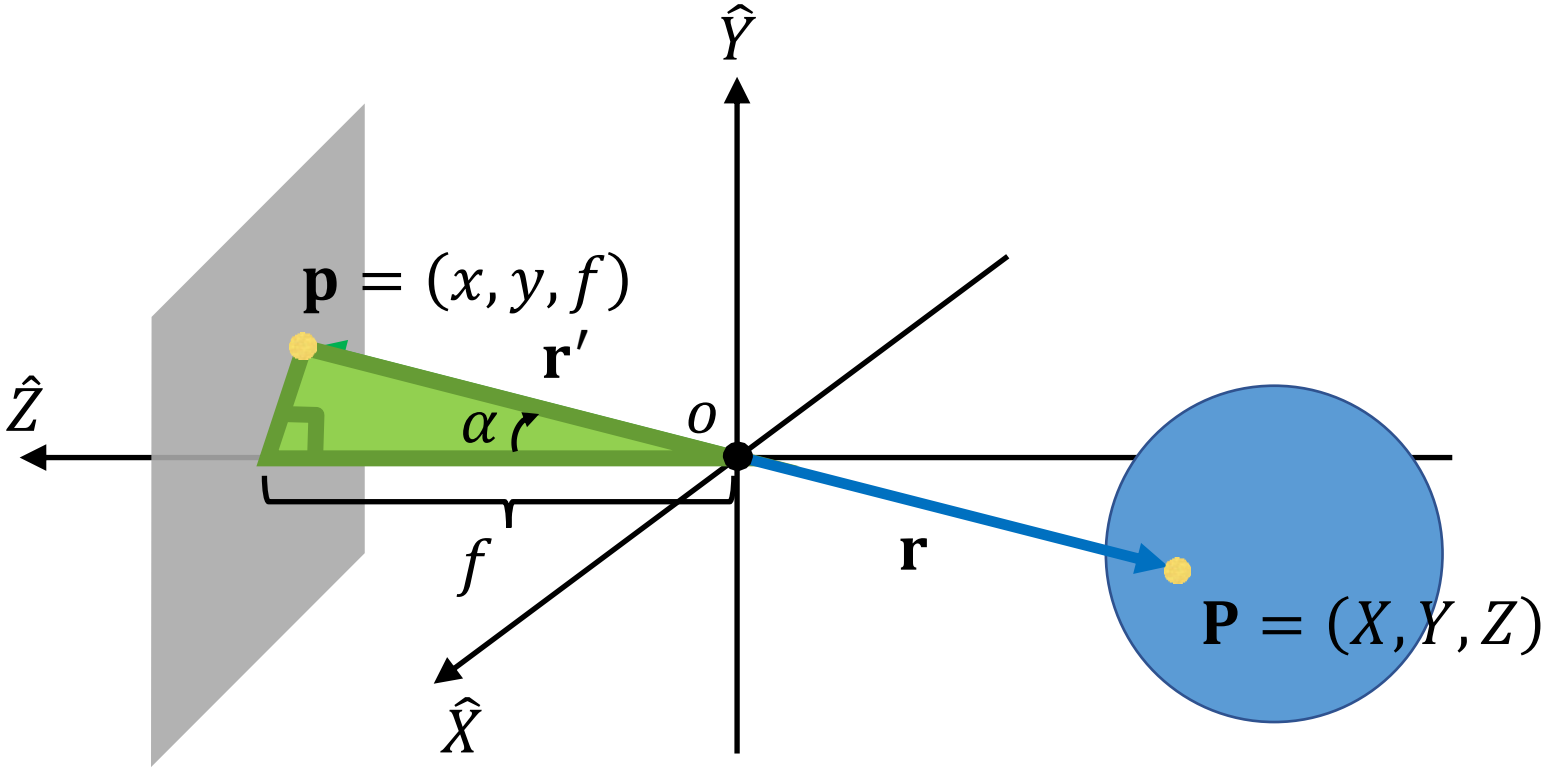
$$\|\mathbf{r}\| = -\frac{Z}{\cos\alpha} = -Z\sec\alpha$$



r的长度? $\cos\alpha = -\frac{Z}{\|\mathbf{r}\|}$

$$\|\mathbf{r}\| = -\frac{Z}{\cos\alpha} = -Z\sec\alpha$$

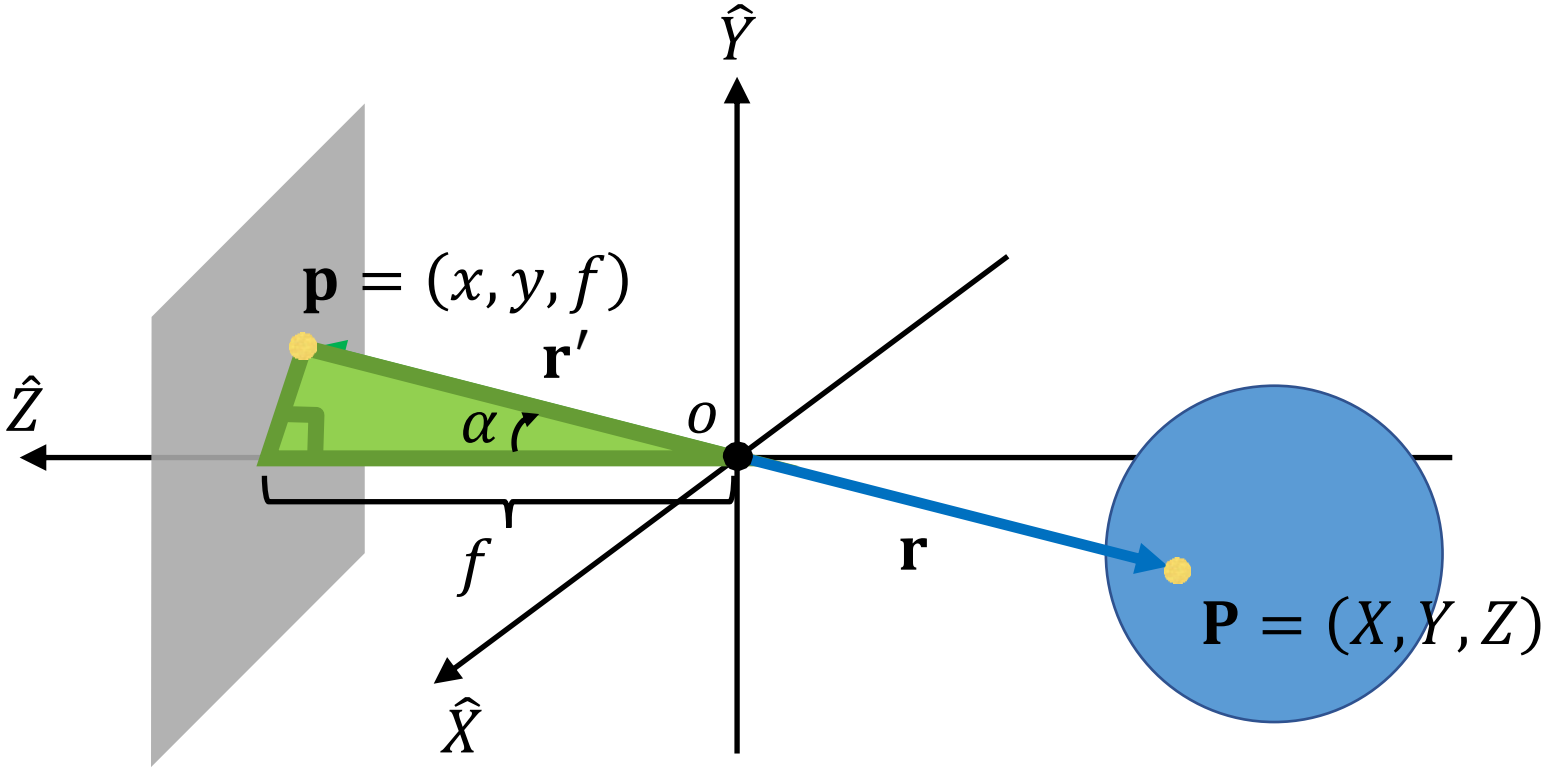
r′的长度?



r的长度? $\cos\alpha = -\frac{Z}{\|\mathbf{r}\|}$

$$\|\mathbf{r}\| = -\frac{Z}{\cos\alpha} = -Z\sec\alpha$$

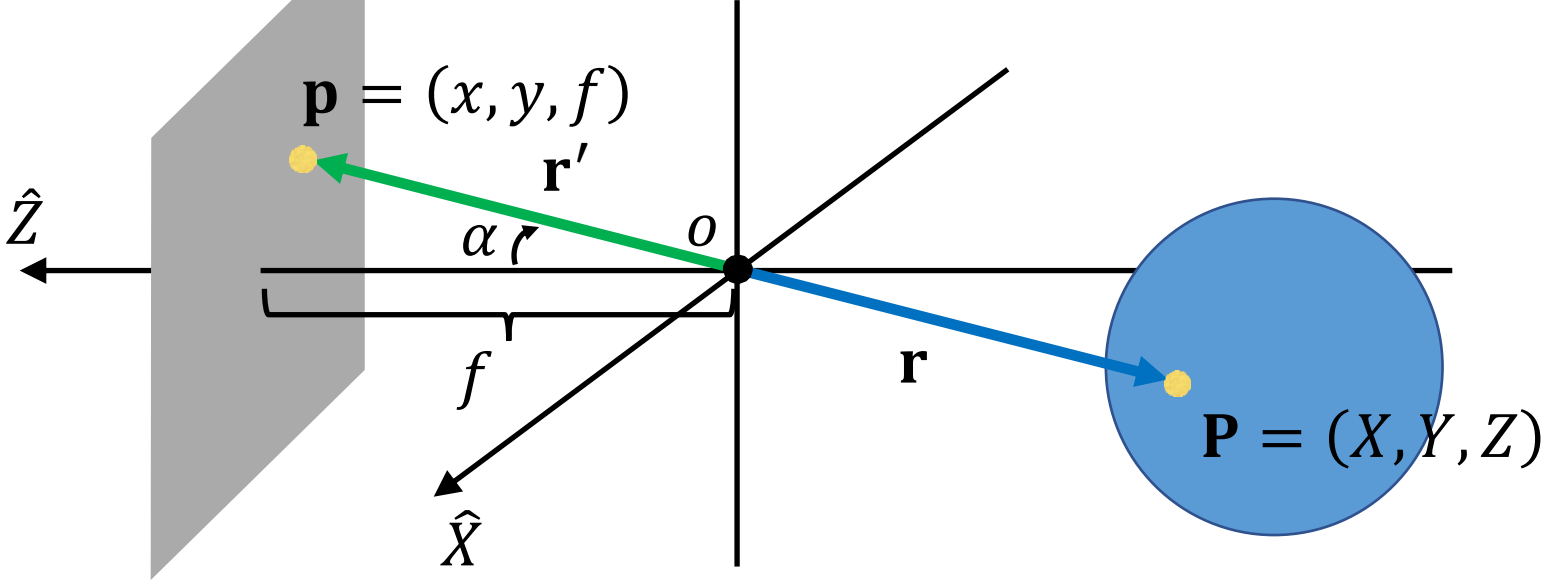
r′的长度?



r的长度? $\cos\alpha = -\frac{Z}{\|\mathbf{r}\|}$

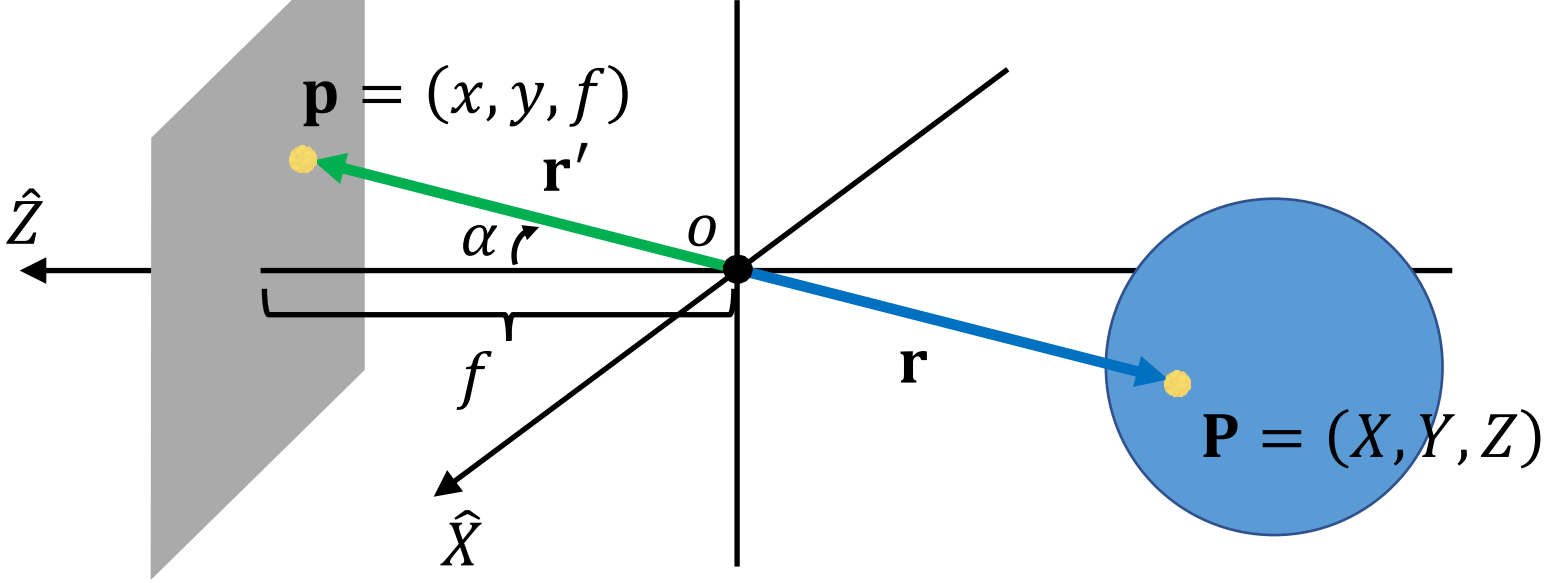
$$\|\mathbf{r}\| = -\frac{Z}{\cos\alpha} = -Z\sec\alpha$$

r′的长度? $\|\mathbf{r}'\| = f\sec\alpha$



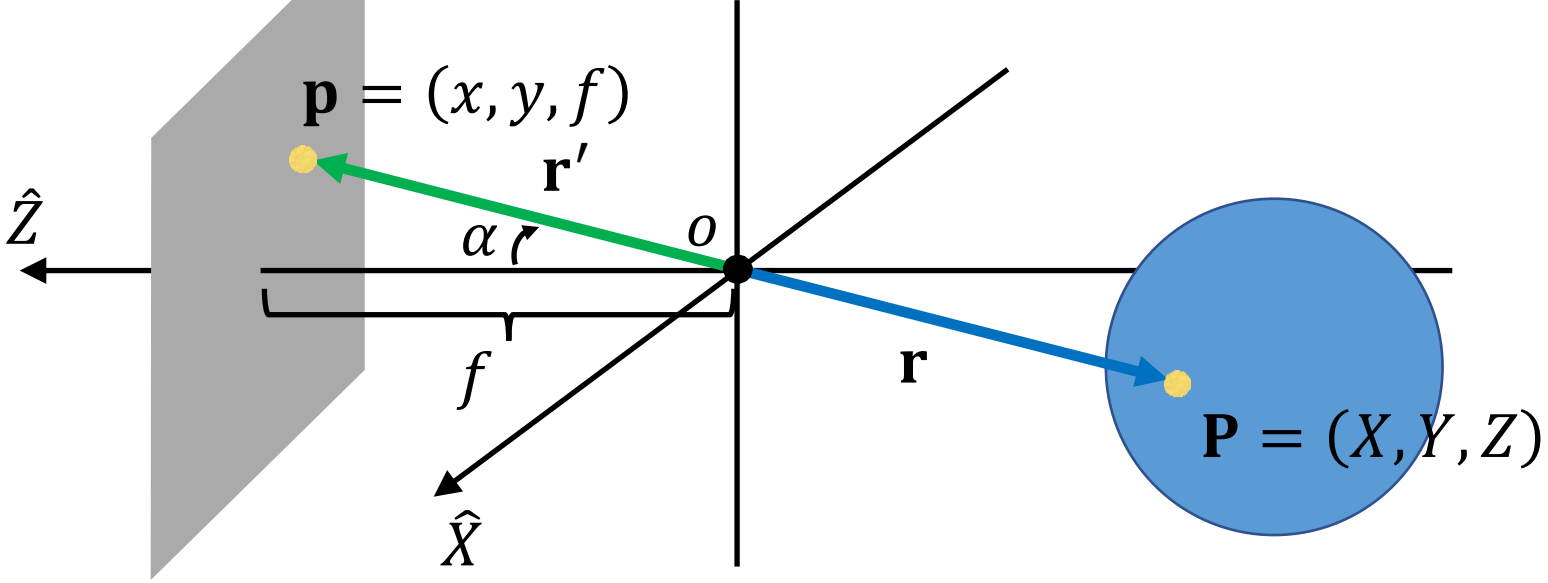
r的长度? $\|\mathbf{r}\| = -Z \sec \alpha$

r′的长度? $\|\mathbf{r}'\| = f \sec \alpha$



r的长度? $\|\mathbf{r}\| = -Z\sec\alpha$ **r′的长度?** $\|\mathbf{r}'\| = f\sec\alpha$

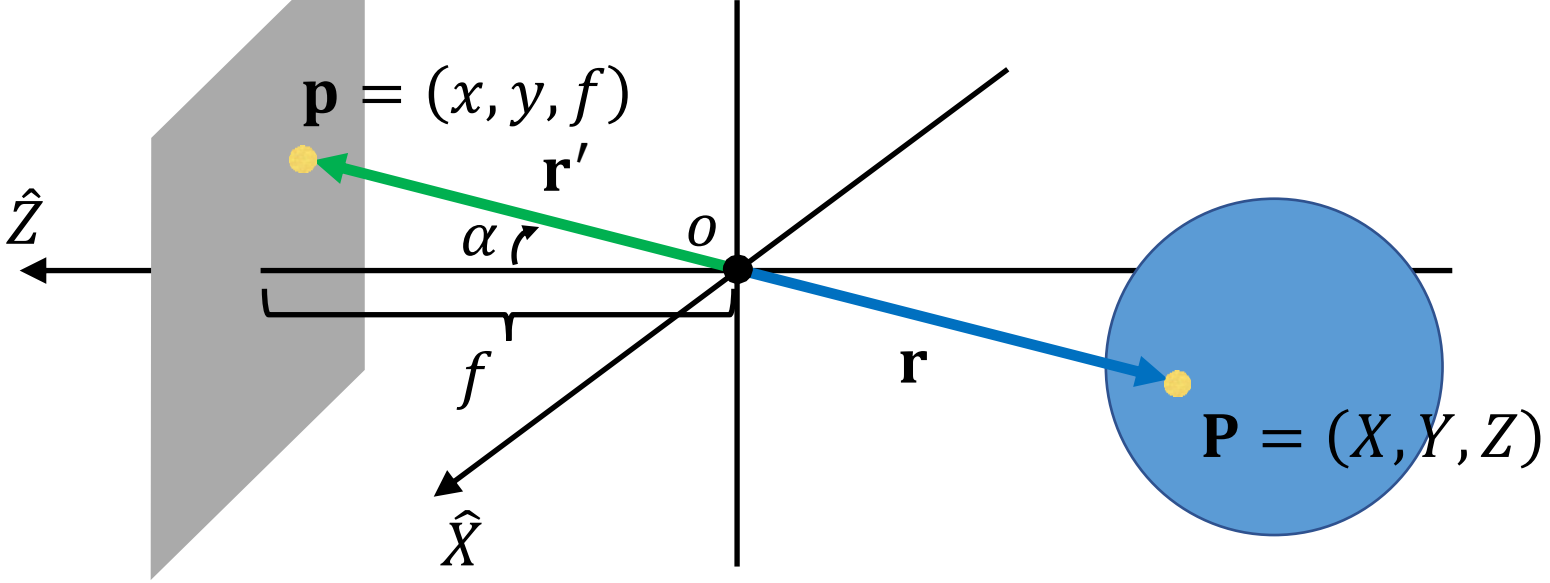
回顾: $\dfrac{\mathbf{r}'}{\|\mathbf{r}'\|} = -\dfrac{\mathbf{r}}{\|\mathbf{r}\|}$



$\mathbf{r}$的长度？ $\|\mathbf{r}\| = -Z \sec\alpha$　　$\mathbf{r}'$的长度？ $\|\mathbf{r}'\| = f \sec\alpha$

回顾： $$\frac{\mathbf{r}'}{\|\mathbf{r}'\|} = -\frac{\mathbf{r}}{\|\mathbf{r}\|}$$

$$\frac{1}{f}\mathbf{r}' = \frac{1}{Z}\mathbf{r}$$

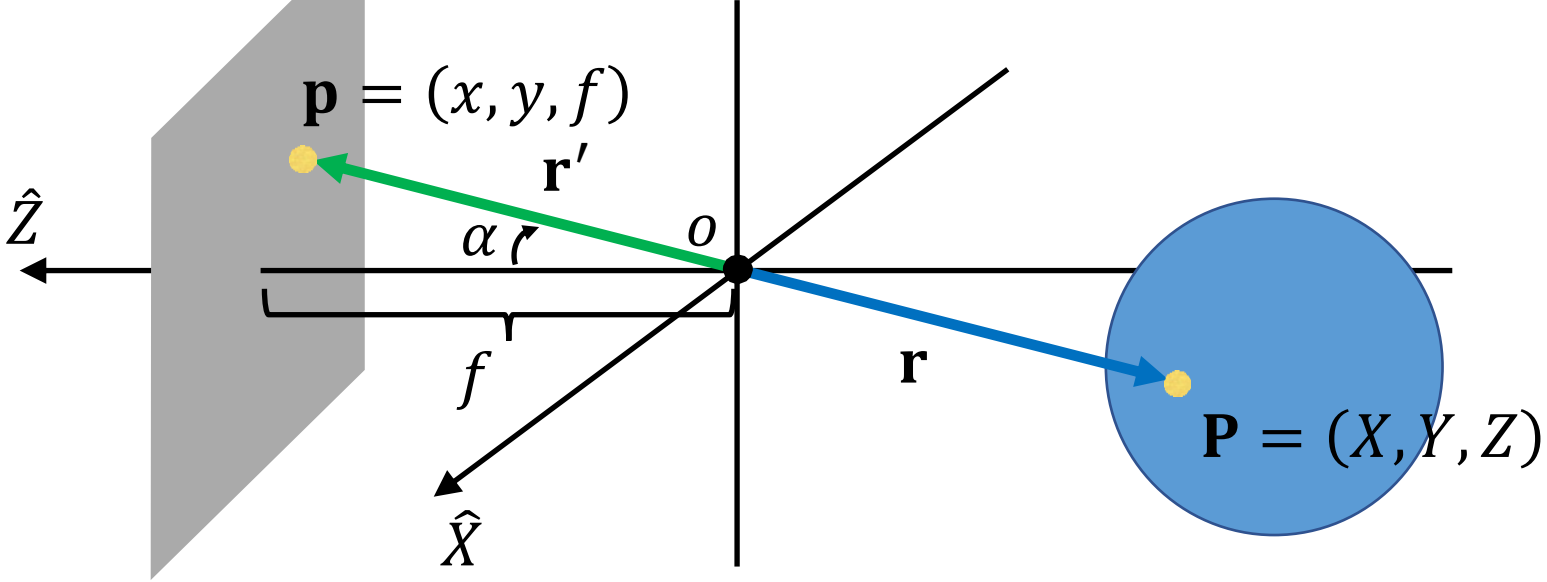


r的长度? $\|\mathbf{r}\| = -Z\sec\alpha$ **r′的长度?** $\|\mathbf{r}'\| = f\sec\alpha$

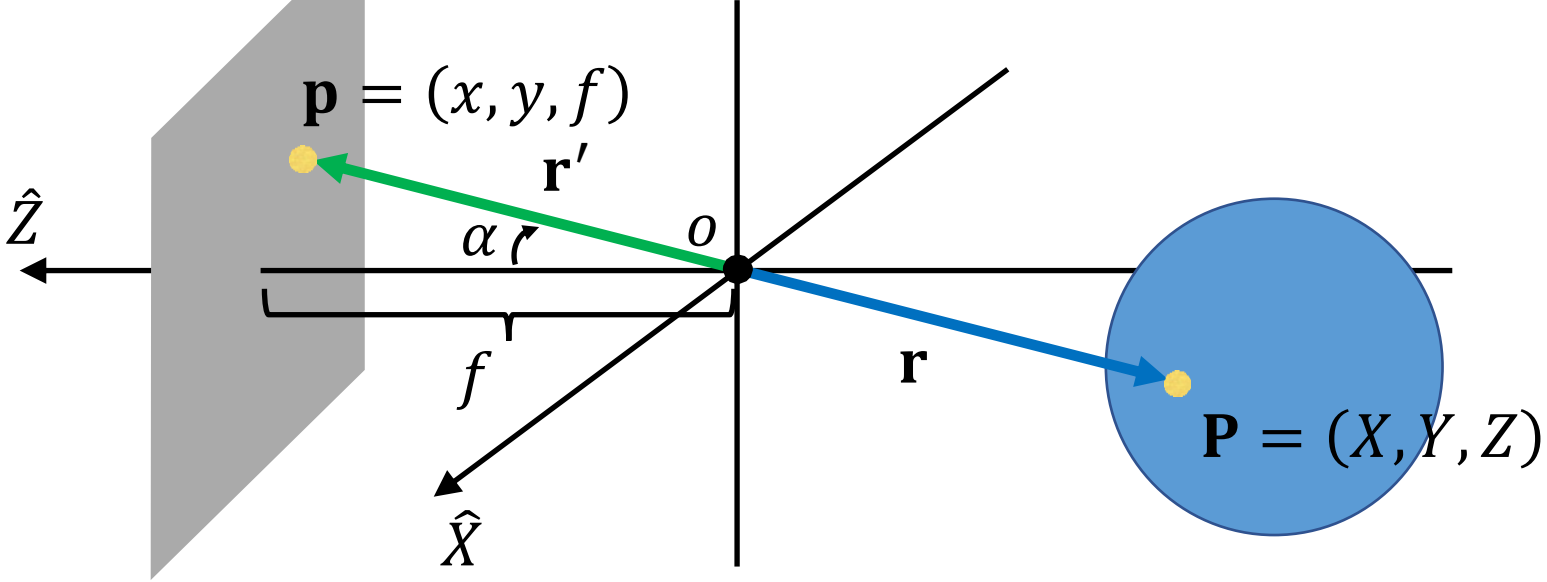
回顾: $$\frac{\mathbf{r}'}{\|\mathbf{r}'\|} = -\frac{\mathbf{r}}{\|\mathbf{r}\|}$$

$$\frac{1}{f}\mathbf{r}' = \frac{1}{Z}\mathbf{r}$$

$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

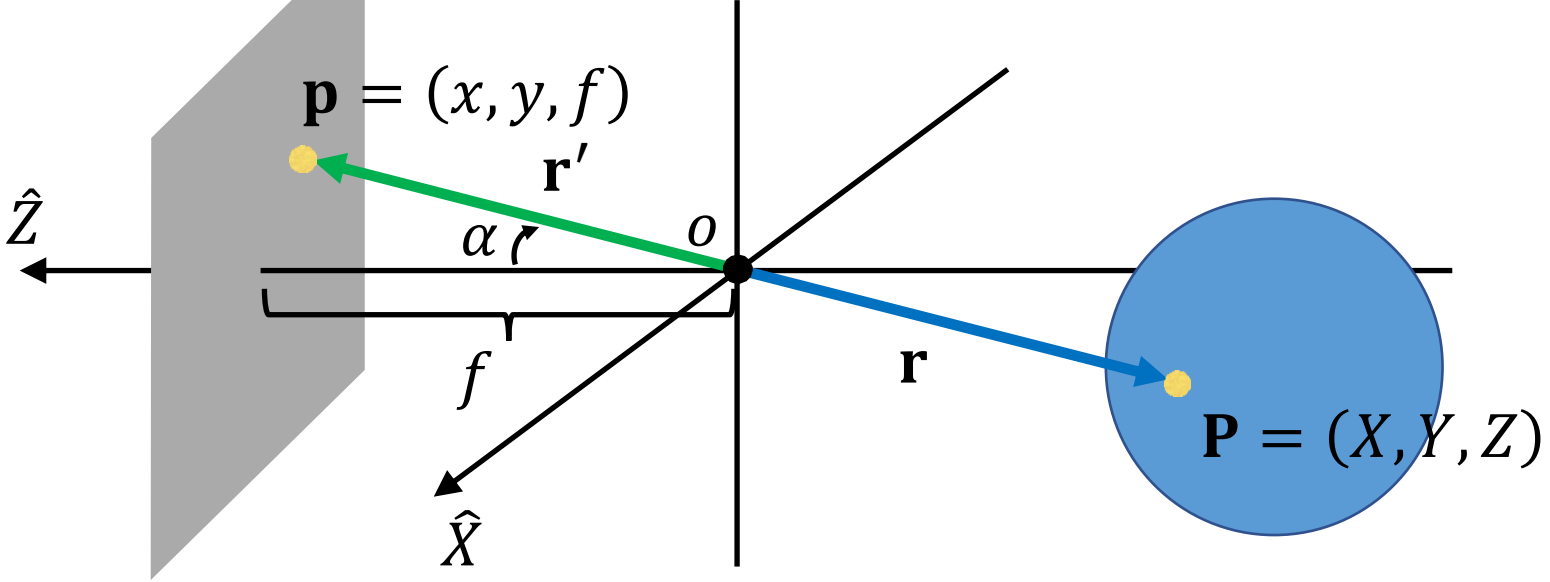


$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$



$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

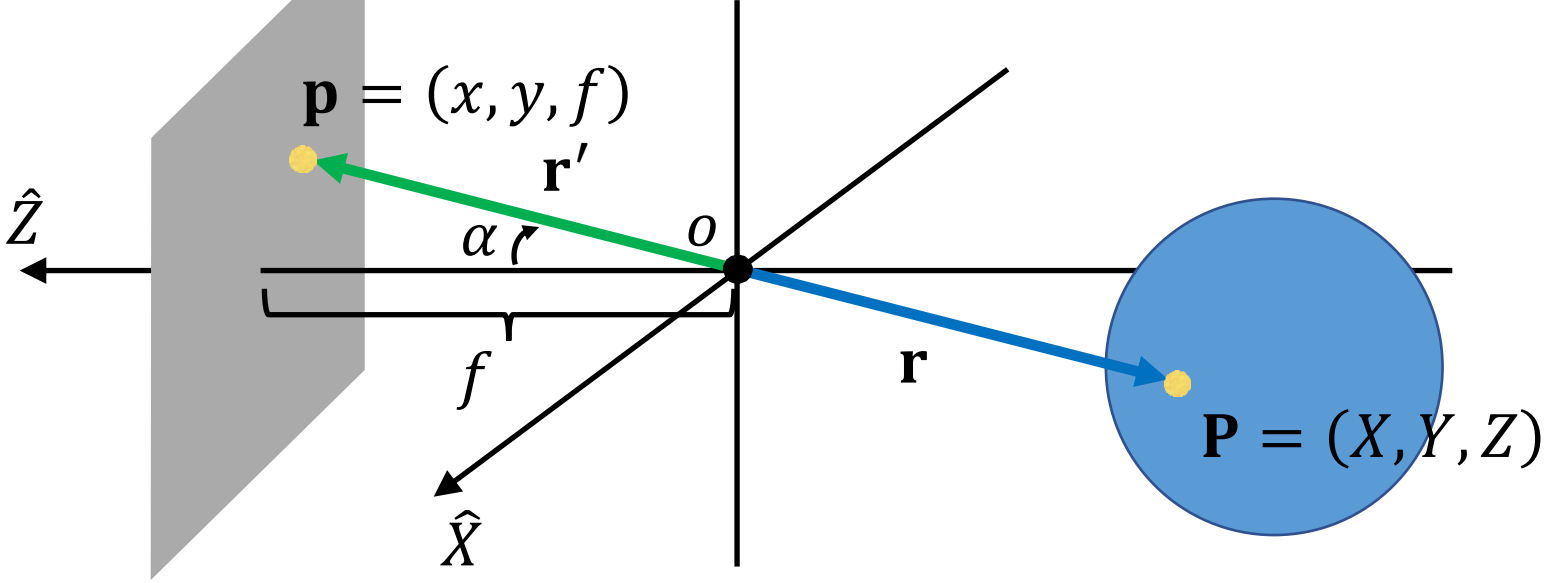
透视投影： $x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$



$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

透视投影： $x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$

注意到投影图像有什么问题吗？

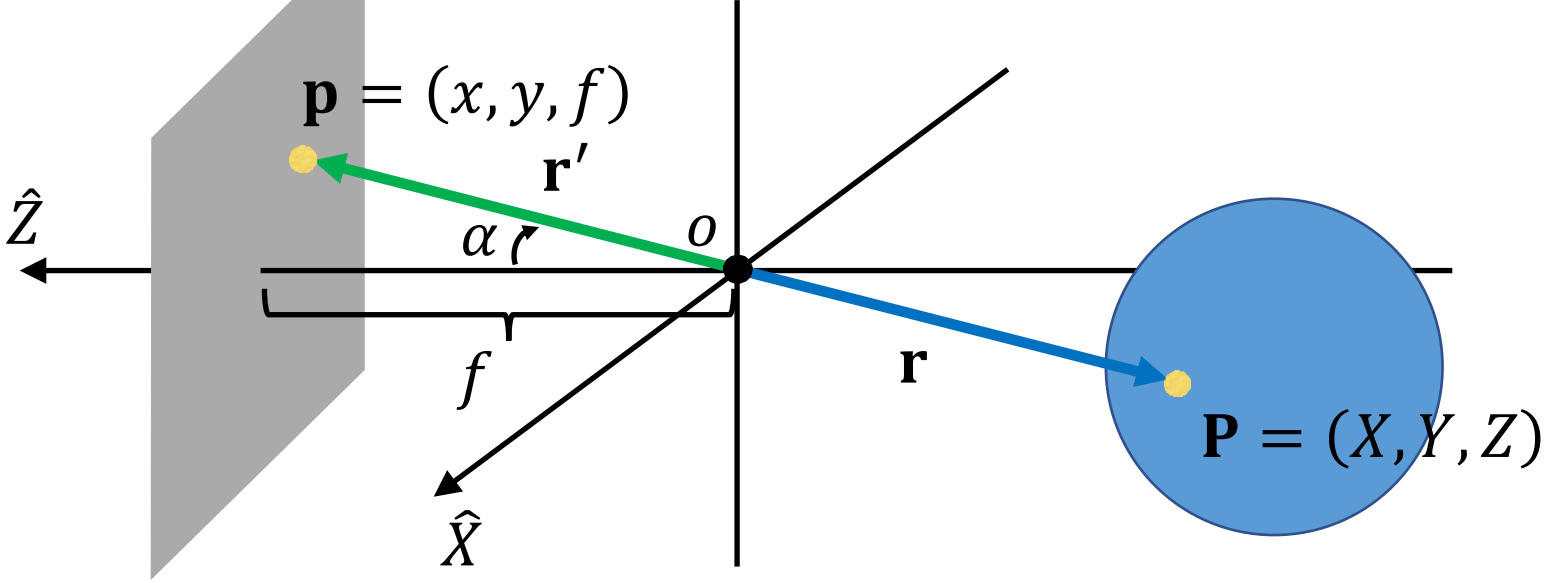


$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

透视投影： $x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$

注意到投影图像有什么问题吗？

颠倒了！



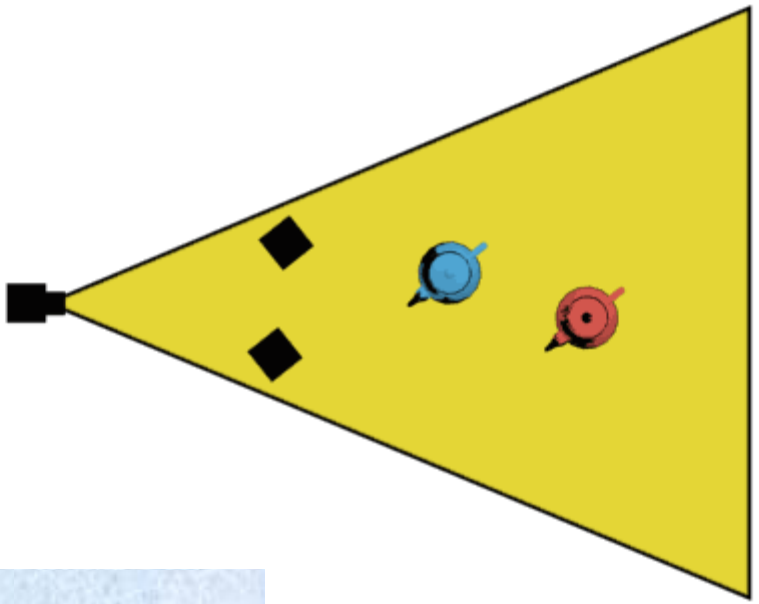
$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

透视投影： $x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$

透视投影（颠倒的）： $x = -f\frac{X}{Z}, \quad y = -f\frac{Y}{Z}$

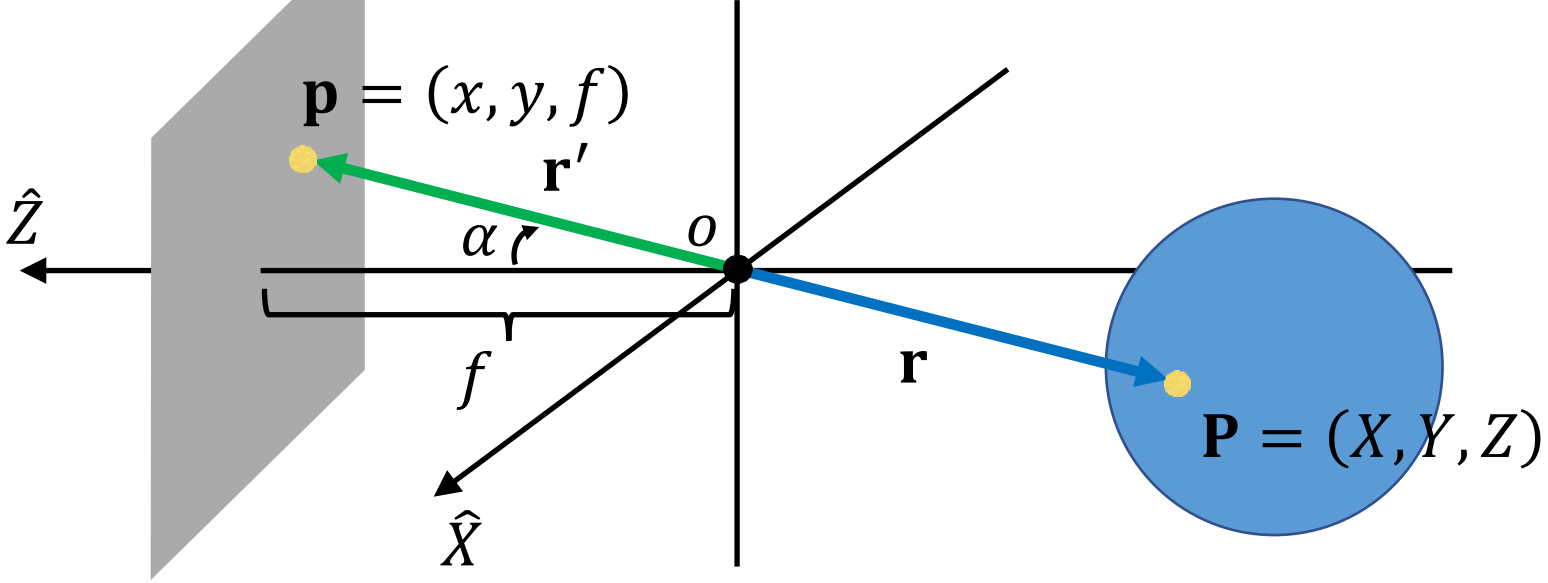
滑动变焦


JAMES STEWART
KIM NOVAK
IN ALFRED HITCHCOCK'S
MASTERPIECE
'VERTIGO'



透视投影：$x = f\frac{X}{Z}$，$y = f\frac{Y}{Z}$

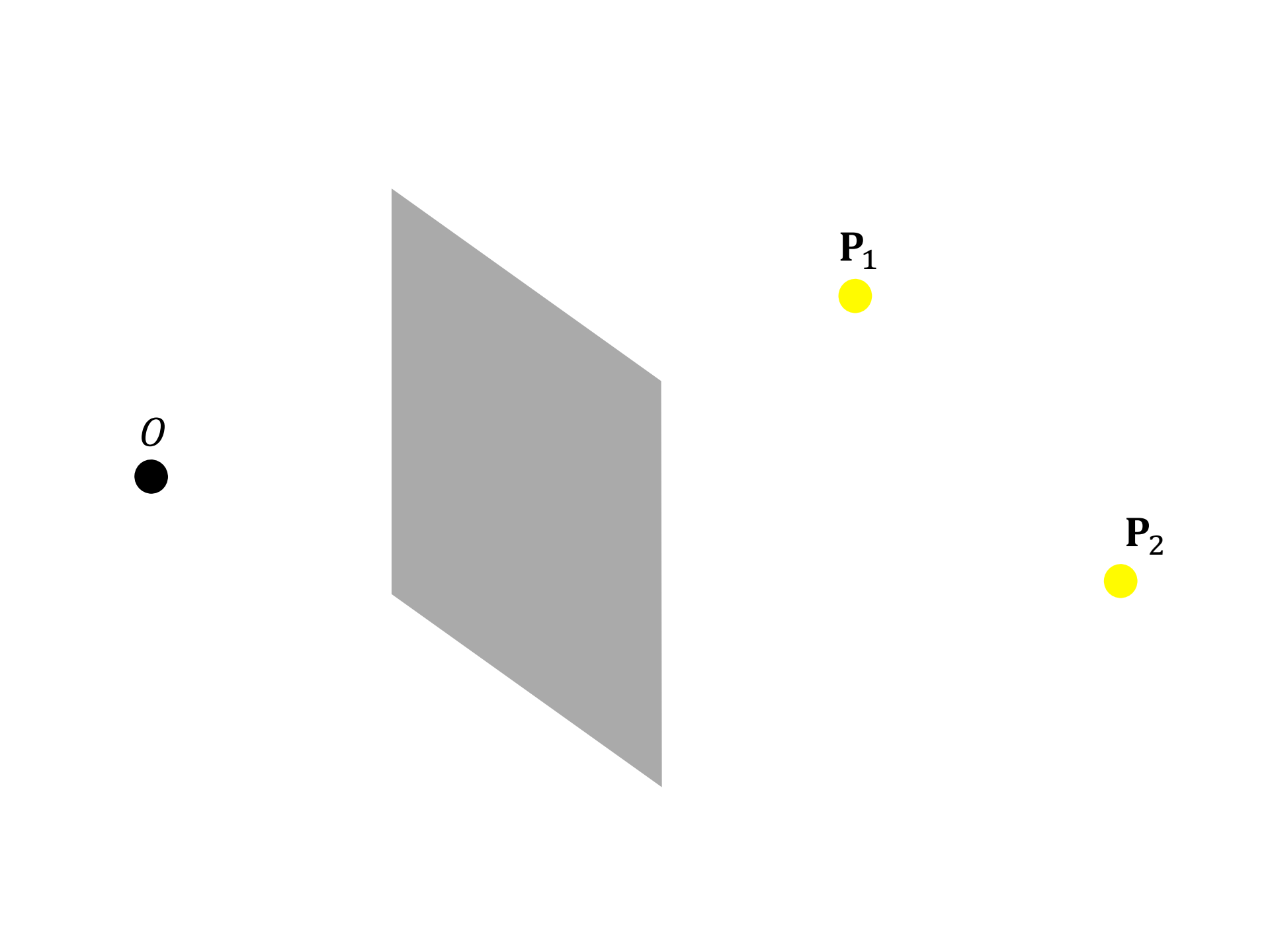
VashiVisuals.com/blog

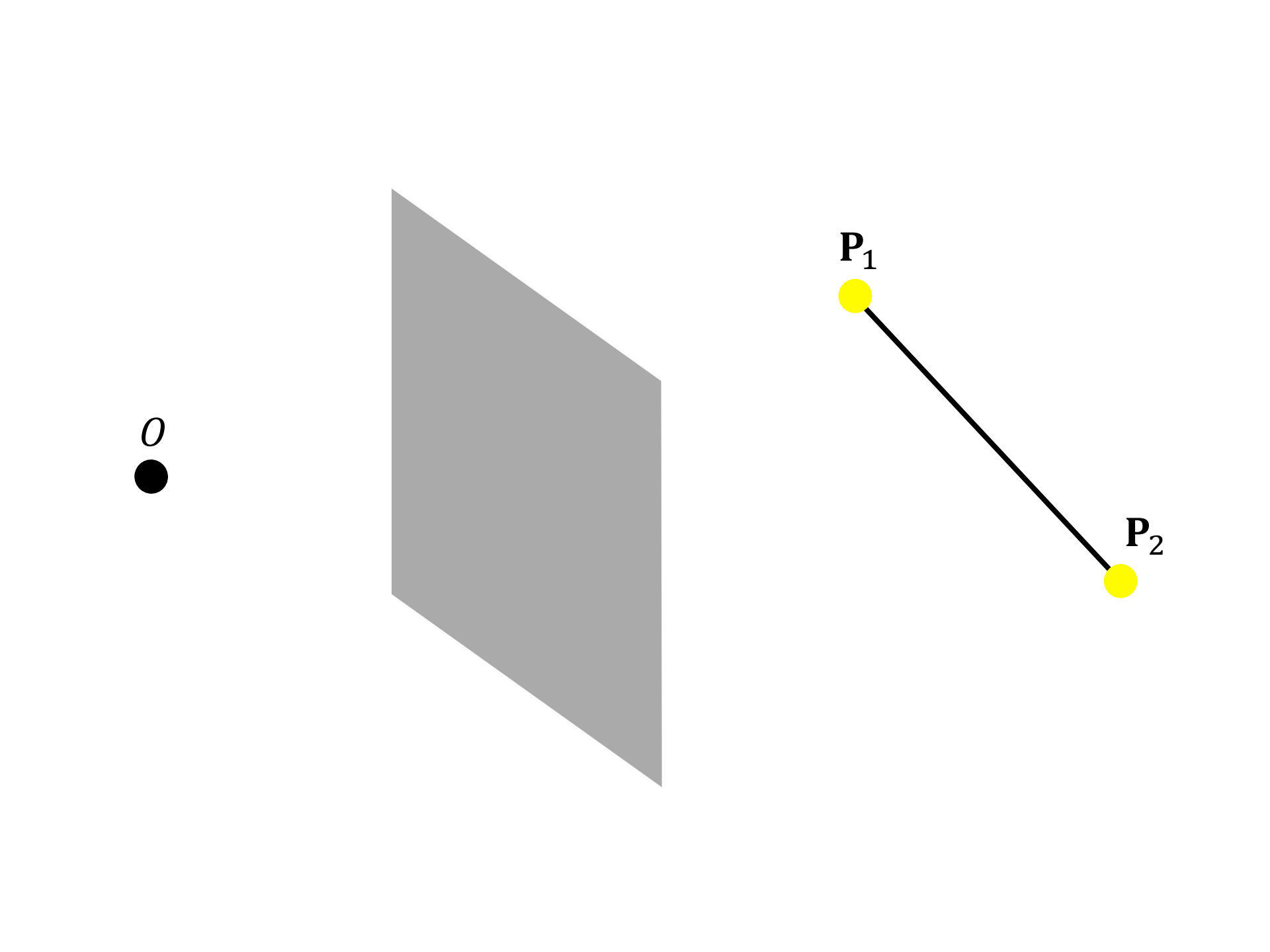


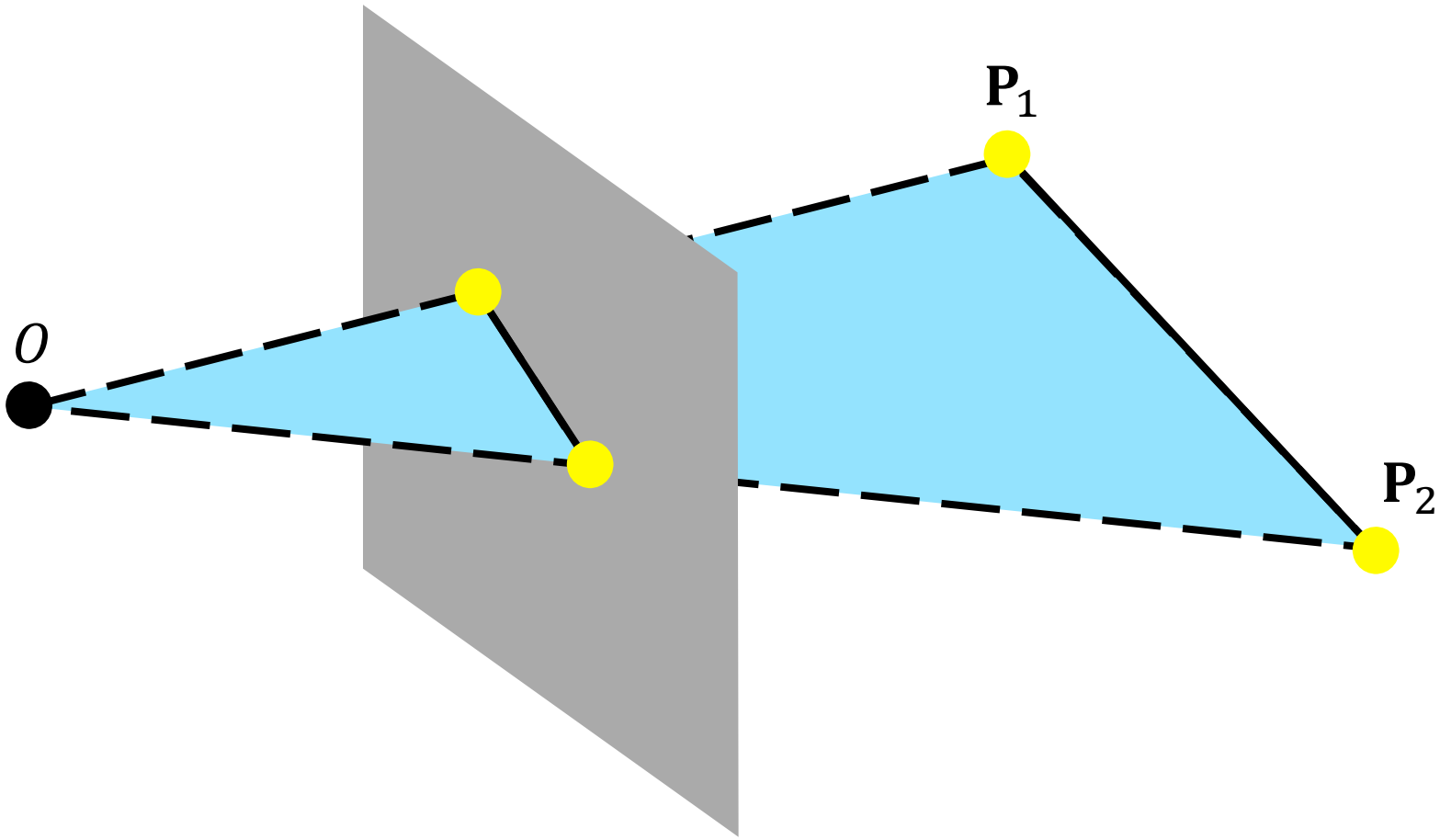
$$\frac{x}{f} = \frac{X}{Z}, \quad \frac{y}{f} = \frac{Y}{Z}$$

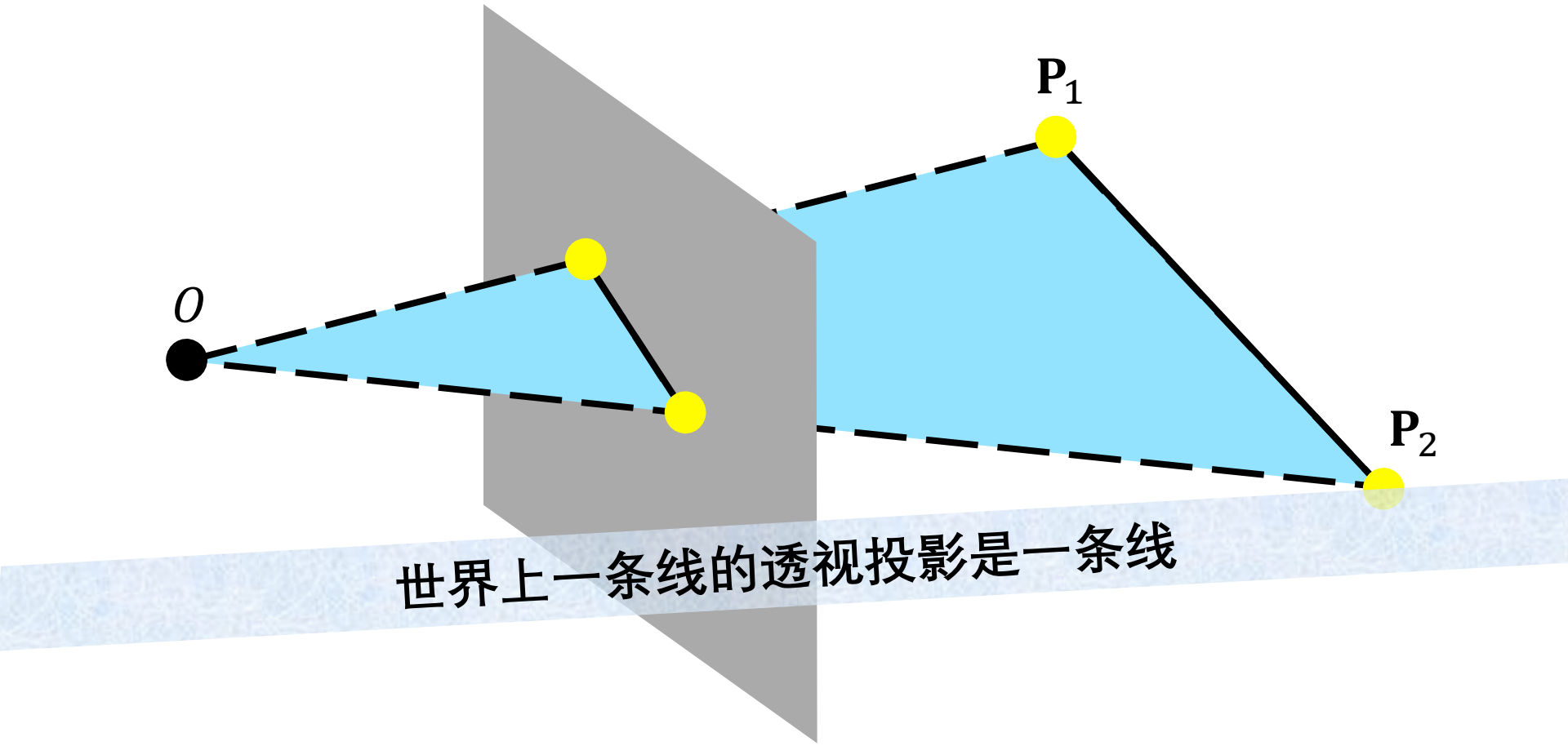
透视投影： $x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}$

透视投影（颠倒的）： $x = -f\frac{X}{Z}, \quad y = -f\frac{Y}{Z}$

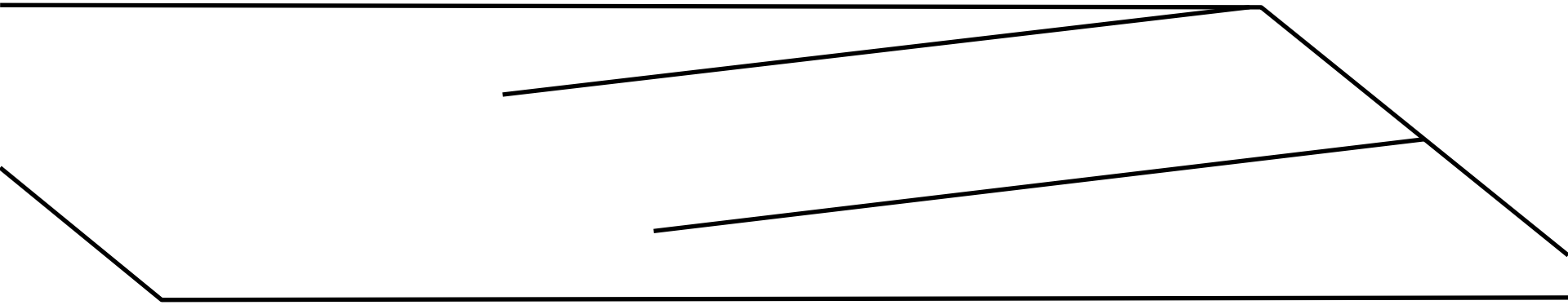

$\mathbf{P}_1$
O
$\mathbf{P}_2$

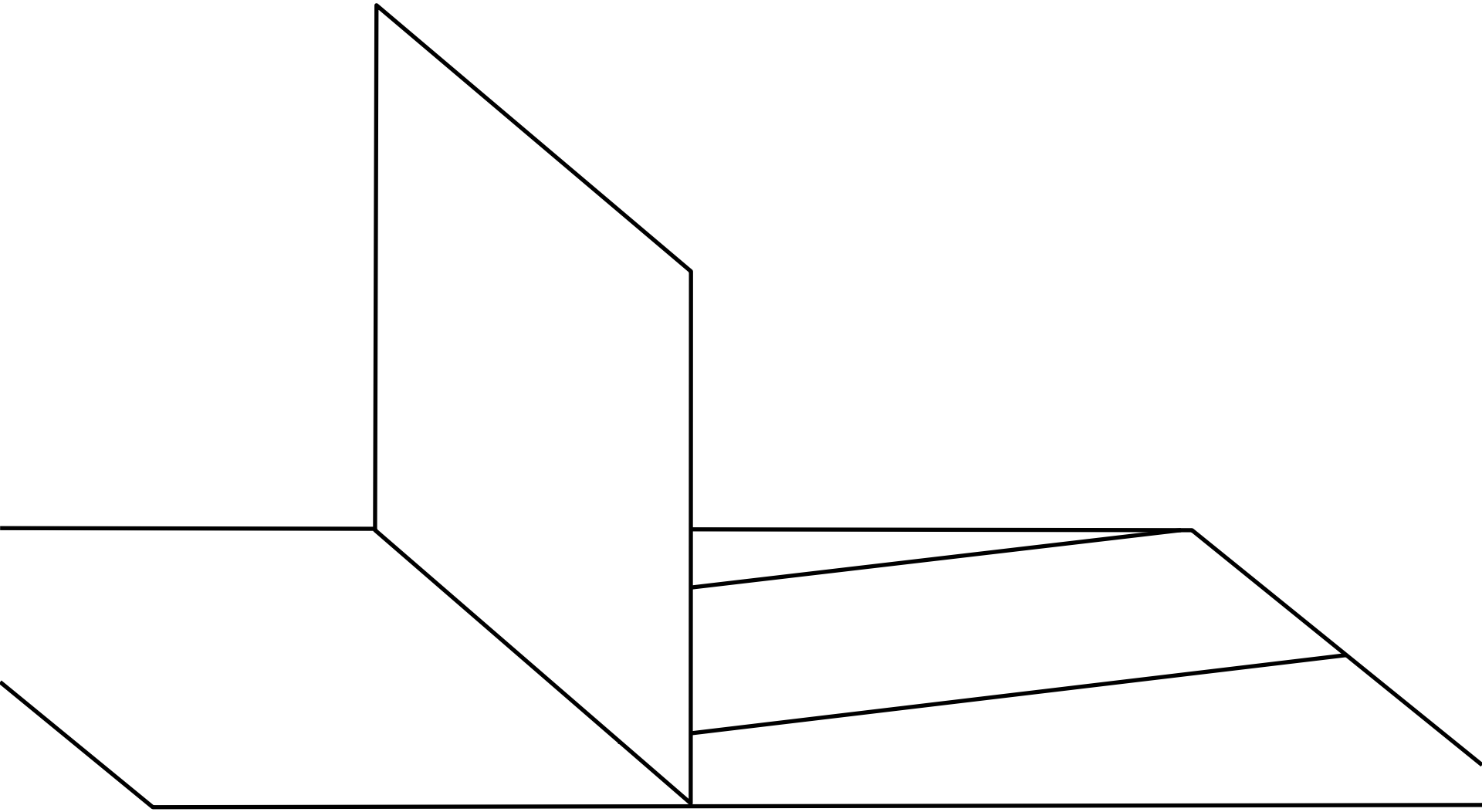

O
P1
P2

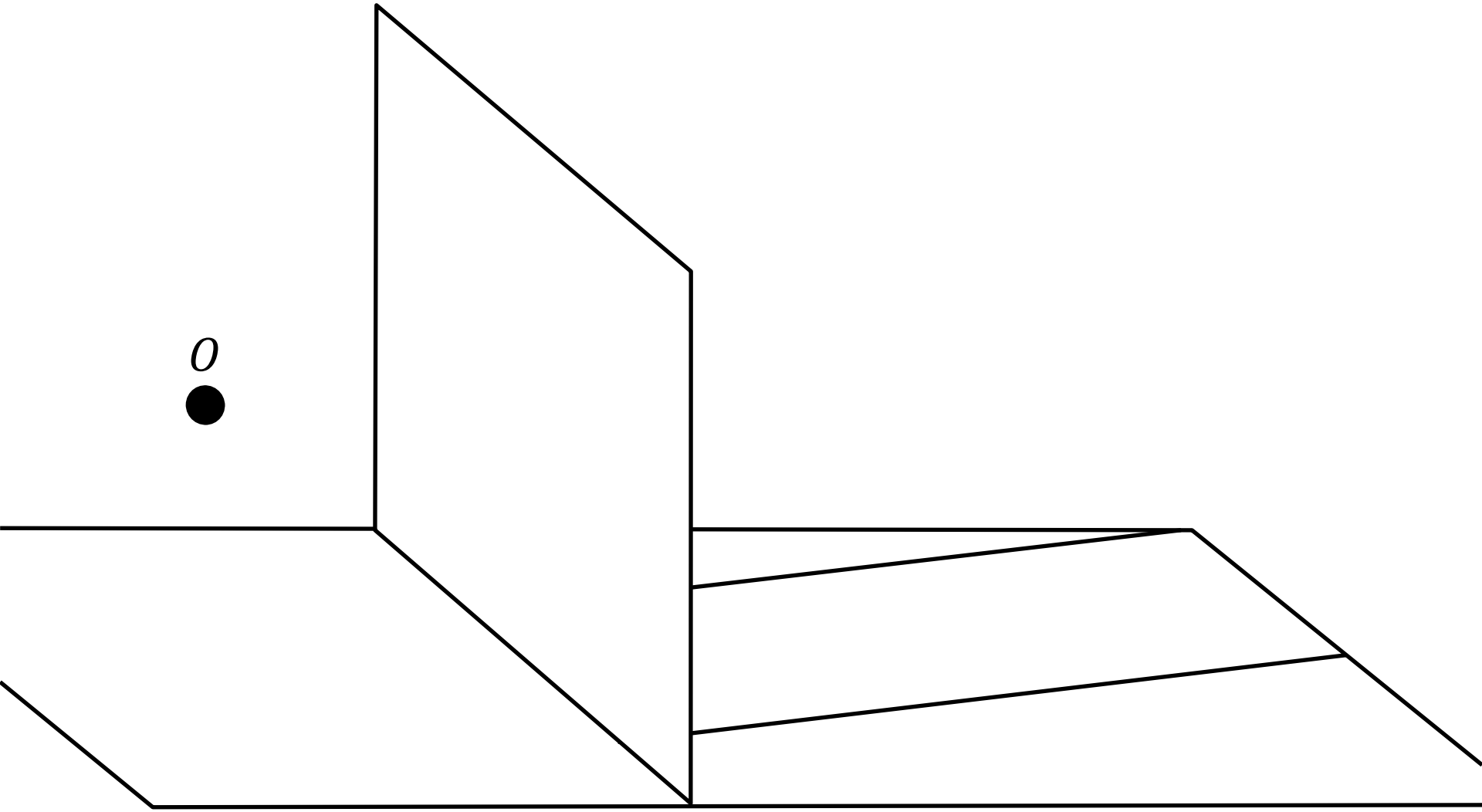

$\mathbf{P}_1$
O
$\mathbf{P}_2$

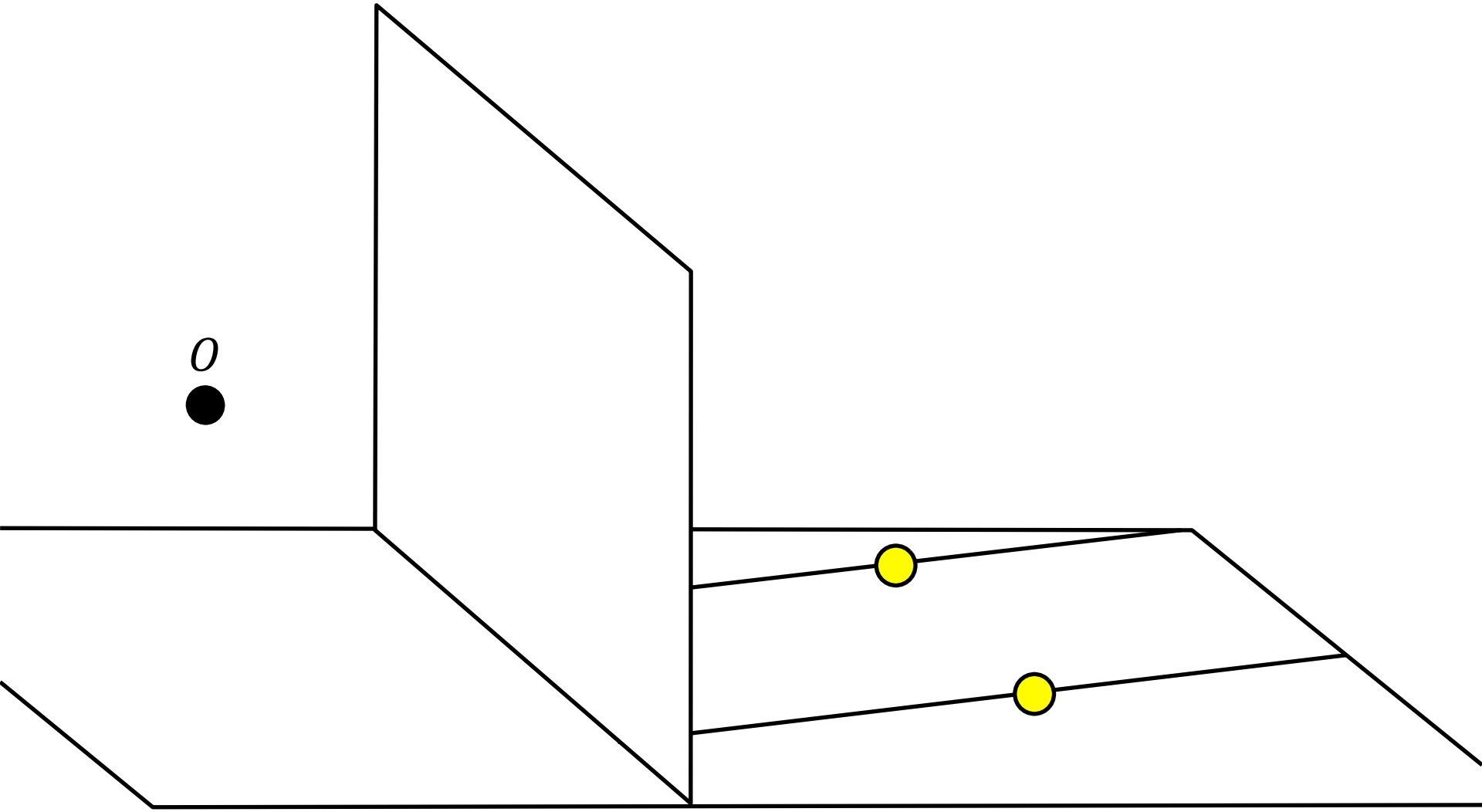

P_1
O
P_2
世界上一条线的透视投影是一条线

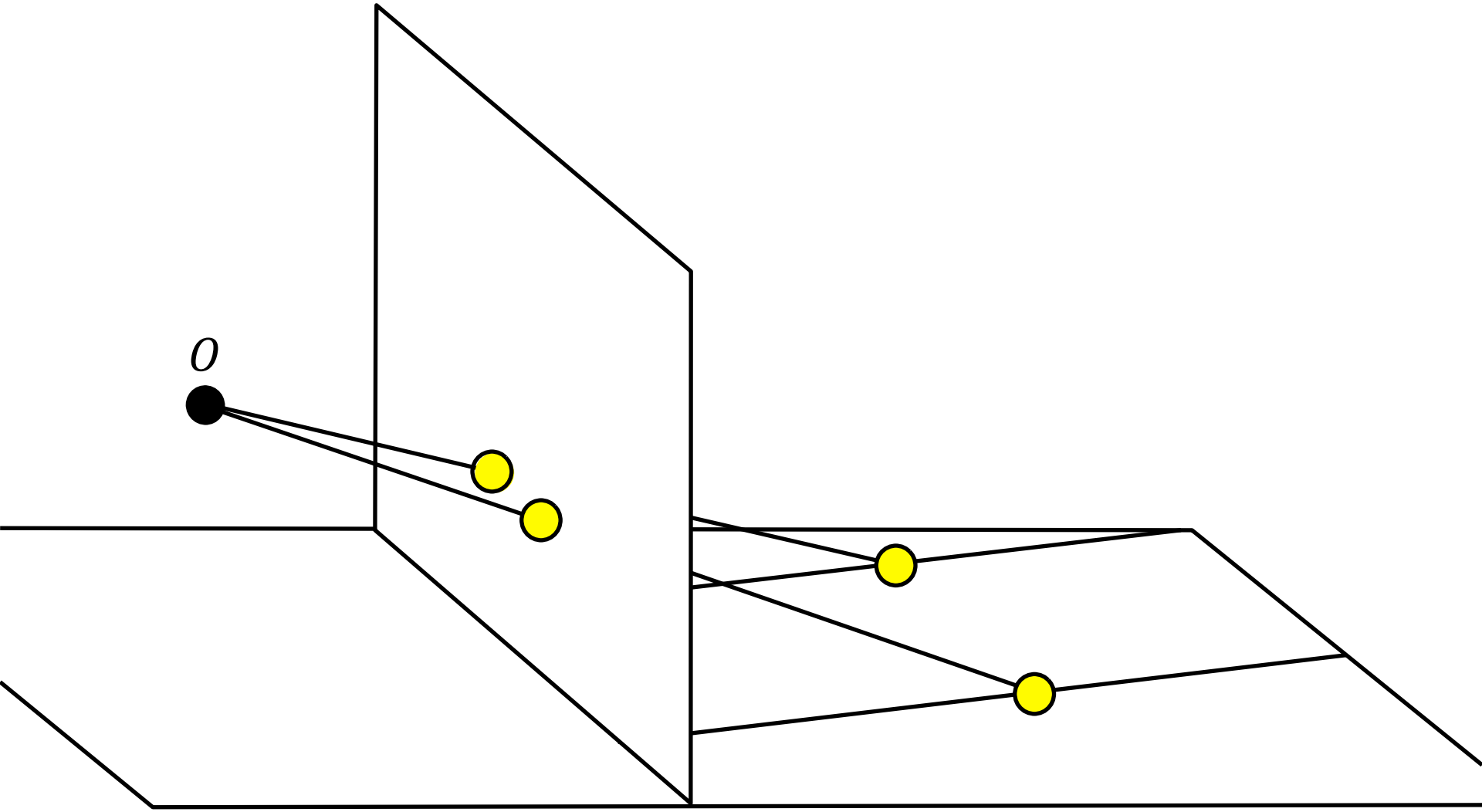

消失点

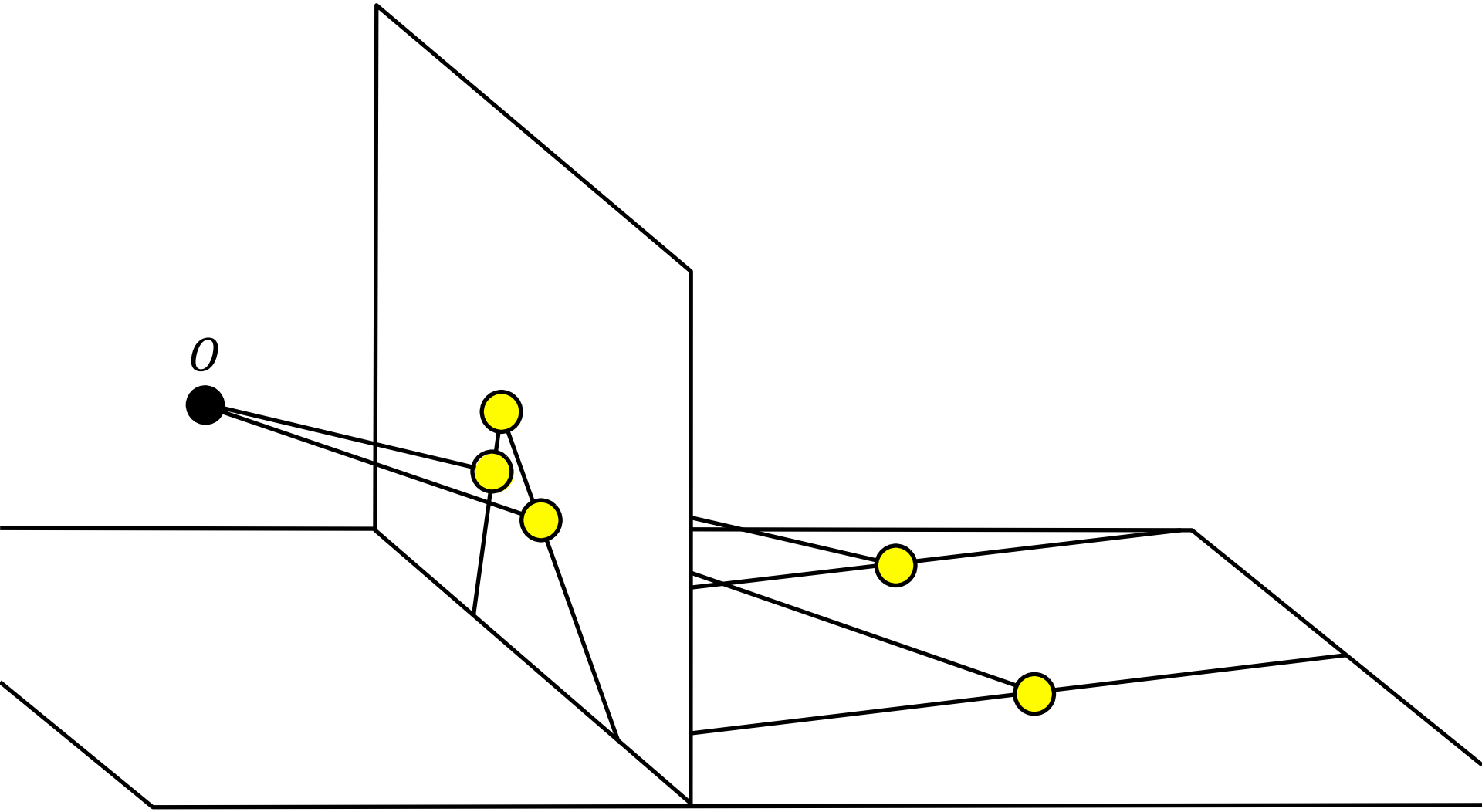


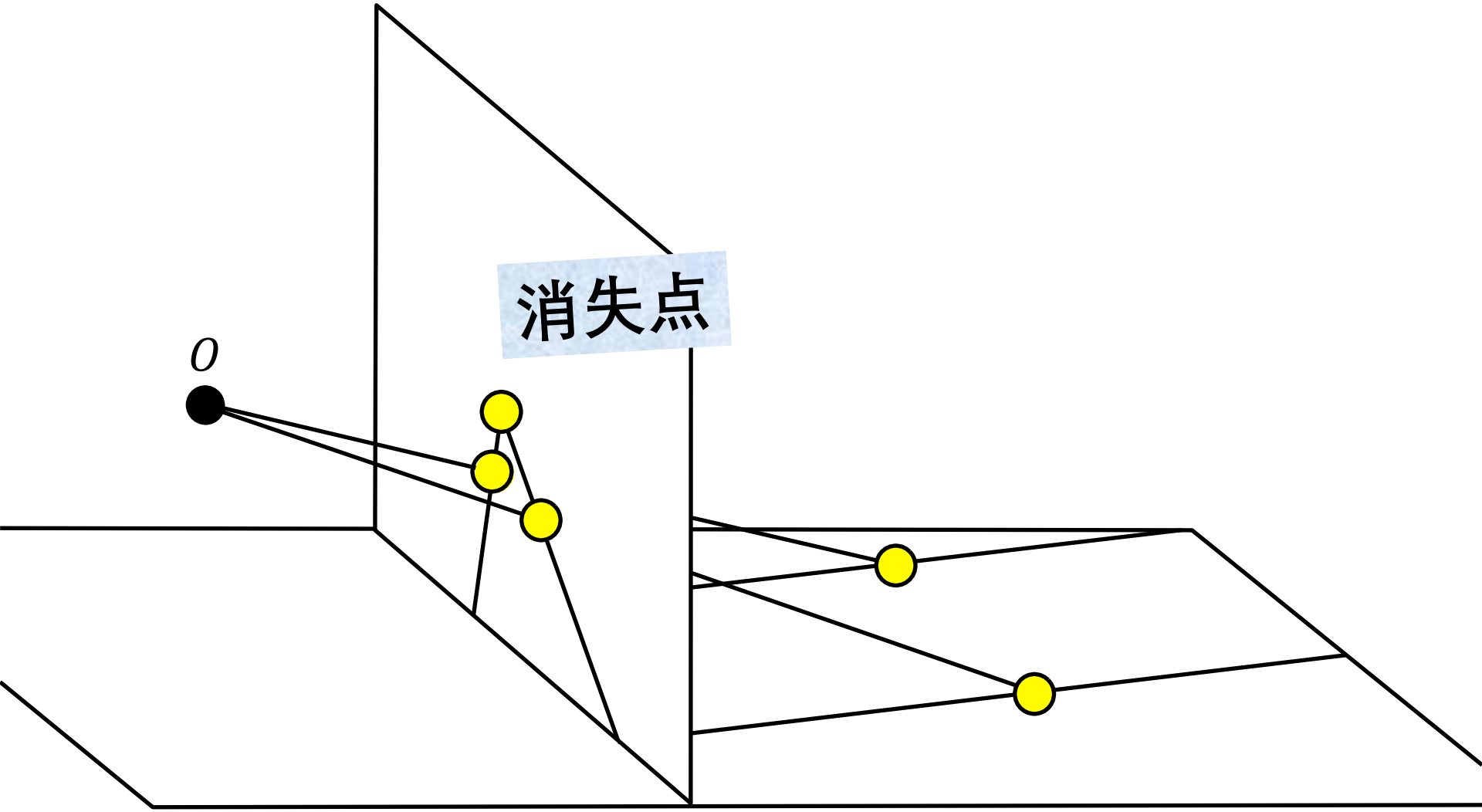



O

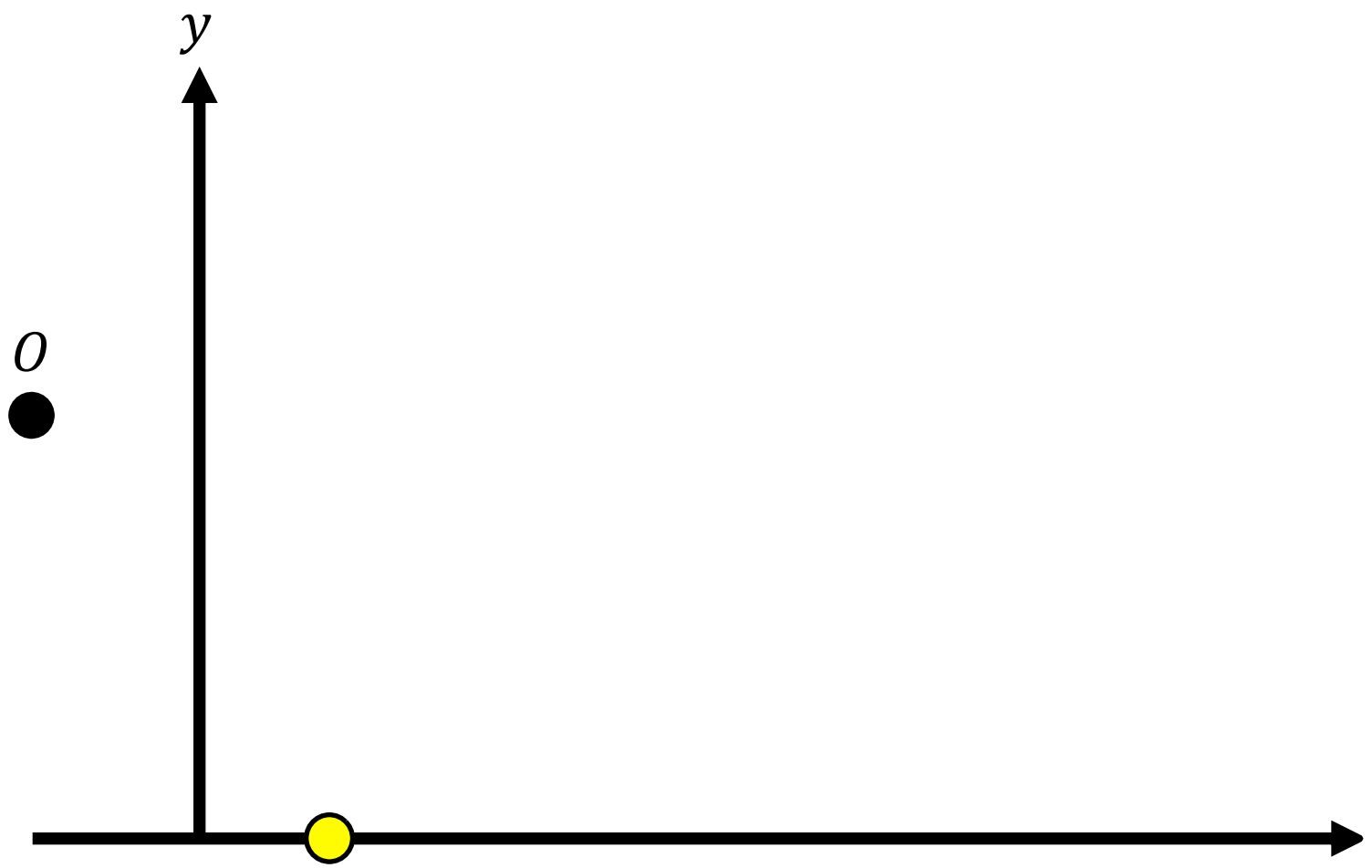

O

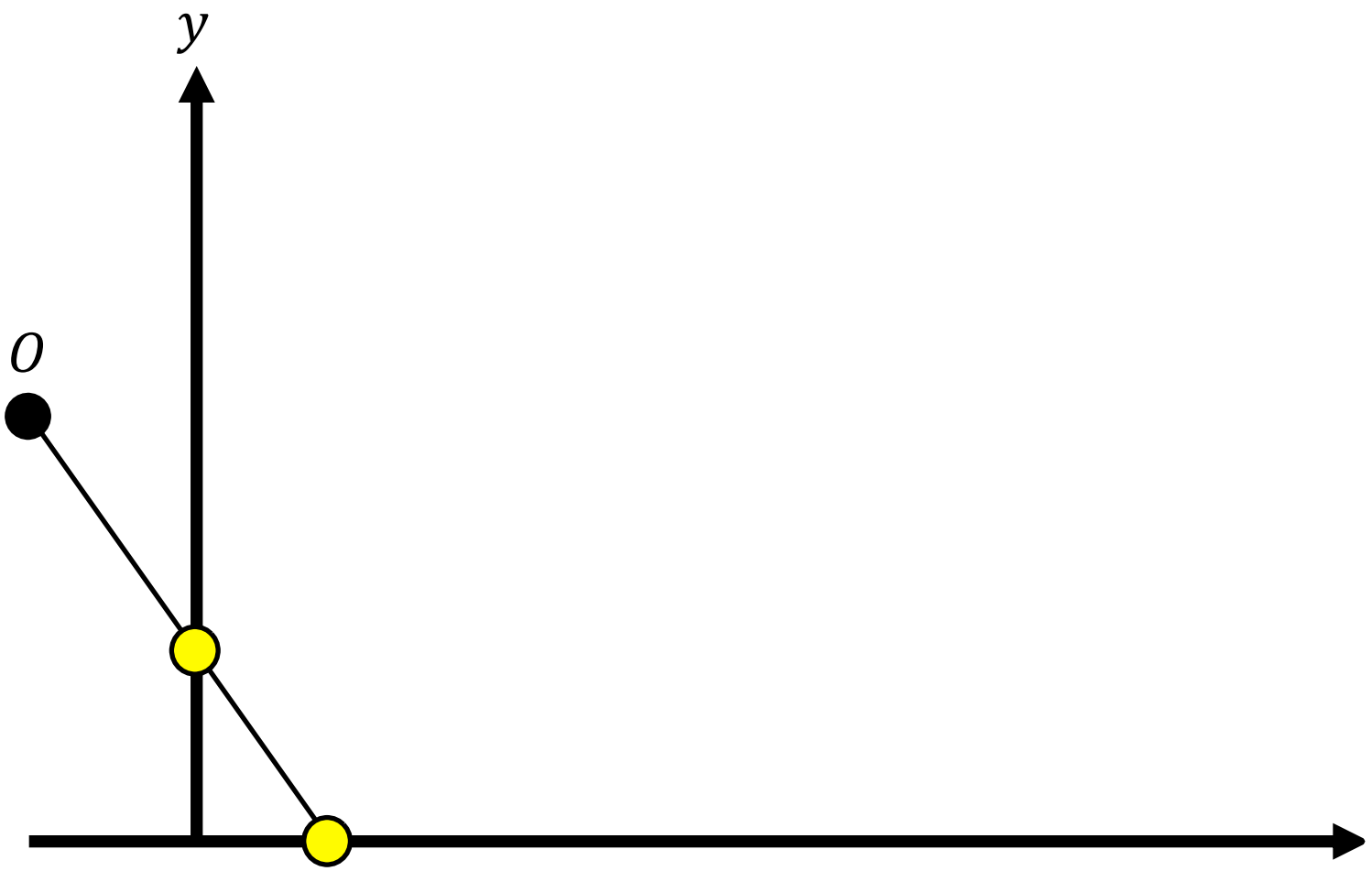

O

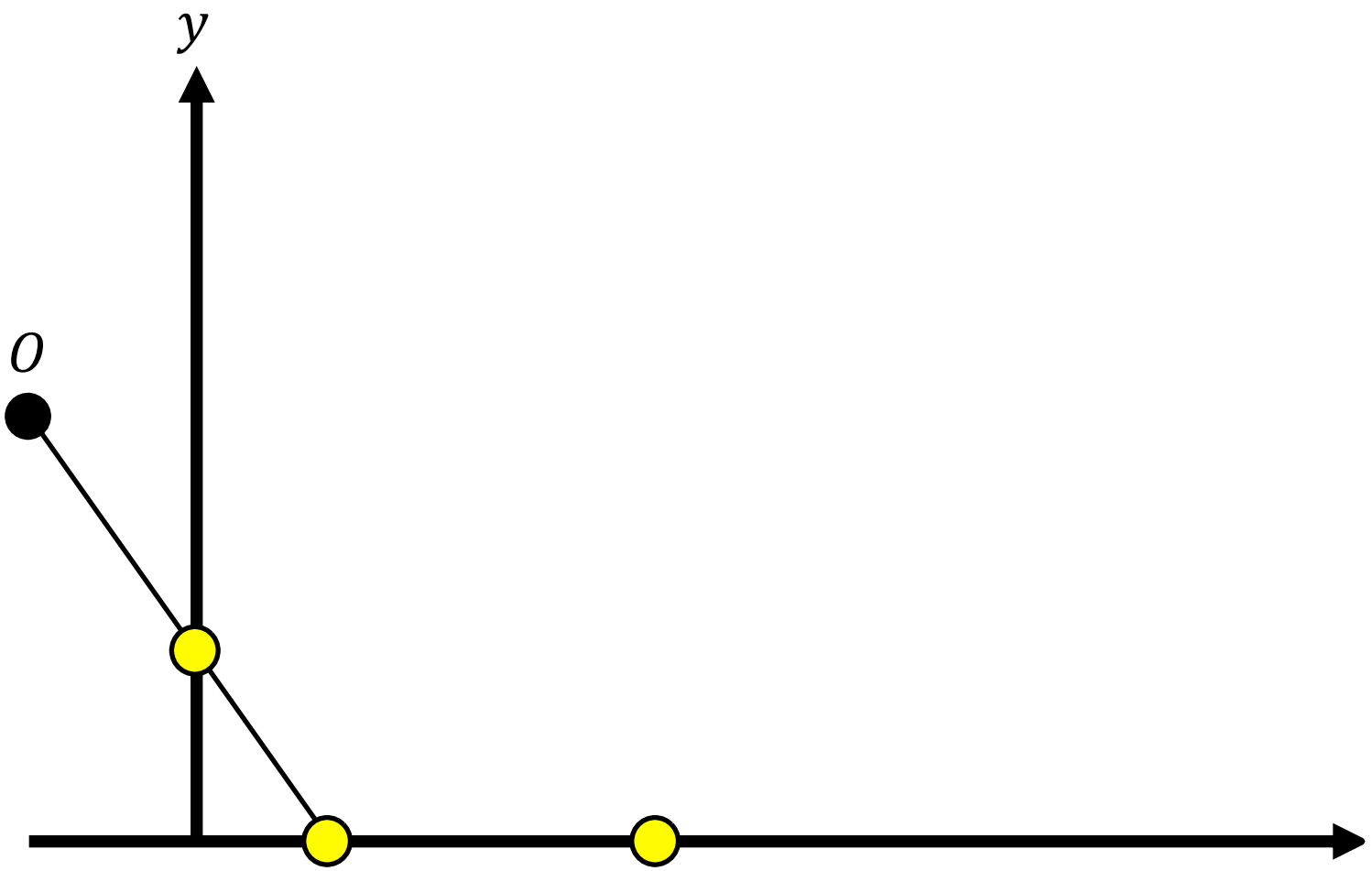

O

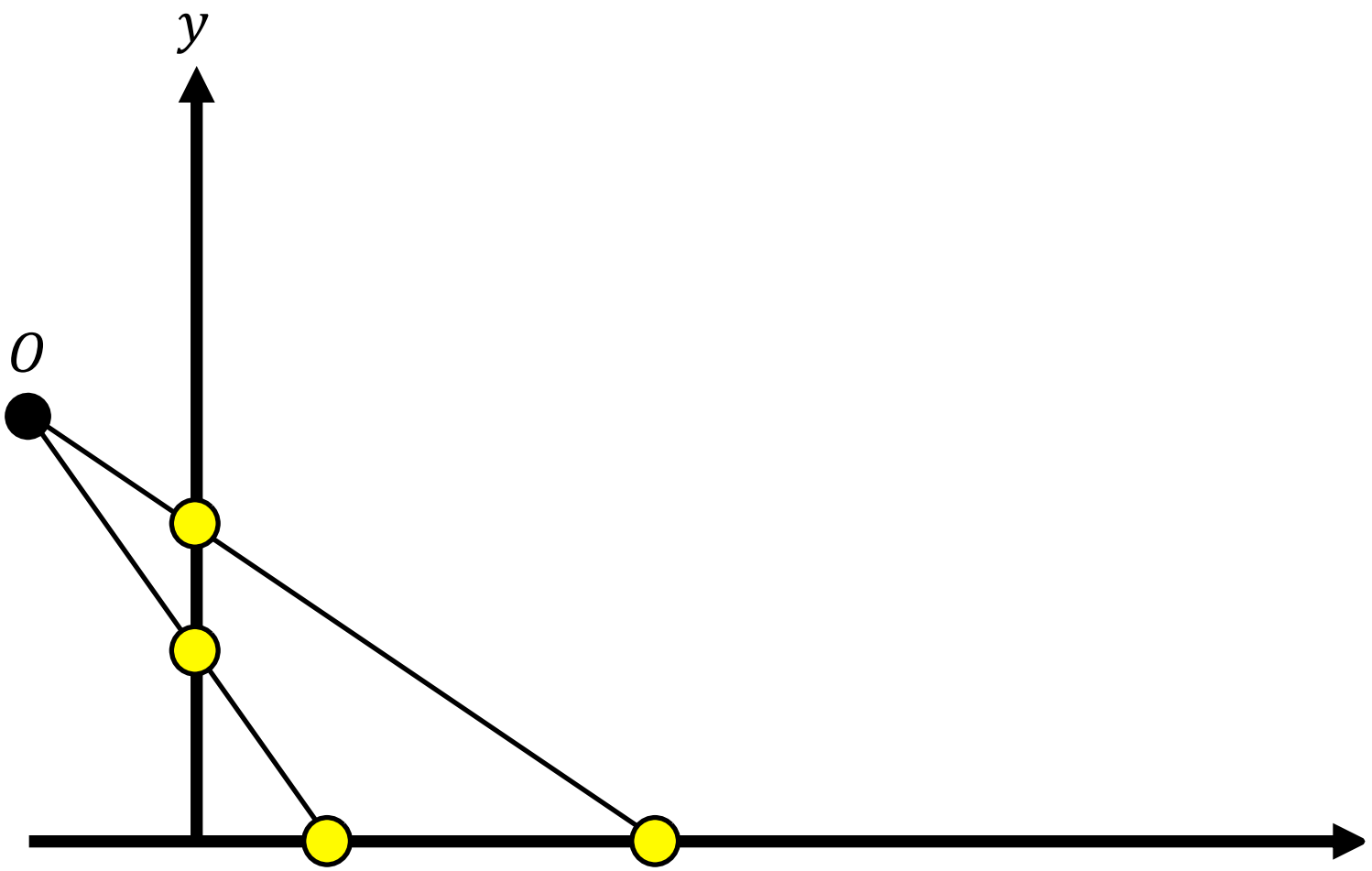

消失点
O

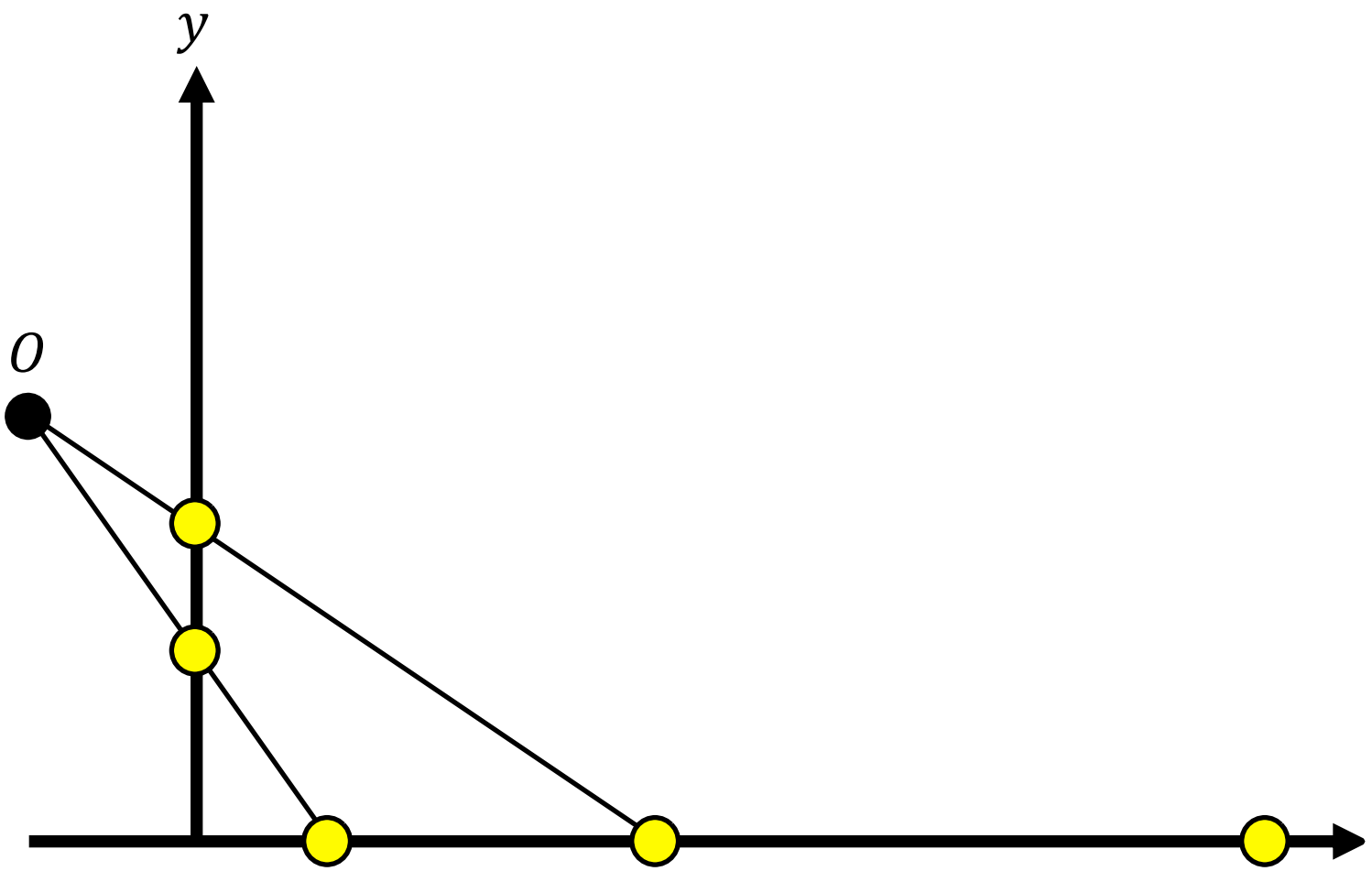

y
O

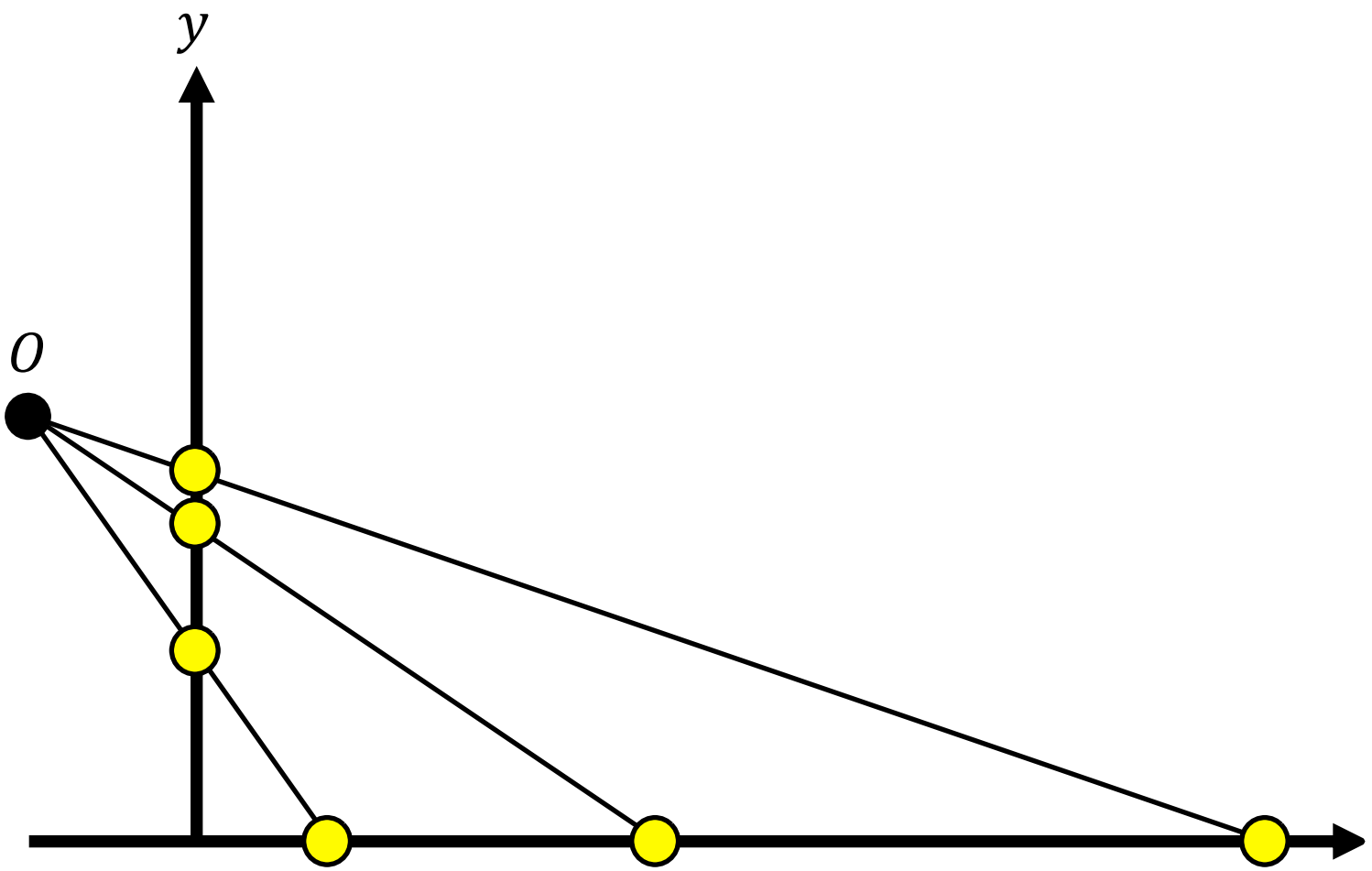

y
O

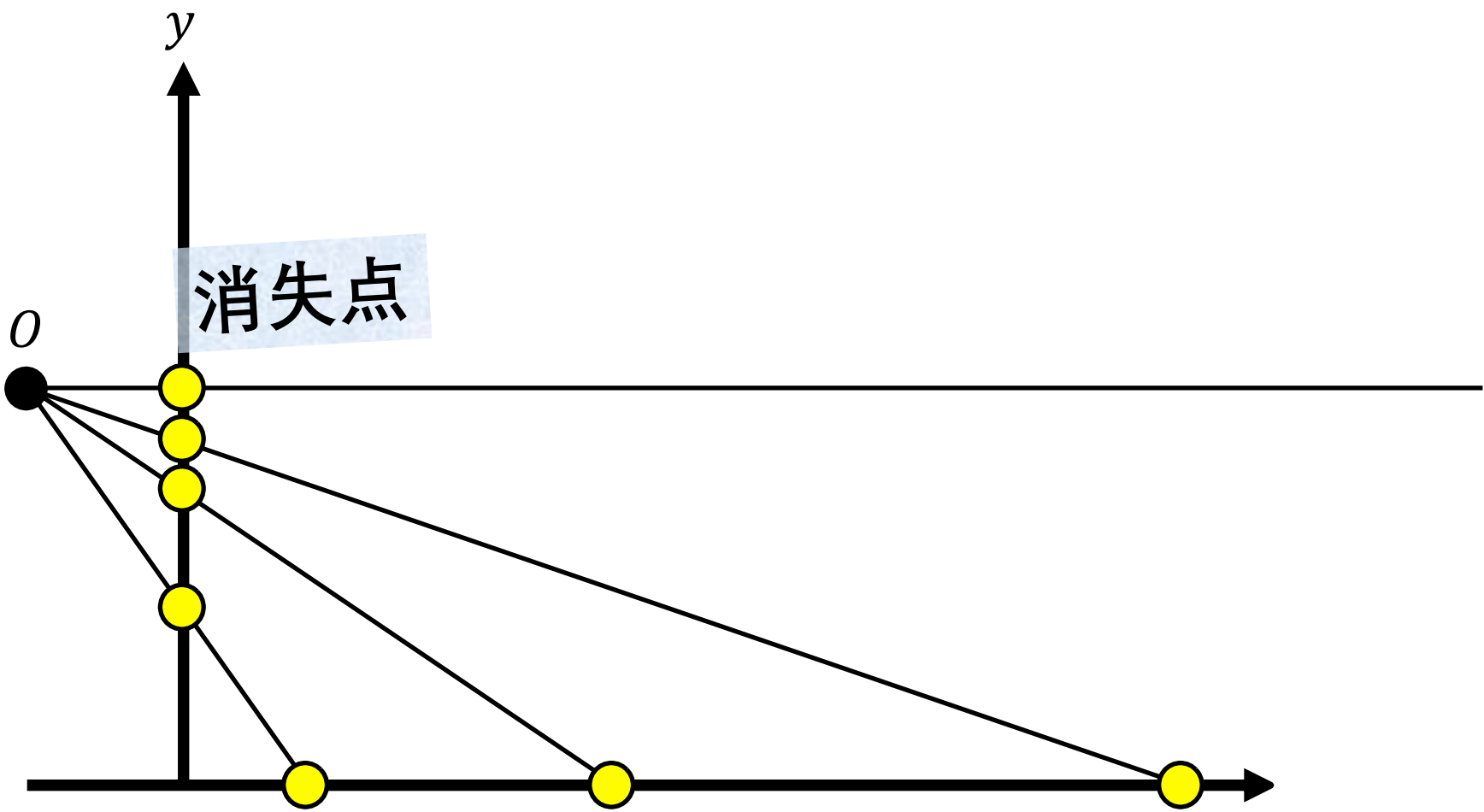

y
O


y
O


y
O


y
O


y
O


y
消失点
O


VISITOR
PARKING
PICK UP
PERMIT
FROM
RECEPTIONIST


VISITOR
PARKING
PICK UP
PERMIT
FROM
RECEPTIONIST

消失点推导

消失点推导

定义：

消失点推导

定义：

直线上任意一点

$\mathbf{P} = (X_1, X_2, X_3)^{\mathrm{T}}$

消失点推导

定义：

直线上任意一点

$\mathbf{P} = (X_1, X_2, X_3)^{\mathrm{T}}$

方向向量

$\mathbf{d} = (d_1, d_2, d_3)^{\mathrm{T}}$

消失点推导

定义：

直线上任意一点

$$\mathbf{P} = (X_1, X_2, X_3)^{\mathrm{T}}$$

方向向量

$$\mathbf{d} = (d_1, d_2, d_3)^{\mathrm{T}}$$

世界空间中的直线

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{\frac{X_1}{\lambda} + d_1}{\frac{X_3}{\lambda} + d_3}, f\frac{\frac{X_2}{\lambda} + d_2}{\frac{X_3}{\lambda} + d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda \to \infty} \left(f\frac{\frac{X_1}{\lambda} + d_1}{\frac{X_3}{\lambda} + d_3}, f\frac{\frac{X_2}{\lambda} + d_2}{\frac{X_3}{\lambda} + d_3} \right)$$

消失点推导

$$\mathbf{X}(\lambda) = \mathbf{P} + \lambda\mathbf{d}$$

投影到图像中

$$\mathbf{x}(\lambda) = \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda\to\infty} \left(f\frac{X_1 + \lambda d_1}{X_3 + \lambda d_3}, f\frac{X_2 + \lambda d_2}{X_3 + \lambda d_3} \right)$$

$$\mathbf{v} = \lim_{\lambda\to\infty} \left(f\frac{\frac{X_1}{\lambda} + d_1}{\frac{X_3}{\lambda} + d_3}, f\frac{\frac{X_2}{\lambda} + d_2}{\frac{X_3}{\lambda} + d_3} \right)$$

消失点：$v_1 = f\frac{d_1}{d_3}$，$v_2 = f\frac{d_2}{d_3}$

消失点：$v_1 = f\dfrac{d_1}{d_3}$，$v_2 = f\dfrac{d_2}{d_3}$

观察：

消失点：$v_1 = f\dfrac{d_1}{d_3}$，$v_2 = f\dfrac{d_2}{d_3}$

观察：

消失点只取决于方向向量

消失点：$v_1 = f\frac{d_1}{d_3}$，$v_2 = f\frac{d_2}{d_3}$

观察：

消失点只取决于方向向量

∴同一方向的直线具有相同的消失点

消失点：$v_1 = f\frac{d_1}{d_3}$，$v_2 = f\frac{d_2}{d_3}$

观察：

消失点只取决于方向向量

∴同一方向的直线具有相同的消失点

越近的物体显得越大

越近的物体显得越大

令 L为路灯的高度

越近的物体显得越大

令 L为路灯的高度

越近的物体显得越大

令 L为路灯的高度

h为相机的高度

越近的物体显得越大

令 L为路灯的高度

h为相机的高度

路灯底：$\mathbf{b} = (X, -h, Z)$

越近的物体显得越大

令 L为路灯的高度

h为相机的高度

路灯底：$\mathbf{b} = (X, -h, Z)$

路灯顶：$\mathbf{t} = (X, -h + L, Z)$

越近的物体显得越大

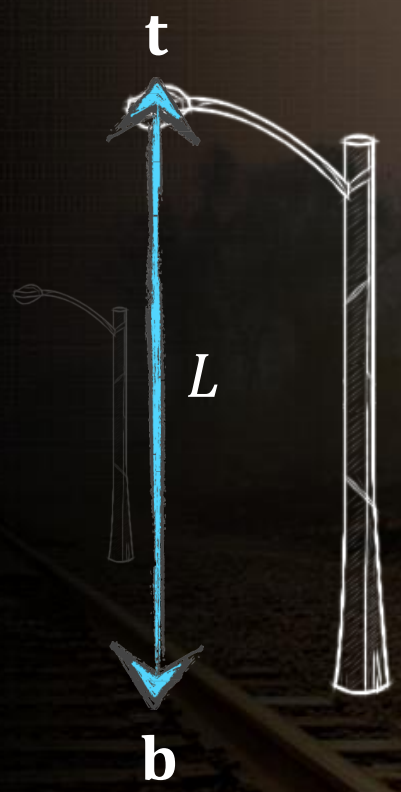
令 L为路灯的高度

h为相机的高度

路灯底：$\mathbf{b} = (X, -h, Z)$

路灯顶：$\mathbf{t} = (X, -h + L, Z)$

底部投影：$\left(f\frac{X}{Z}, -f\frac{h}{Z}\right)$



越近的物体显得越大

令 L为路灯的高度

h为相机的高度

路灯底：$\mathbf{b} = (X, -h, Z)$

路灯顶：$\mathbf{t} = (X, -h + L, Z)$

底部投影：$\left(f\frac{X}{Z}, -f\frac{h}{Z}\right)$

顶部投影：$\left(f\frac{X}{Z}, f\frac{L-h}{Z}\right)$

越近的物体显得越大

令 L为路灯的高度

h为相机的高度

路灯底：$\mathbf{b} = (X, -h, Z)$

路灯顶：$\mathbf{t} = (X, -h + L, Z)$

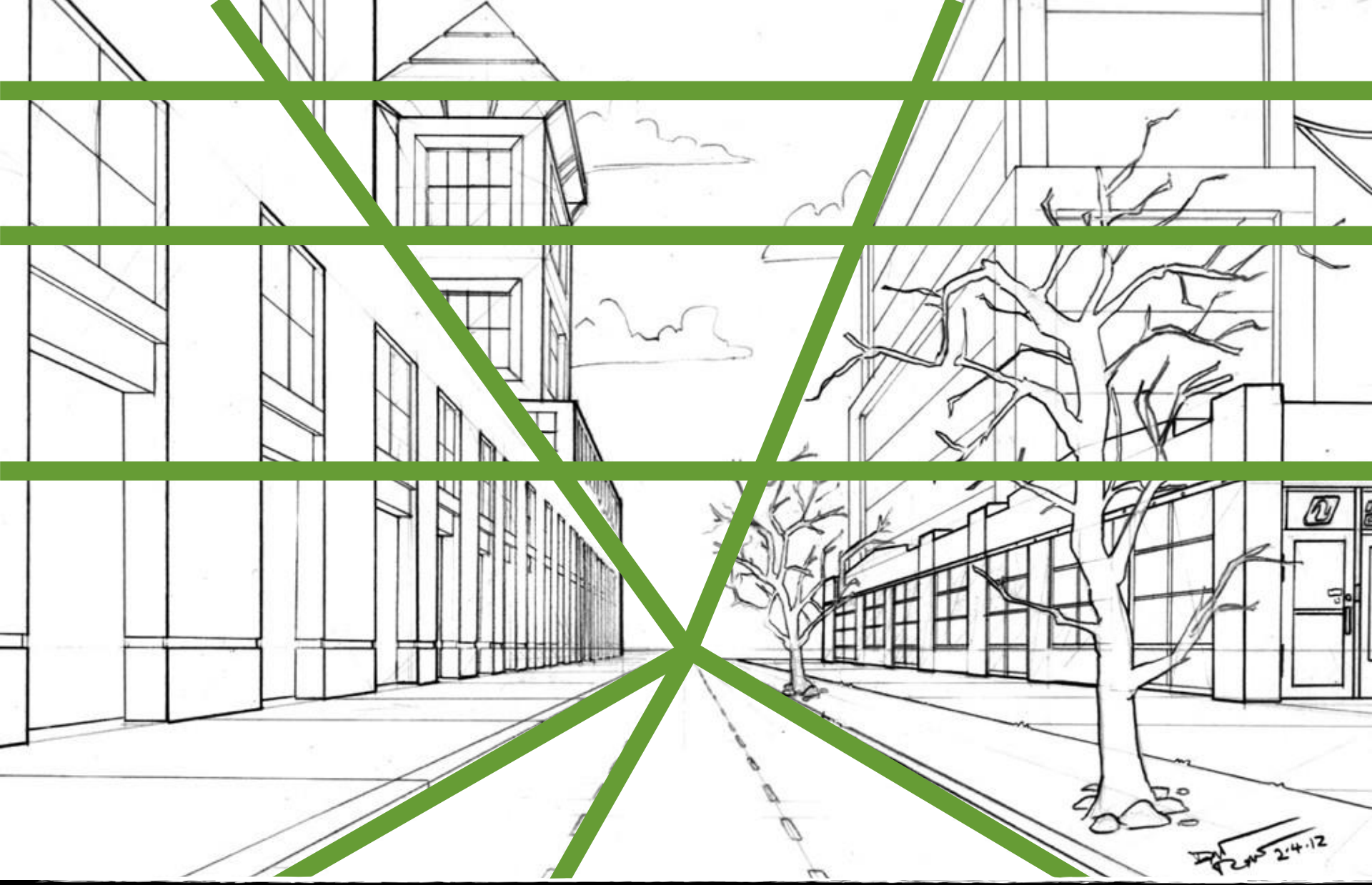
底部投影：$\left(f\frac{X}{Z}, -f\frac{h}{Z}\right)$

顶部投影：$\left(f\frac{X}{Z}, f\frac{L-h}{Z}\right)$

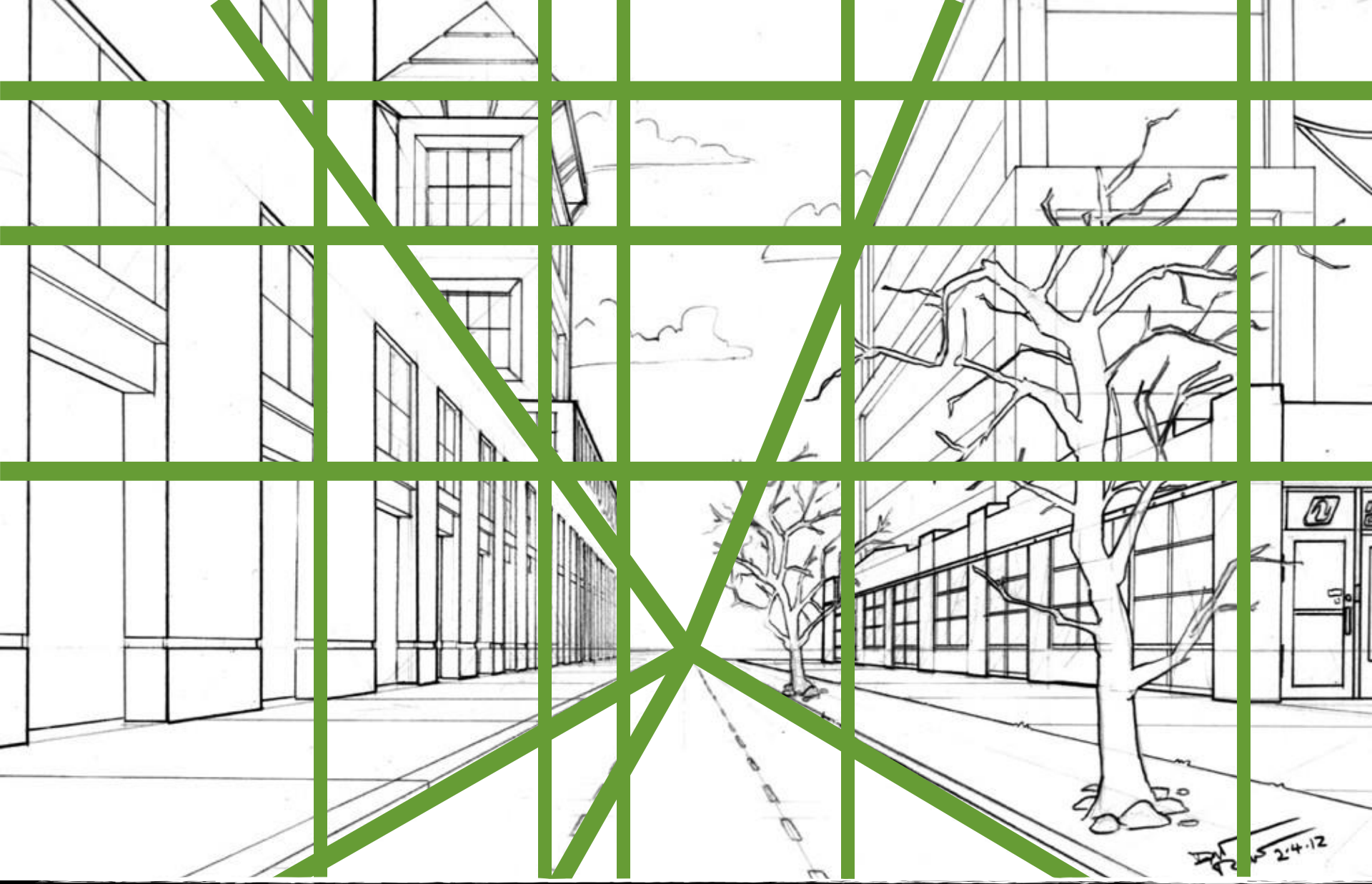
图像长度：$f\frac{L}{Z}$

一点透视

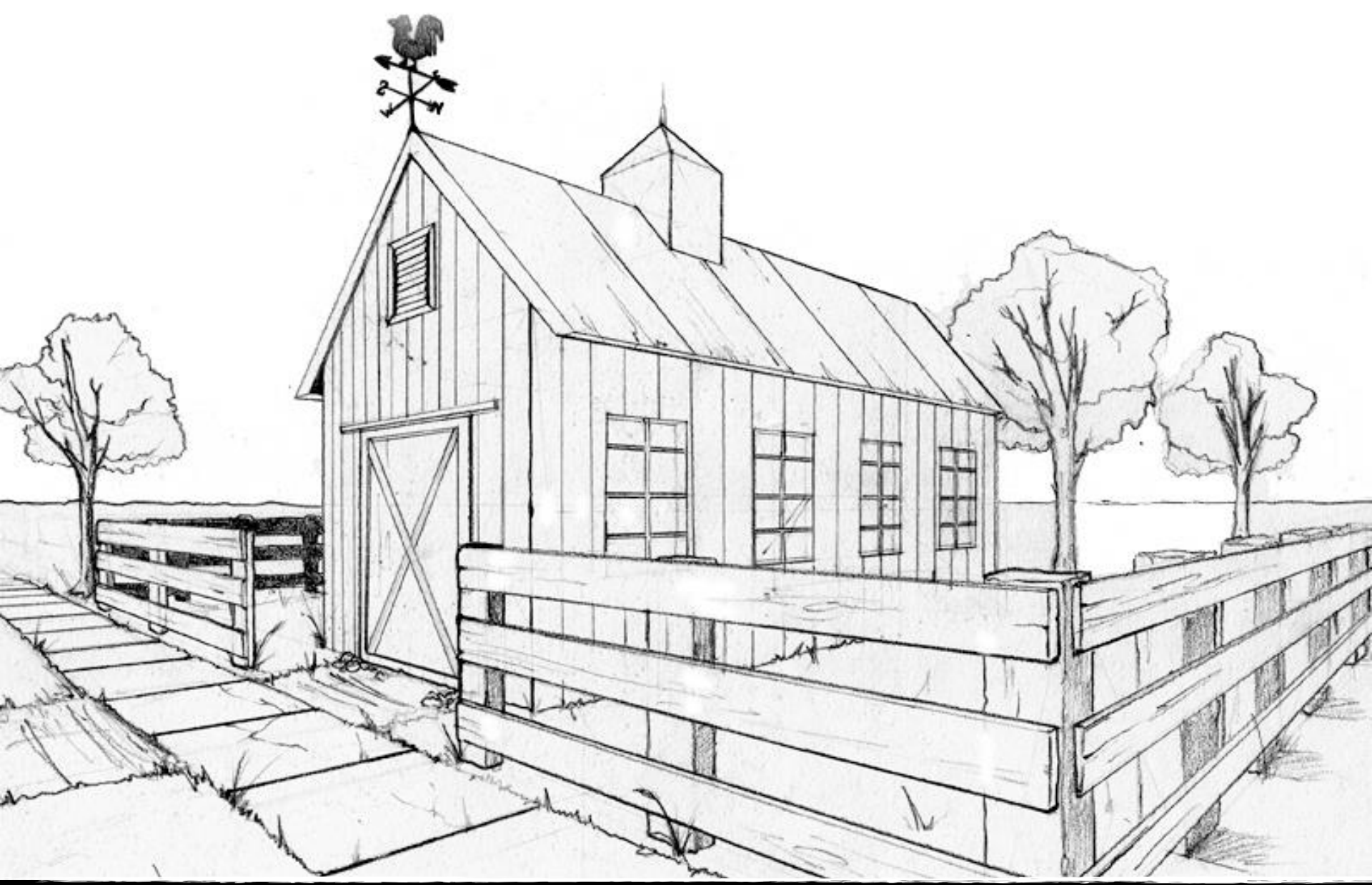
一点透视



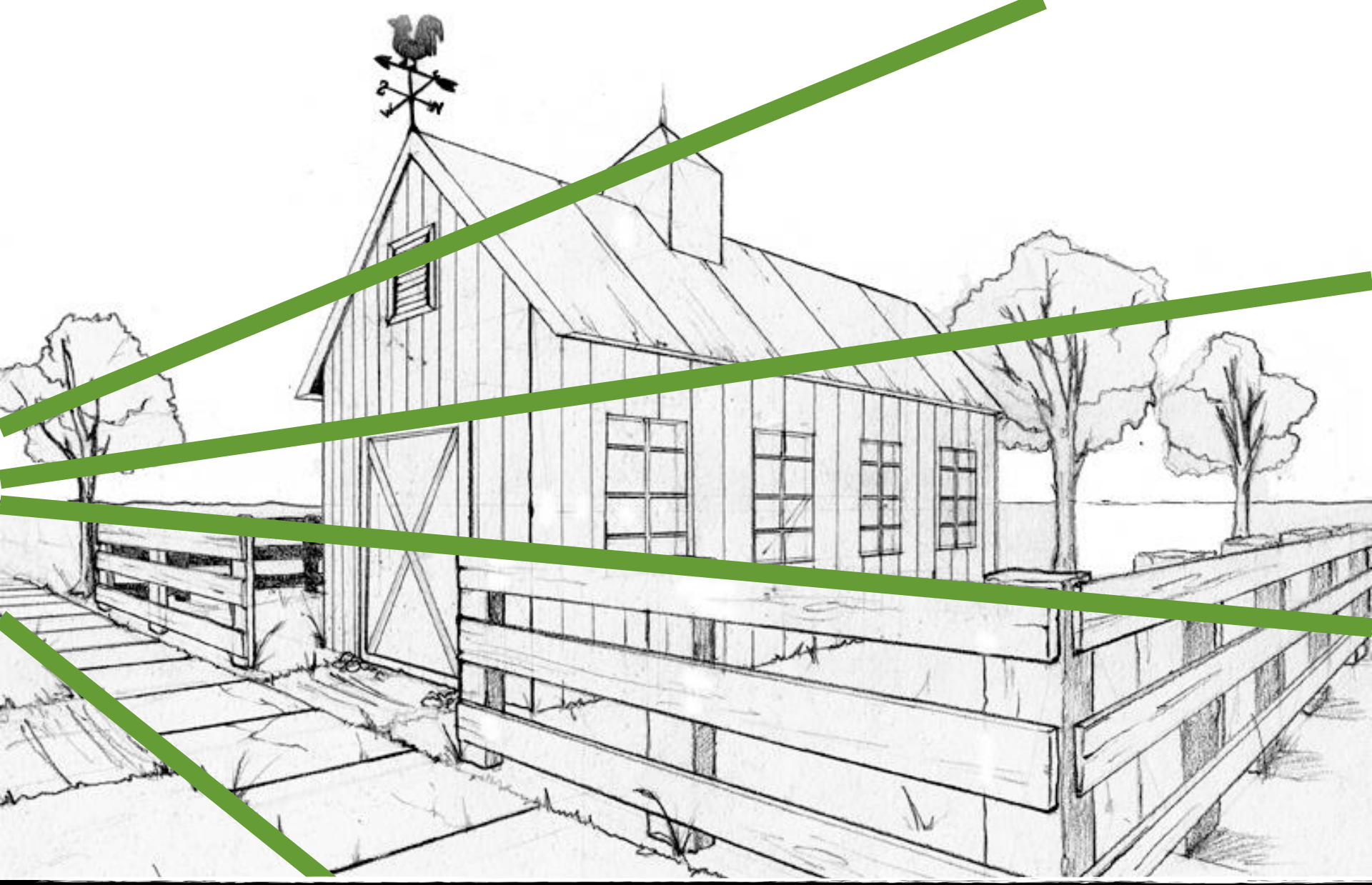
一点透视



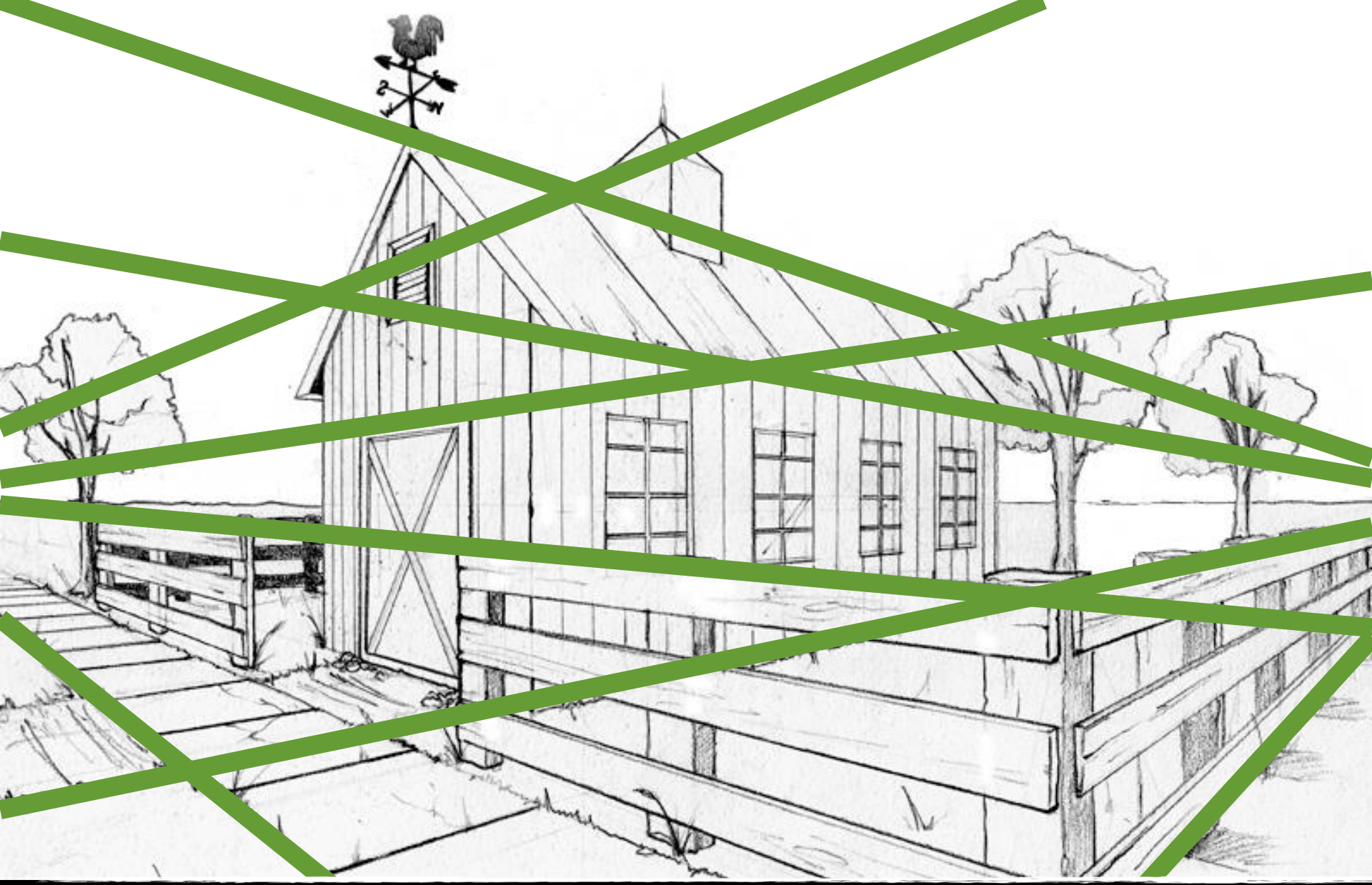
一点透视



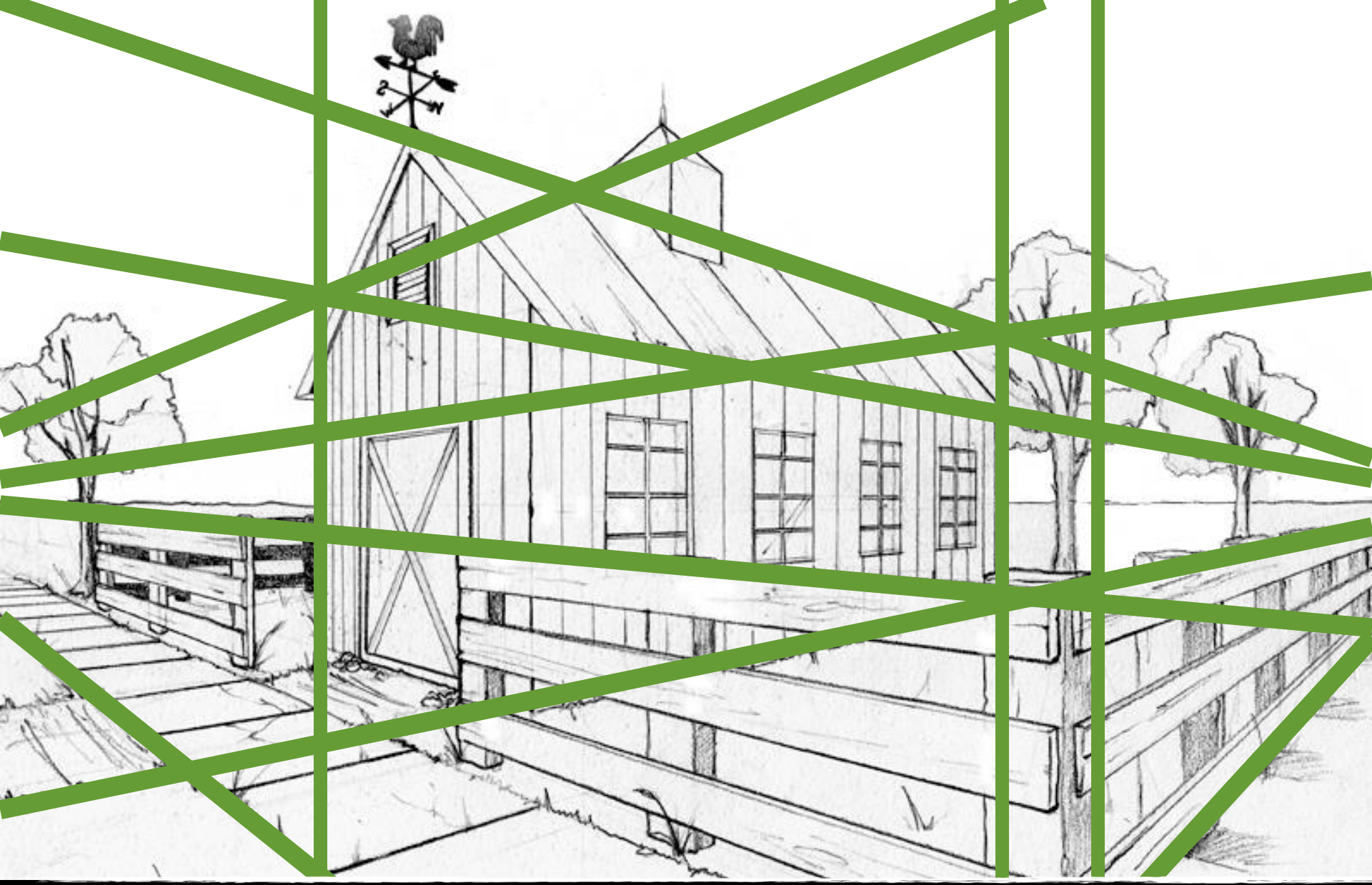
两点透视



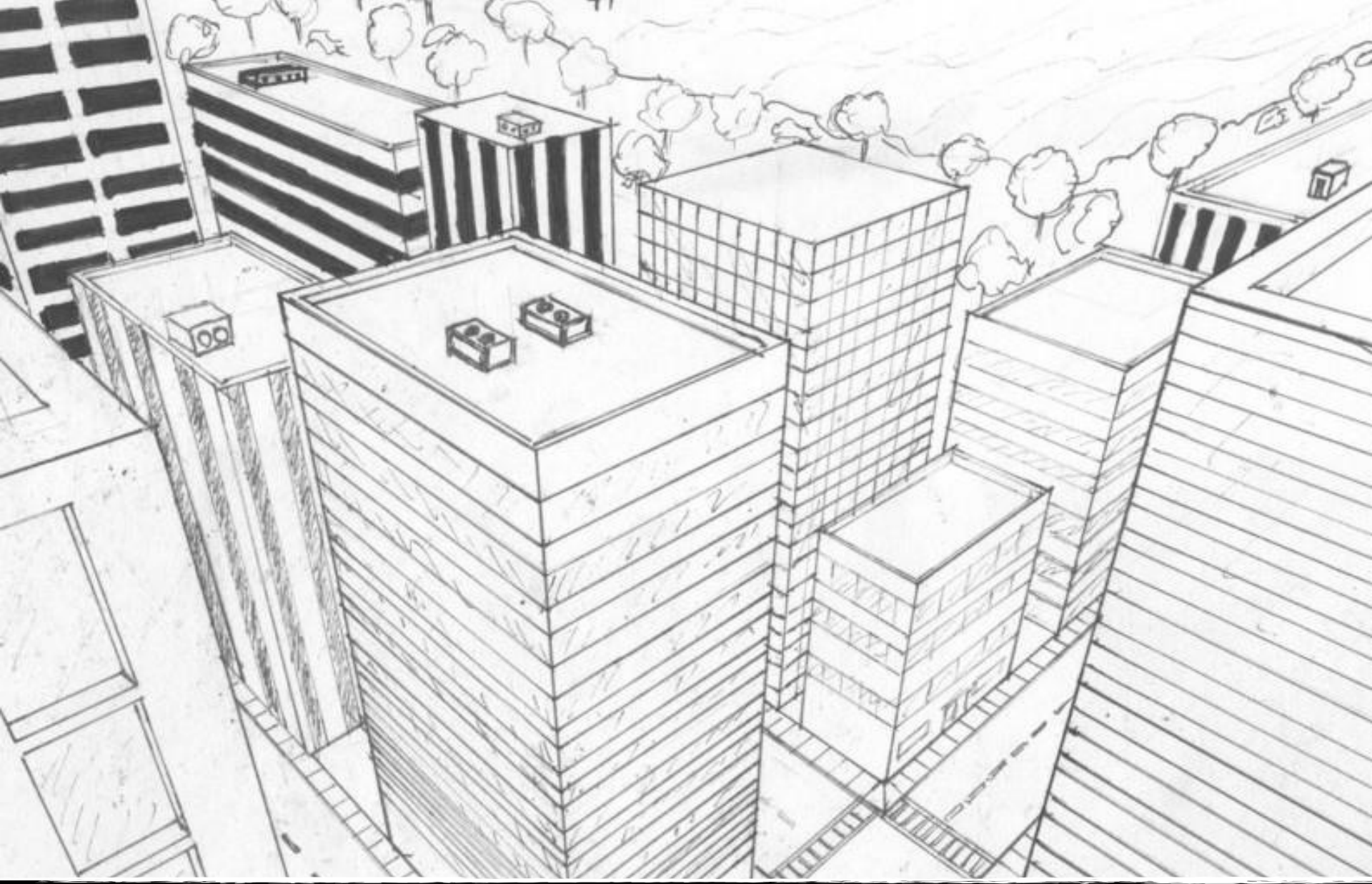
两点透视



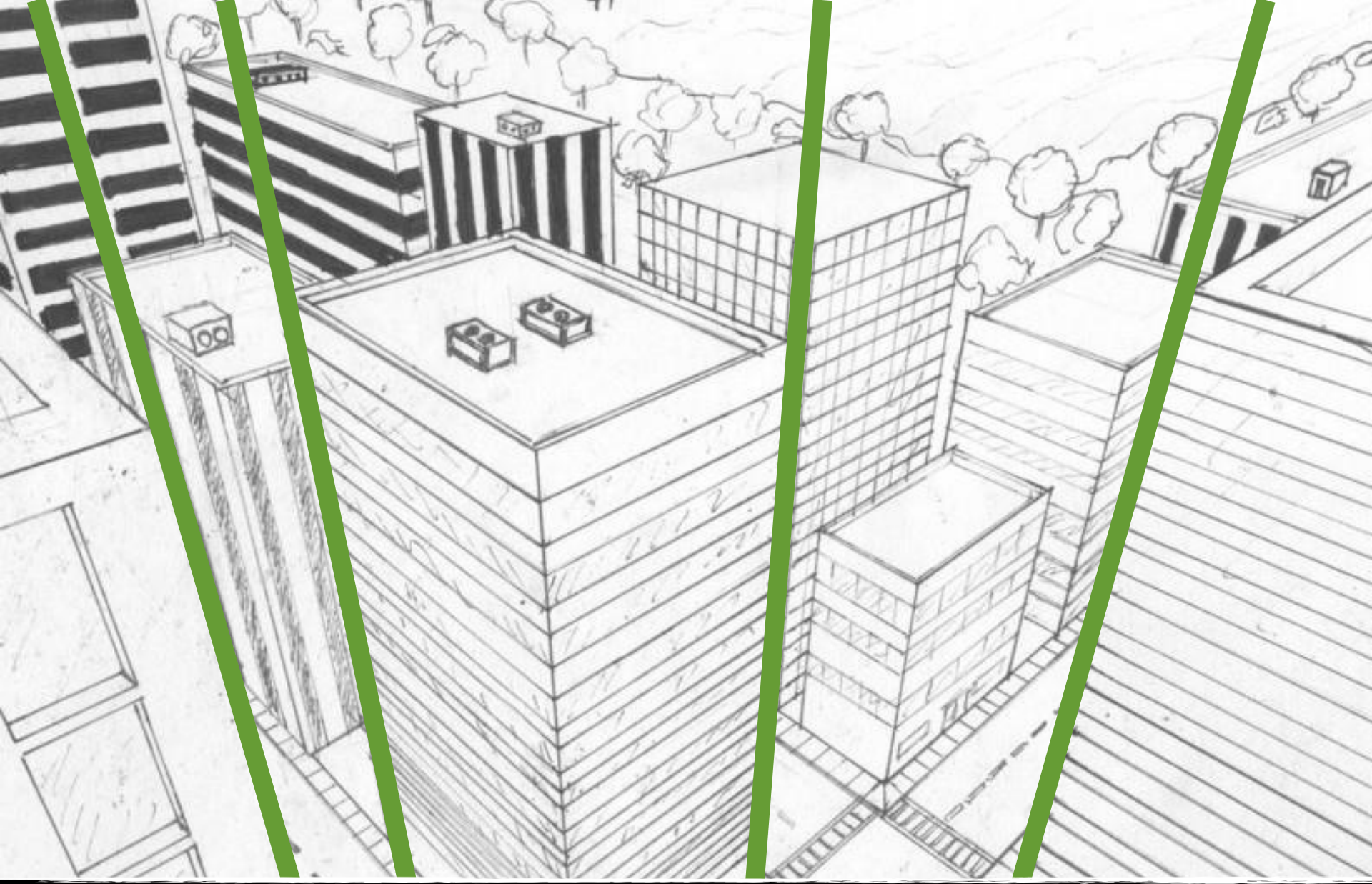
两点透视



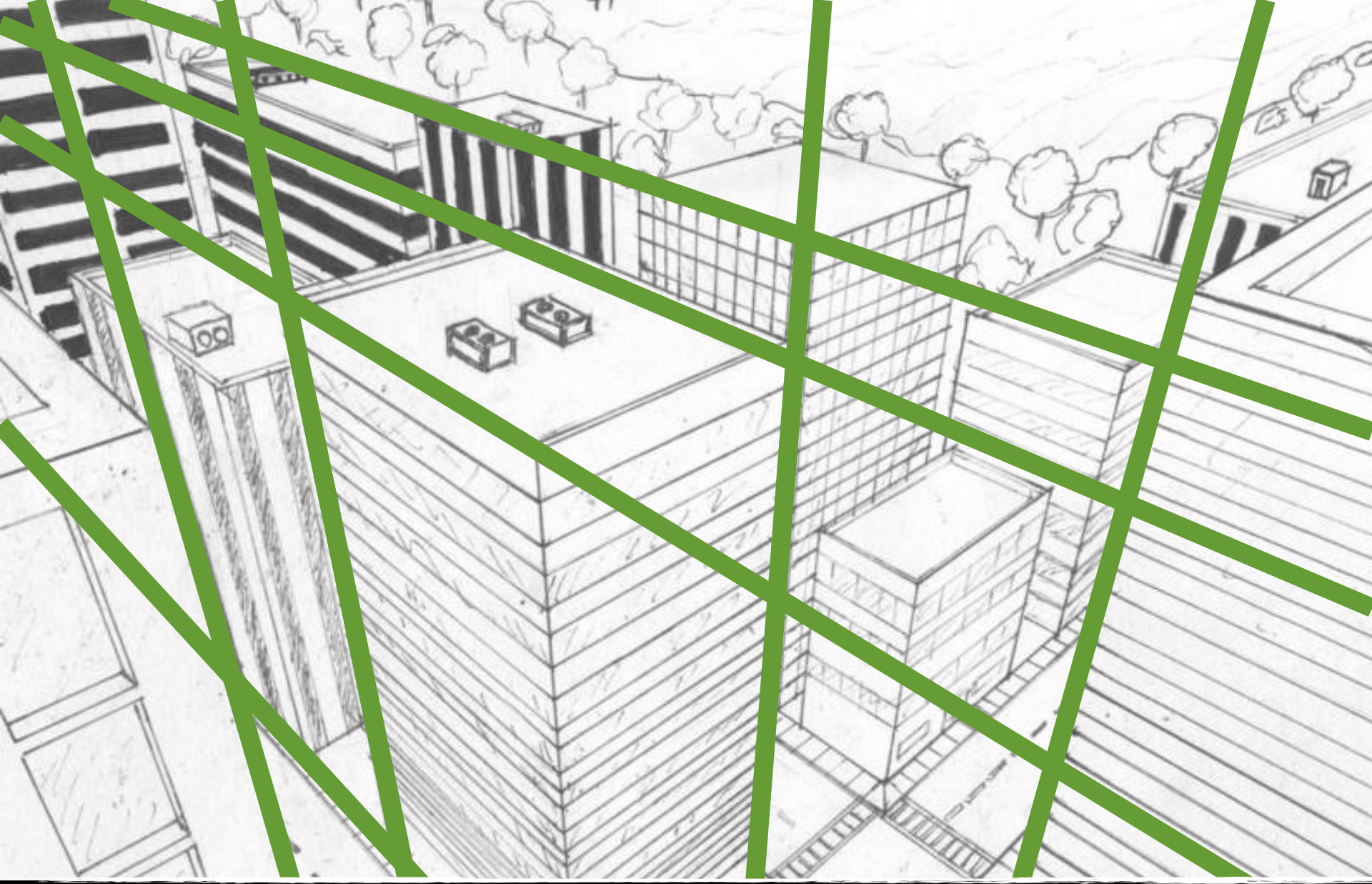
两点透视



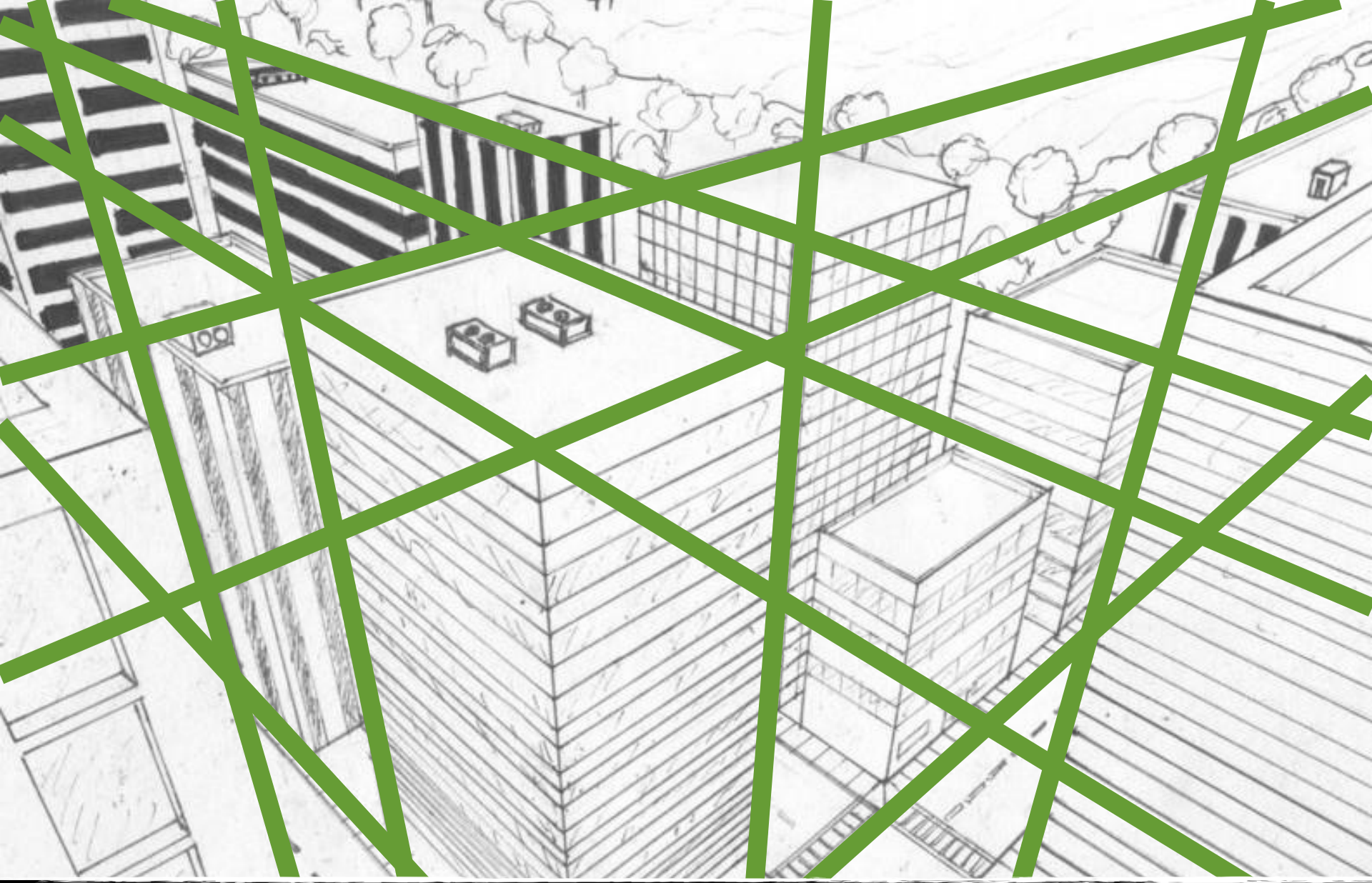
三点透视



三点透视



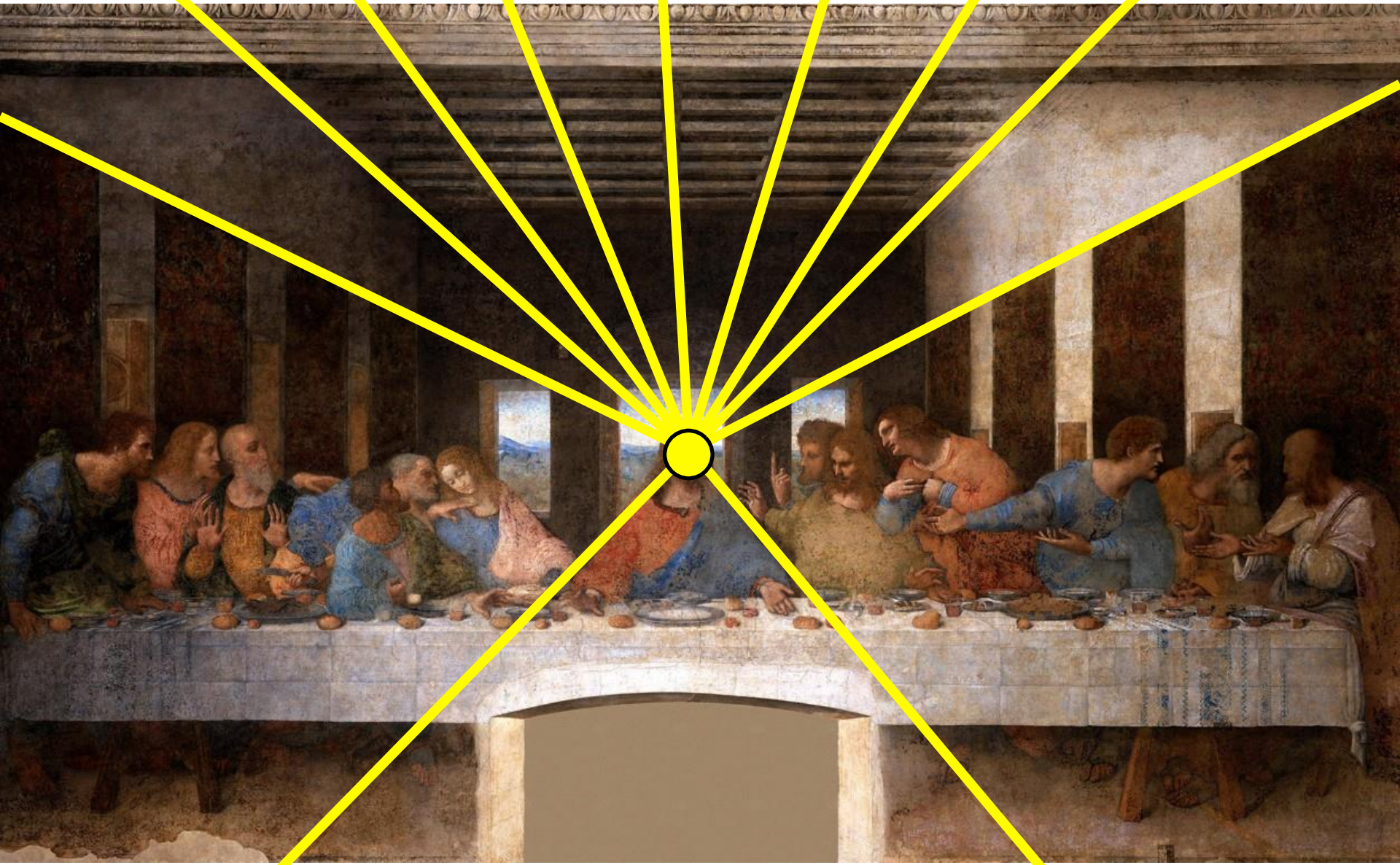
三点透视



三点透视

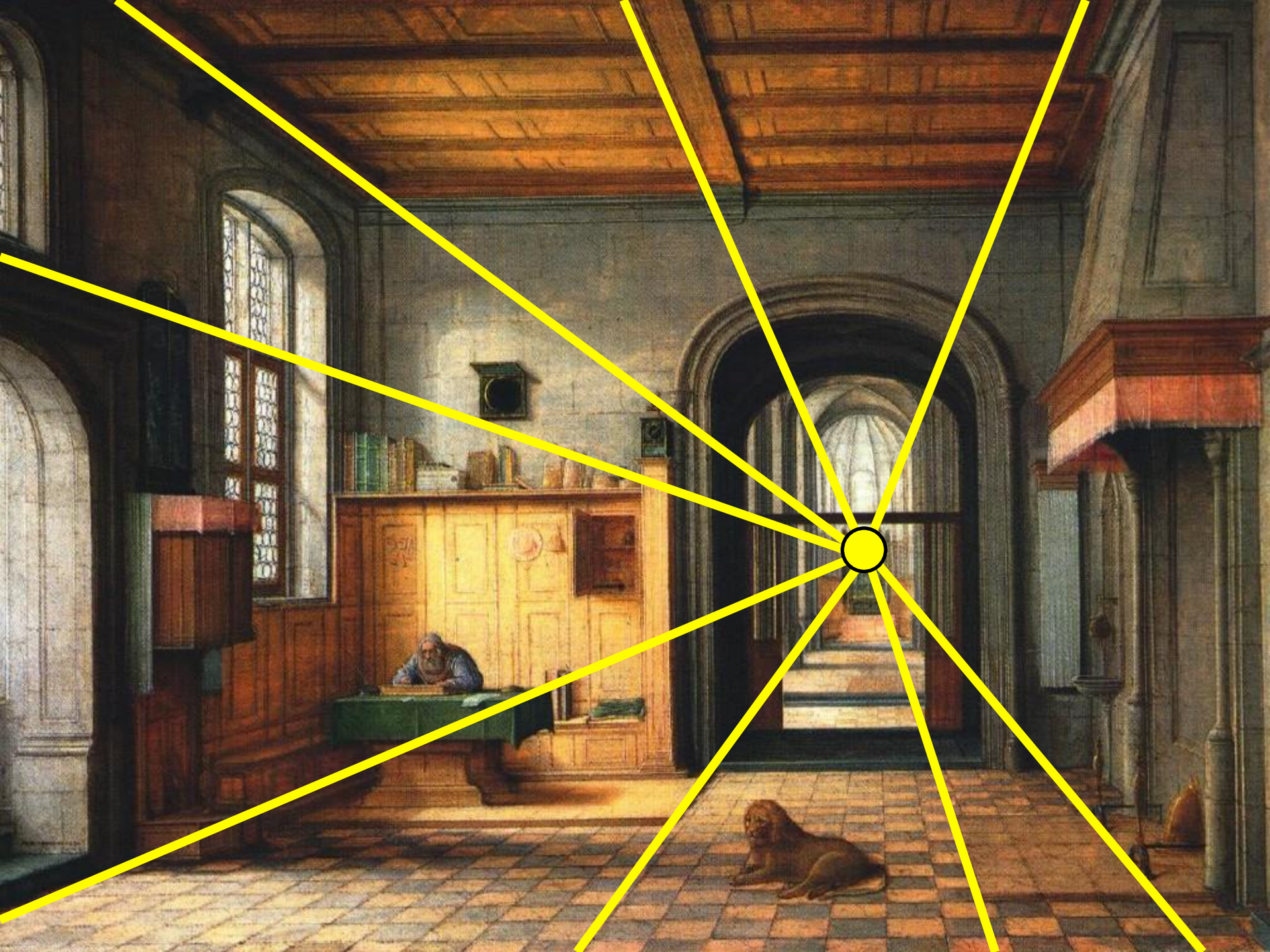
最后的晚餐

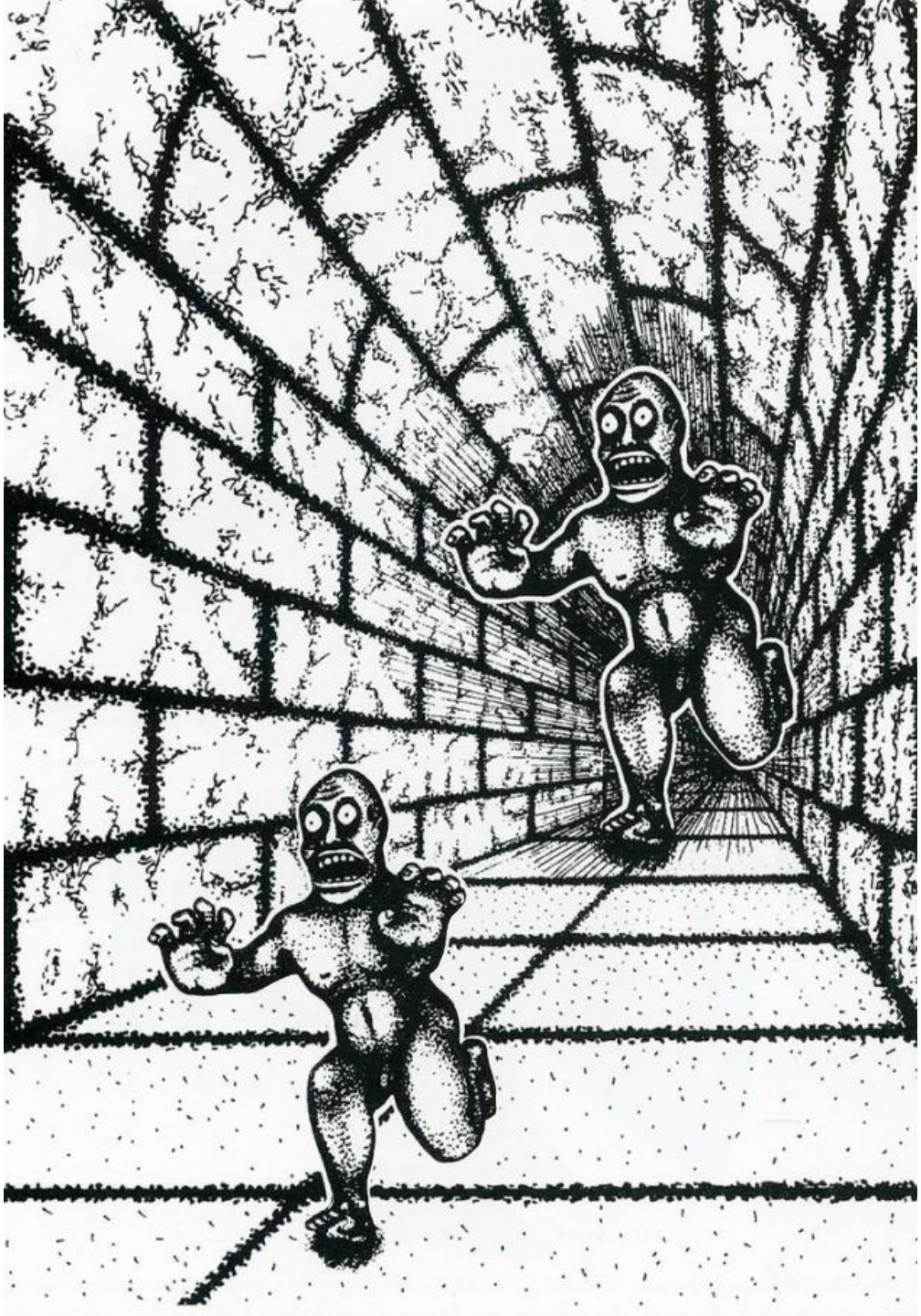
——列奥纳多·达·芬奇

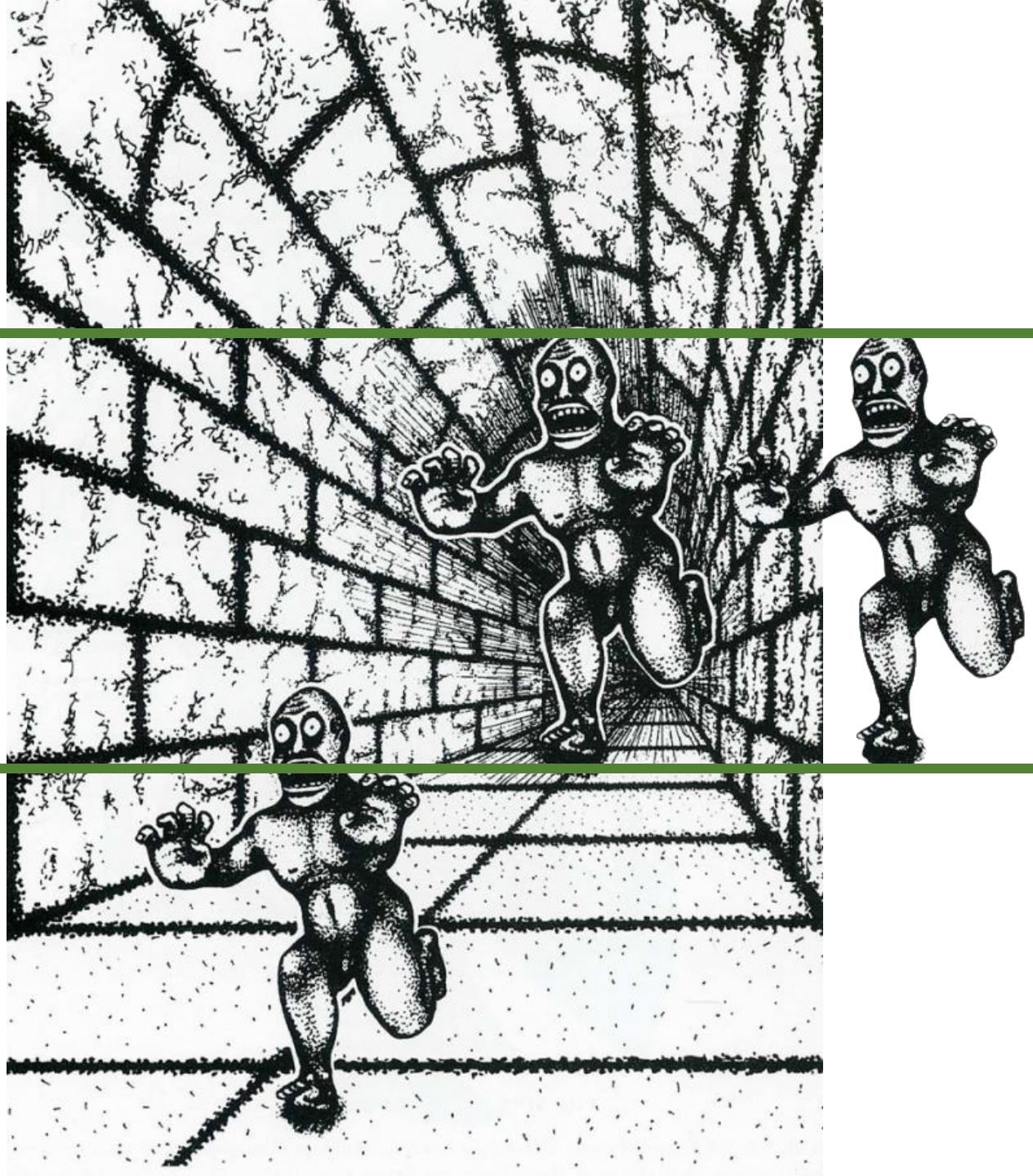


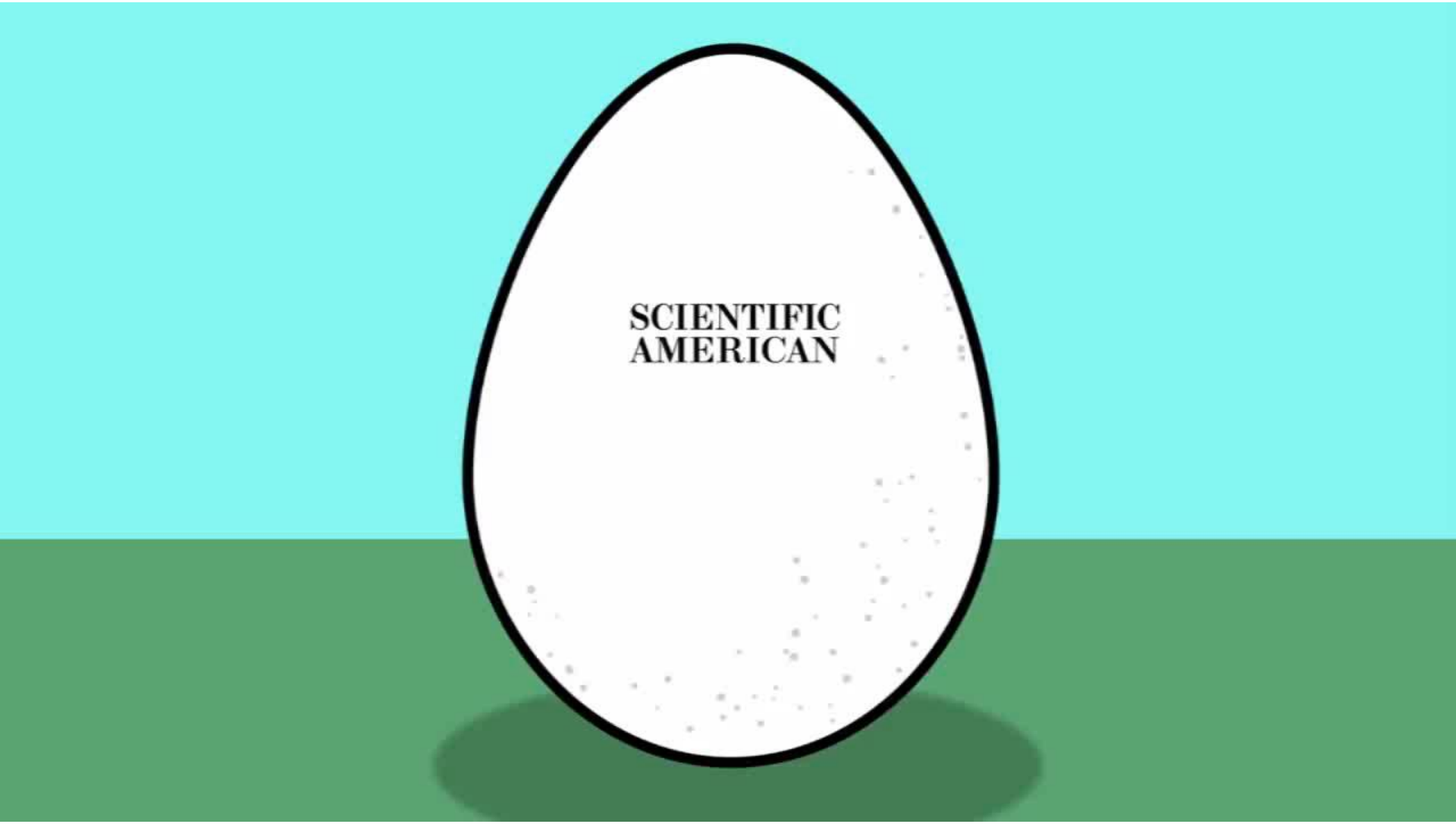
圣杰罗姆在他的书房

——亨德里克·范·斯坦威克

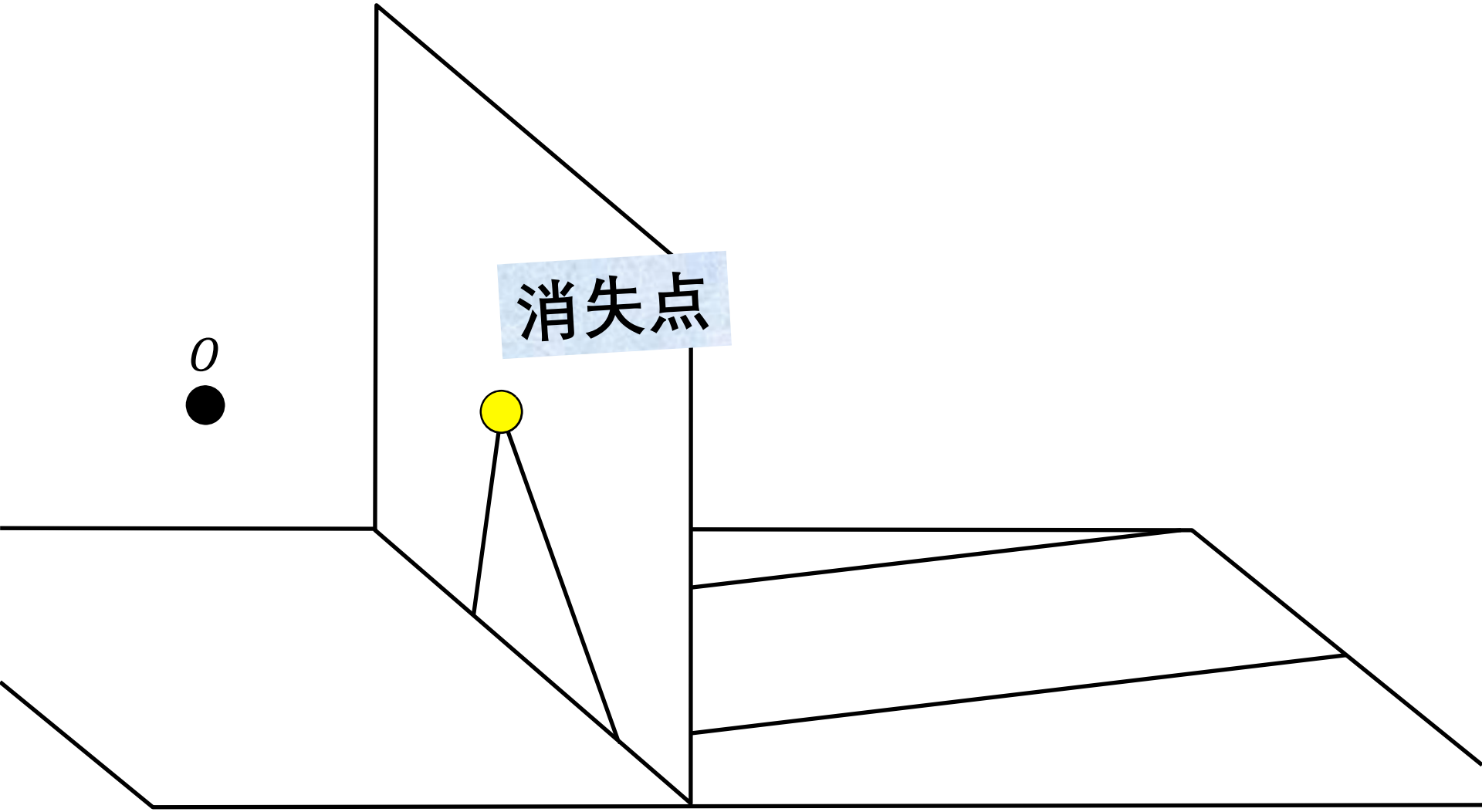


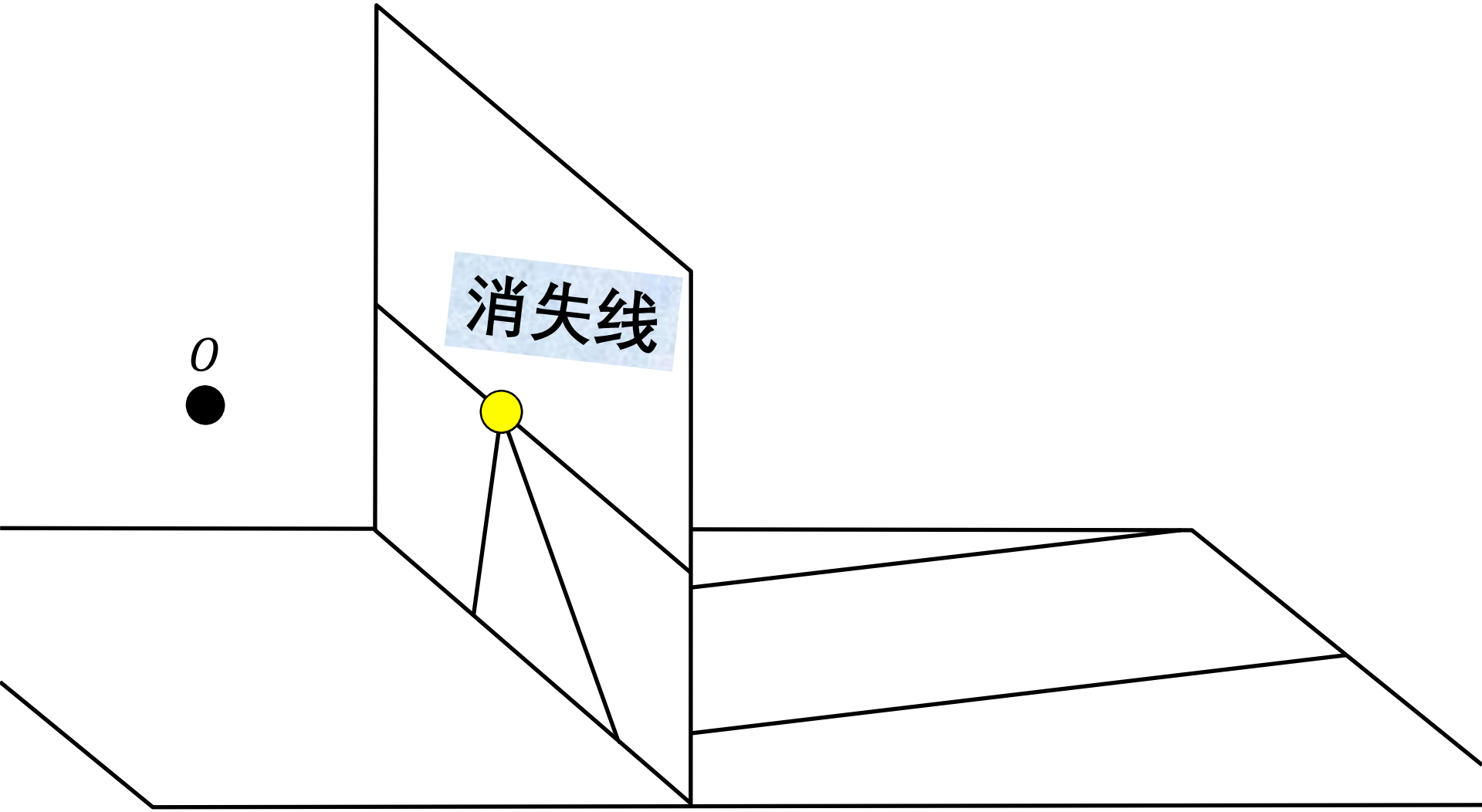





SCIENTIFIC
AMERICAN


消失点
消失线


消失点
O


消失线
O

假设相机与地平线是平齐的

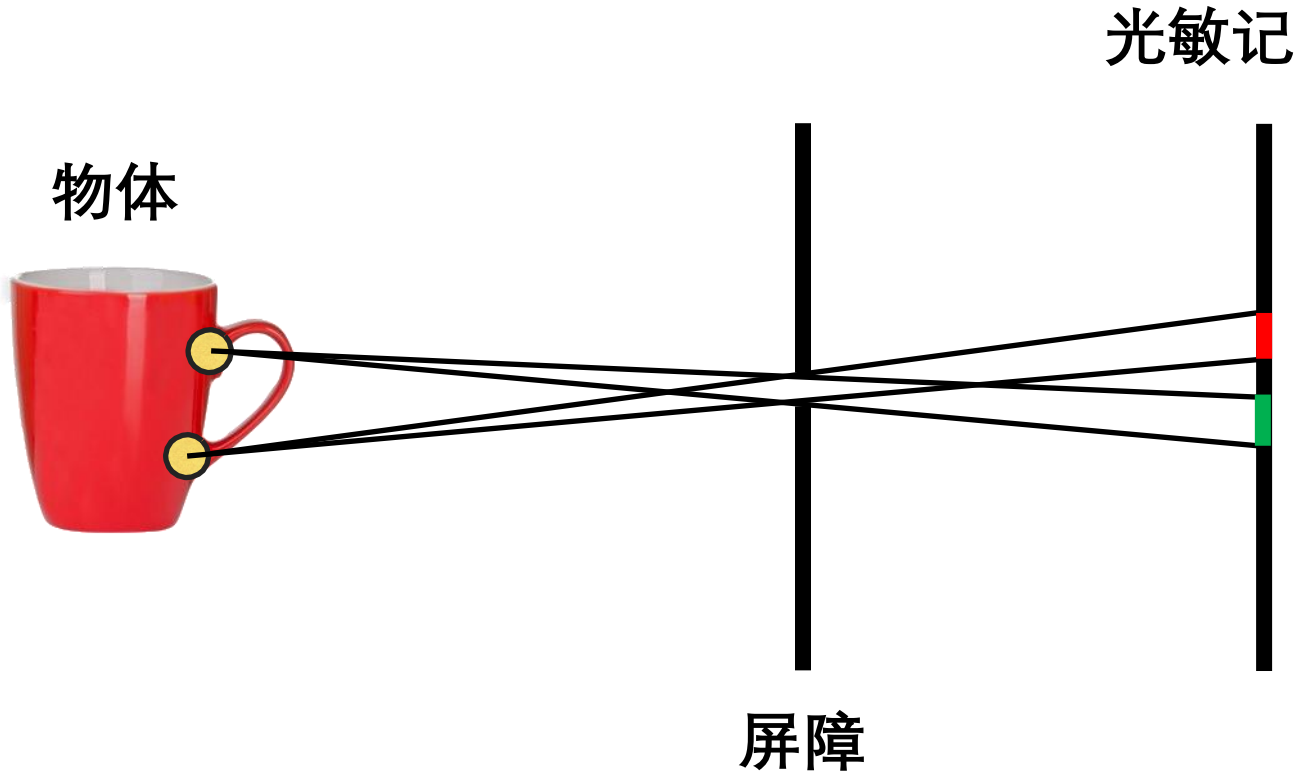
相机跟气球哪个高呢？

相机跟气球哪个高呢？

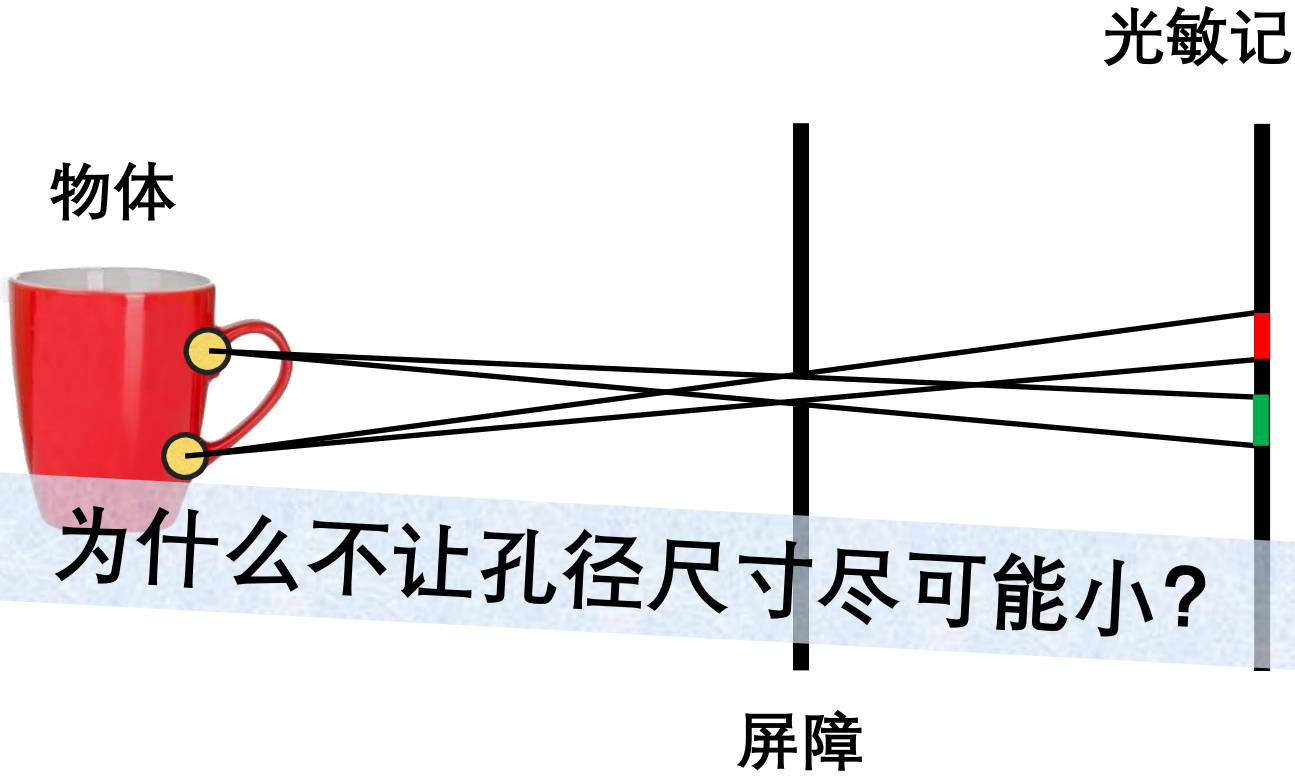
来源：http://www.pauldebevec.com/Pinhole/

为什么这张图像模糊了？

针孔相机

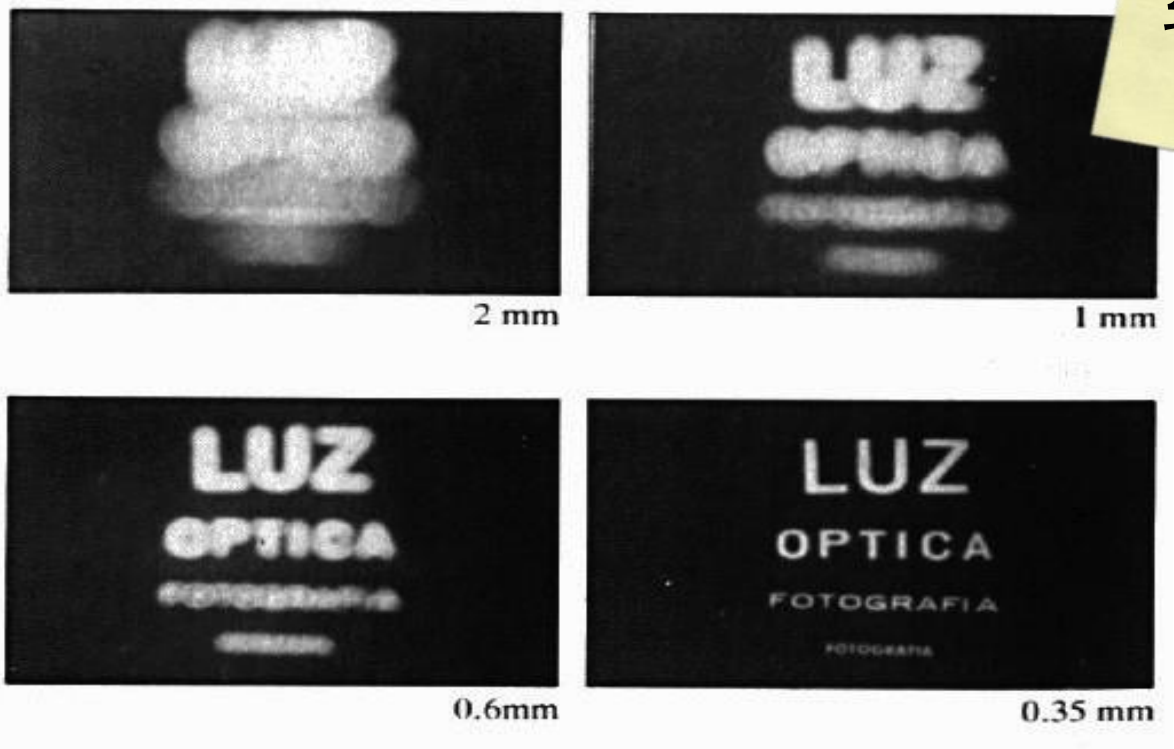


针孔相机

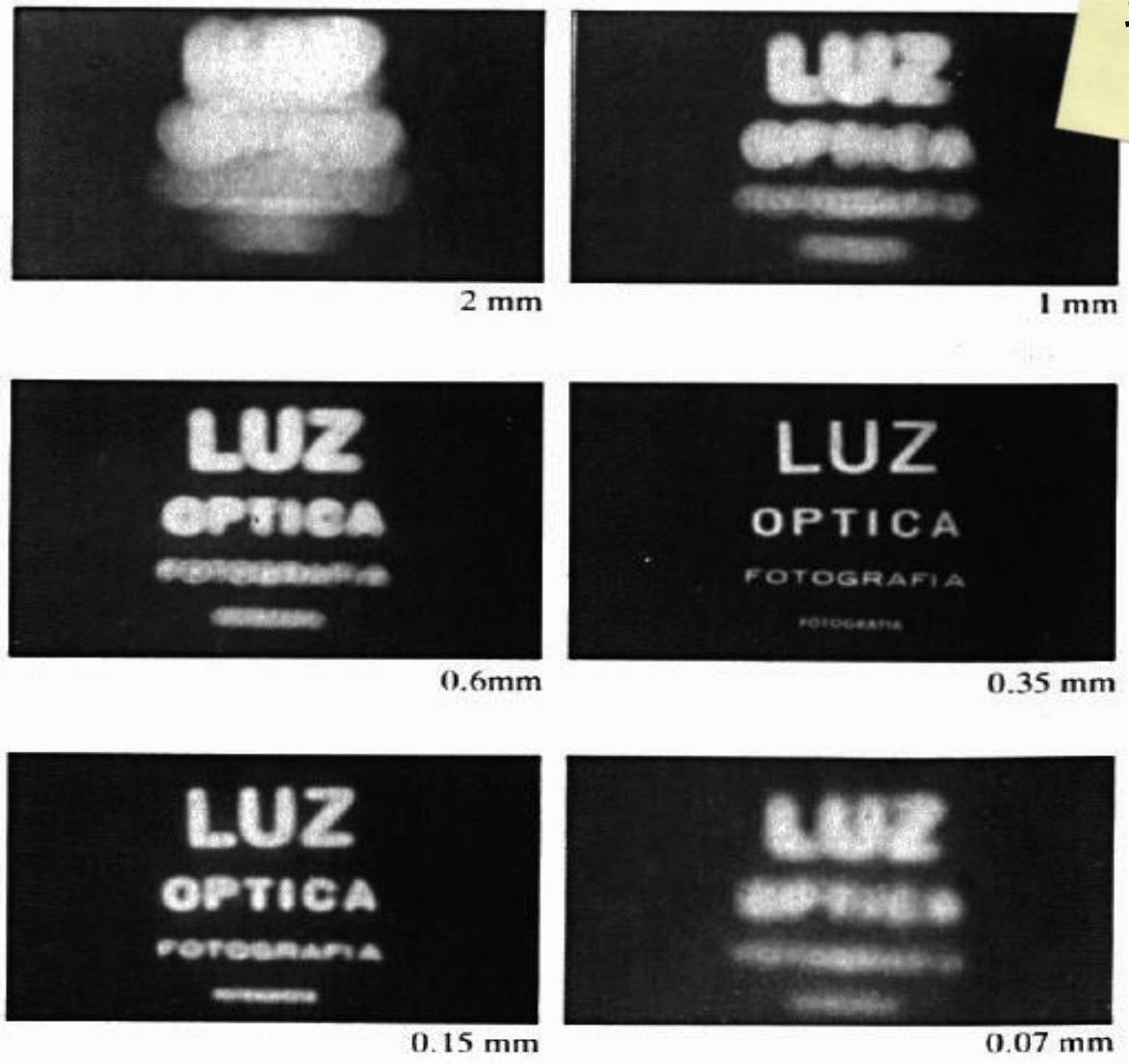


为什么不让孔径尺寸尽可能小？

缩小孔径



缩小孔径

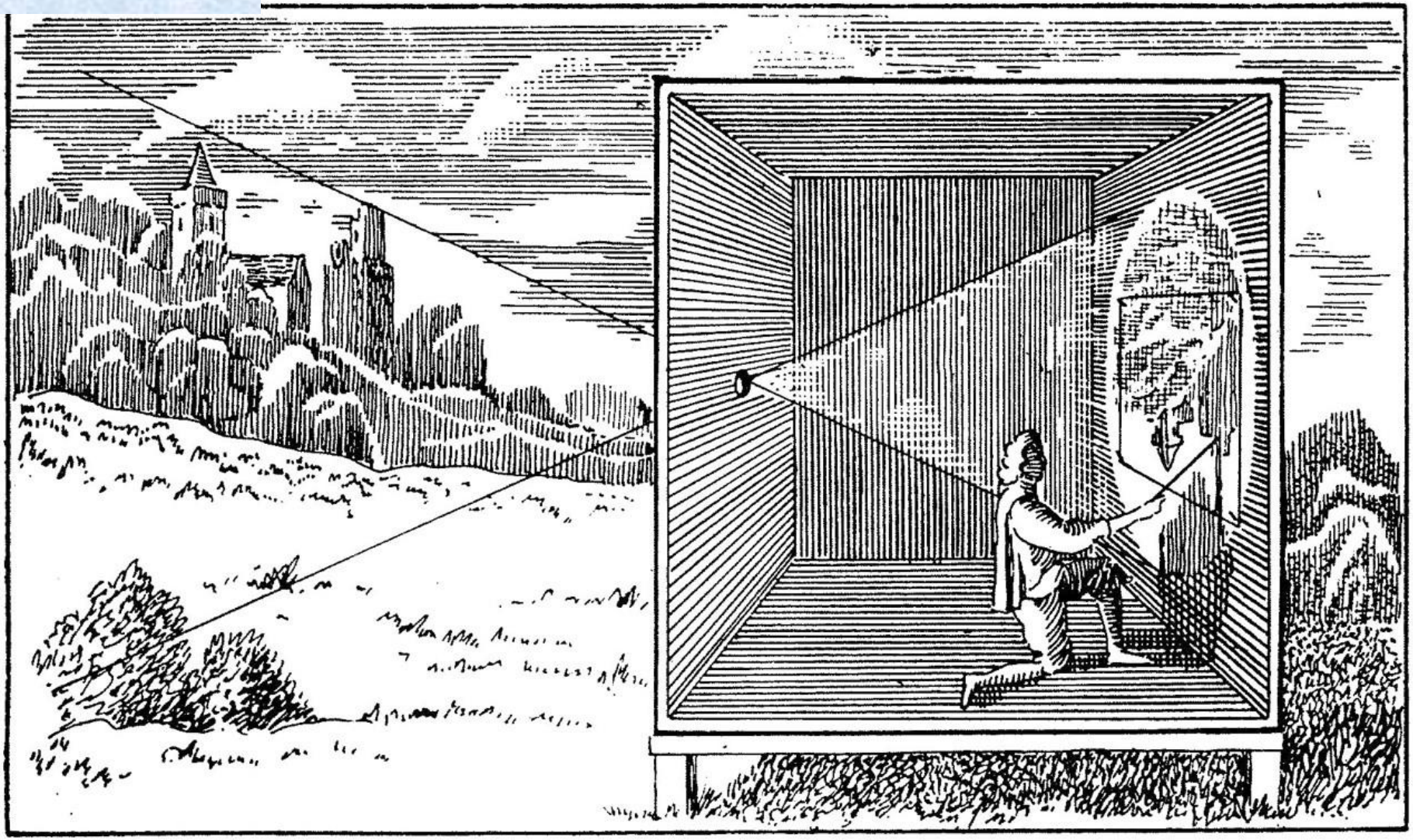

2 mm
1 mm
LUZ
OPTICA
0.6mm
LUZ
OPTICA
FOTOGRAFIA
0.35 mm
LUZ
OPTICA
FOTOGRAFIA
0.15 mm
0.07 mm


2 mm
1 mm
LUZ
OPTICA
FOTOGRAFIA
0.6mm
LUZ
OPTICA
FOTOGRAFIA
0.35 mm
LUZ
OPTICA
FOTOGRAFIA
0.15 mm
mm

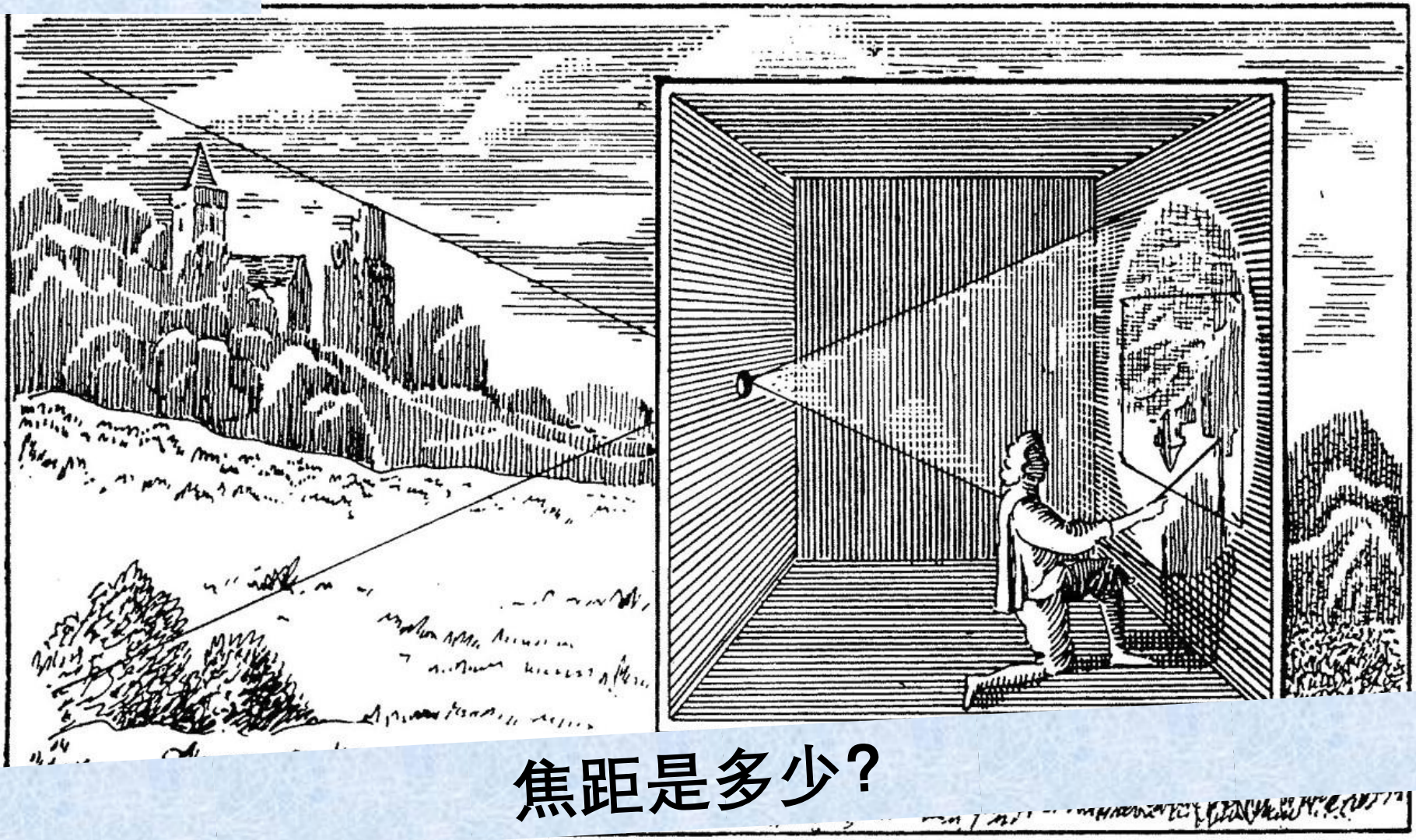
缩小孔径

衍射效应

相机暗盒

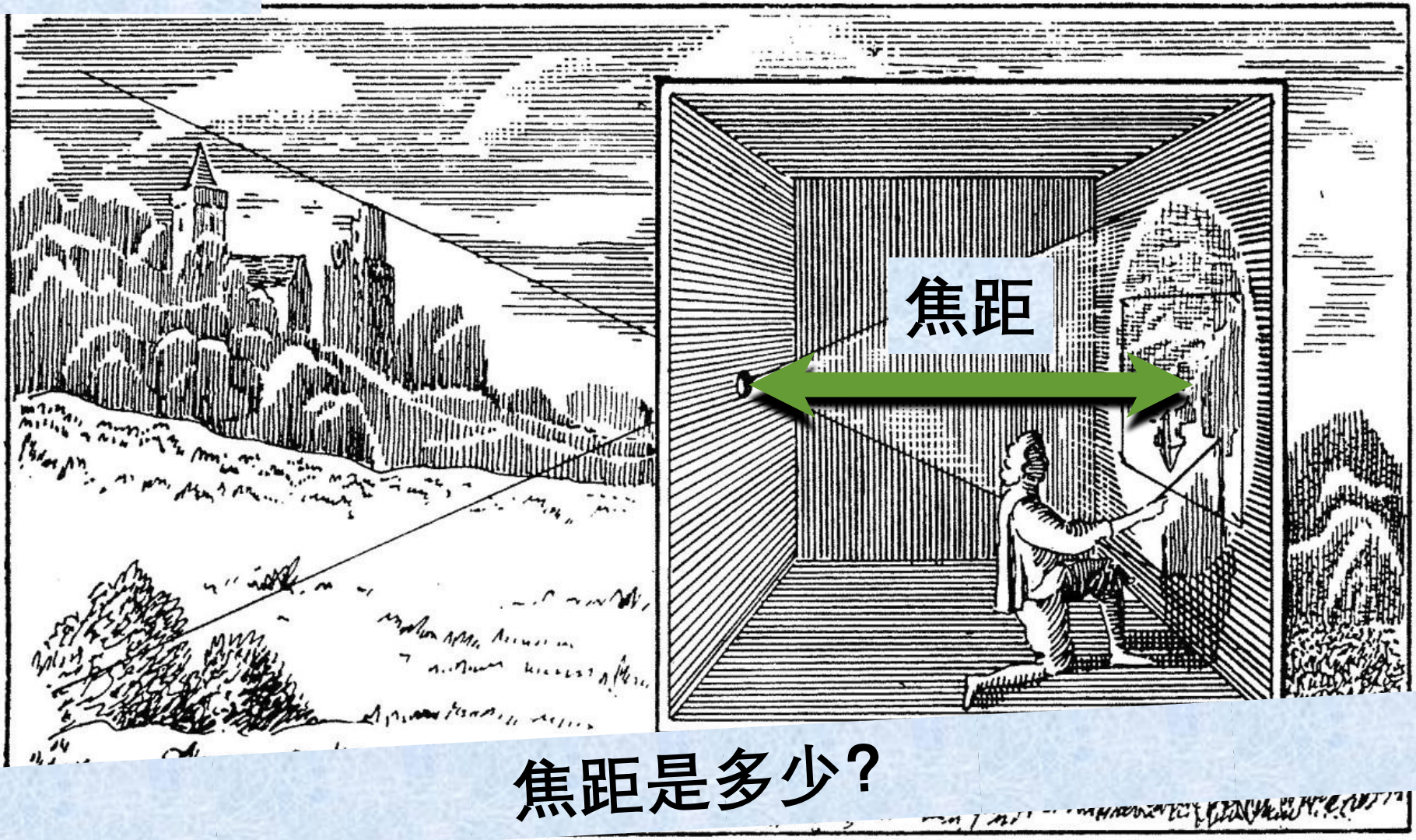


相机暗盒



焦距是多少?

相机暗盒



焦距是多少?


鸣谢：阿贝拉多·莫瑞尔

АГЕНТСТВО
КРЕАТИВНЫХ
КОММУНИКАЦИЙ
YELLOWBOOK

头戴相机暗室

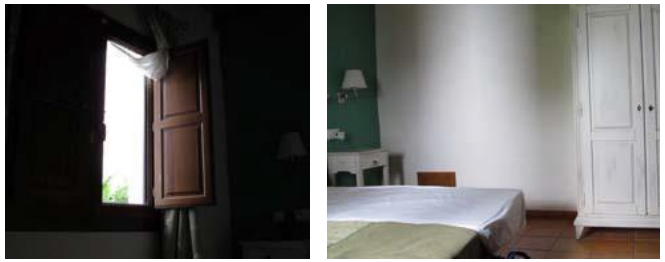
Accidental pinhole and pinspeck cameras: revealing the scene outside the picture

Antonio Torralba, William T. Freeman
Computer Science and Artificial Intelligence Laboratory (CSAIL)
MIT
torralba@mit.edu, billf@mit.edu

IEEE Computer Vision and Pattern Recognition (CVPR), 2012

Abstract

We identify and study two types of "accidental" images that can be formed in scenes. The first is an accidental pinhole camera image. These images are often mistaken for shadows, but can reveal structures outside a room, or the unseen shape of the light aperture into the room. The second class of accidental images are "inverse" pinhole camera images, formed by subtracting an image with a small occluder present from a reference image without the occluder. The reference image can be an earlier frame of a video sequence. Both types of accidental images happen in a variety of different situations (an indoor scene illuminated by natural light, a street with a person walking under the shadow of a building, etc.). Accidental cameras can reveal information about the scene outside the image, the lighti



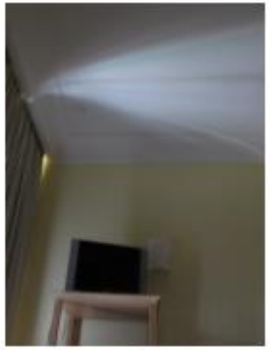
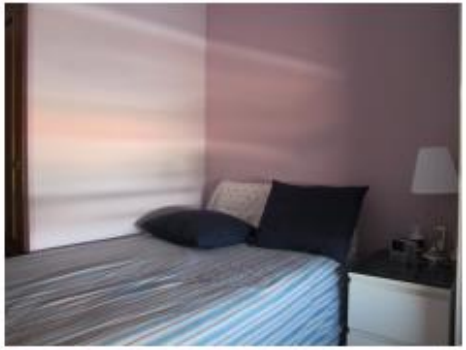
a) b)

Figure 1. a) Light enters the room via an open window. b) On the wall opposite the window, we can see a projected pattern of light and shadow. But, are the dark regions shadows? See Fig. 2

one restricts the set of light rays falli

ght from only a pinhole falling on it. A second way to view an image when looking at a surface is to restrict the reflected rays from the surface by looking at a mirror surface. All rays impinge on the surface, but only those from a particular direction reflect properly into our eye and so we again see an image when viewing a surface.

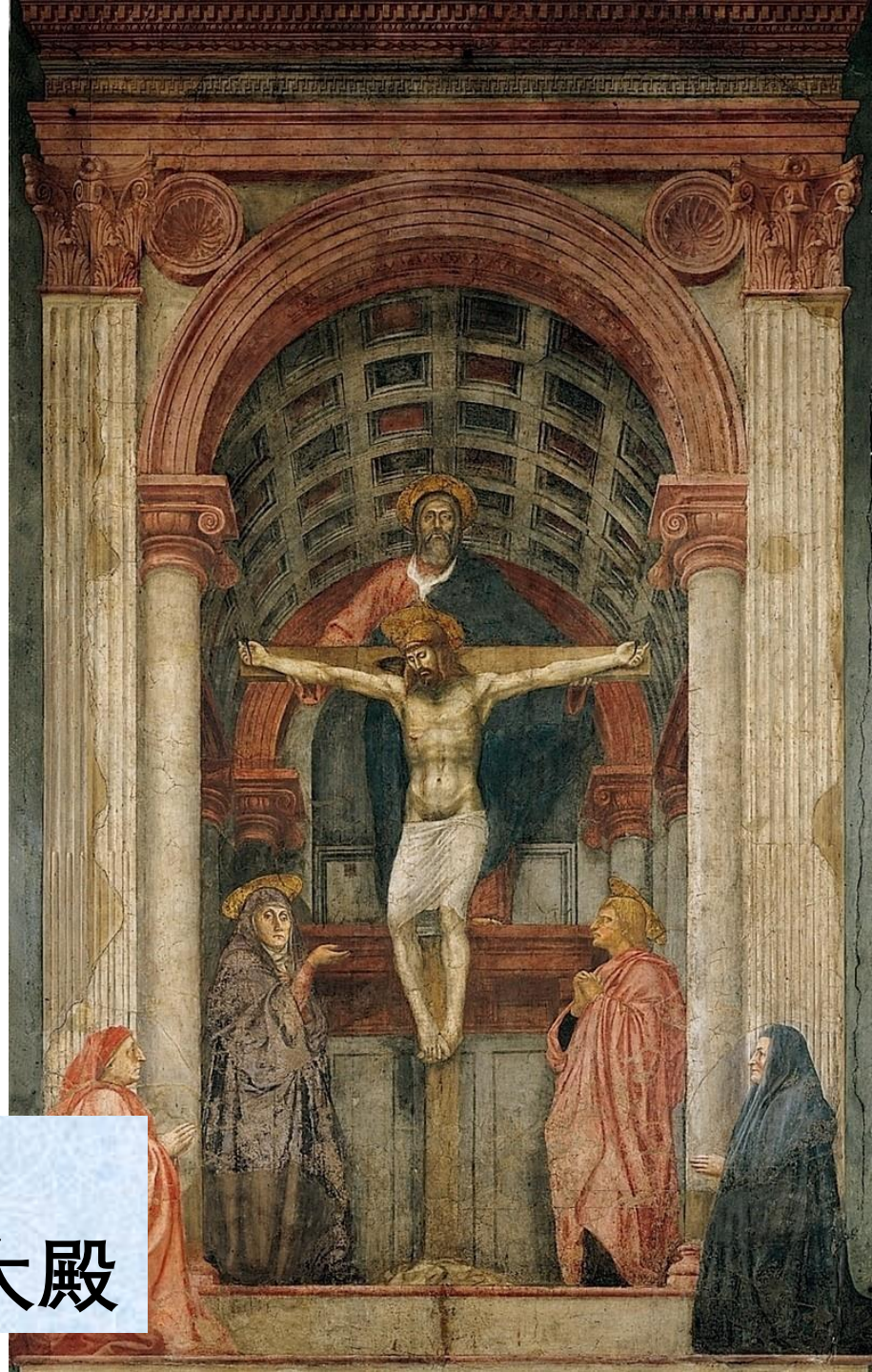
Researchers in computer vision have explored numerous ways to form images, including novel lenses, mirrors, coded apertures, and light sources (e.g. [1, 2, 7, 10]). The novel

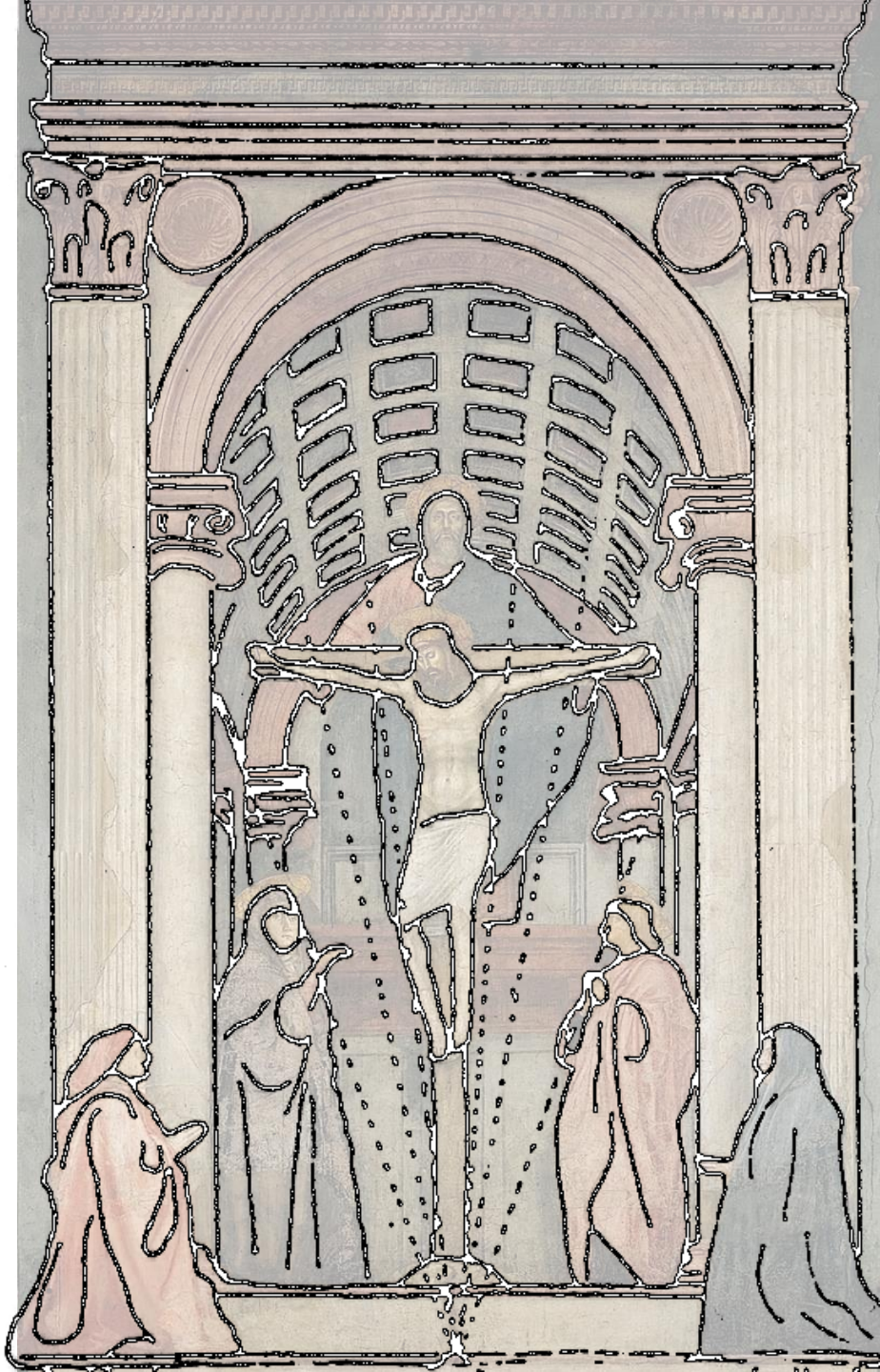


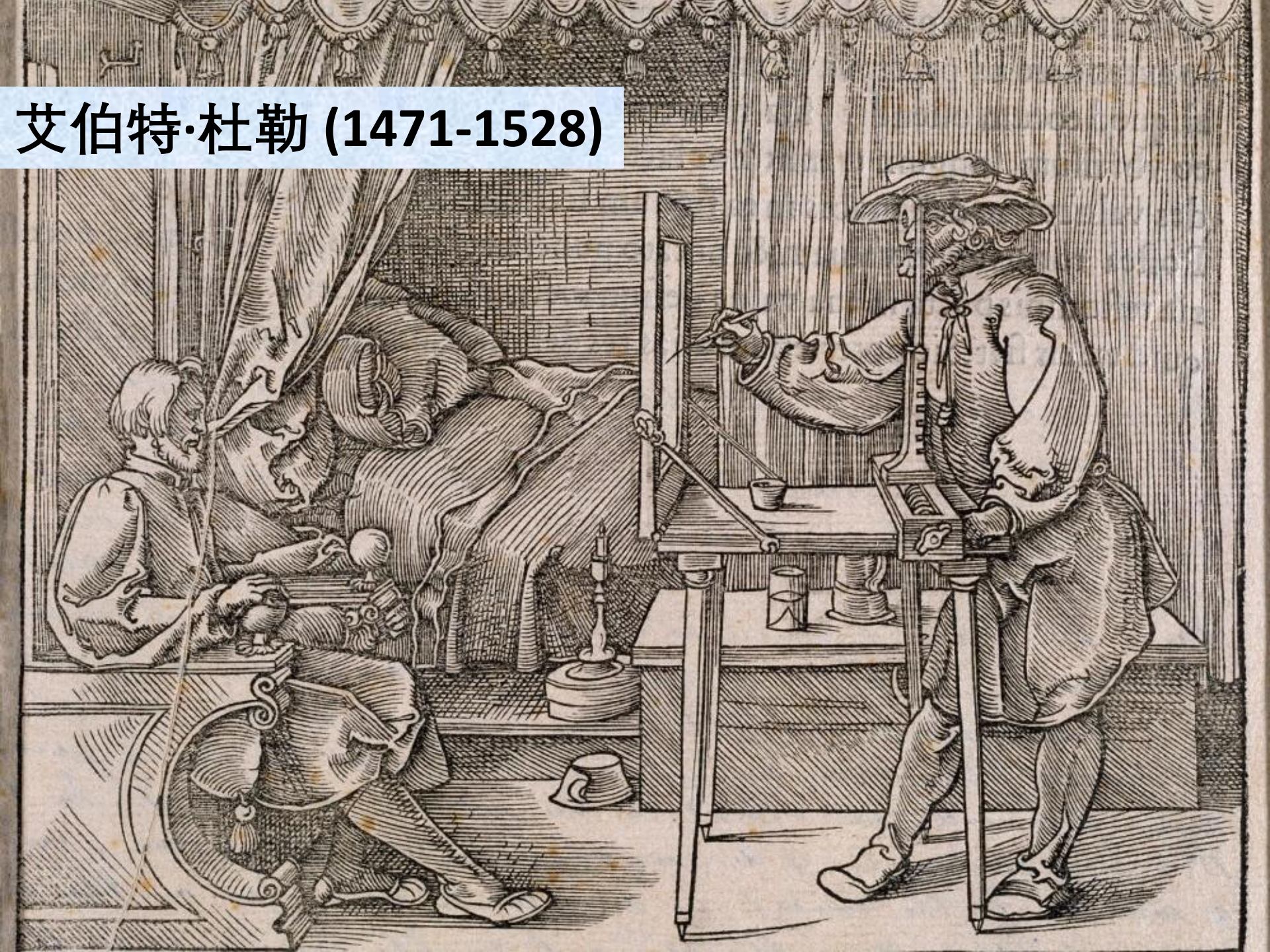


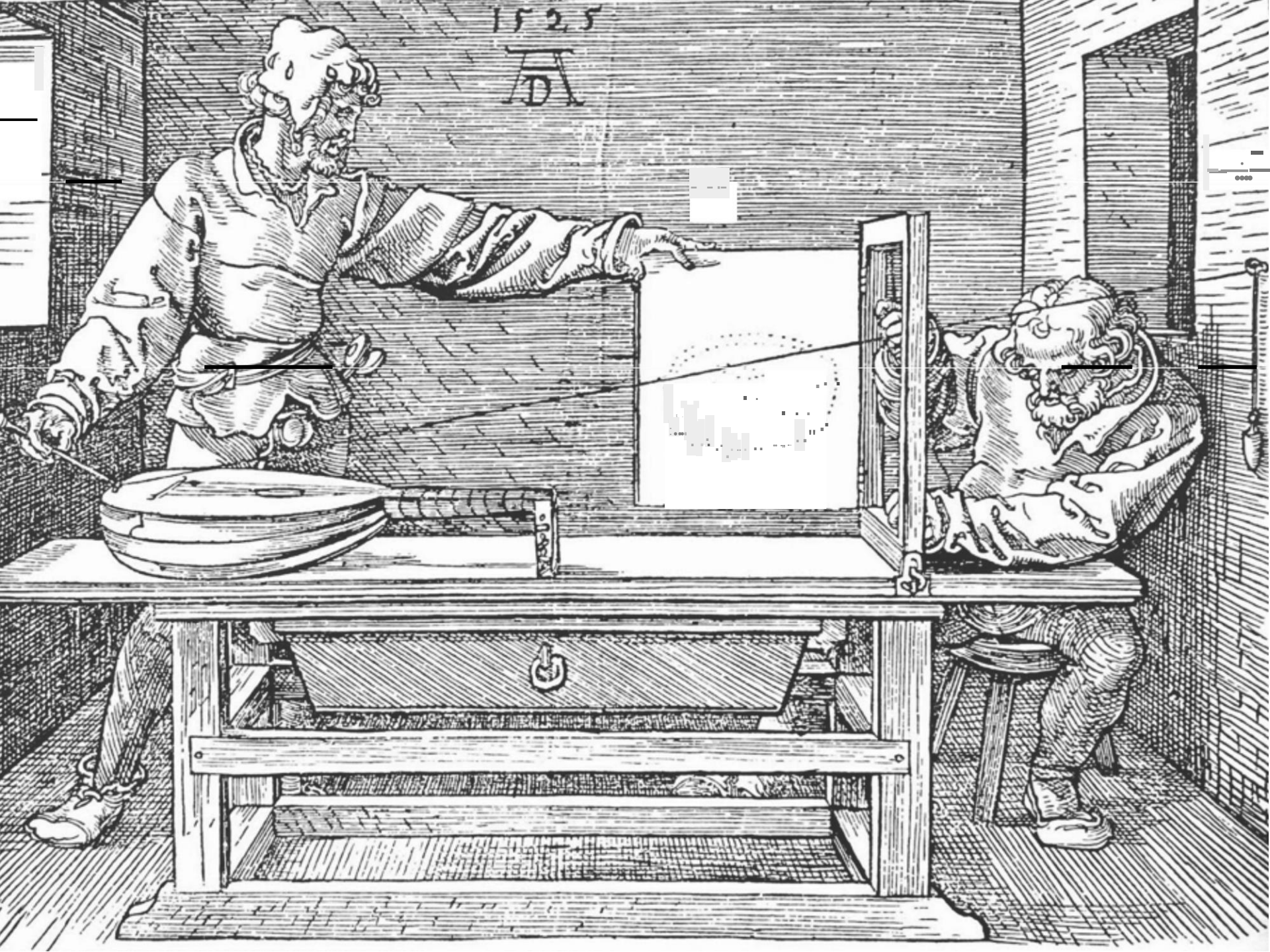


圣三位一体 (c. 1427-1428)
马萨乔，弗洛伦萨新圣母大殿

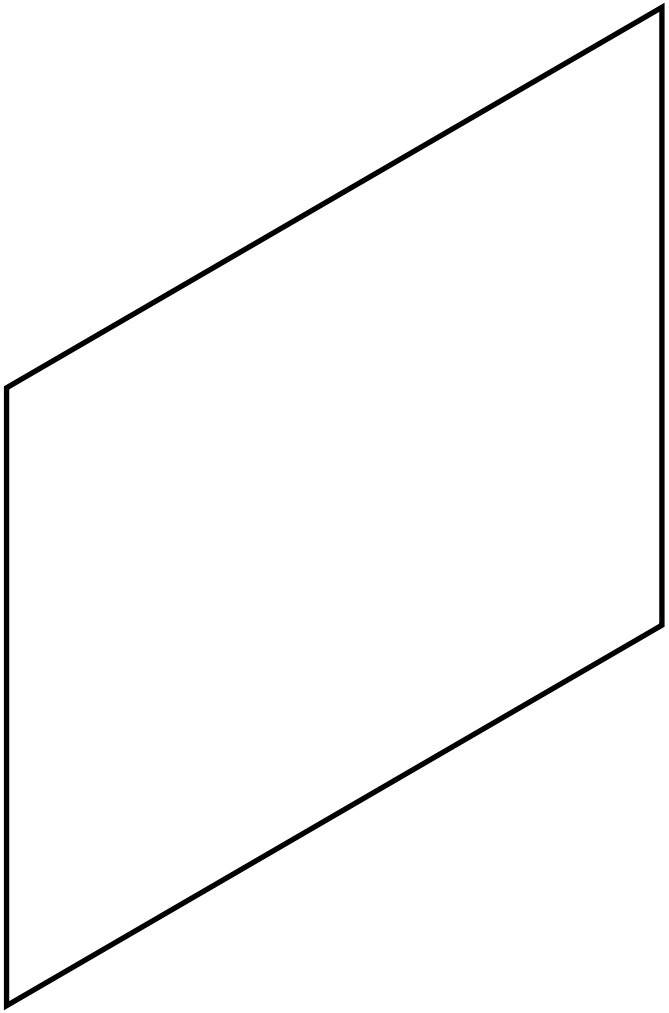


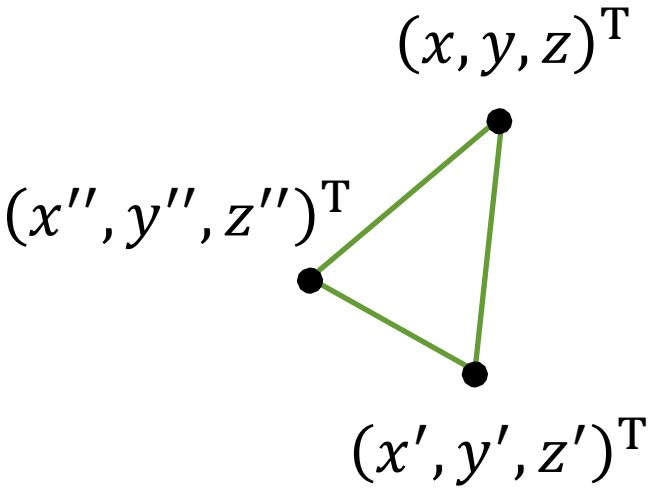

艾伯特·杜勒 (1471-1528)


1525
AD

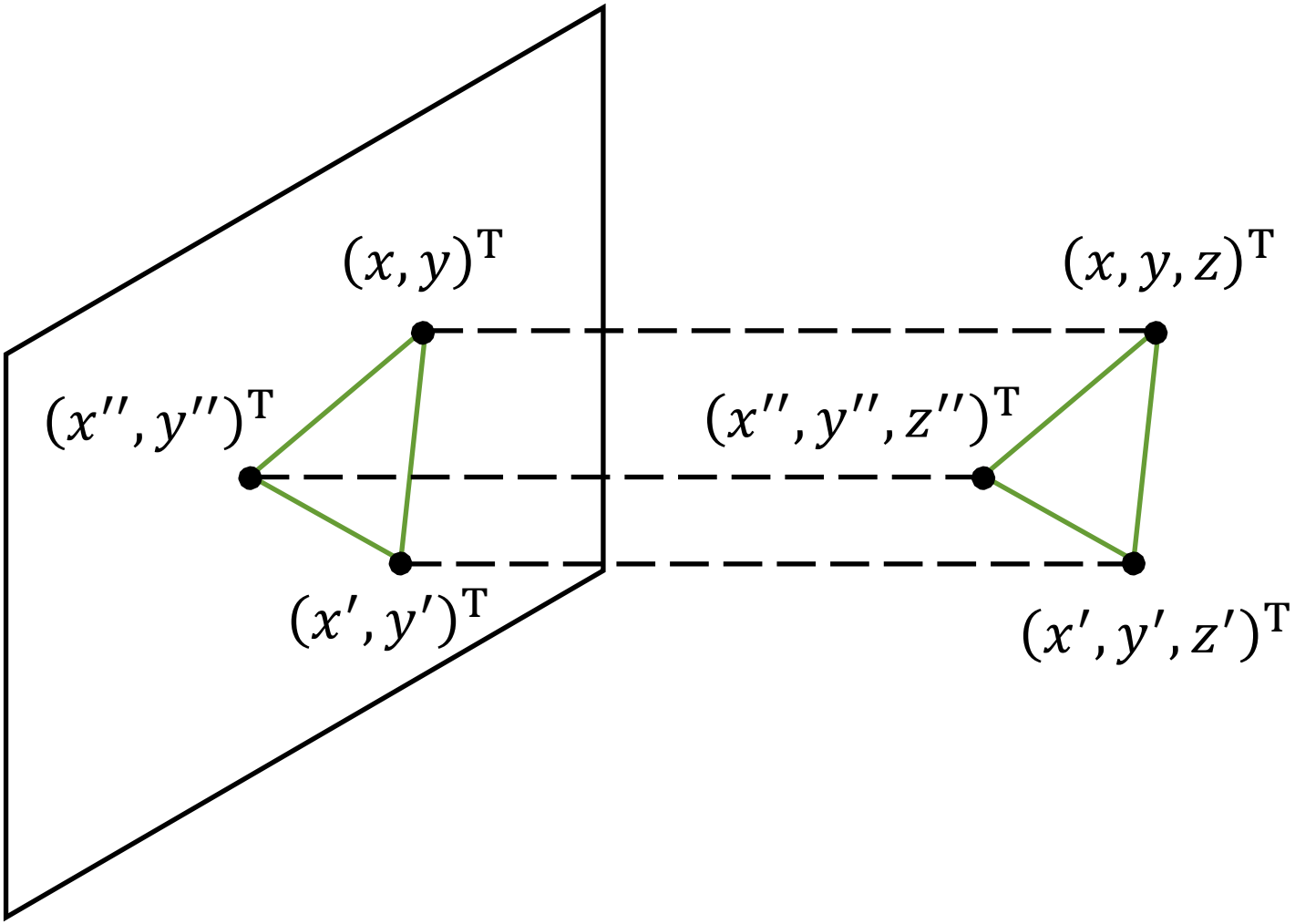
其他投影模型

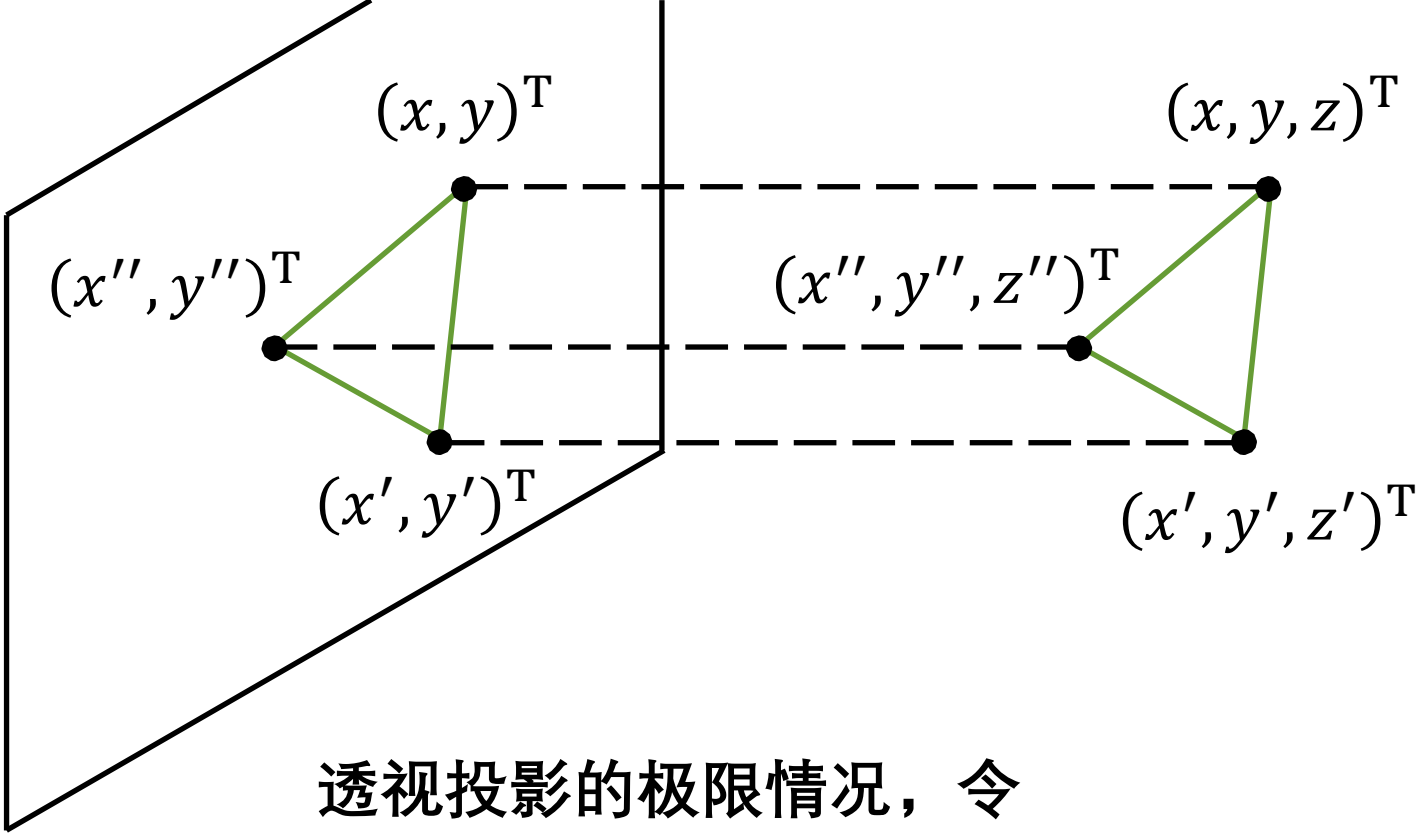
正投影





正投影


$(x, y)^{\mathrm{T}}$
$(x, y, z)^{\mathrm{T}}$
$(x'', y'')^{\mathrm{T}}$
$(x'', y'', z'')^{\mathrm{T}}$
$(x', y')^{\mathrm{T}}$
$(x', y', z')^{\mathrm{T}}$



透视投影的极限情况，令

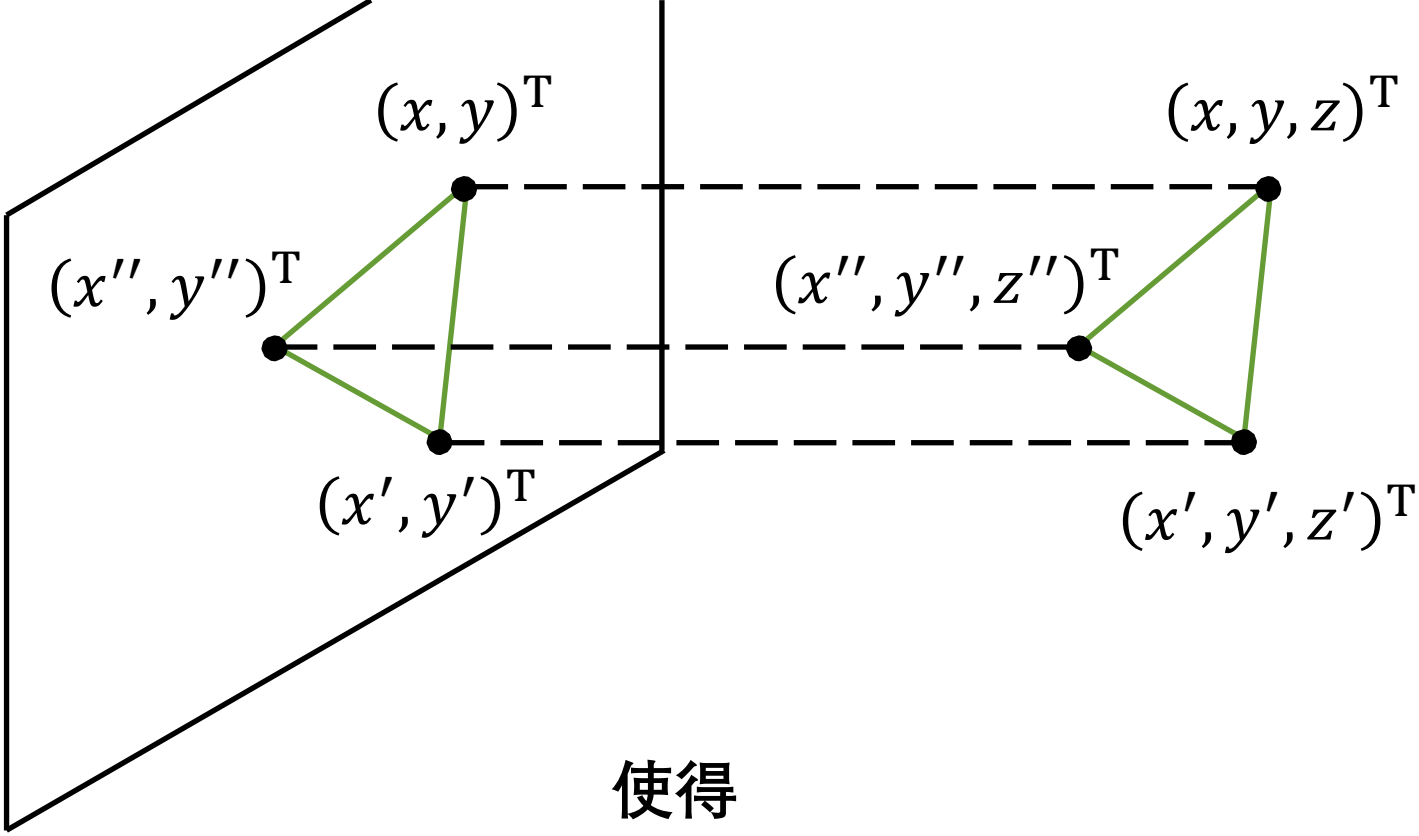
$$f \to \infty$$

相应地

$$z \to \infty$$

使得

$$f/z \to 1$$



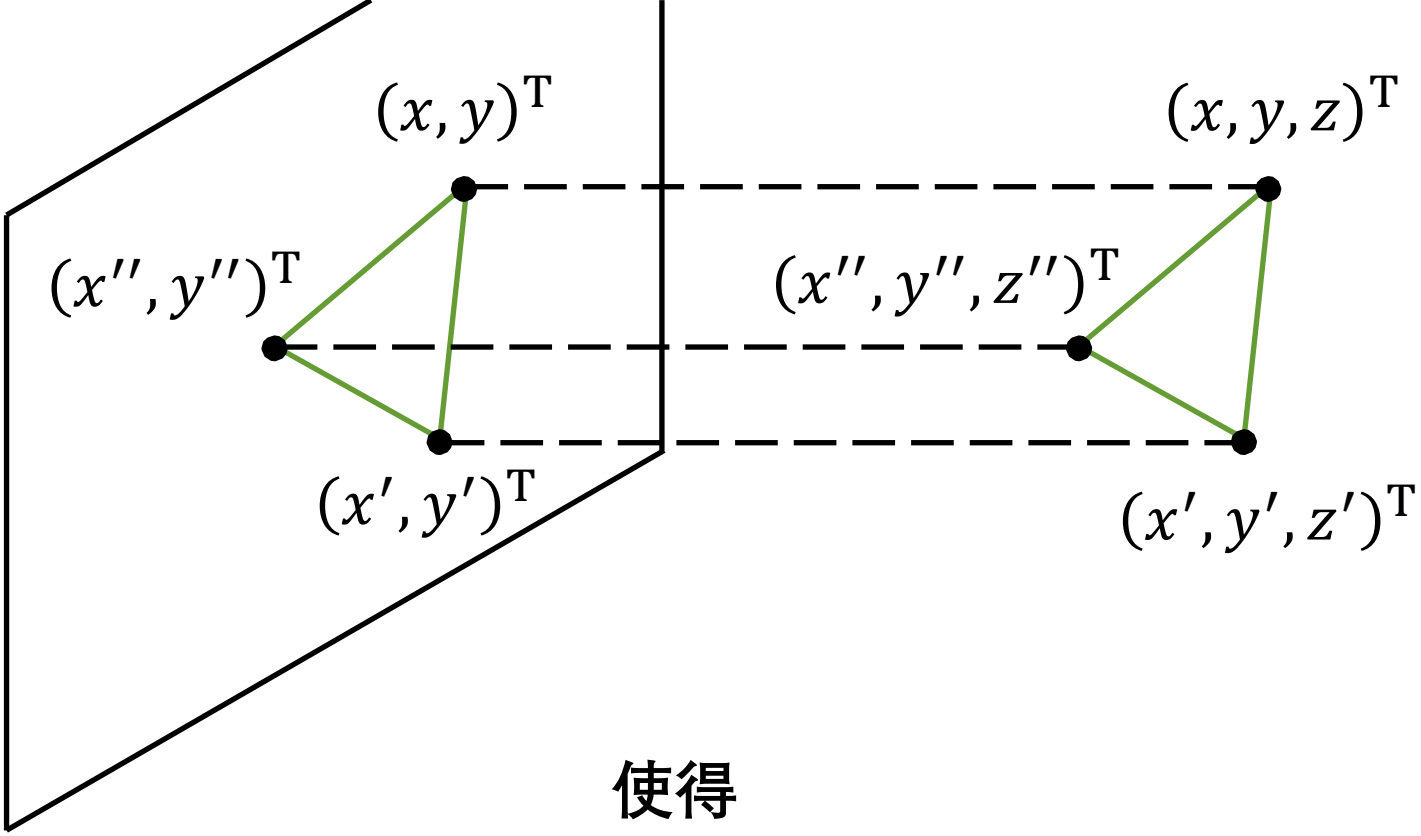
使得

$$f/z \to 1$$

透视方程可以近似为

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

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



使得

$$f/z \to 1$$

透视方程可以近似为

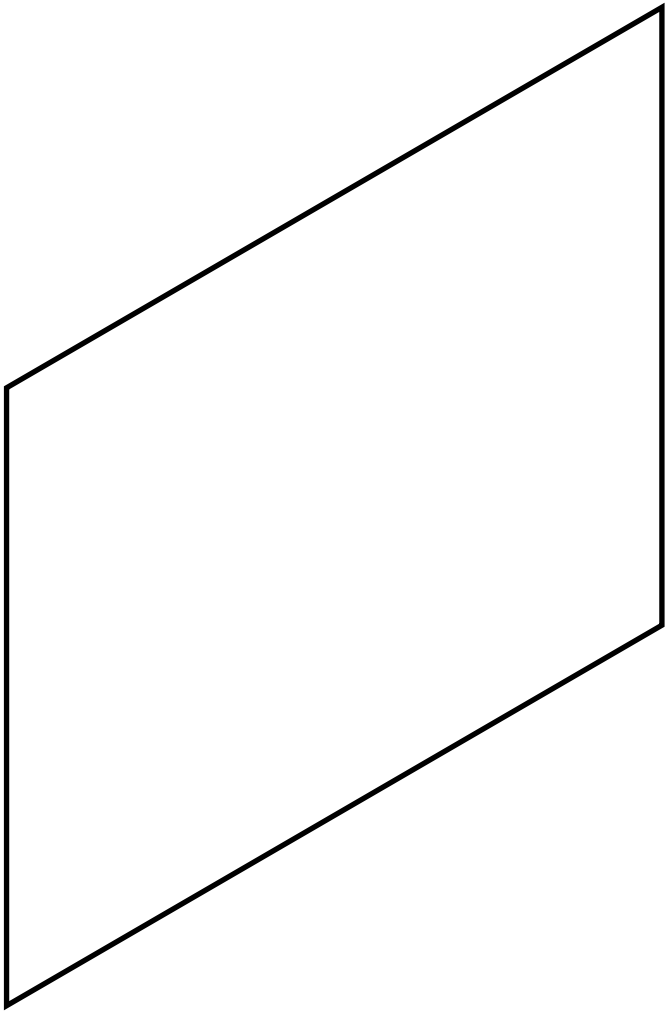
$$x = f\frac{X}{Z} = \frac{f}{Z}X \approx X$$

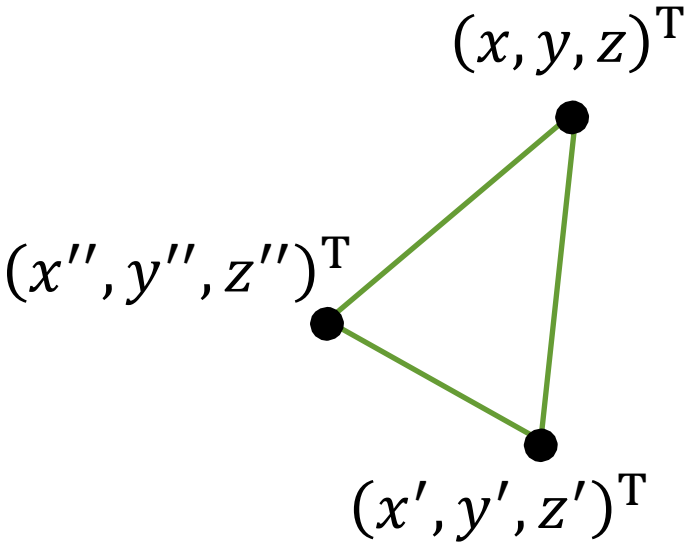
$$y = f\frac{Y}{Z} = \frac{f}{Z}Y \approx Y$$

康熙南巡图(c. 1427-1428)
王翬、杨晋等

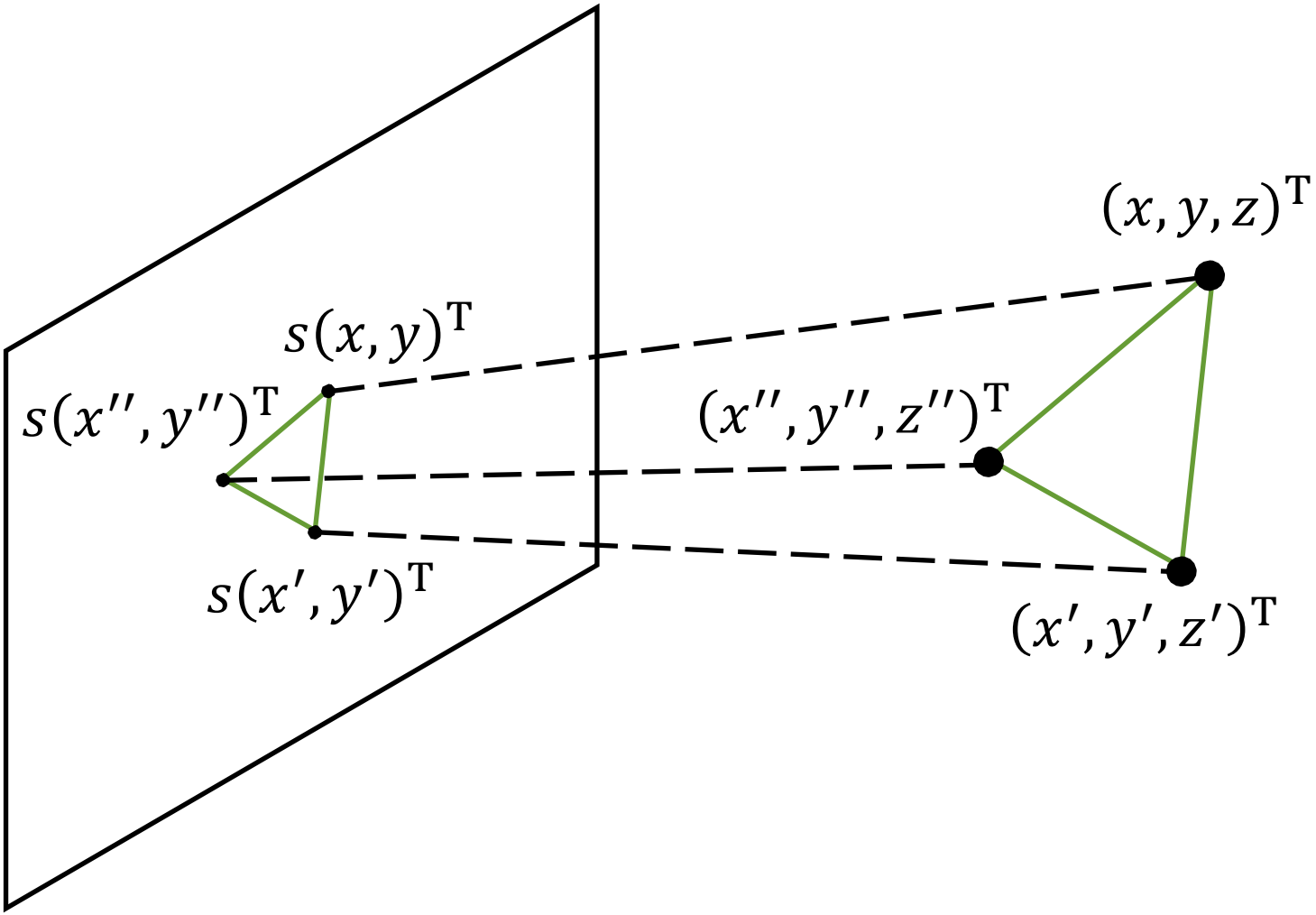

模拟城市4

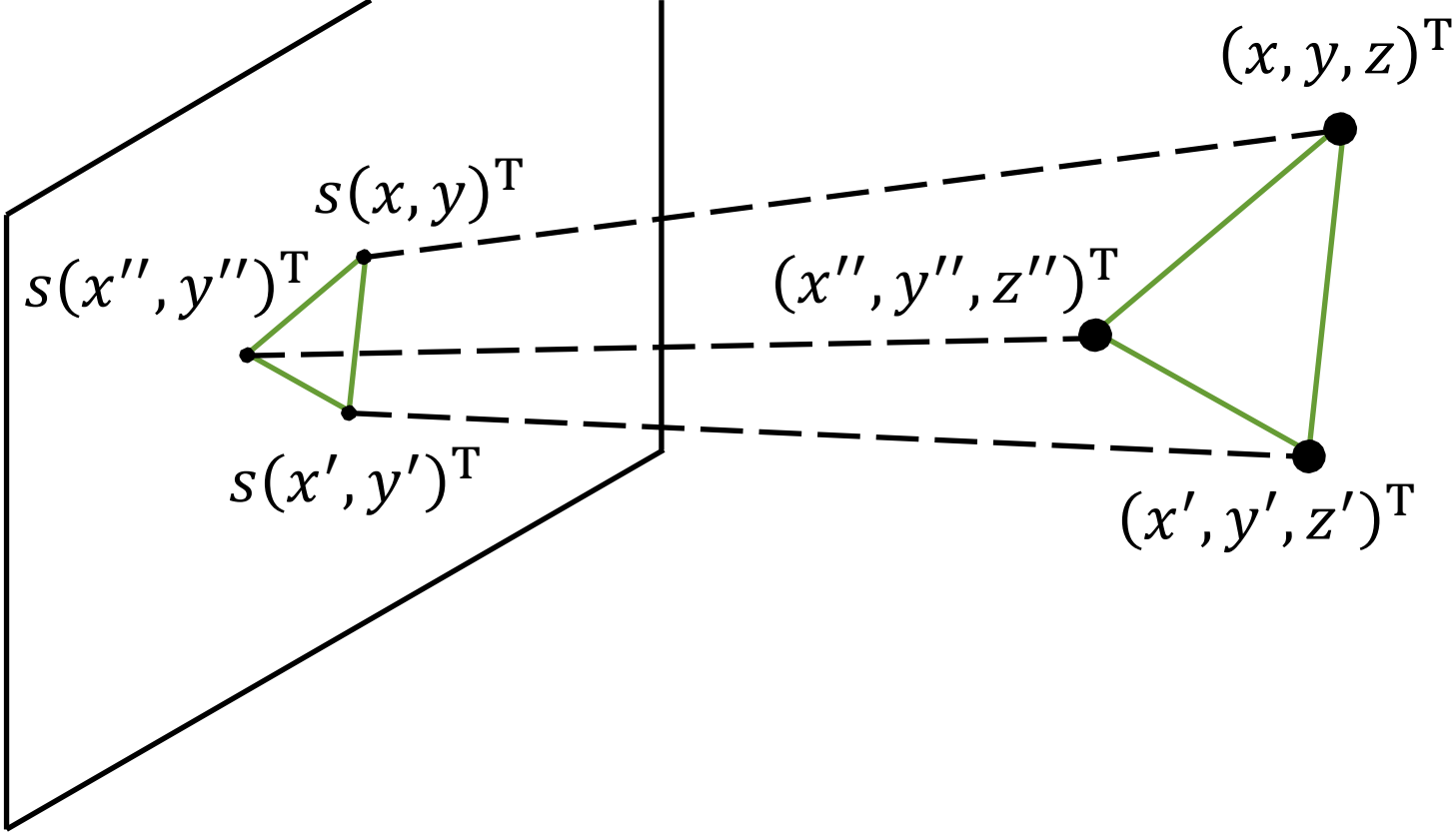
弱透视投影



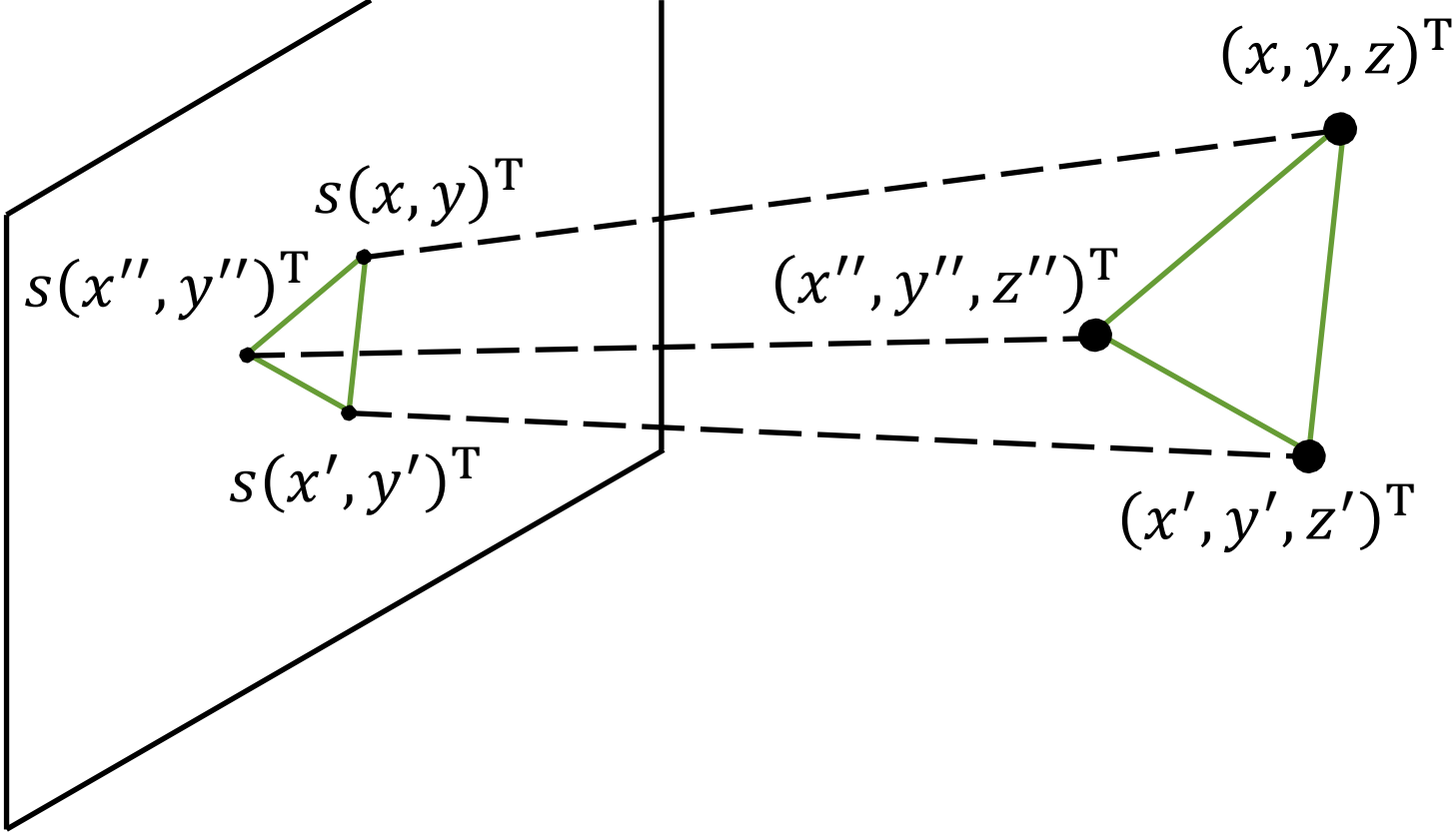


弱透视投影


$(x, y, z)^{\mathrm{T}}$
$s(x, y)^{\mathrm{T}}$
$s(x'', y'')^{\mathrm{T}}$
$(x'', y'', z'')^{\mathrm{T}}$
$s(x', y')^{\mathrm{T}}$
$(x', y', z')^{\mathrm{T}}$



假设场景沿着光轴的距离变化$d\bar{Z}$，和平均距离$\bar{Z}$相比很小



假设场景沿着光轴的距离变化$d\bar{Z}$，和平均距离$\bar{Z}$相比很小

透视方程可以近似为

$$x = f\frac{X}{Z} \approx f\frac{X}{\bar{Z}} = sX \qquad y = f\frac{Y}{Z} \approx f\frac{X}{\bar{Z}} = sY$$

新超级马里奥兄弟 (任天堂Wii)

Python时间

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

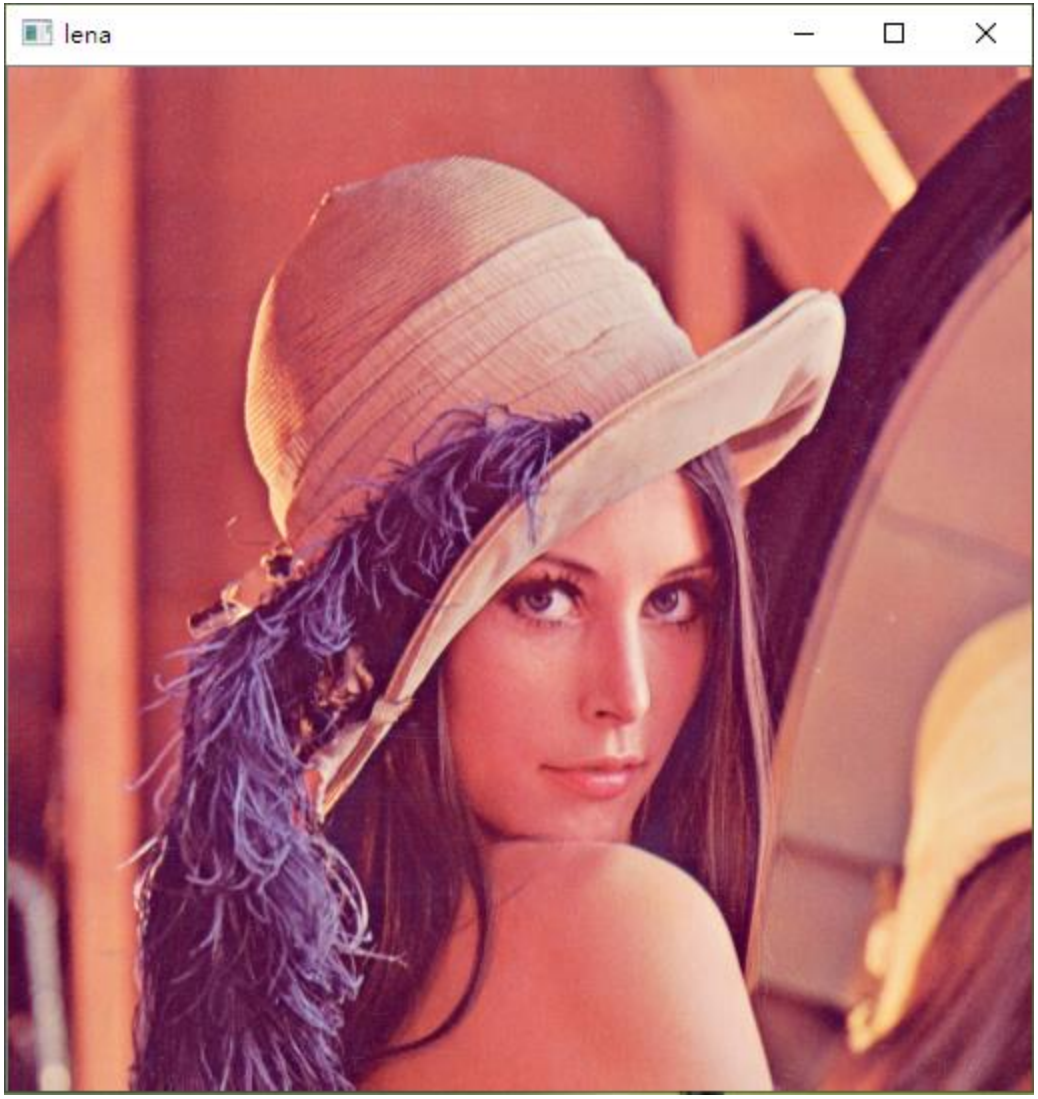
```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```


lena

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```


lena_grayscale

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

OpenCV默认颜色排列为蓝绿红

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```

Python中索引从0开始

图像基本操作

```
# import libraries
import numpy as np
import cv2

# read image and display
im = cv2.imread('lena.png', cv2.IMREAD_COLOR)
cv2.imshow('lena', im)
cv2.waitKey(0)
cv2.destroyWindow('lena')

# convert to gray scale
gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)

# convert uint8 to float
gray = gray.astype(float) / 255.0

# index specifical pixel
gray[row, col]
```