计算机视觉

立体视觉 下

中国传媒大学
COMMUNICATION UNIVERSITY OF CHINA

本节主题：

三角测量

本节主题：

三角测量

立体匹配

本节主题：

三角测量

立体匹配

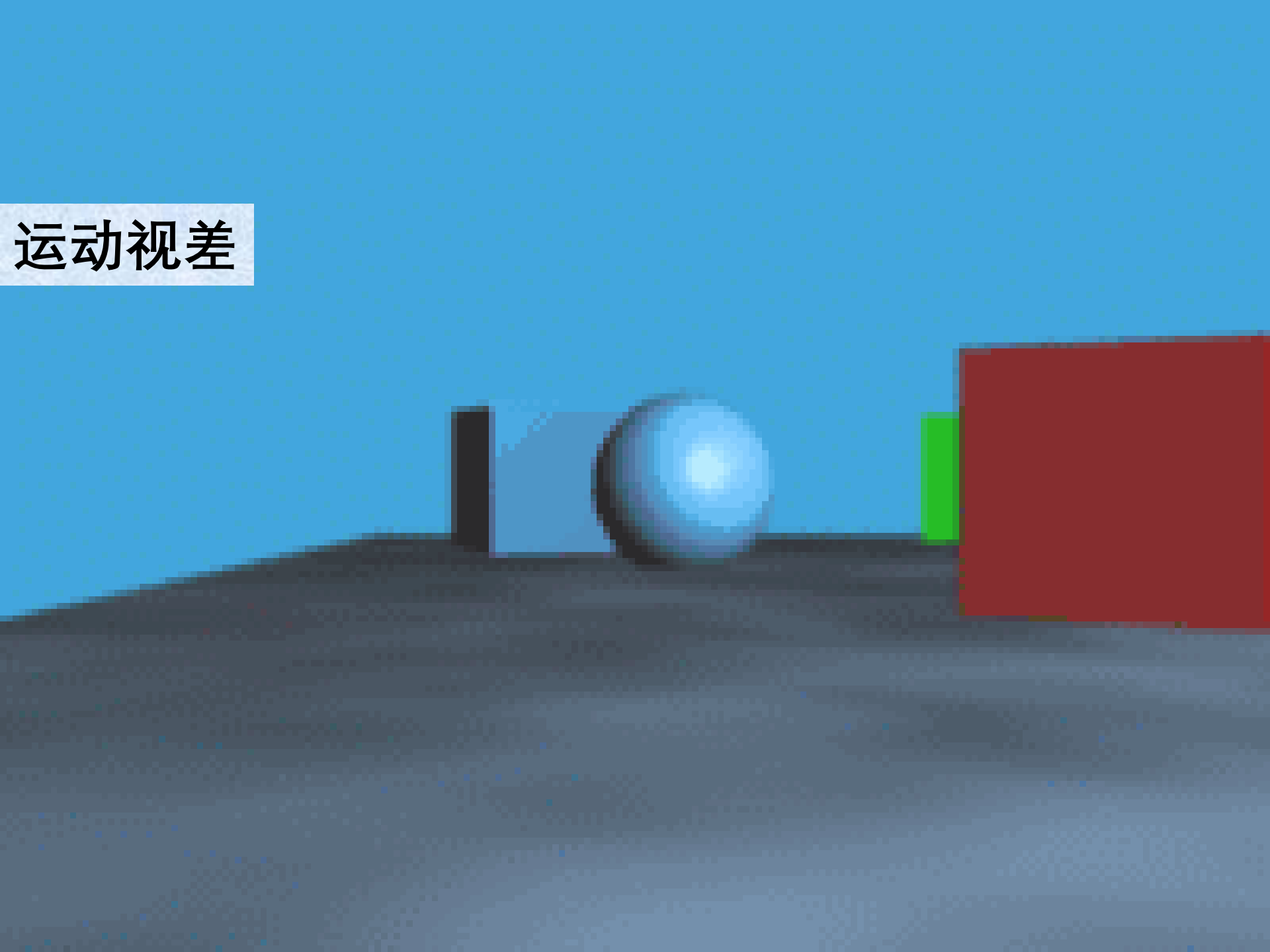
平面扫描

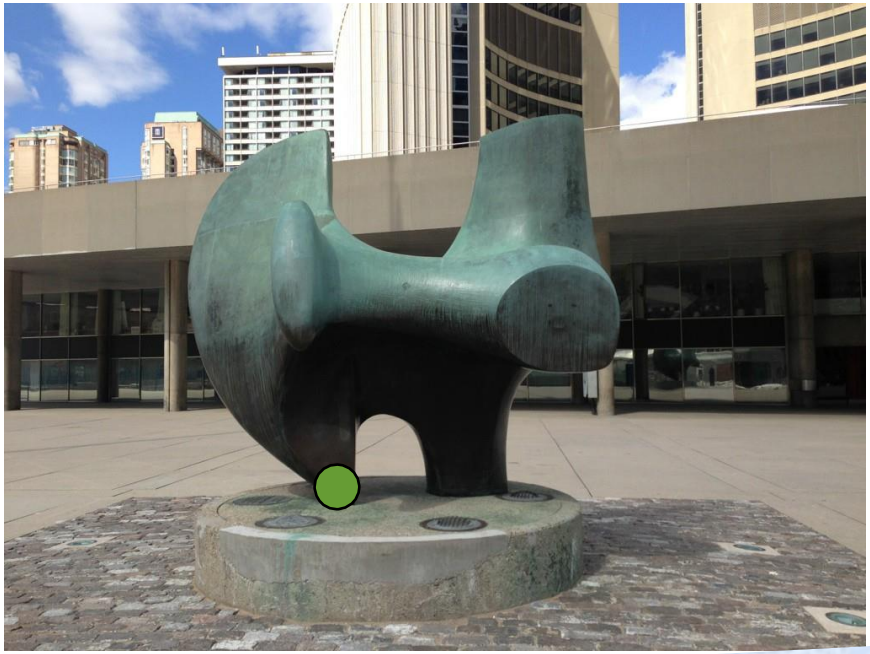
我们如何从图像中恢复3D几何结构?


单视点是有歧义的

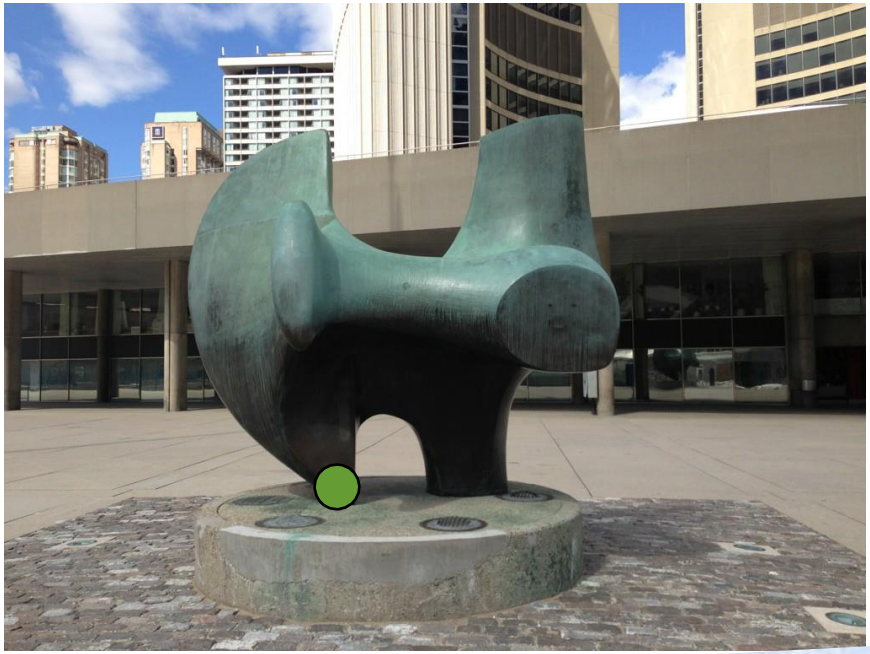

多视点

运动视差



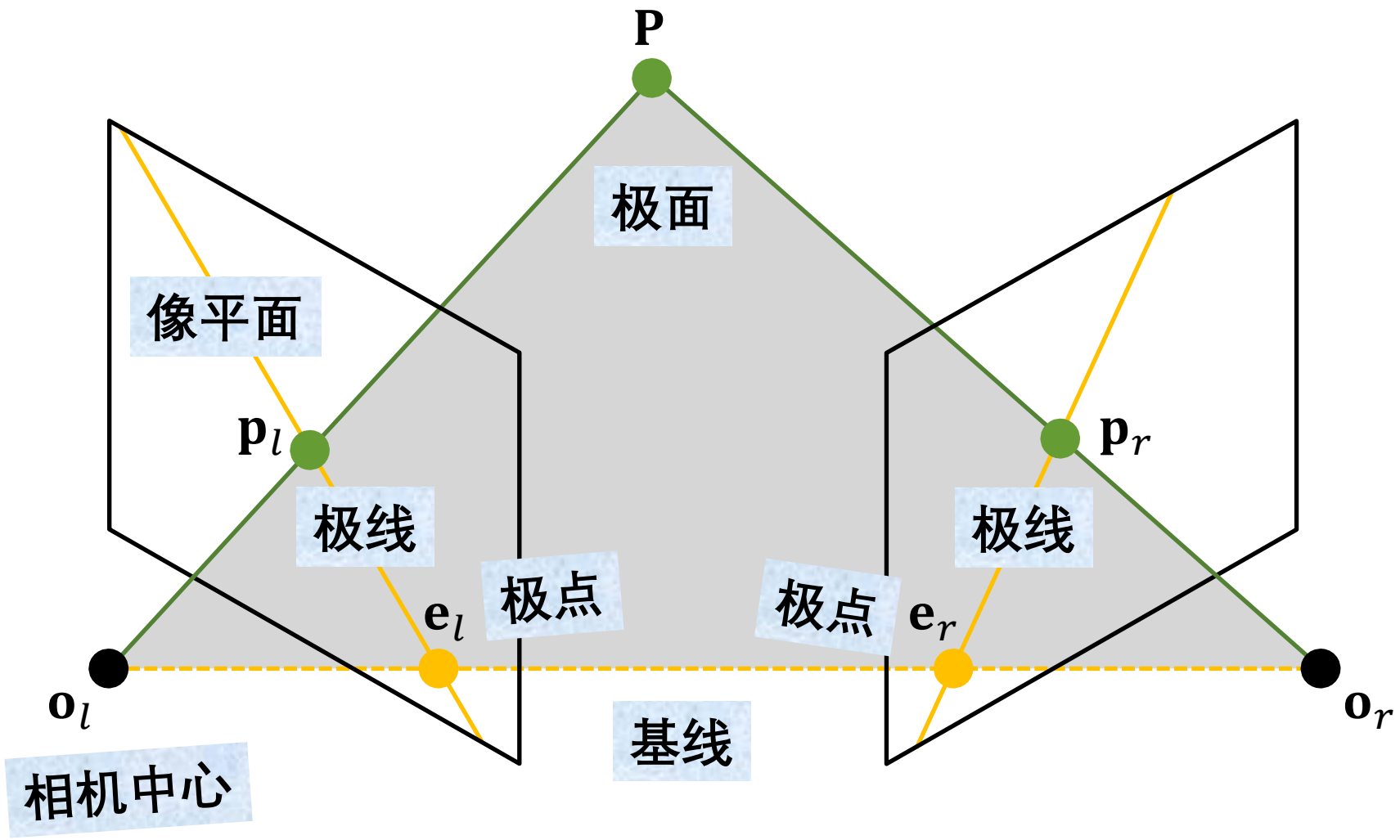


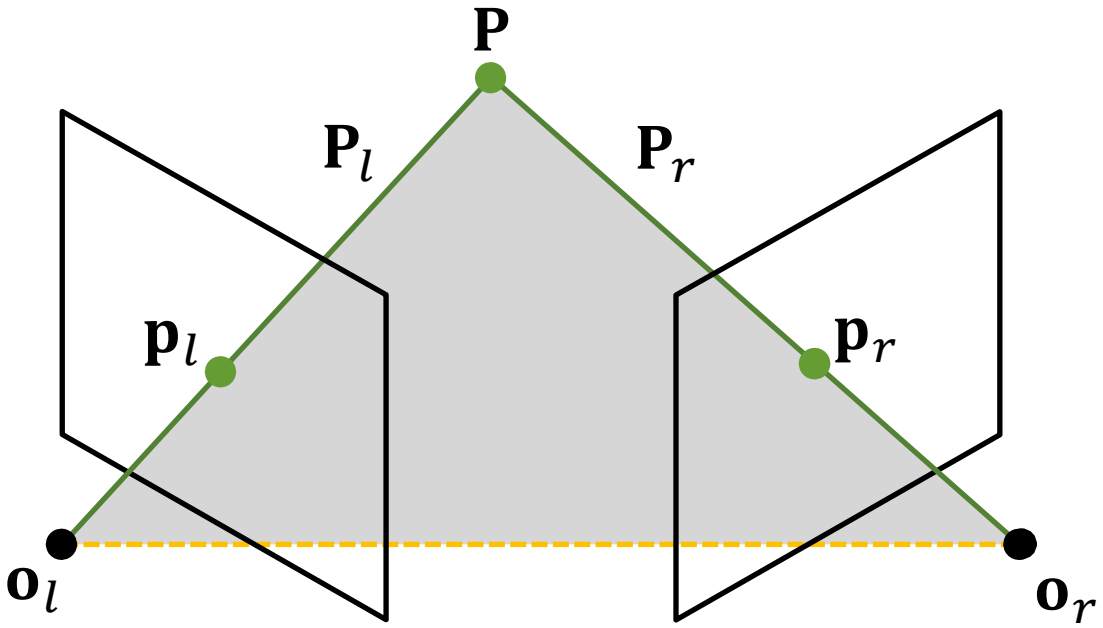
给定左图中的一点



给定左图中的一点

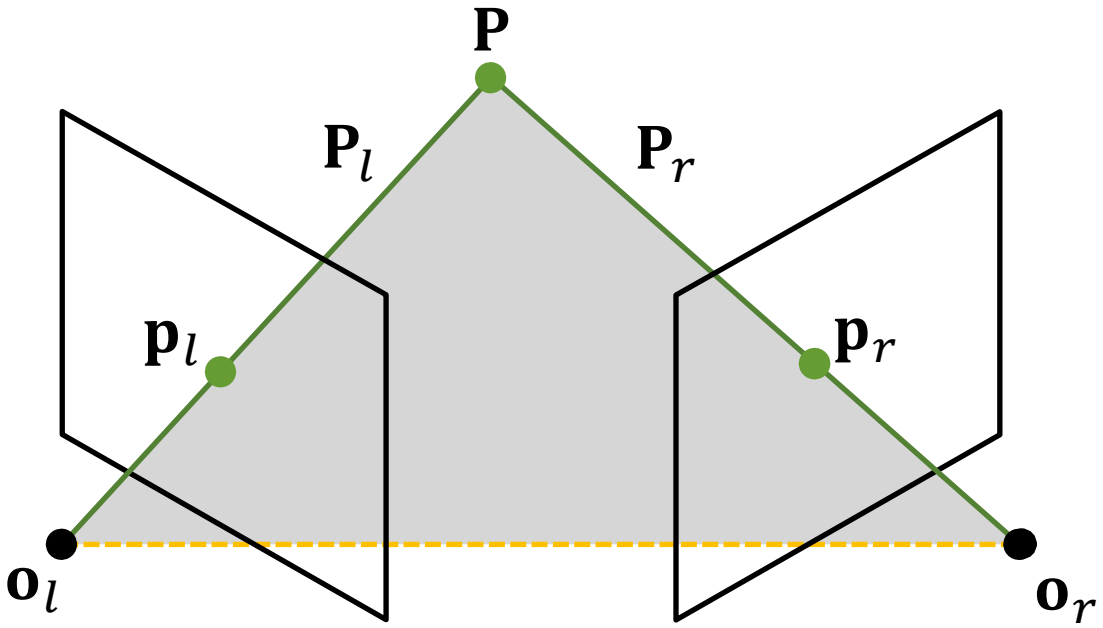
如何找出它在右图中的对应点


P
极面
像平面
$\mathbf{p}_l$
$\mathbf{p}_r$
极线
极线
极点
极点
$\mathbf{e}_l$
$\mathbf{e}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$
基线
相机中心



基础矩阵：$\mathbf{F} = \mathbf{M}_{\text{int},r}^{-\mathrm{T}} \mathbf{E} \mathbf{M}_{\text{int},l}^{-1}$

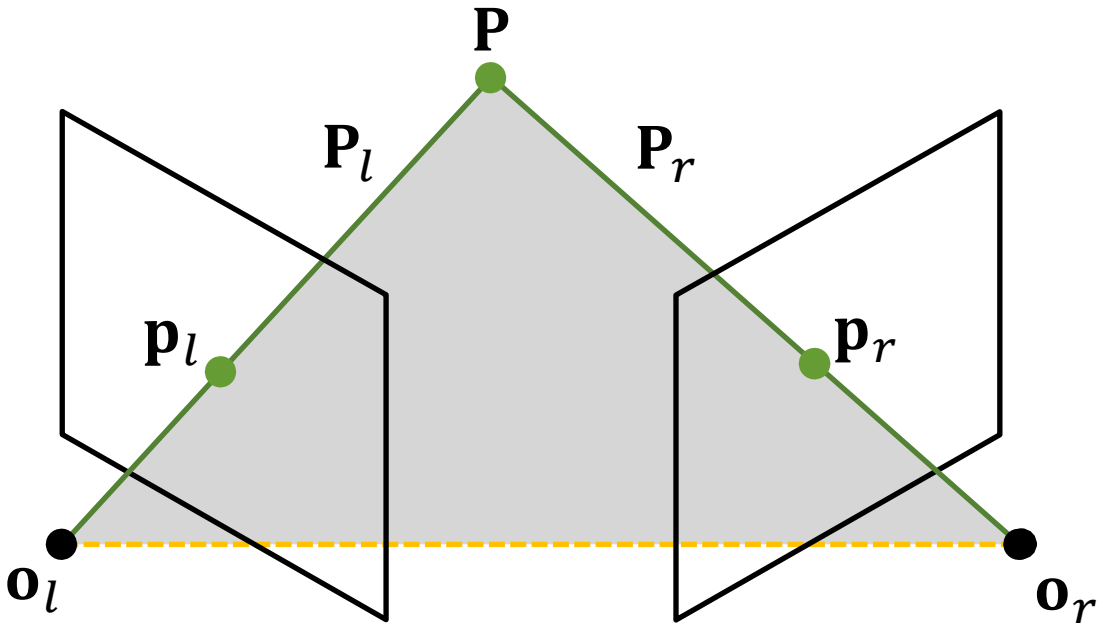
基础矩阵约束：$\tilde{\mathbf{p}}_r^{\mathrm{T}} \mathbf{F} \tilde{\mathbf{p}}_l = 0$



基础矩阵：$\mathbf{F} = \mathbf{M}_{\text{int},r}^{-\mathrm{T}}\mathbf{E}\mathbf{M}_{\text{int},l}^{-1}$

基础矩阵约束：$\tilde{\mathbf{p}}_r^{\mathrm{T}}\mathbf{F}\tilde{\mathbf{p}}_l = 0$

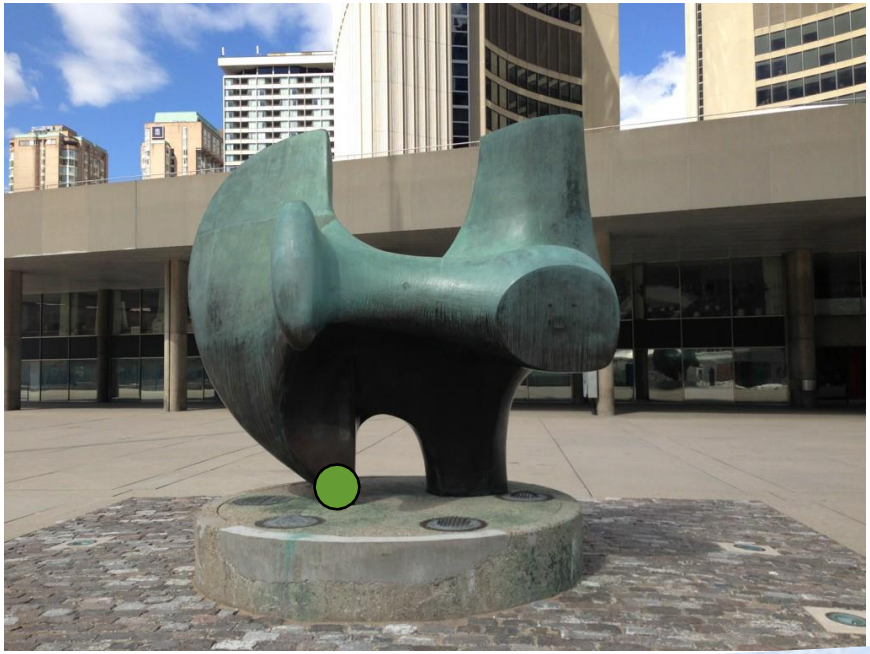
7个自由度



基础矩阵：$\mathbf{F} = \mathbf{M}_{\text{int},r}^{-\mathrm{T}}\mathbf{E}\mathbf{M}_{\text{int},l}^{-1}$

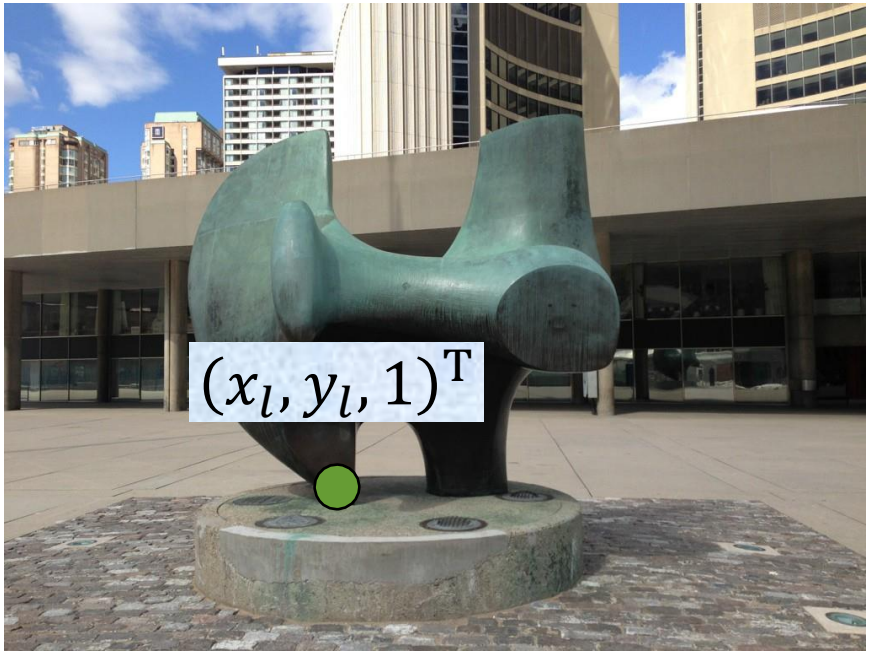
基础矩阵约束：$\tilde{\mathbf{p}}_r^{\mathrm{T}}\mathbf{F}\tilde{\mathbf{p}}_l = 0$

P在左、右两个视点中的像素坐标之间的约束关系

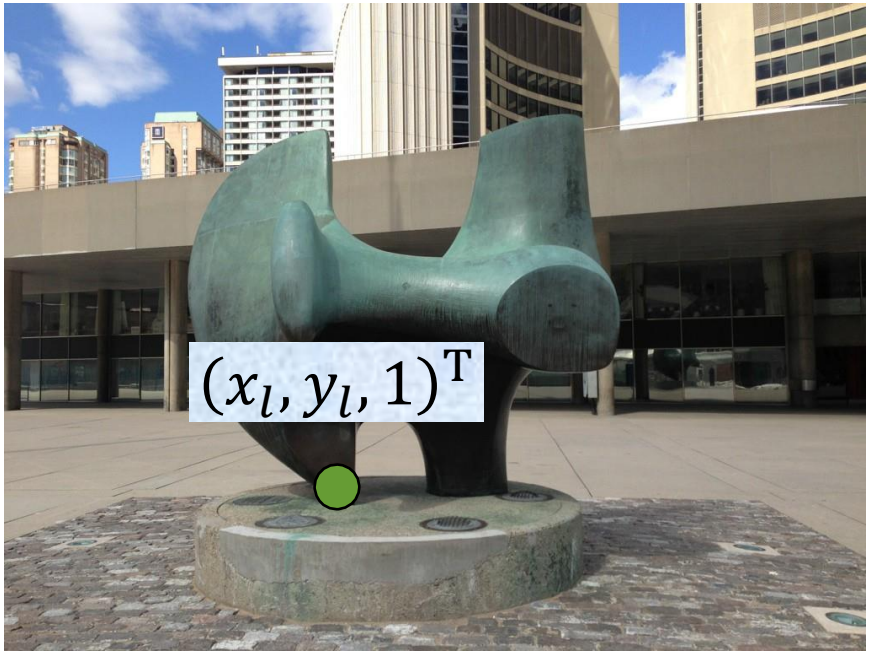


给定左图中的一点

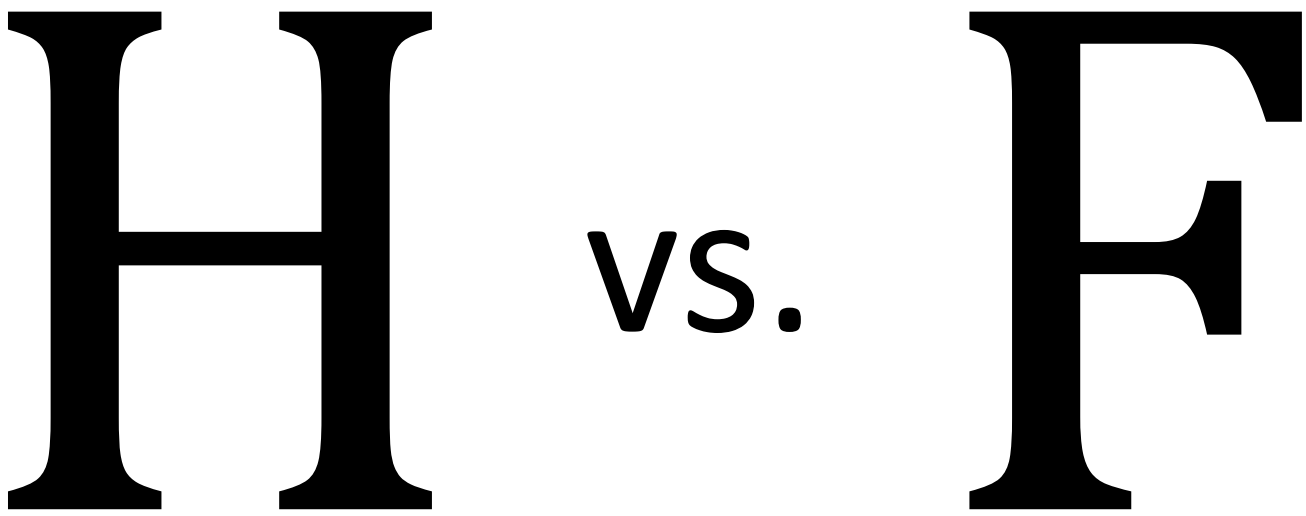
如何找出它在右图中的对应点


$(x_l, y_l, 1)^{\mathrm{T}}$


极线：$\mathbf{l}' = \mathbf{F}\mathbf{x}_l$



搜索空间被缩小到一条直线



单应矩阵　　基础矩阵

$$\mathbf{x}' = \mathbf{H}\mathbf{x}$$

单应矩阵将一个点映射到一个点

$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$

基础矩阵将一个点映射到一条直线

回顾:
8点算法

A computer algorithm for reconstructing a scene from two projections

H. C. Longuet-Higgins

Laboratory of Experimental Psychology, University of Sussex,
Brighton BN1 9QG, UK

8点算法

A simple algorithm for computing the three-dimensional structure of a scene from a correlated pair of perspective projections is described here, when the spatial relationship between the two projections is unknown. This problem is relevant not only to photographic surveying[1] but also to binocular vision[2], where the non-visual information available to the observer about the orientation and focal length of each eye is much less accurate than the optical information supplied by the retinal images

Nature, 1981

回顾：
8点算法

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

n对对应点

回顾：
8点算法

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

矩阵秩为8

回顾：
8点算法

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

矩阵秩为8

8对对应点可得一组非零解

回顾：
8点算法

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

矩阵秩为8

8对对应点可得一组非零解

解是零空间

SVD求解
基础矩阵

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\underbrace{\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix}}_{\mathbf{A}} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

SVD求解
基础矩阵

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\underbrace{\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix}}_{\mathbf{f}} = \mathbf{0}$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\mathbf{f}$

齐次最小二乘问题

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\mathbf{f}$

齐次最小二乘问题

$$\arg\min_{\mathbf{f}} \|\mathbf{Af}\|^2$$

并服从如下约束

$$\|\mathbf{f}\| = 1$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\mathbf{f}$

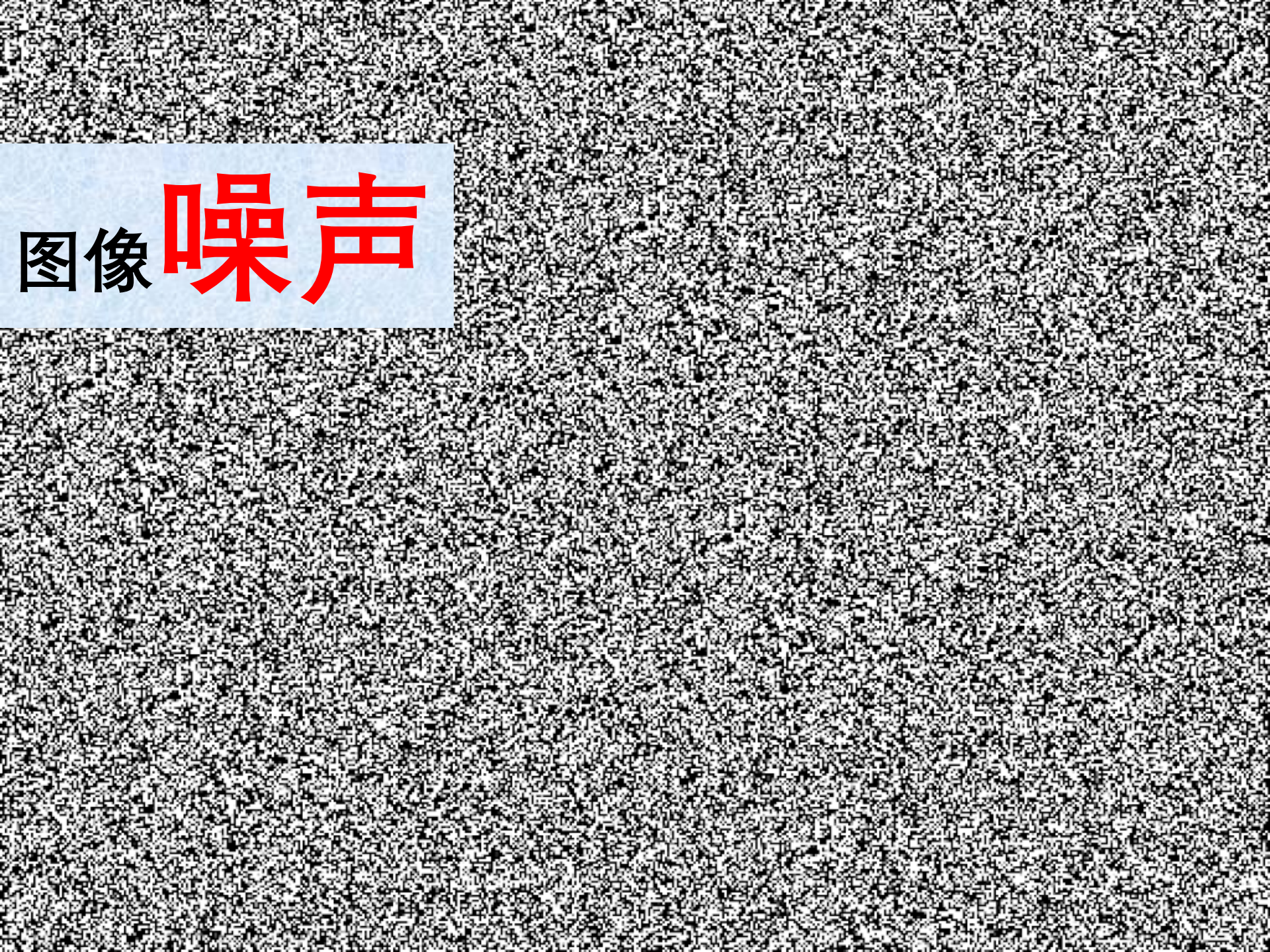
计算A的SVD，解就是V的最后一列

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\mathbf{f}$

计算A的SVD，解就是V的最后一列

强制基础矩阵的秩必须为2


图像噪声

In Defense of the Eight-Point Algorithm

Richard I. Hartley

Abstract—The fundamental matrix is a basic tool in the analysis of scenes taken with two uncalibrated cameras, and the eight-point algorithm is a frequently cited method for computing the fundamental matrix from a set of eight or more point matches. It has the advantage of simplicity of implementation. The prevailing view is, however, that it is extremely susceptible to noise and hence virtually useless for most purposes. This paper challenges that view, by showing that by preceding the algorithm with a very simple normalization (translation and scaling) of the coordinates of the matched points, results are obtained comparable with the best iterative algorithms. This improved performance is justified by theory and verified by extensive experiments on real images.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997

In Defense of the Eight-Point Algorithm

Richard I. Hartley

Abstract—The fundamental matrix is a basic tool in the analysis of scenes taken with two uncalibrated cameras, and the eight-point algorithm is a frequently cited method for computing the fundamental matrix from a set of eight or more point matches. It has the advantage of simplicity of implementation. The prevailing view is, however, that it is extremely susceptible to noise and hence virtually useless for most purposes. This paper challenges that view, by showing that by preceding the algorithm with a very simple normalization (translation and scaling) of the coordinates of the matched points, results are obtained comparable with the best iterative algorithms. This improved performance is justified by theory and verified by extensive experiments on real images.

解决数值计算问题

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\underbrace{\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix}}_{\mathbf{f}} = \mathbf{0}$$

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\qquad$ $\mathbf{f}$

A是病态的

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{F}\mathbf{p}_l = 0$$

$$\begin{pmatrix} x_{r1}x_{l1}, x_{r1}y_{l1}, x_{r1}, y_{r1}x_{l1}, y_{r1}y_{l1}, y_{r1}, x_{l1}, y_{l1}, 1 \\ \vdots \\ x_{rn}x_{ln}, x_{rn}y_{ln}, x_{rn}, y_{rn}x_{ln}, y_{rn}y_{ln}, y_{rn}, x_{ln}, y_{ln}, 1 \end{pmatrix} \begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = \mathbf{0}$$

$\mathbf{A}$ $\mathbf{f}$

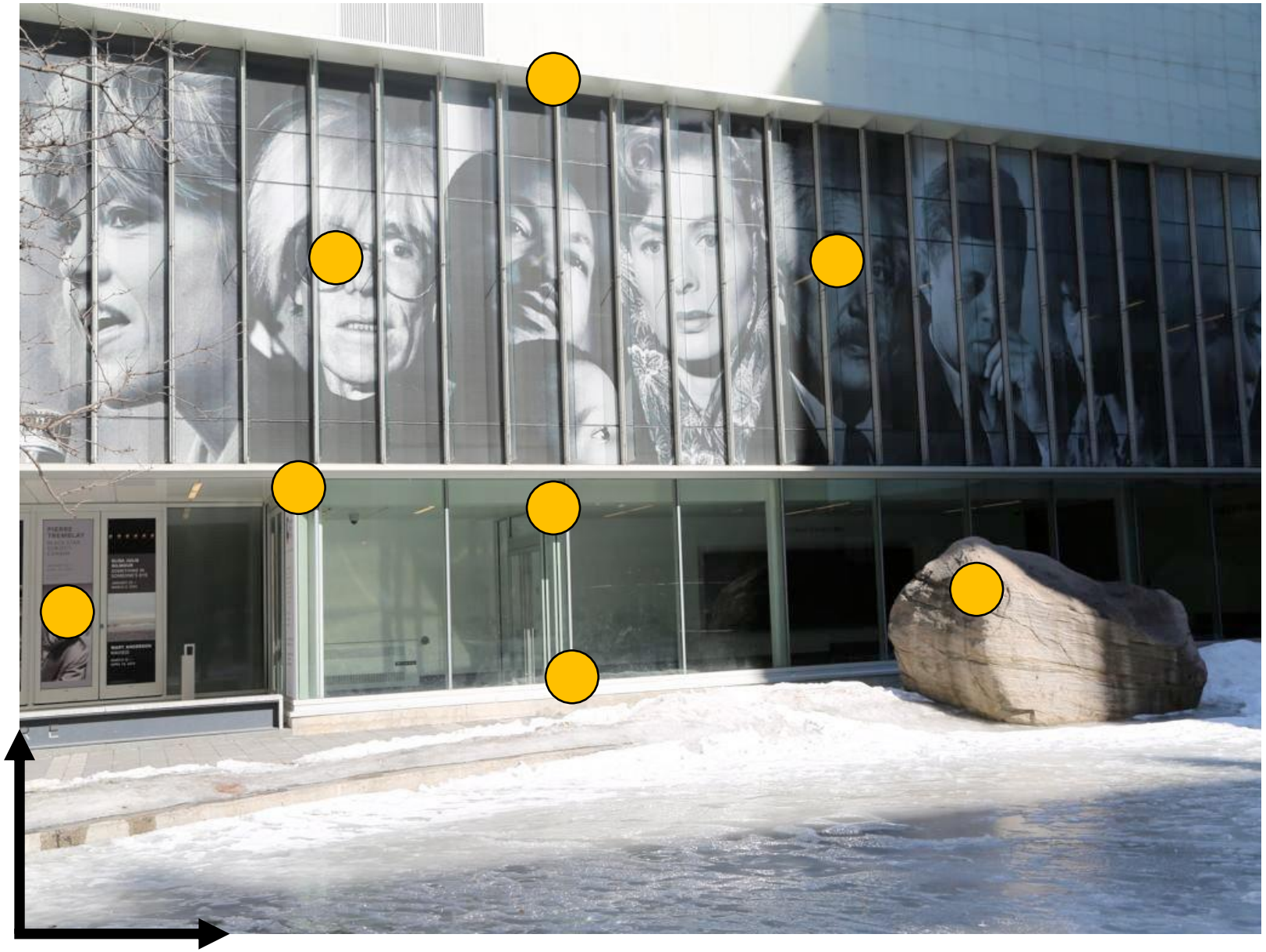
变换对应点的坐标改善A的状态

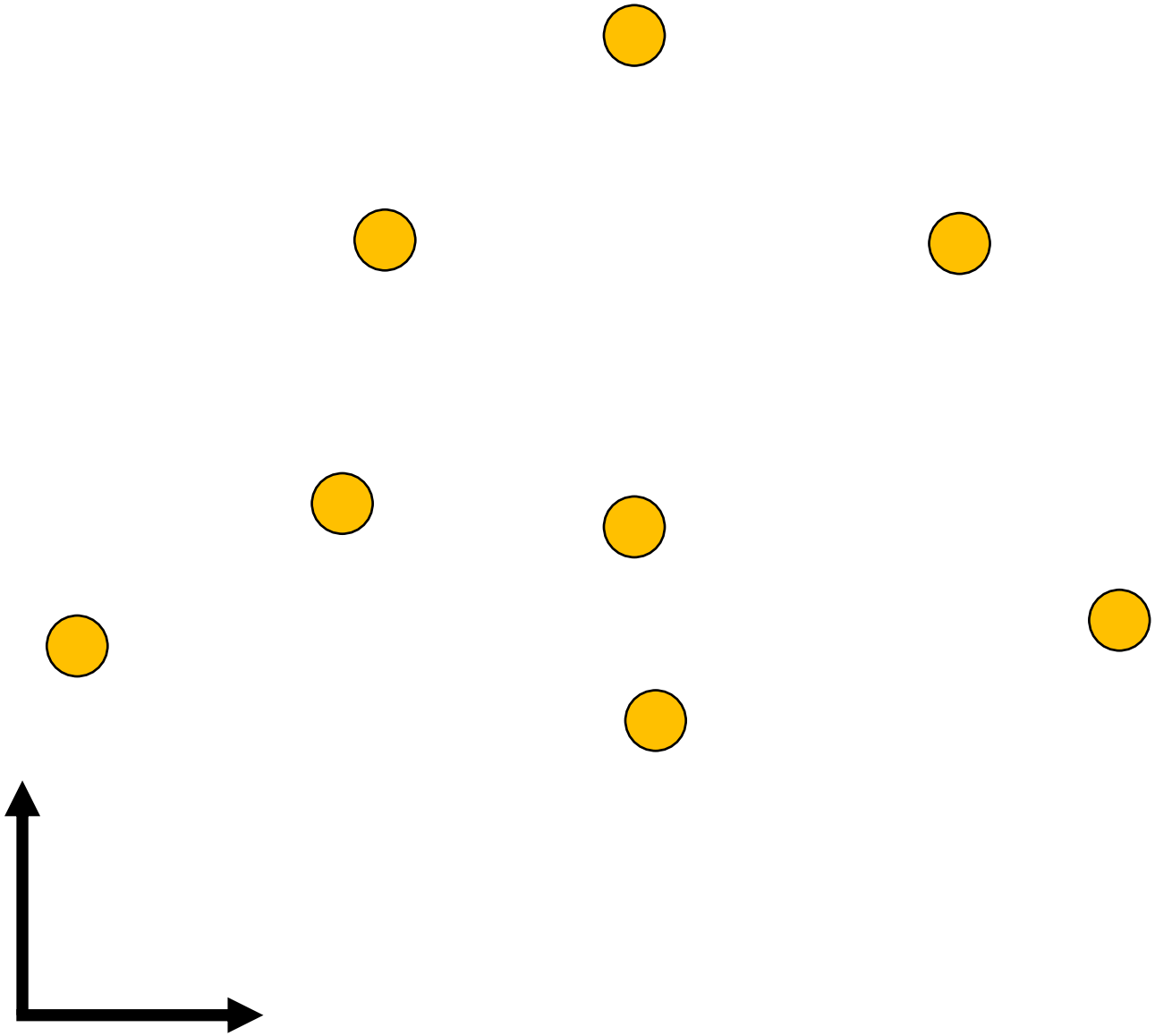
4

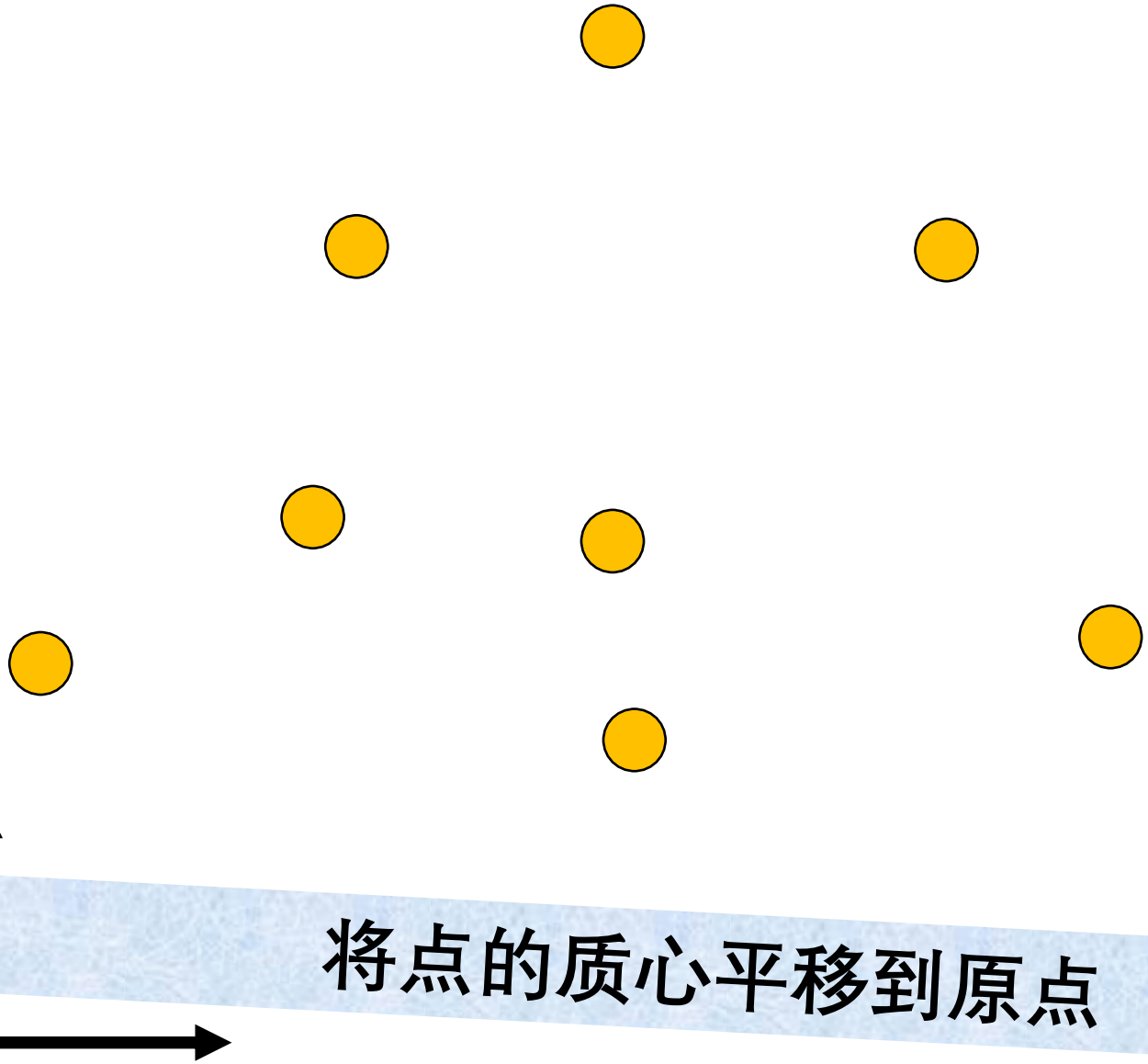
主要步骤

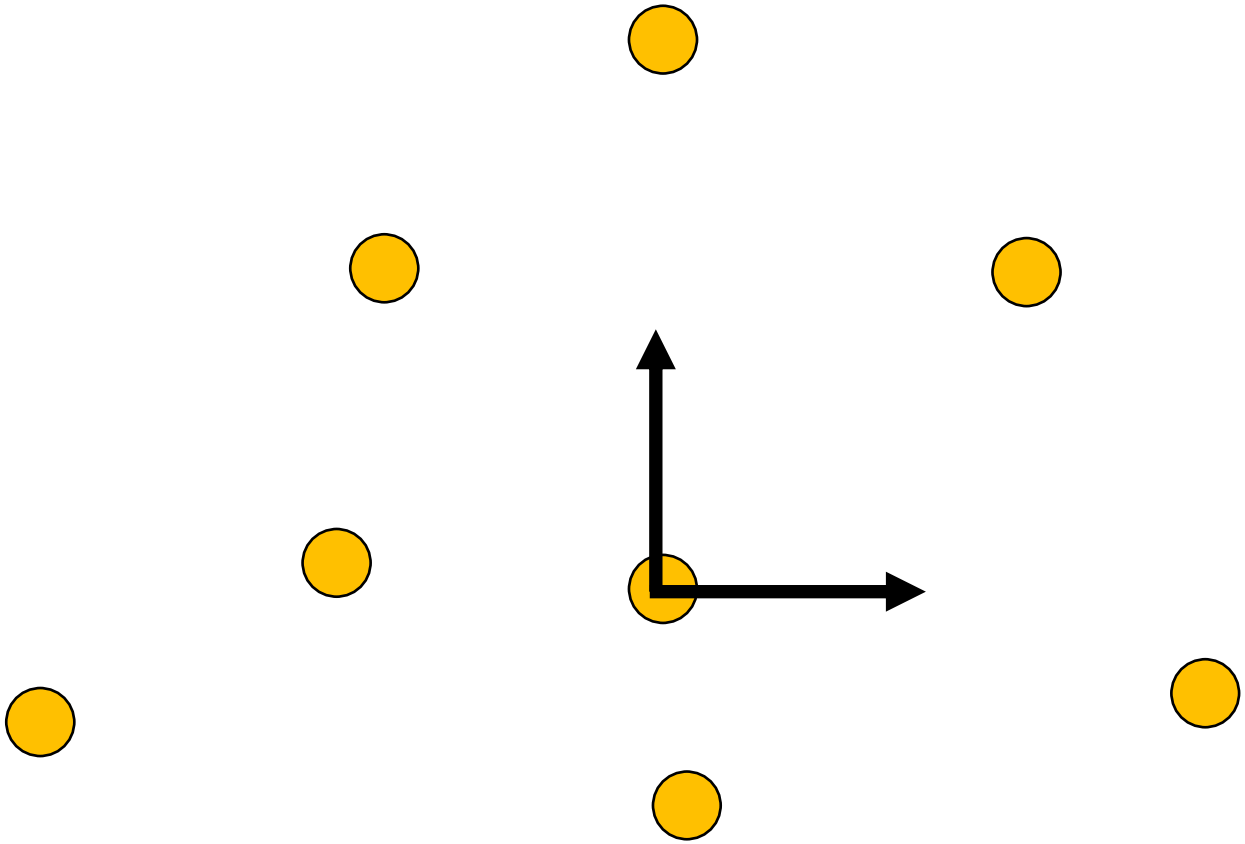
步骤1

将数据点中心化





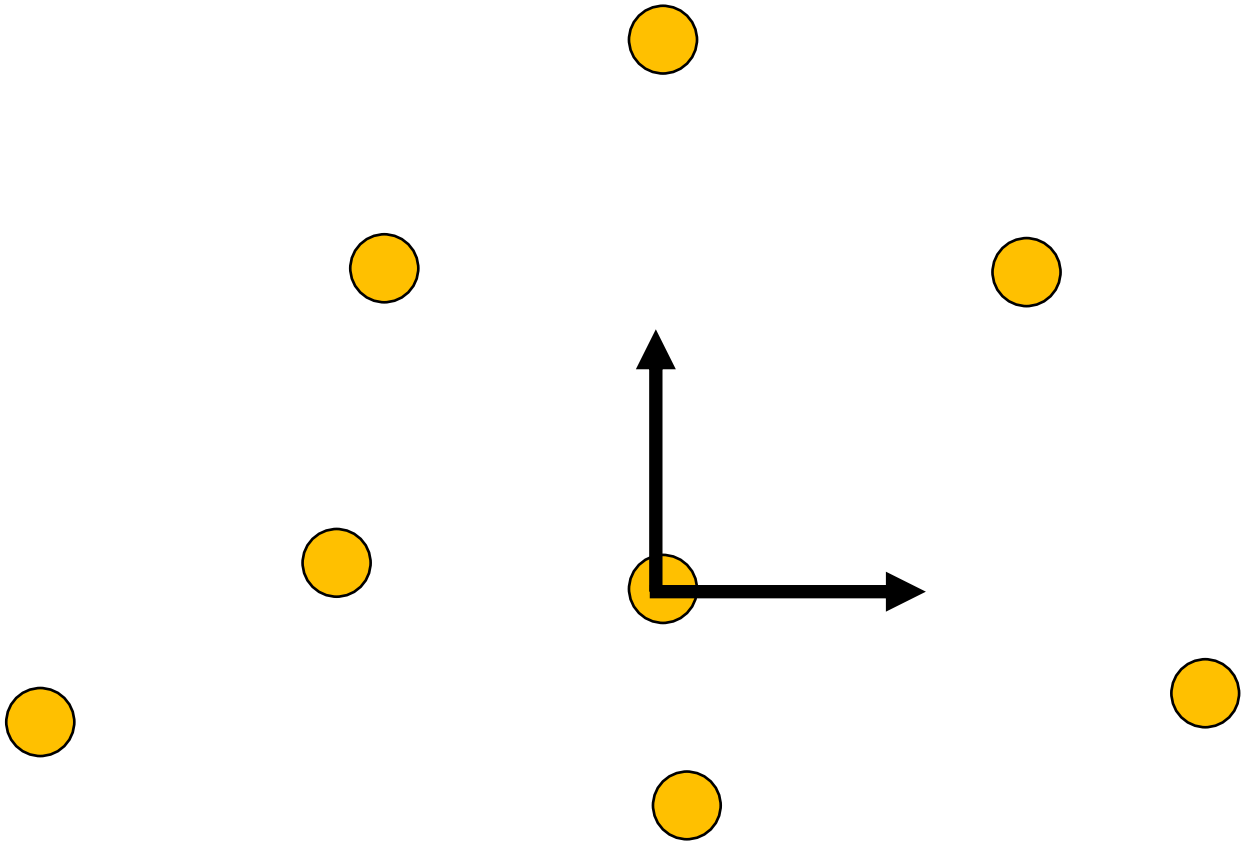

将点的质心平移到原点

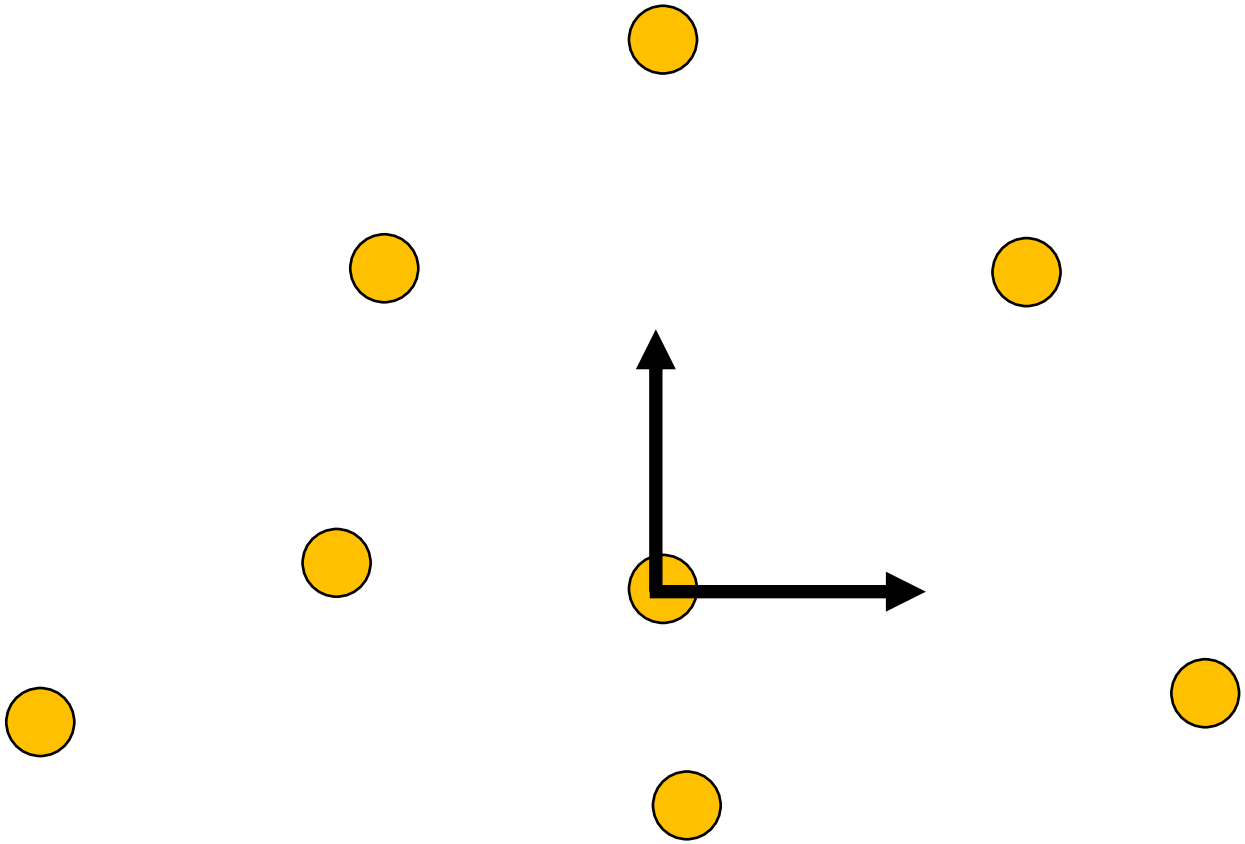


将点的质心平移到原点

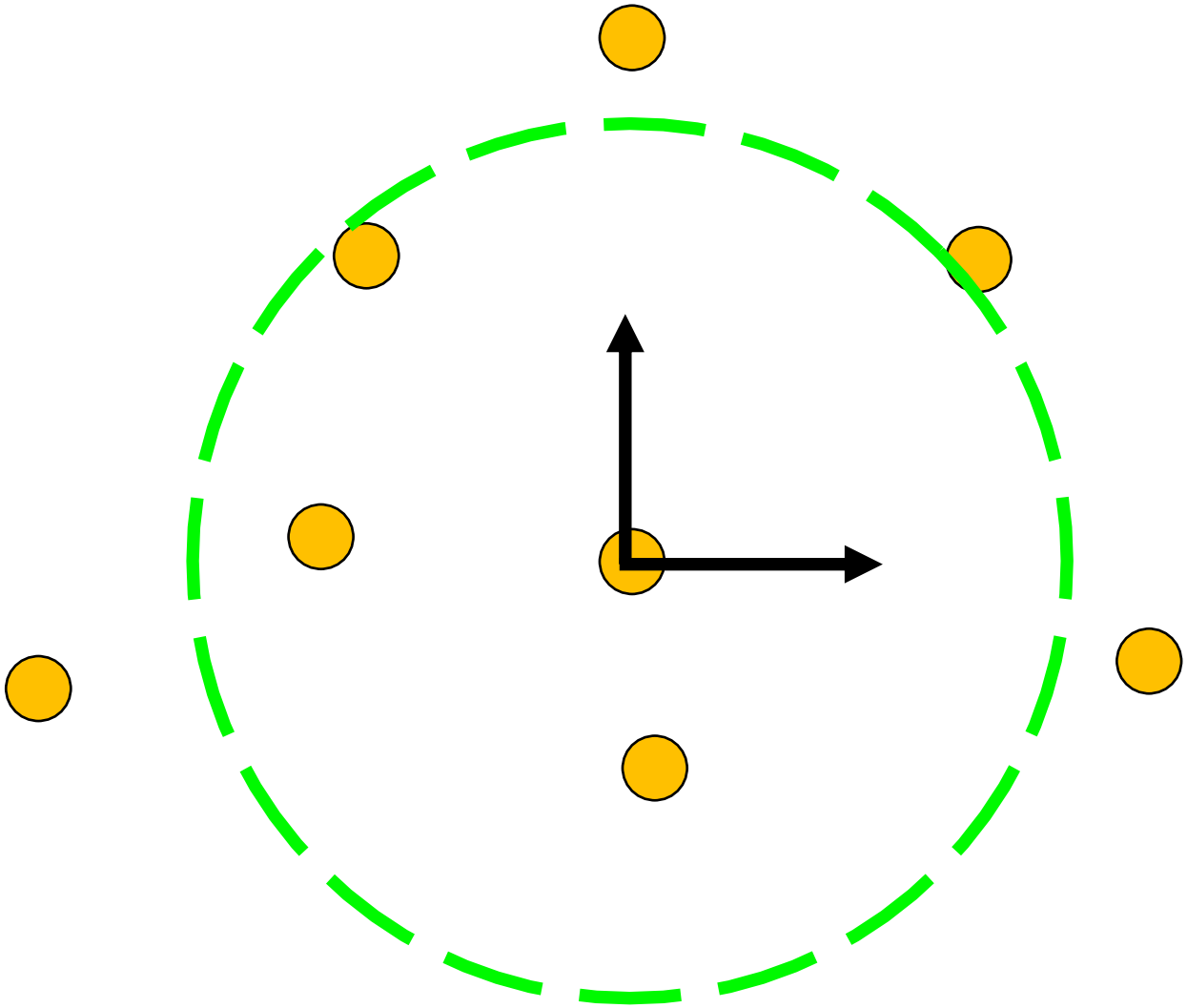
步骤2

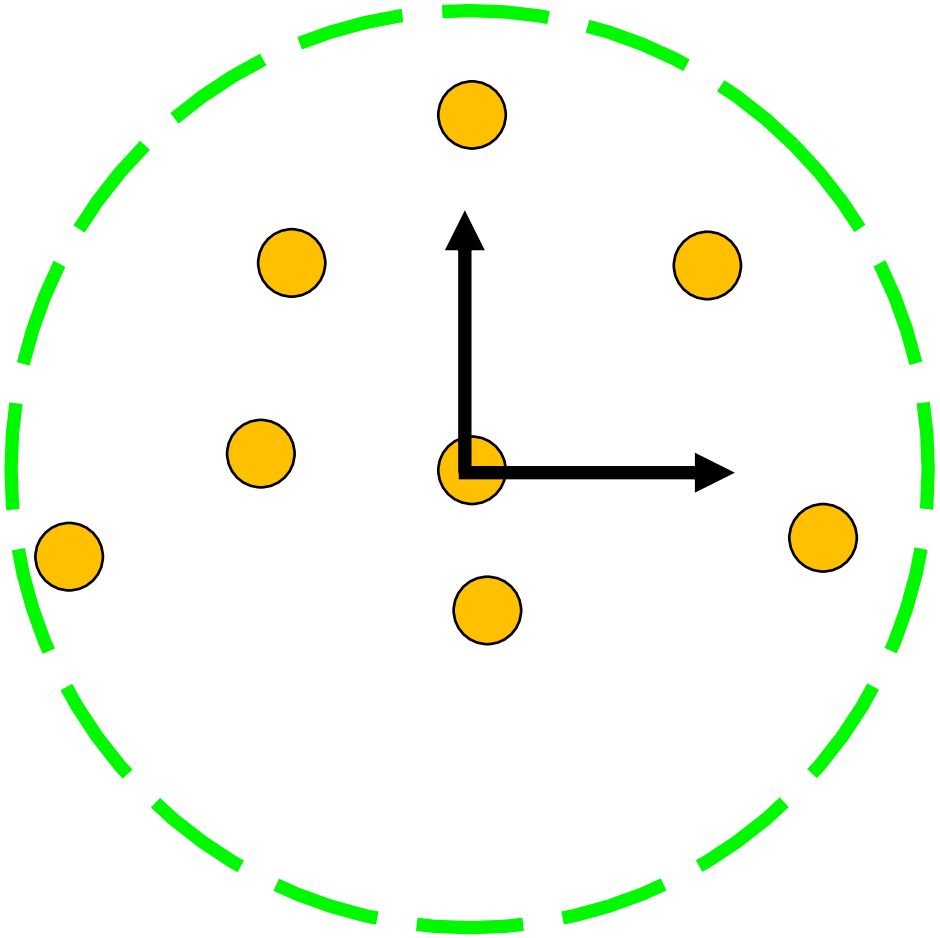
缩放数据





缩放点的坐标使得它们到原点
的平均距离等于2像素





步骤3

计算基础矩阵

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

利用归一化后的对应点对求解基础矩阵

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

强制估计的基础矩阵秩为2

步骤4

去归一化基础矩阵

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

展开

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{T}_r^{\mathrm{T}}\mathbf{F}\mathbf{T}_l\mathbf{p}_l = 0$$

$$(\mathbf{T}_r\mathbf{p}_r)^{\mathrm{T}}\mathbf{F}(\mathbf{T}_l\mathbf{p}_l) = 0$$

展开

$$\mathbf{p}_r^{\mathrm{T}}\mathbf{T}_r^{\mathrm{T}}\mathbf{F}\mathbf{T}_l\mathbf{p}_l = 0$$

$$\mathbf{F} = \mathbf{T}_r^{\mathrm{T}} \hat{\mathbf{F}} \mathbf{T}_l$$

归一化8点算法总结

归一化8点算法总结

1. 将数据点的质心平移到原点

归一化8点算法总结

1. 将数据点的质心平移到原点

2. 缩放点的坐标使得它们到原点的平均距离等于2像素

归一化8点算法总结

1. 将数据点的质心平移到原点

2. 缩放点的坐标使得它们到原点的平均距离等于2像素

3. 按照步骤1、2分别独立处理两张图像，并估计基础矩阵

归一化8点算法总结

1. 将数据点的质心平移到原点

2. 缩放点的坐标使得它们到原点的平均距离等于2像素

3. 按照步骤1、2分别独立处理两张图像，并估计基础矩阵

4. 对所估计的基础矩阵去归一化

如果没有已知的对应关系，
我们如何估计基本矩阵?

RANSAC

RANdom **SA**mple **C**onsensus

随机抽样一致

Communications of the ACM, 1981

Graphics and Image Processing — J. D. Foley, Editor

Random Sample Consensus: A Paradigm for Model Fitting with [illegible] Cartography

Martin A. Fi[illegible] Bo[illegible]
SRI International

A new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landm[illegible] in space [illegible] response to a RANSAC requirement, new results are derived on the minimum number of landmarks needed to obtain a solution, and algorithms are presented for

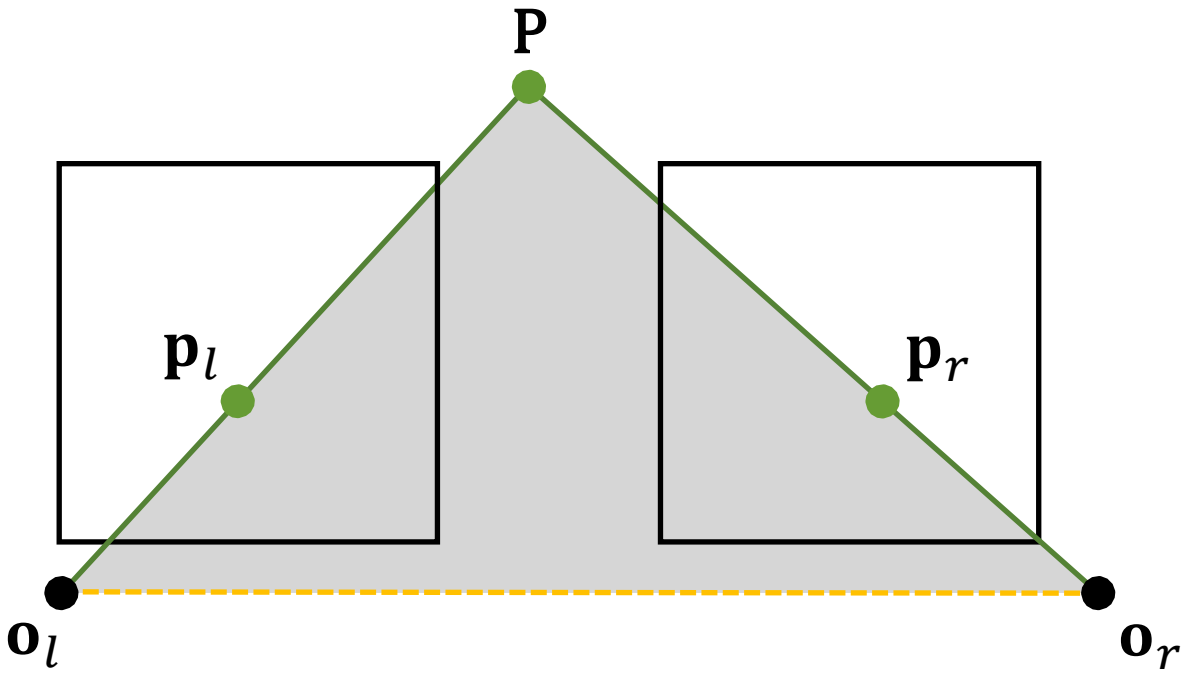
I. Introduction

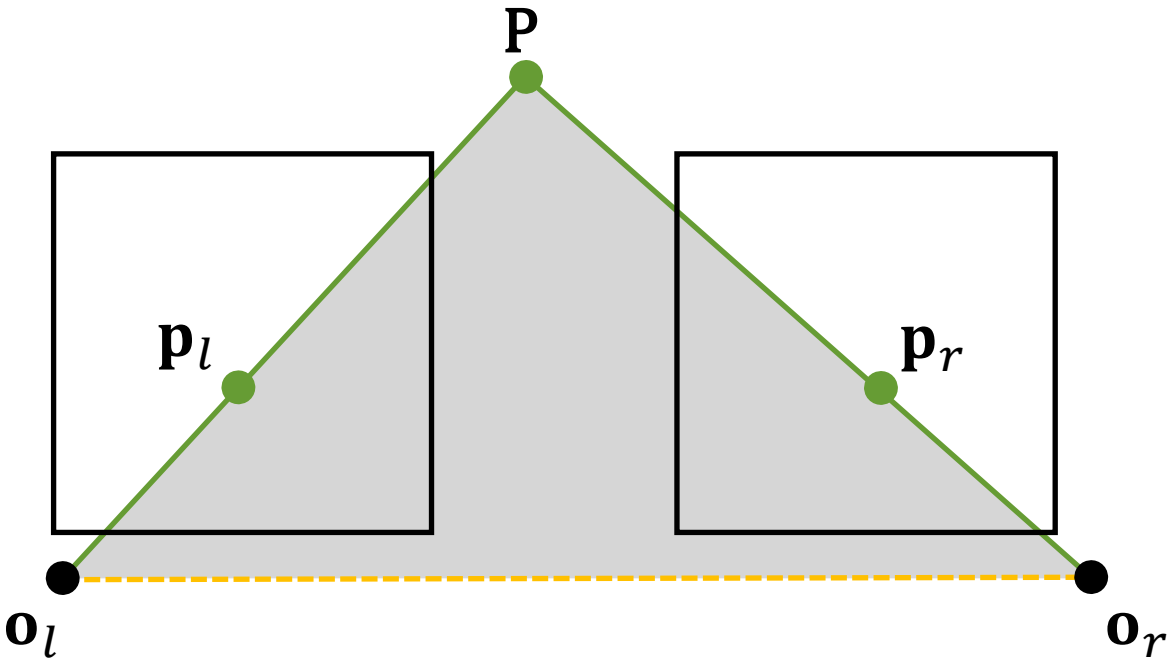
We introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The application discussed, the location [illegible] is [illegible] the RANSAC [illegible] conditions [illegible] presented and [illegible] this problem [illegible] appli[illegible]

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, [illegible]: First, there is the problem of finding the best match between the data and one of the available models (the classification [illegible] there is the problem of computing the best values for [illegible] free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent—a solution to the parameter estimation problem is often required to solve the classification problem.

Classical techniques for parameter estimation, such as least squares, optimize (according to a specified objective function) the fit of a functional description (model) to *all* of the presented data. These techniques have no internal mechanisms for detecting and rejecting gross errors. They [illegible] the maximum expected deviation of any datum from the assumed model is a direct function of the size of the data set, and thus regardless of the size of the data set, there

图像矫正


P
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$



$$\mathbf{R} = \mathbf{I}_{3\times3}$$

$$\mathbf{T} = (B, 0, 0)^{\mathrm{T}}$$

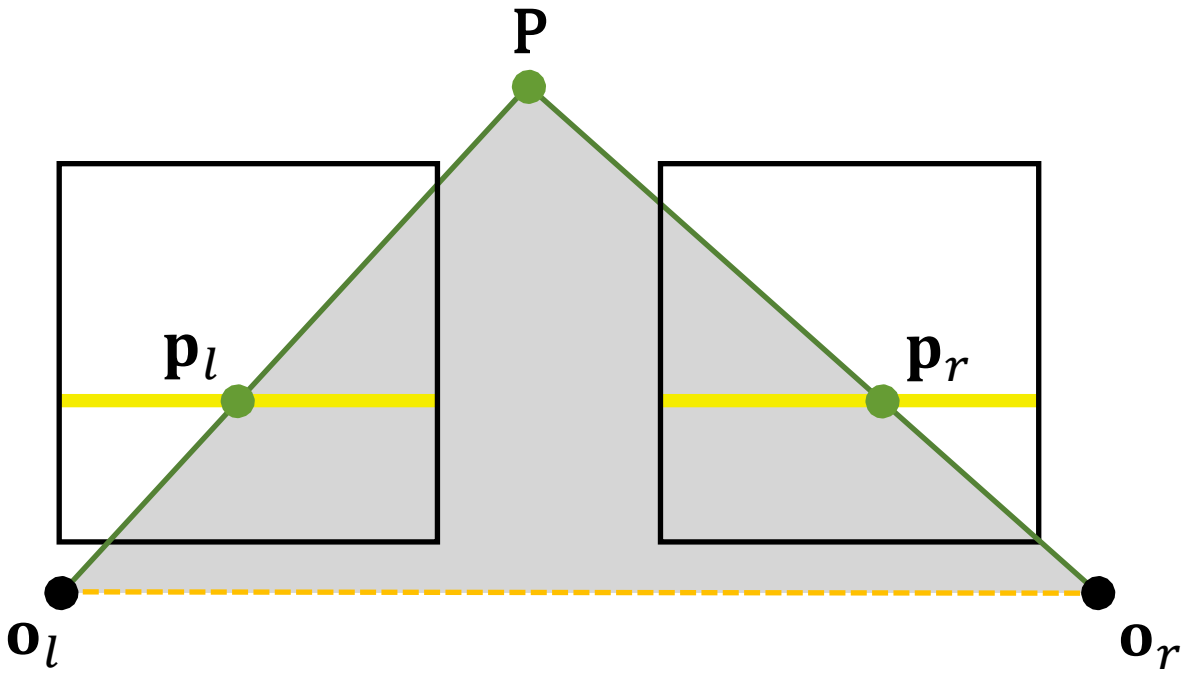
$$\mathbf{p}_r^{\mathrm{T}}\mathbf{E}\mathbf{p}_l = 0$$

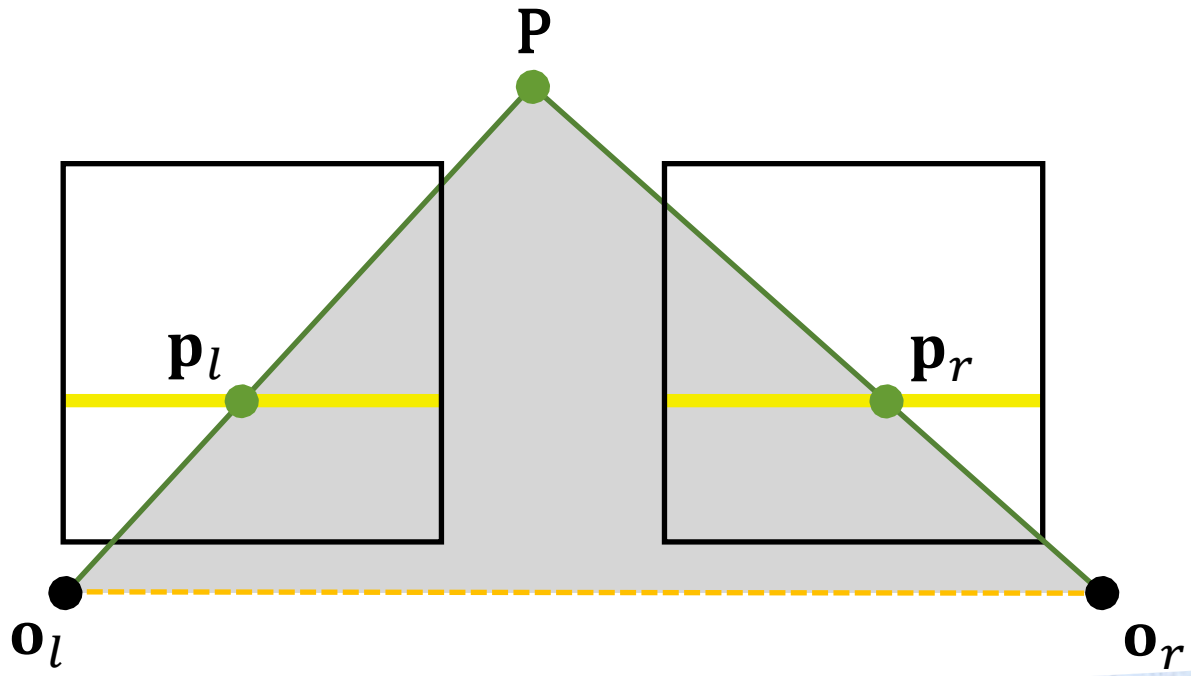
$$(x_r, y_r, 1)[\mathbf{T}_\times](x_l, y_l, 1)^{\mathrm{T}} = 0$$

展开并化简

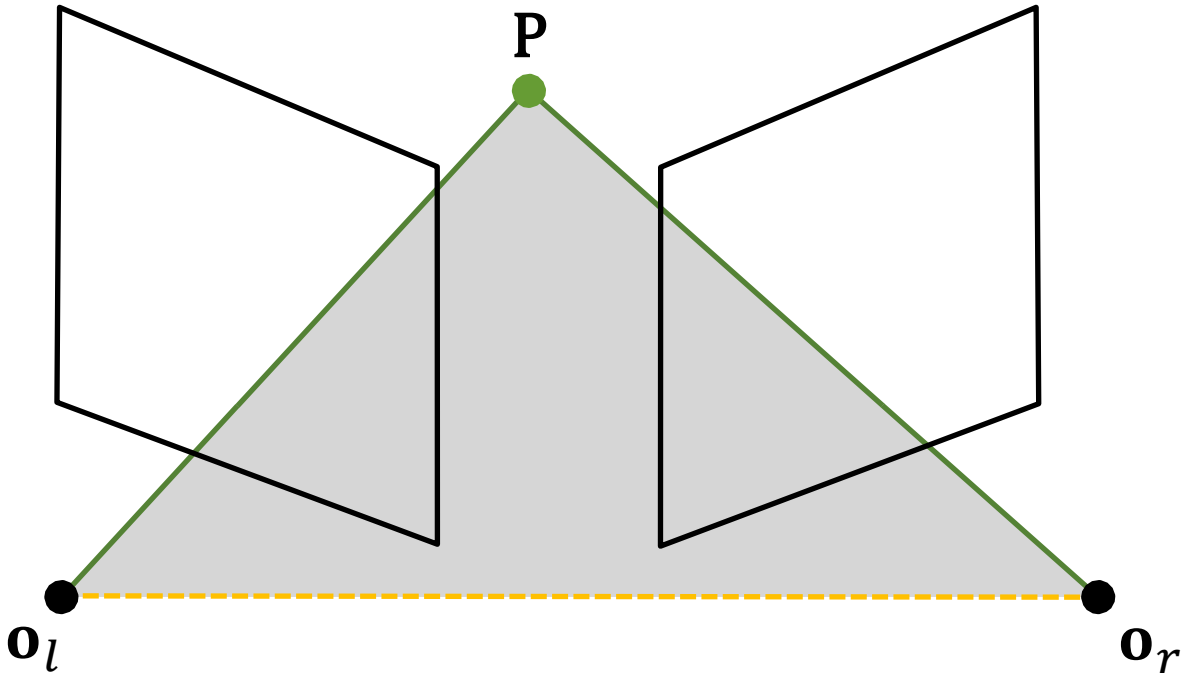
$$y_r = y_l$$

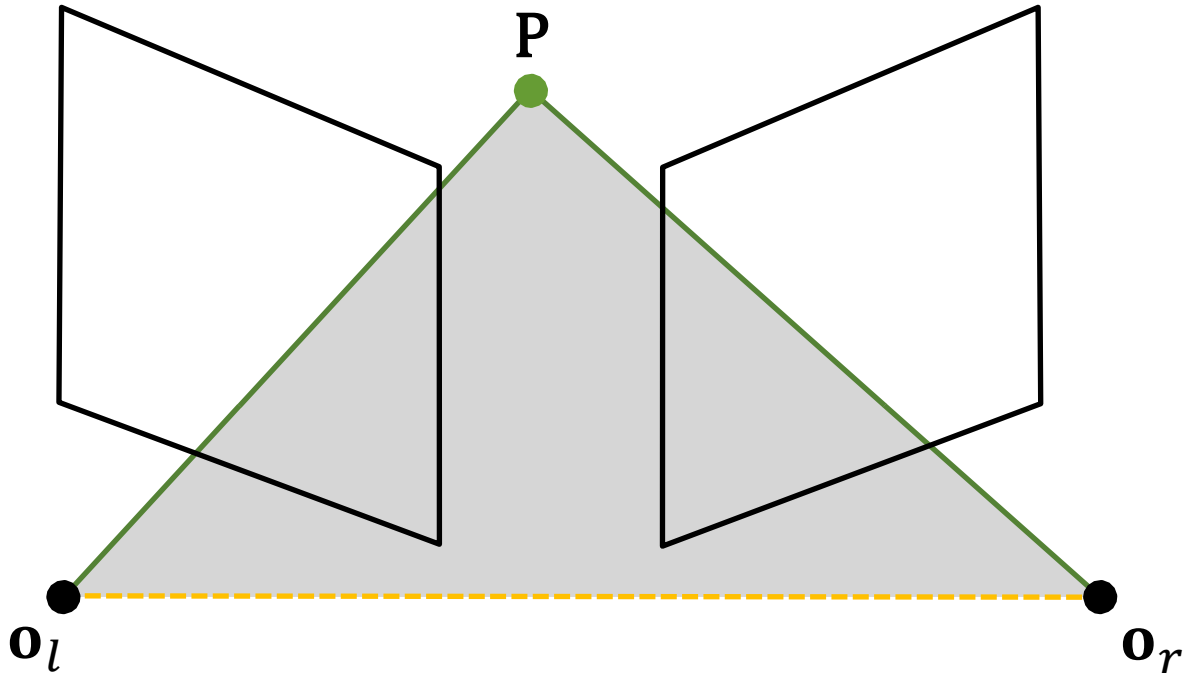
对应点位于相同一条水平直线上


P
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$

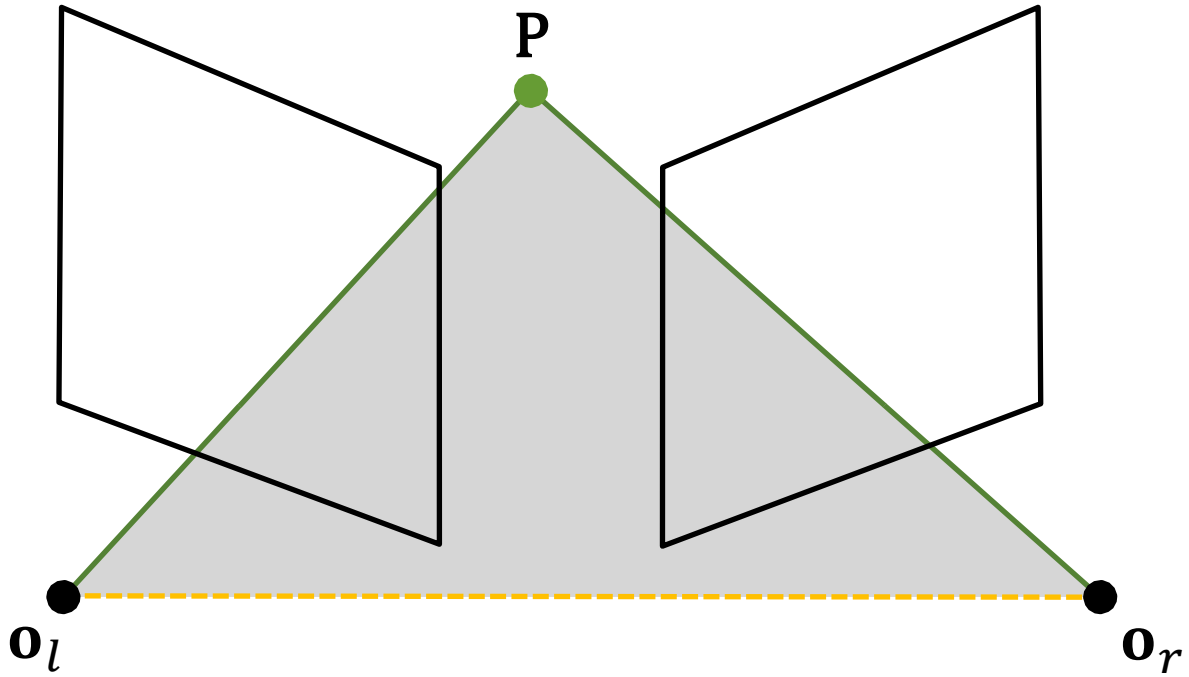


平行的像平面可以简化立体匹配问题


P
$\mathbf{o}_l$
$\mathbf{o}_r$

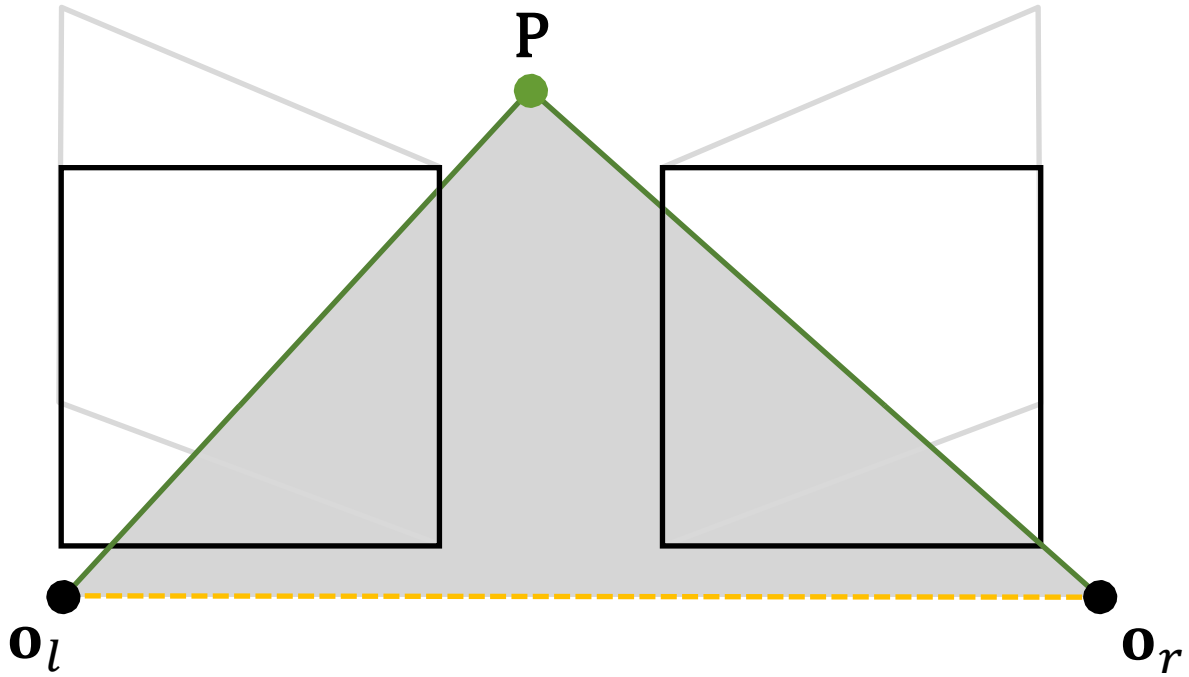


将图像重投影到与基线平行的公共平面上



将图像重投影到与基线平行的公共平面上

对每幅图像应用单应变换



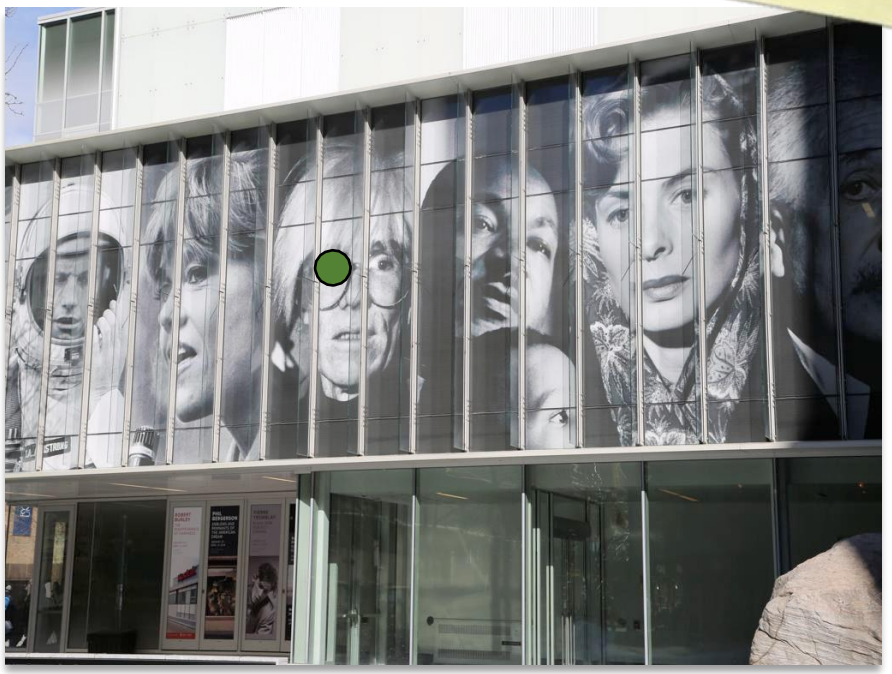
将图像重投影到与基线平行的公共平面上

对每幅图像应用单应变换

原始
图像对

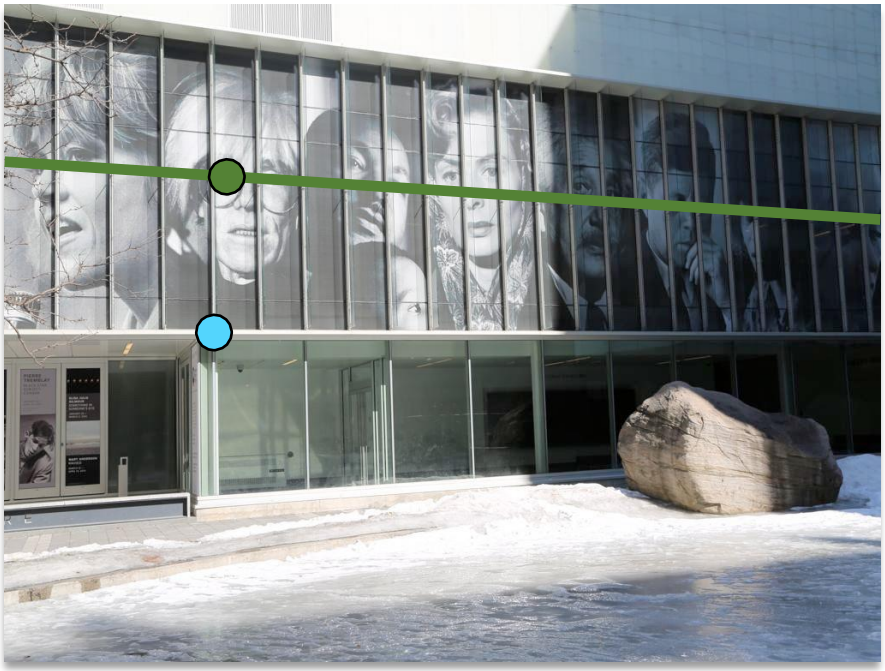
原始
图像对

原始
图像对

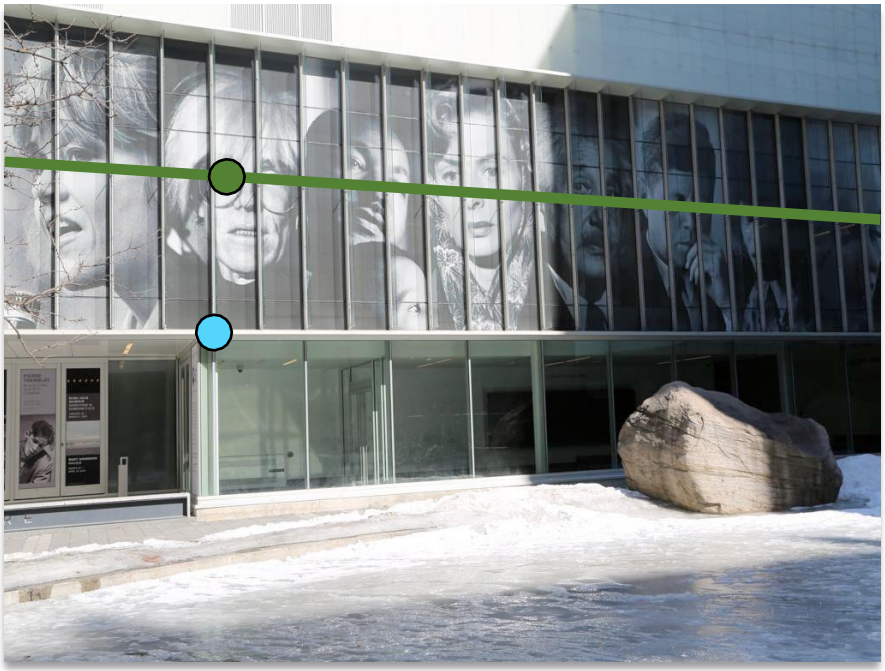


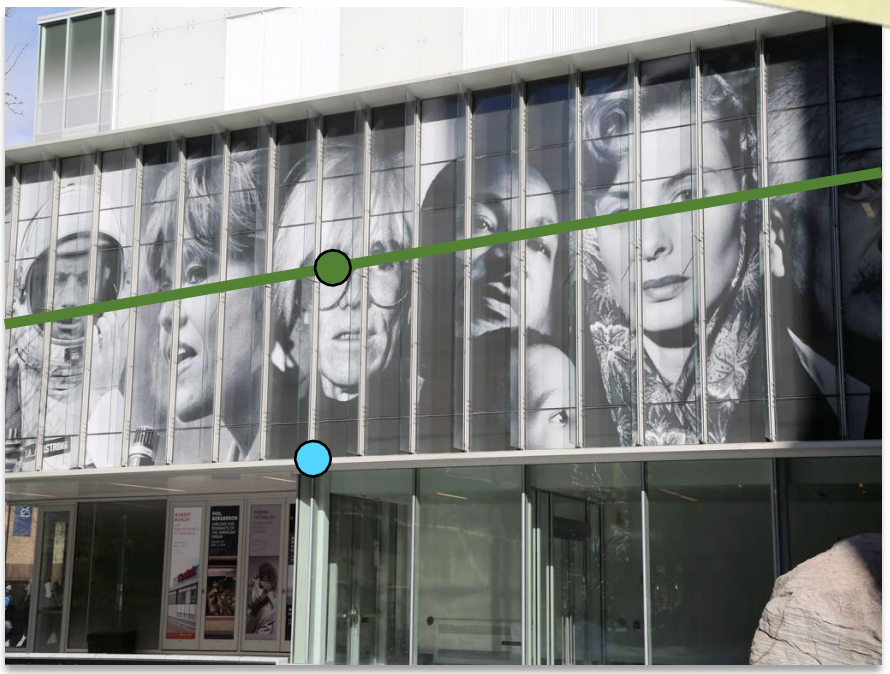
原始
图像对

原始
图像对

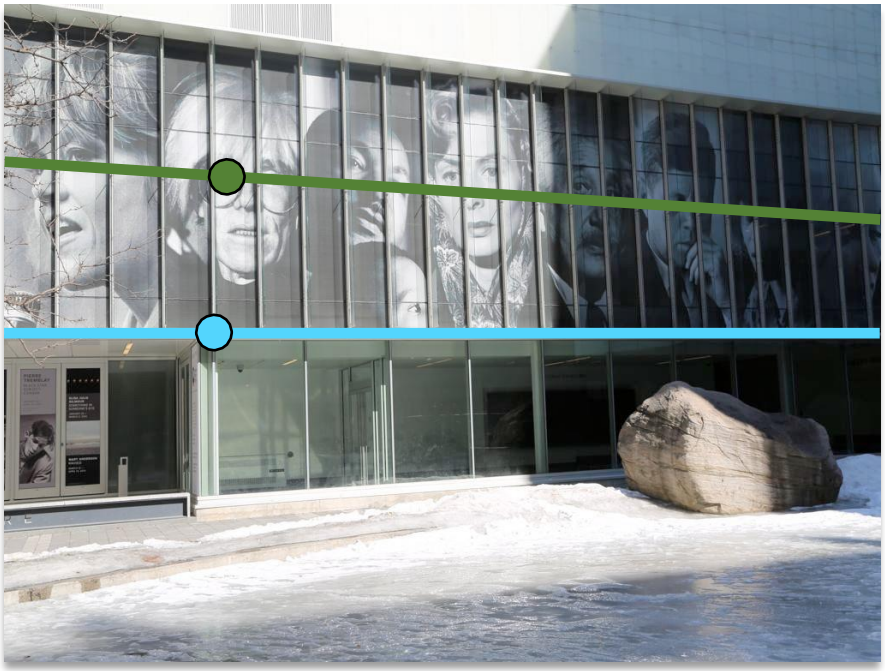


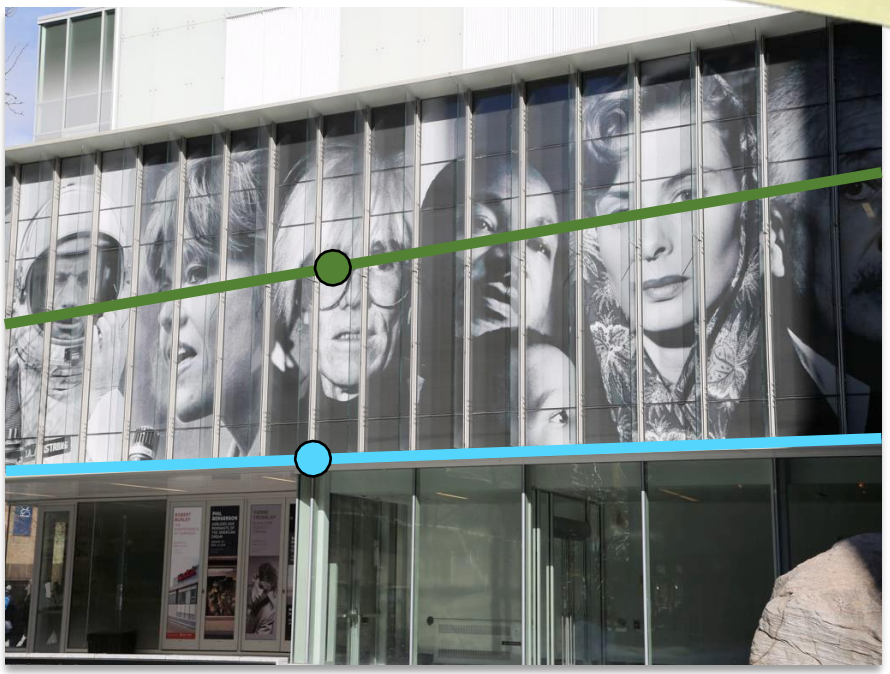
原始
图像对



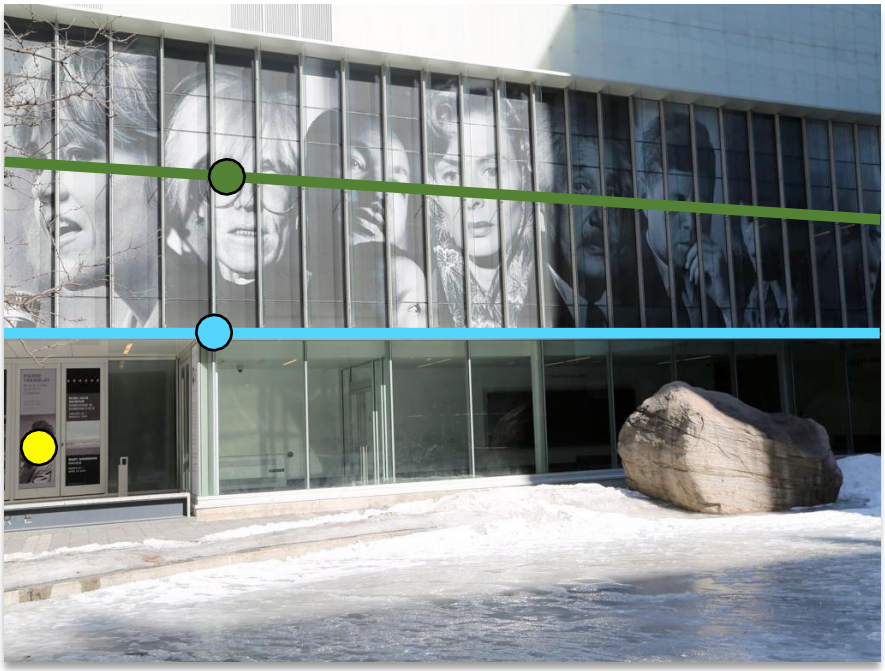


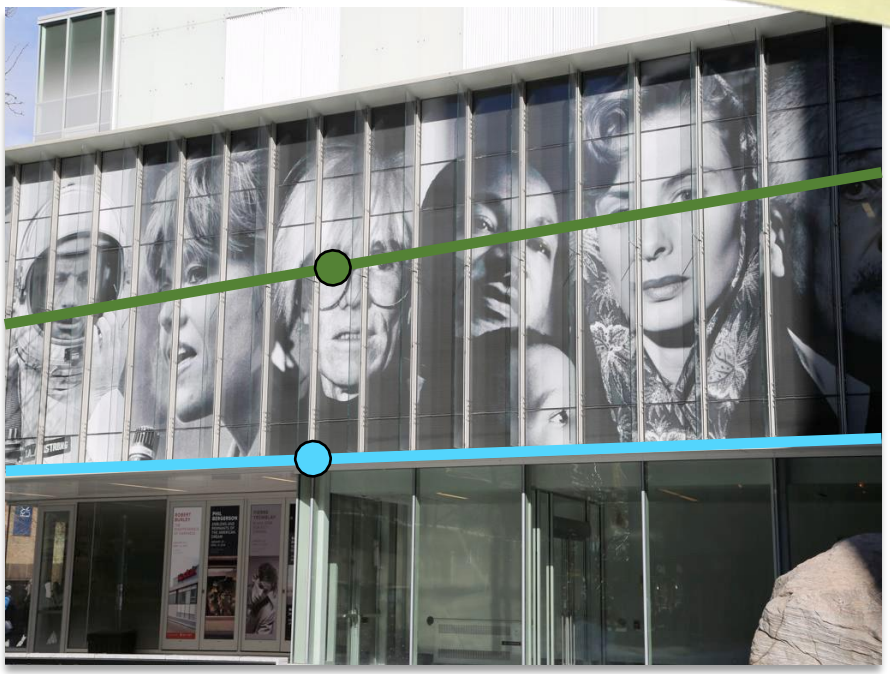
原始
图像对



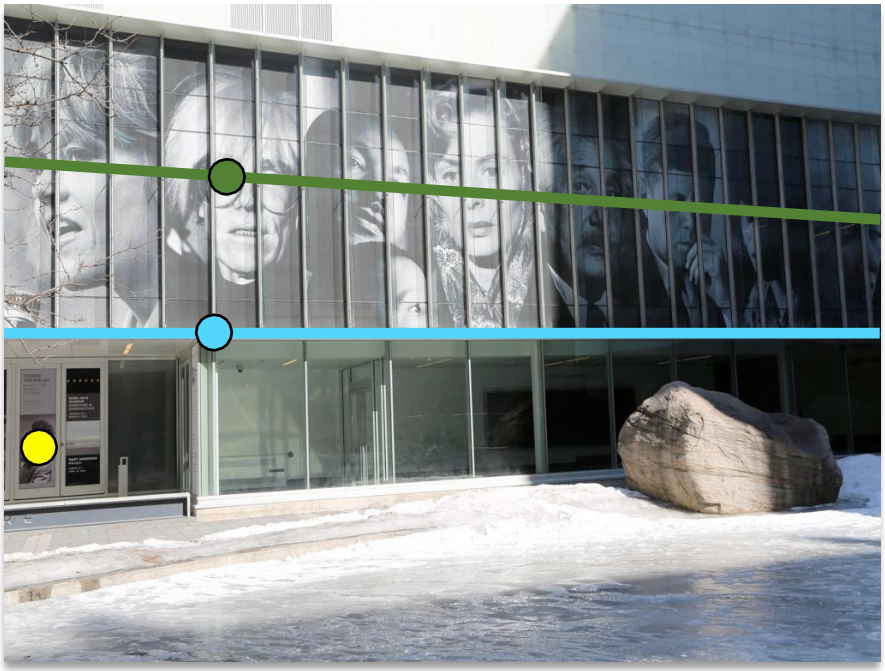


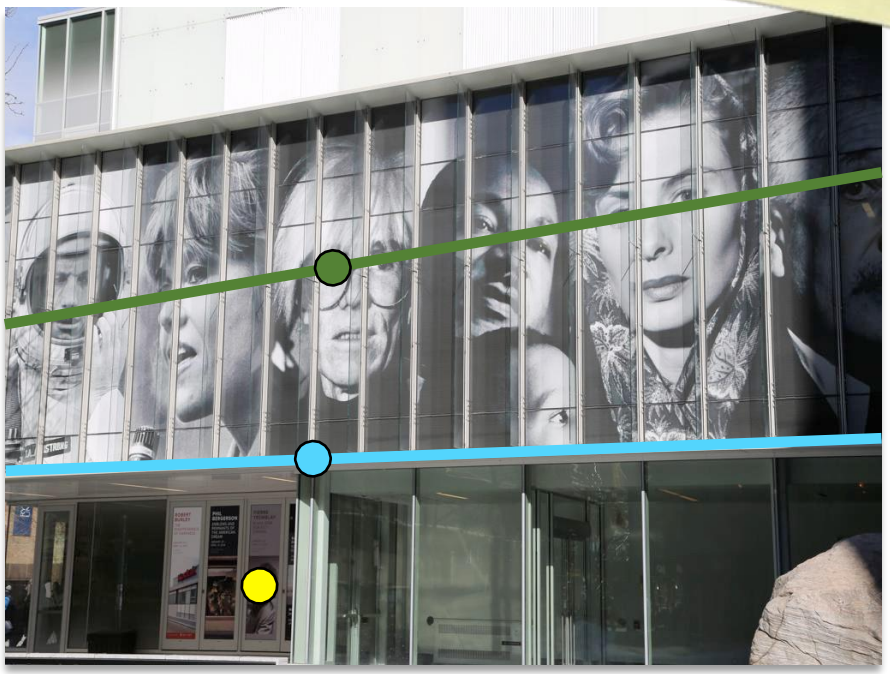
原始
图像对



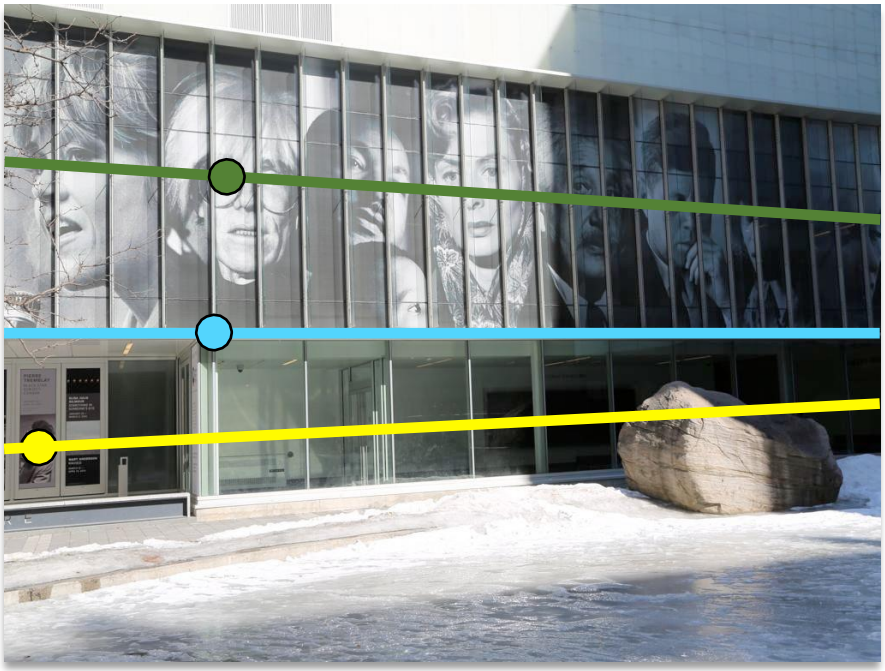


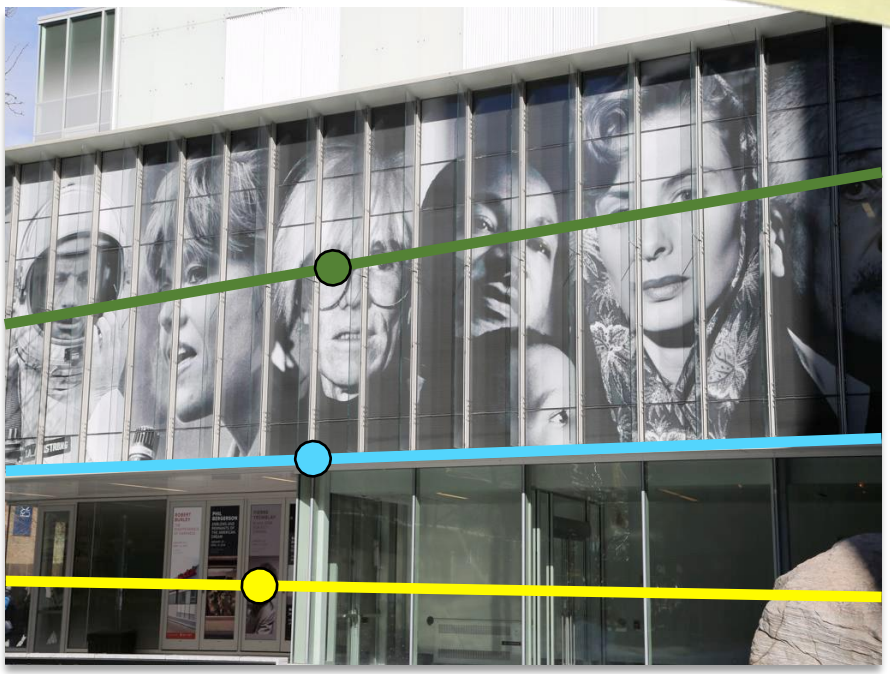
原始
图像对





原始
图像对



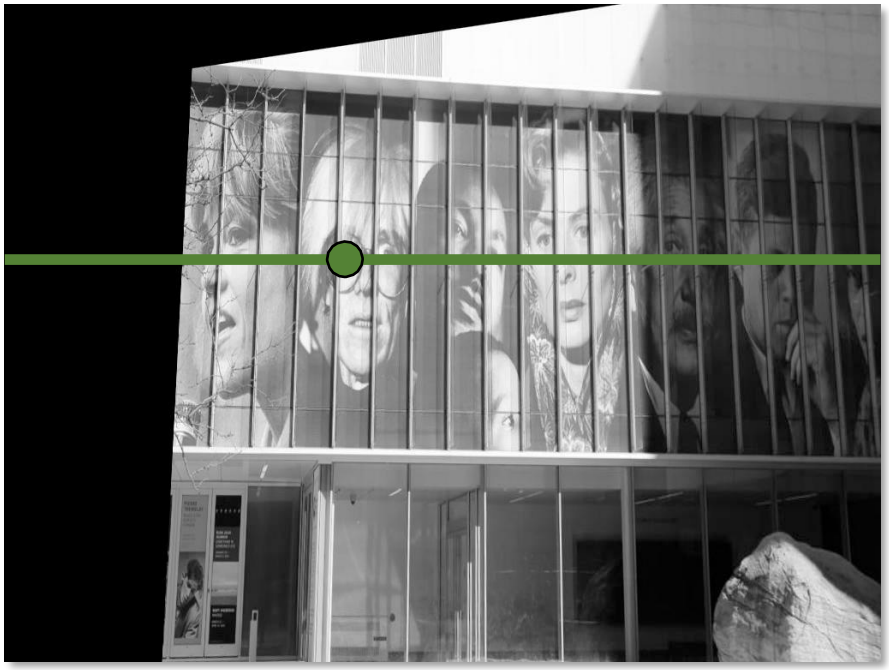


矫正后
图像对

矫正后
图像对

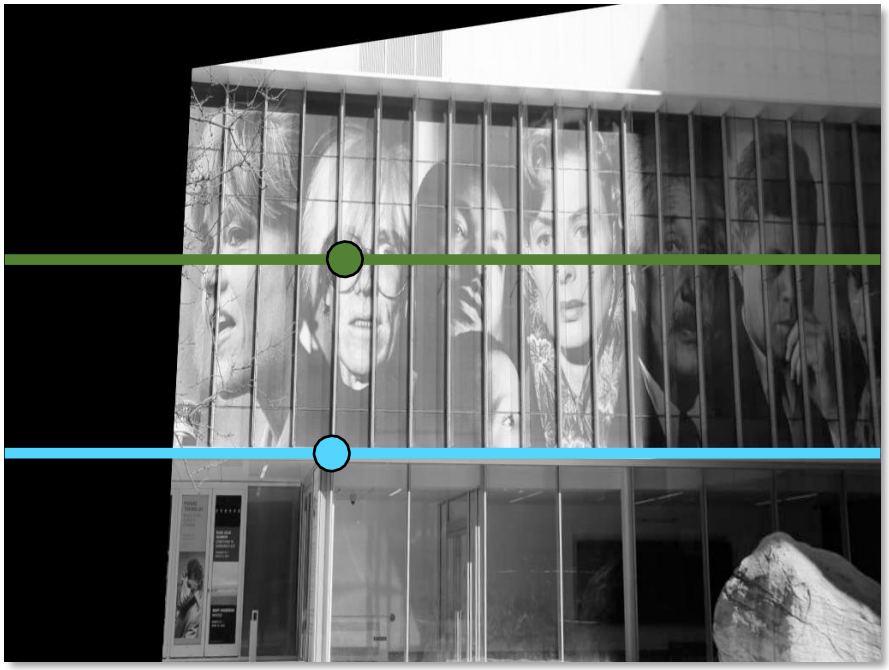
对应点在同一条扫描线上

矫正后
图像对



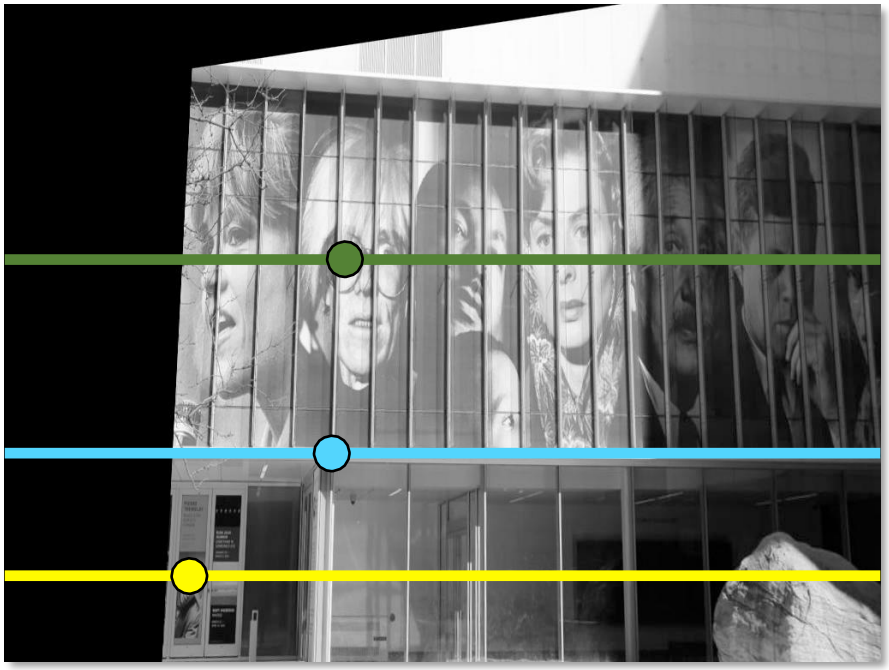
对应点在同一条扫描线上

矫正后
图像对



对应点在同一条扫描线上

矫正后
图像对



对应点在同一条扫描线上

Python时间

图像矫正

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

关键点

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

特征描述子

```
# Load stereo image pair and convert to grayscale
I1 = cv2.imread('left.png', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('right.png', cv2.IMREAD_GRAYSCALE)

# Find the keypoints and descriptors with SIFT
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(I1, None)
kp2, des2 = sift.detectAndCompute(I2, None)

# Visualize keypoints
I1_sift = cv2.drawKeypoints(I1, kp1, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
I2_sift = cv2.drawKeypoints(I2, kp2, None,
   flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

I1_I2_sift = np.concatenate((I1_sift, I2_sift), axis=1)
cv2.imshow('Image SIFT keypoints', I1_I2_sift)
```

寻找
对应点

```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []

for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```

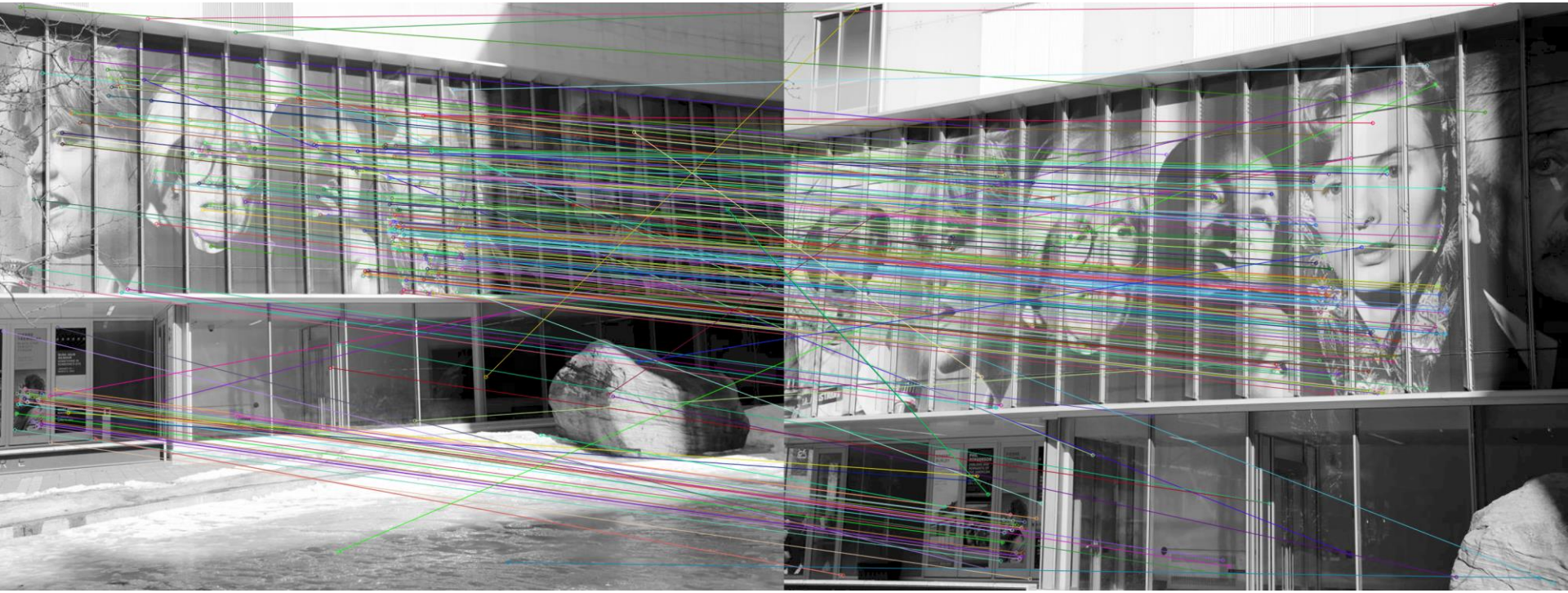
```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []

for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```

```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []

for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```

```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []

for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```

```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []
for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```

```
# Match keypoints in both images
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
flann = cv2.FlannBasedMatcher(index_params, {})
matches = flann.knnMatch(des1, des2, k=2)

# Keep good matches: calculate distinctive image features
good, pts1, pts2 = [], [], []

for i, (m, n) in enumerate(matches):
    if m.distance < 0.7 * n.distance:
        good.append([m])
        pts1.append(kp1[m.queryIdx].pt)
        pts2.append(kp2[m.trainIdx].pt)

keypoint_matches = cv2.drawMatchesKnn(I1, kp1, I2, kp2, good, None,
    flags=cv2.DRAW_MATCHES_FLAGS_NOT_DRAW_SINGLE_POINTS)
cv2.imshow('Keypoint matches', keypoint_matches)
```



```
# Calculate the fundamental matrix for the cameras
pts1 = np.float32(pts1)
pts2 = np.float32(pts2)

fundamental_matrix, inliers = cv2.findFundamentalMat(
    pts1, pts2, cv2.FM_RANSAC,
    ransacReprojThreshold=0.9, confidence=0.99
)

# Select only inlier points
pts1 = pts1[inliers.ravel() == 1]
pts2 = pts2[inliers.ravel() == 1]
```

计算
基础矩阵

```
# Calculate the fundamental matrix for the cameras
pts1 = np.float32(pts1)
pts2 = np.float32(pts2)

fundamental_matrix, inliers = cv2.findFundamentalMat(
    pts1, pts2, cv2.FM_RANSAC,
    ransacReprojThreshold=0.9, confidence=0.99
)

# Select only inlier points
pts1 = pts1[inliers.ravel() == 1]
pts2 = pts2[inliers.ravel() == 1]
```

```
# Calculate the fundamental matrix for the cameras
pts1 = np.float32(pts1)
pts2 = np.float32(pts2)

fundamental_matrix, inliers = cv2.findFundamentalMat(
    pts1, pts2, cv2.FM_RANSAC,
    ransacReprojThreshold=0.9, confidence=0.99
)

# Select only inlier points
pts1 = pts1[inliers.ravel() == 1]
pts2 = pts2[inliers.ravel() == 1]
```

```
# Calculate the fundamental matrix for the cameras
pts1 = np.float32(pts1)
pts2 = np.float32(pts2)

fundamental_matrix, inliers = cv2.findFundamentalMat(
    pts1, pts2, cv2.FM_RANSAC,
    ransacReprojThreshold=0.9, confidence=0.99
)

# Select only inlier points
pts1 = pts1[inliers.ravel() == 1]
pts2 = pts2[inliers.ravel() == 1]
```

```
# Calculate the fundamental matrix for the cameras
pts1 = np.float32(pts1)
pts2 = np.float32(pts2)

fundamental_matrix, inliers = cv2.findFundamentalMat(
    pts1, pts2, cv2.FM_RANSAC,
    ransacReprojThreshold=0.9, confidence=0.99
)

# Select only inlier points
pts1 = pts1[inliers.ravel() == 1]
pts2 = pts2[inliers.ravel() == 1]
```

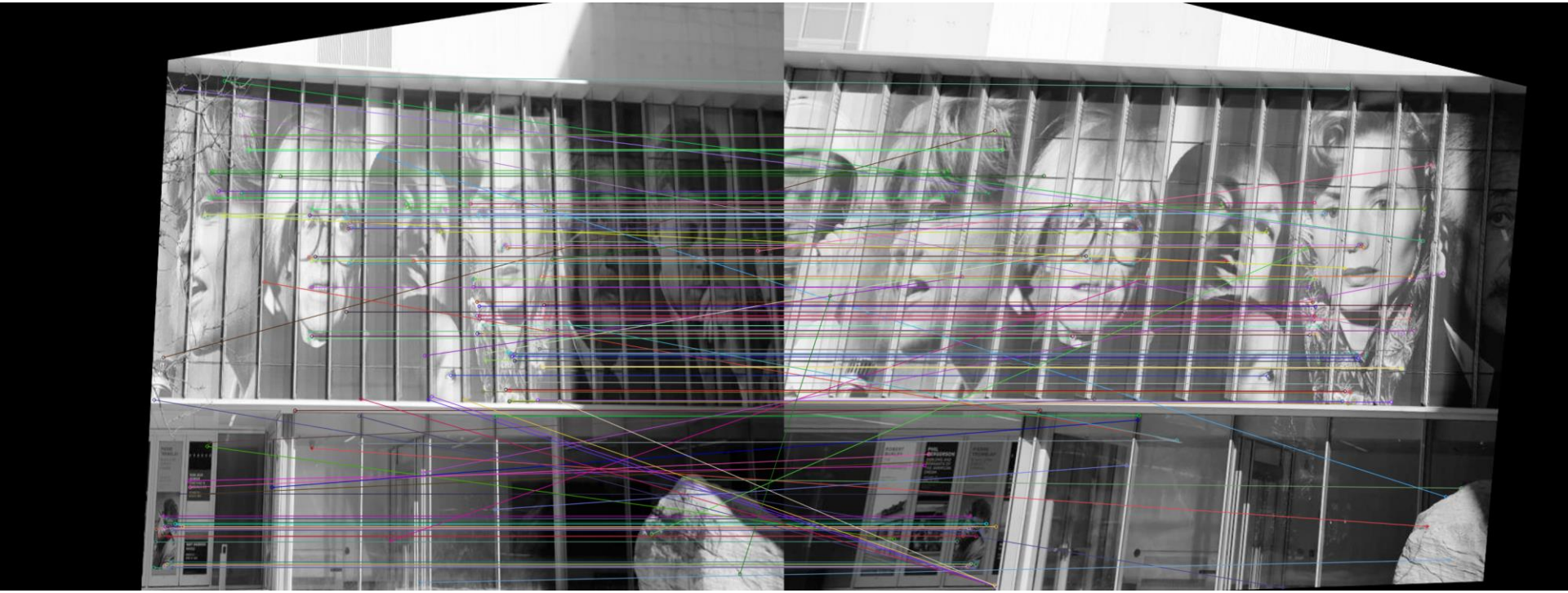
```
# Stereo rectification (uncalibrated variant)
h1, w1 = img1.shape
h2, w2 = img2.shape
_, H1, H2 = cv2.stereoRectifyUncalibrated(
      pts1, pts2, fundamental_matrix, imgSize=(w1, h1)
)

# Rectify the images
I1_rect = cv2.warpPerspective(I1, H1, (w1, h1))
I2_rect = cv2.warpPerspective(I2, H2, (w2, h2))

# Visualize rectified images
I1_I2_rect = np.concatenate((I1_rect, I2_rect), axis=1)
cv2.imshow('Rectified images', I1_I2_rect), cv2.waitKey(0)
```

矫正图像

```
# Stereo rectification (uncalibrated variant)
h1, w1 = img1.shape
h2, w2 = img2.shape
_, H1, H2 = cv2.stereoRectifyUncalibrated(
      pts1, pts2, fundamental_matrix, imgSize=(w1, h1)
)

# Rectify the images
I1_rect = cv2.warpPerspective(I1, H1, (w1, h1))
I2_rect = cv2.warpPerspective(I2, H2, (w2, h2))

# Visualize rectified images
I1_I2_rect = np.concatenate((I1_rect, I2_rect), axis=1)
cv2.imshow('Rectified images', I1_I2_rect), cv2.waitKey(0)
```

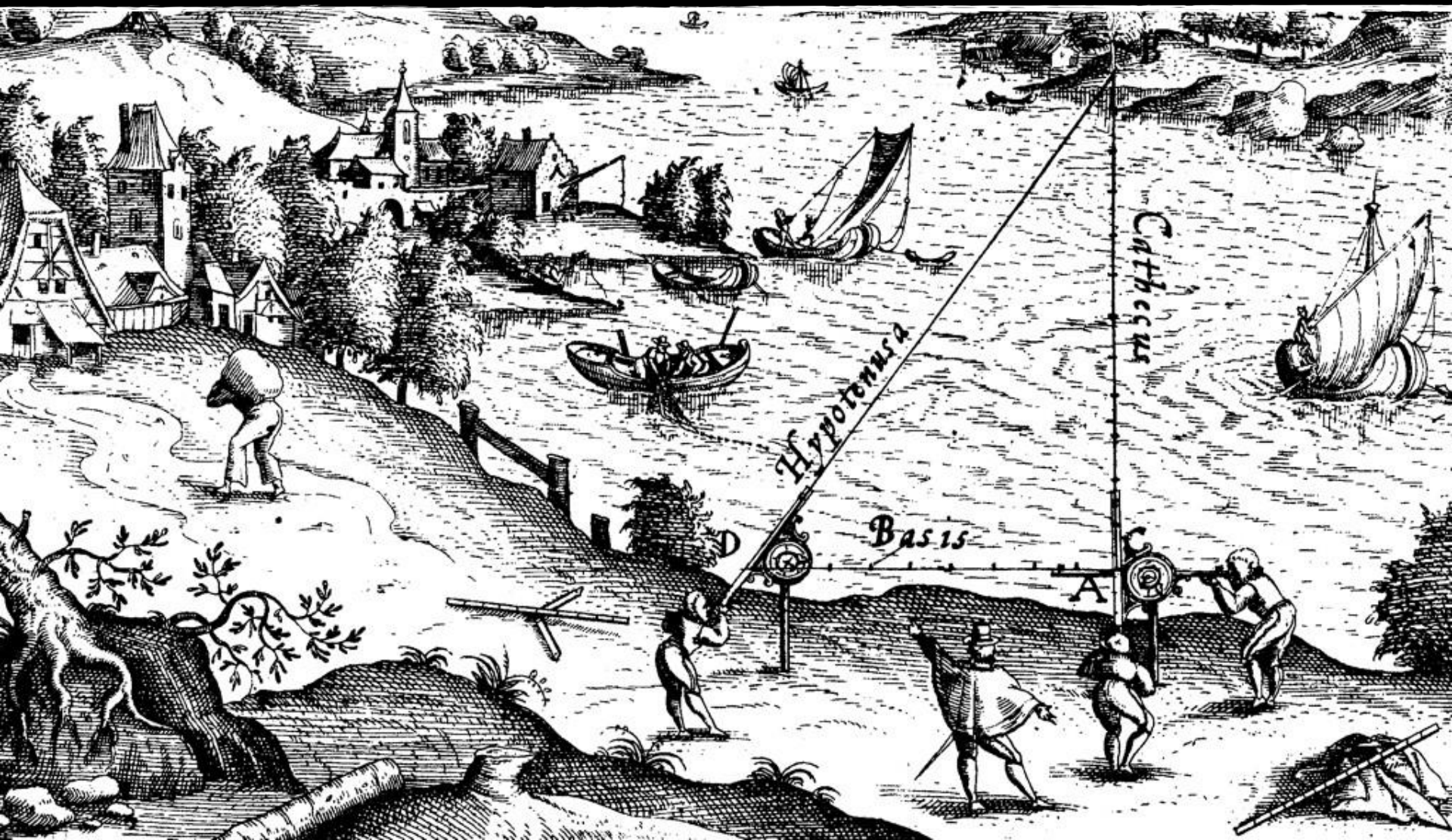
```
# Stereo rectification (uncalibrated variant)
h1, w1 = img1.shape
h2, w2 = img2.shape
_, H1, H2 = cv2.stereoRectifyUncalibrated(
      pts1, pts2, fundamental_matrix, imgSize=(w1, h1)
)

# Rectify the images
I1_rect = cv2.warpPerspective(I1, H1, (w1, h1))
I2_rect = cv2.warpPerspective(I2, H2, (w2, h2))

# Visualize rectified images
I1_I2_rect = np.concatenate((I1_rect, I2_rect), axis=1)
cv2.imshow('Rectified images', I1_I2_rect), cv2.waitKey(0)
```

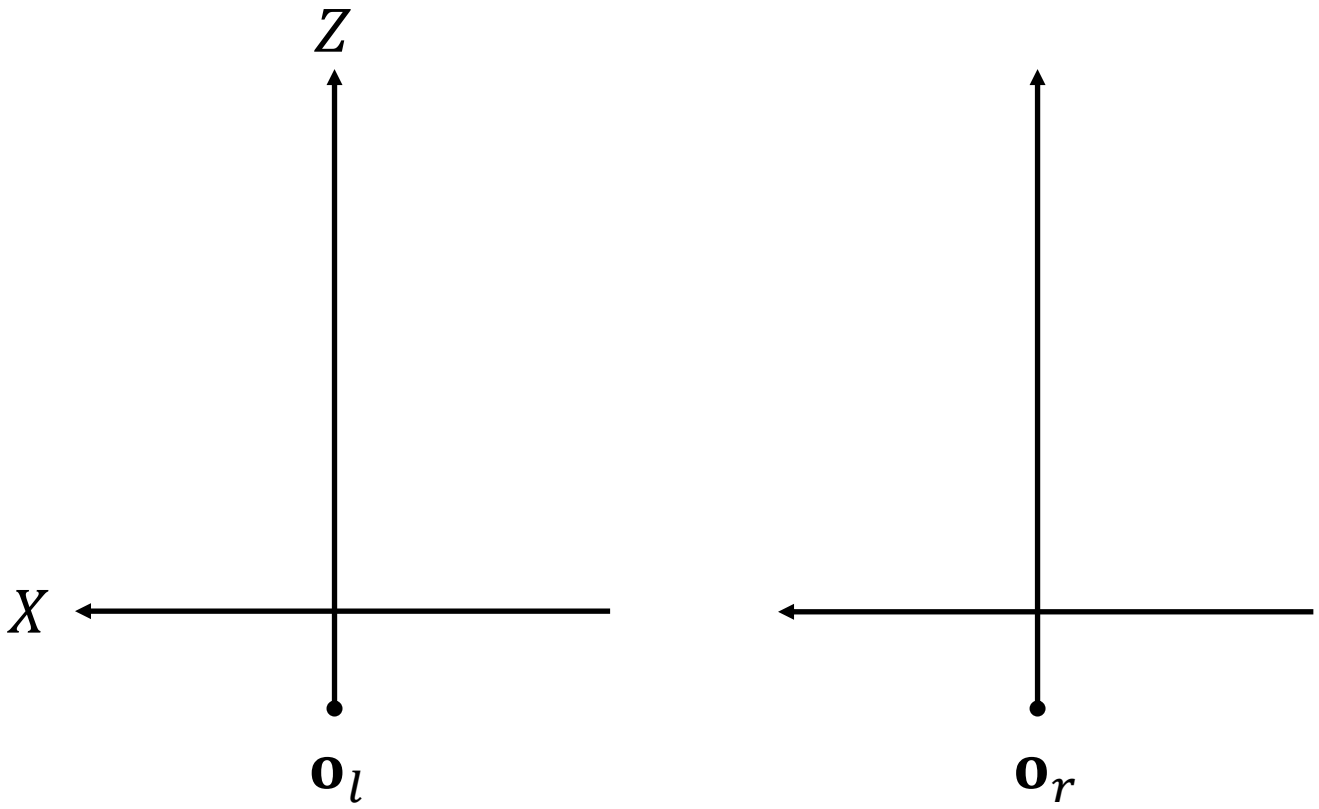
```
# Stereo rectification (uncalibrated variant)
h1, w1 = img1.shape
h2, w2 = img2.shape
_, H1, H2 = cv2.stereoRectifyUncalibrated(
      pts1, pts2, fundamental_matrix, imgSize=(w1, h1)
)

# Rectify the images
I1_rect = cv2.warpPerspective(I1, H1, (w1, h1))
I2_rect = cv2.warpPerspective(I2, H2, (w2, h2))

# Visualize rectified images
I1_I2_rect = np.concatenate((I1_rect, I2_rect), axis=1)
cv2.imshow('Rectified images', I1_I2_rect), cv2.waitKey(0)
```

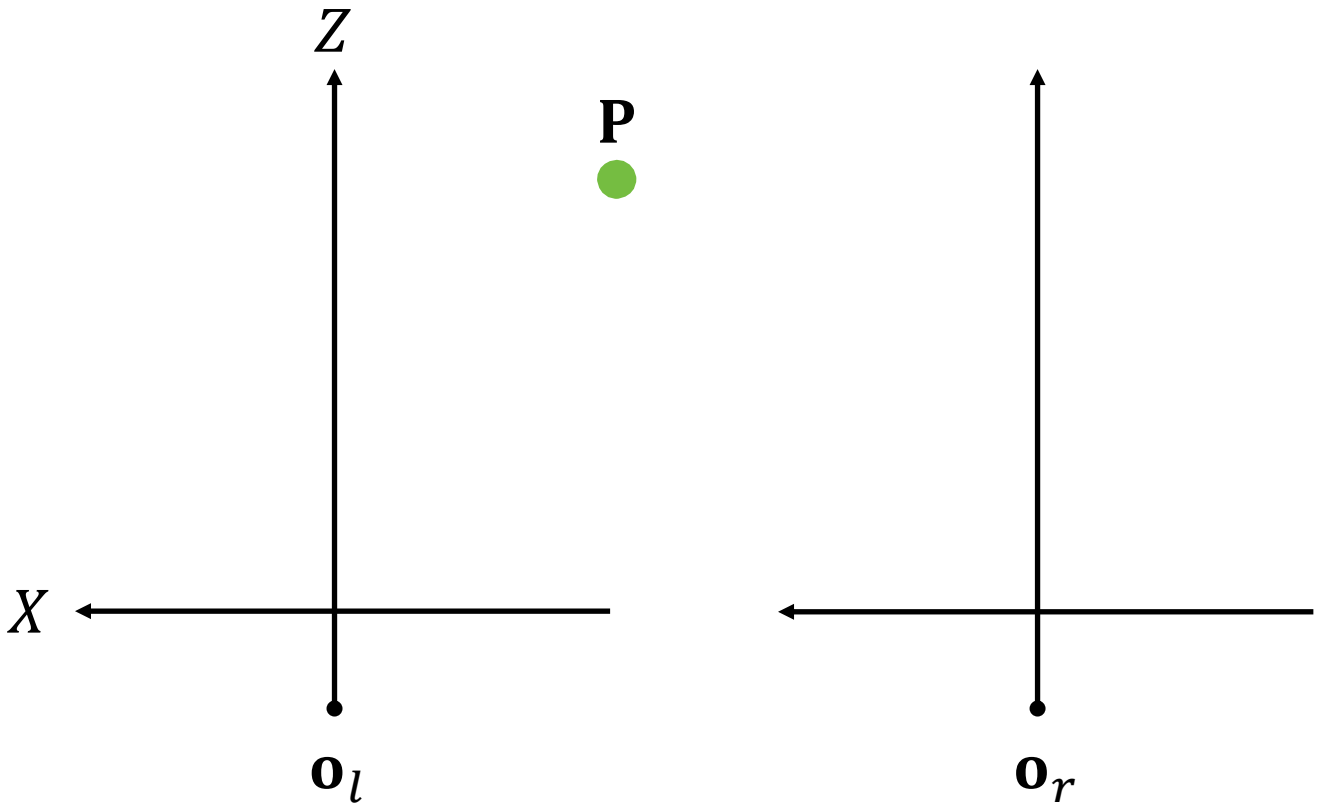


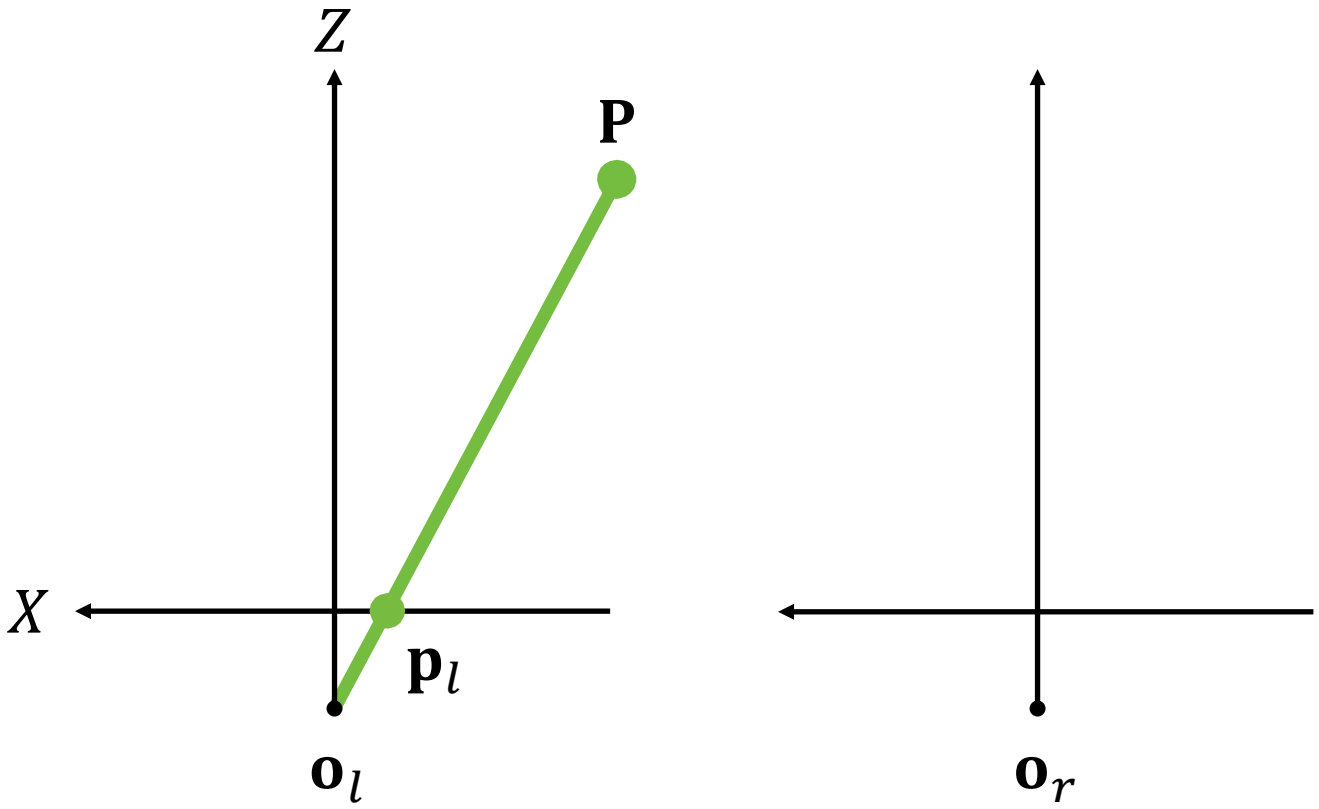
Python时间

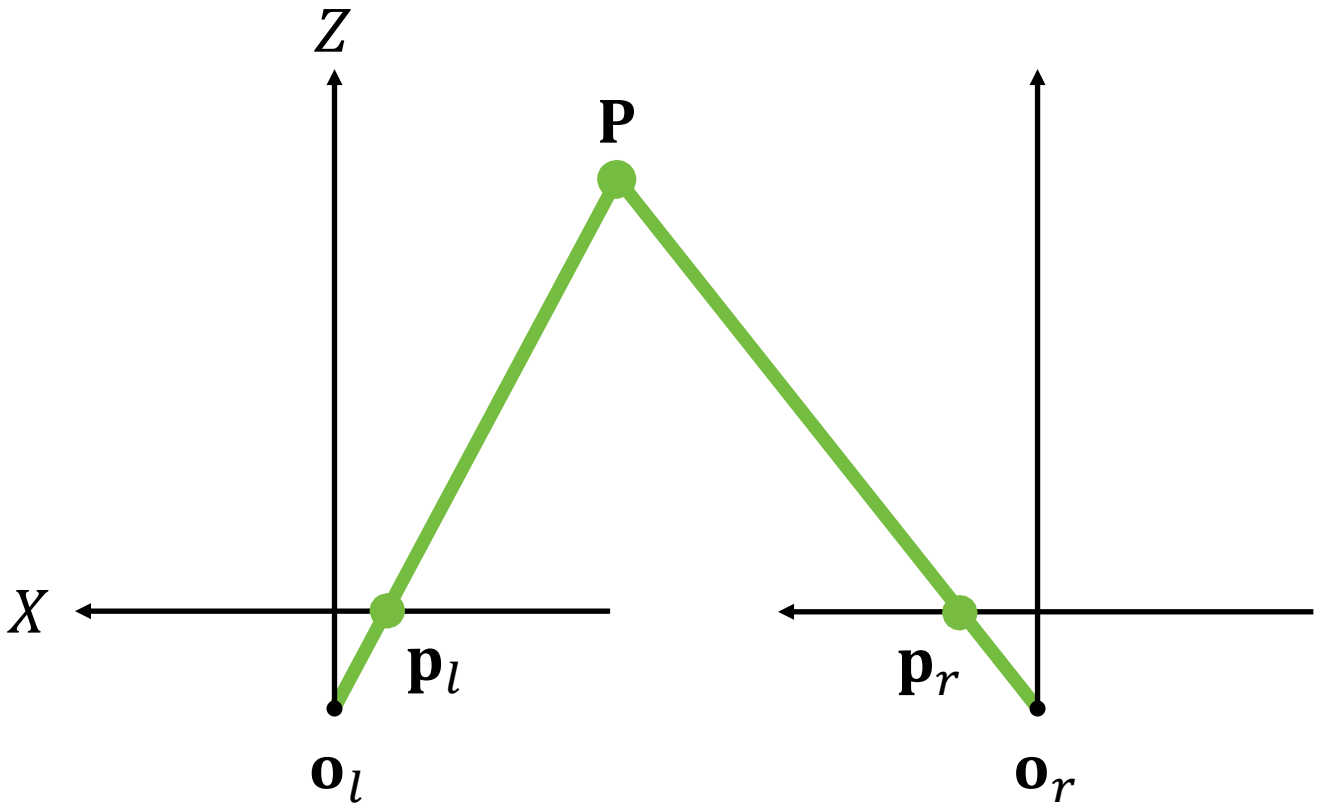
三角測量

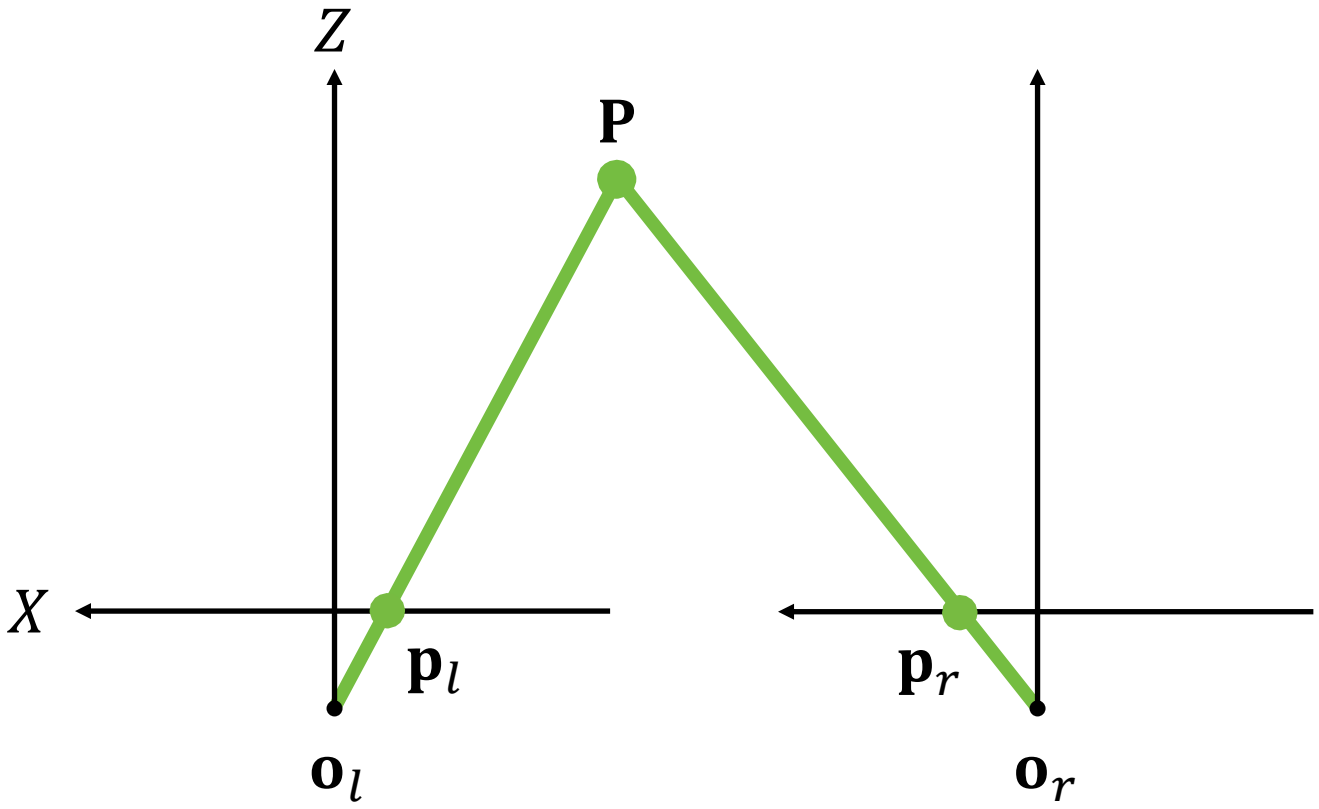


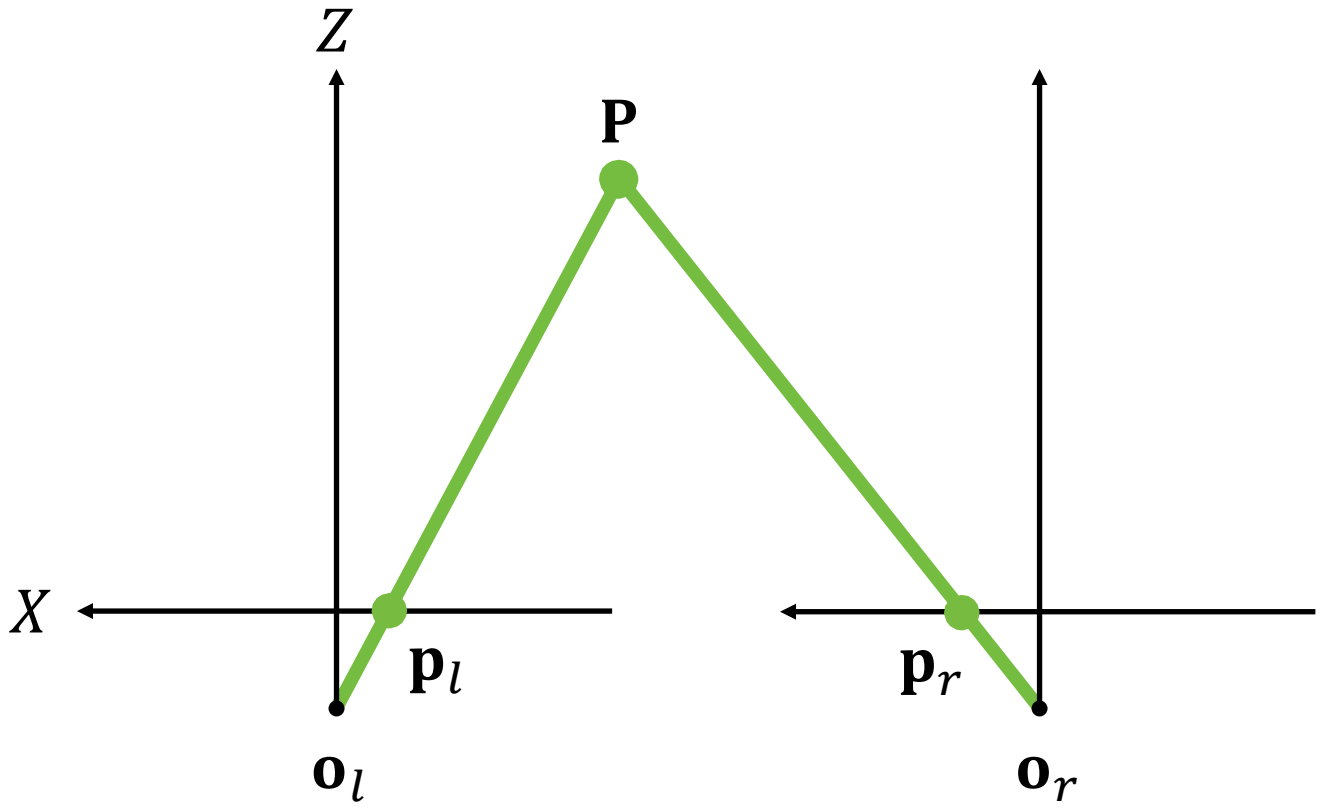

Z
X
$\mathbf{o}_l$
$\mathbf{o}_r$


Z
P
X
$\mathbf{o}_l$
$\mathbf{o}_r$

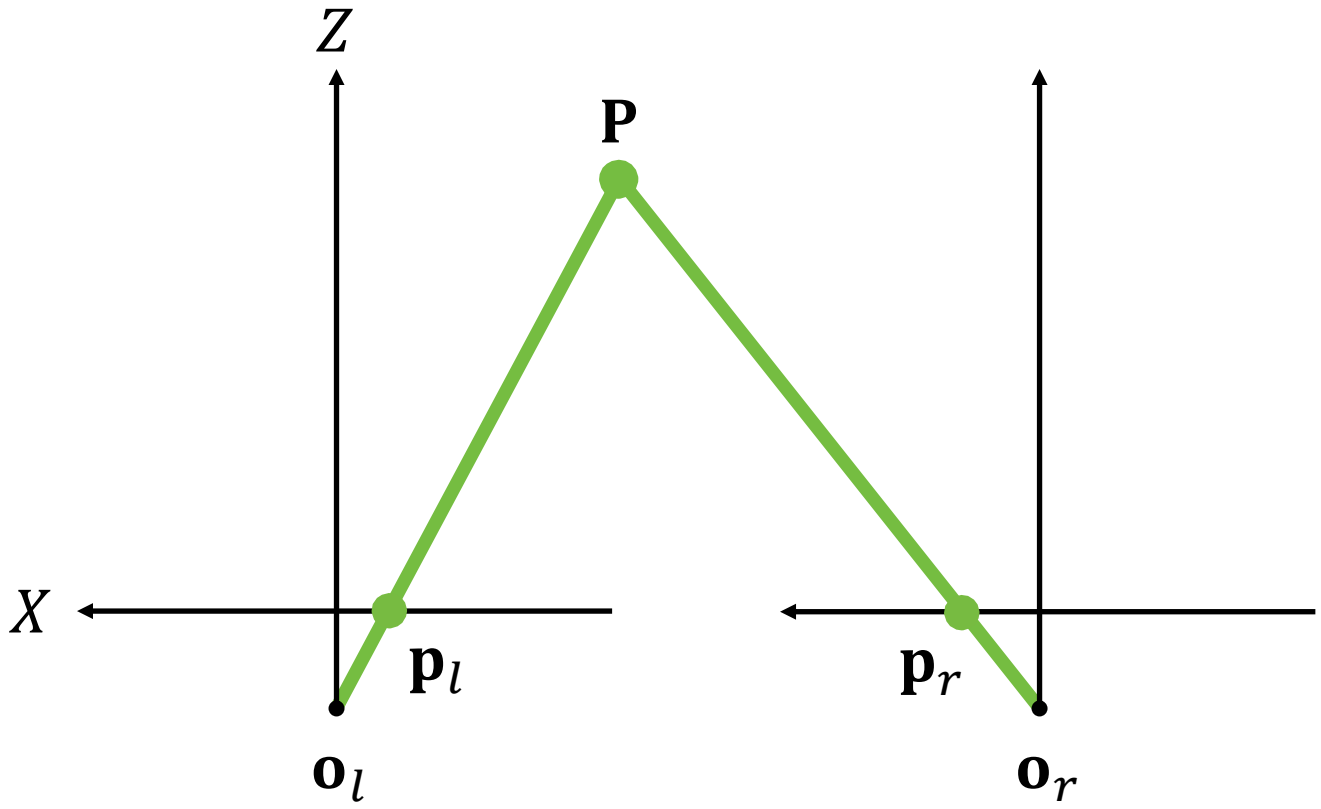

Z
P
X
$\mathbf{p}_l$
$\mathbf{o}_l$
$\mathbf{o}_r$


Z
P
X
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$


Z
P
X
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$



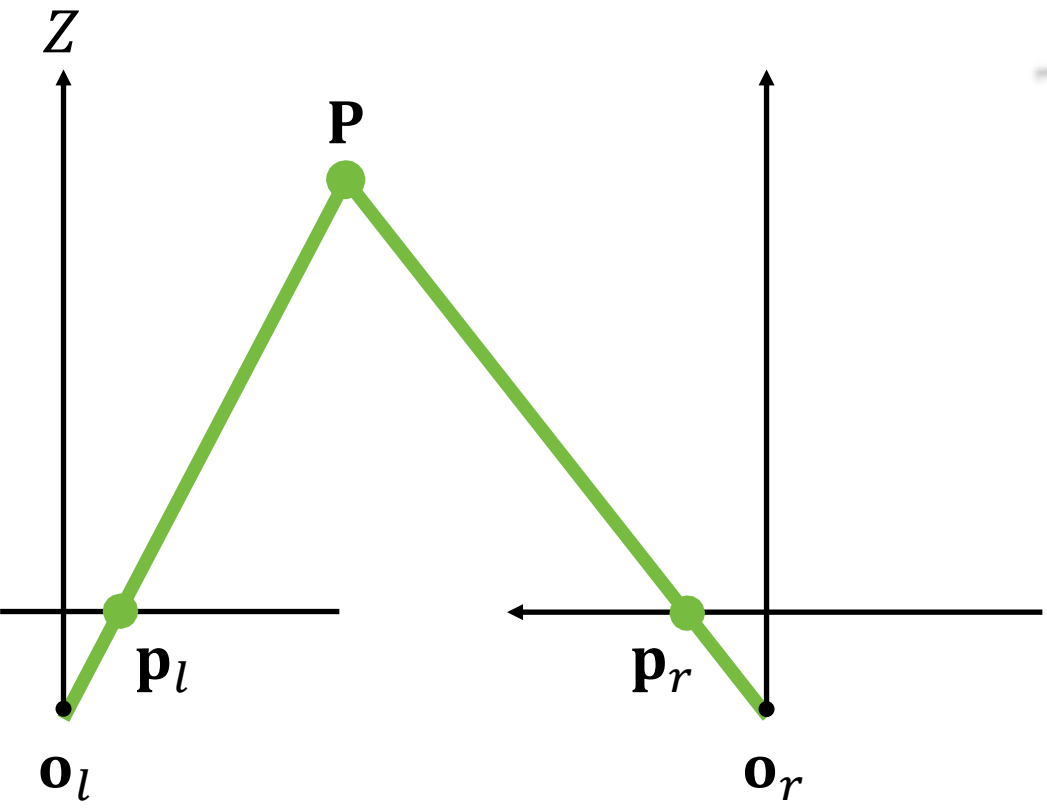
P的3D坐标可以通过射线的交点确定



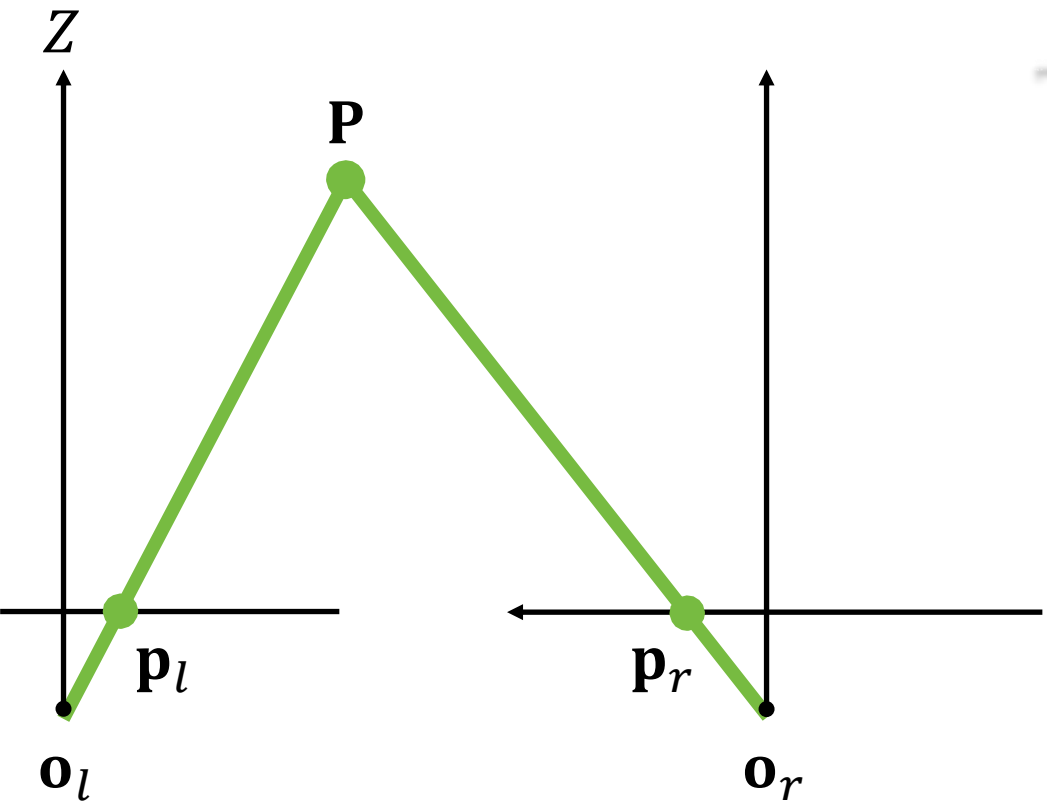
P的3D坐标可以通过射线的交点确定

这个过程称为三角测量

三角测量推导

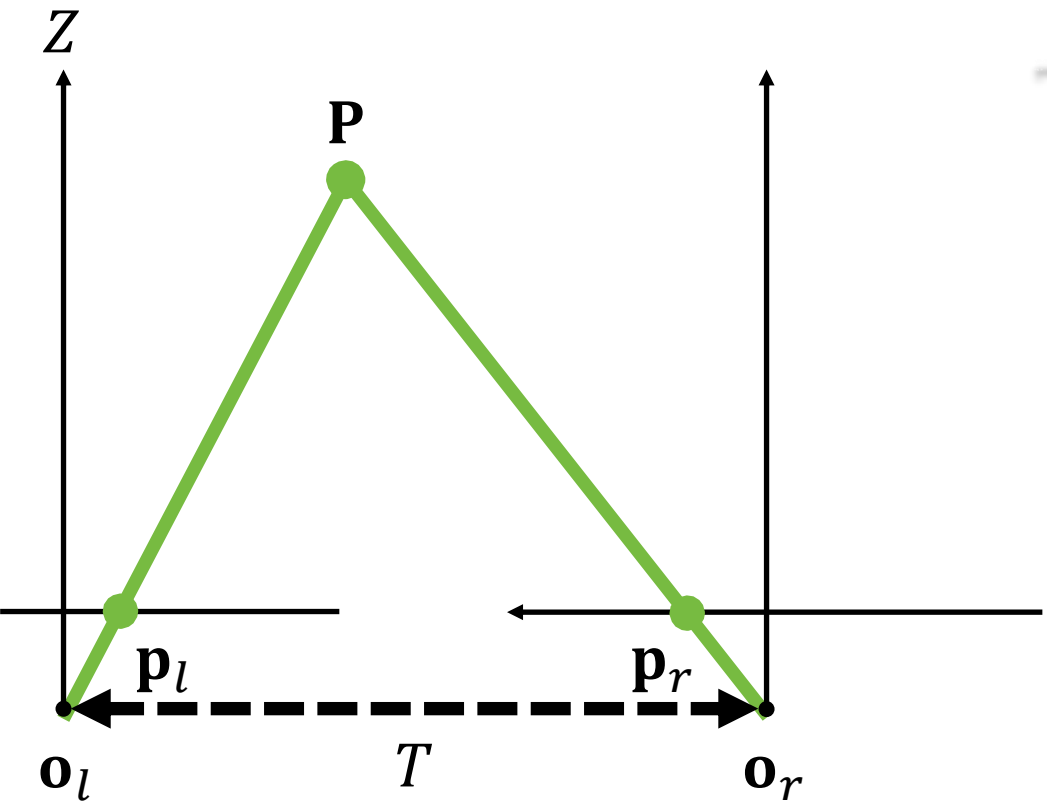


三角测量推导



定义

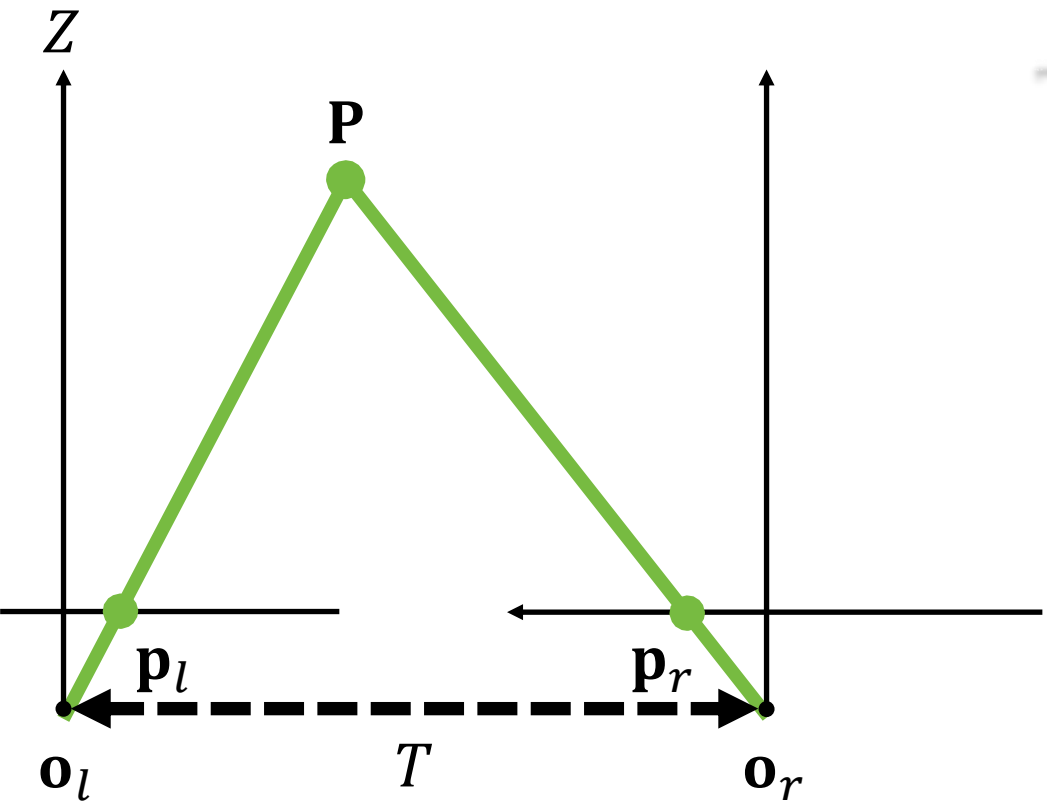
三角测量推导



定义

T相机中心距离

三角测量推导

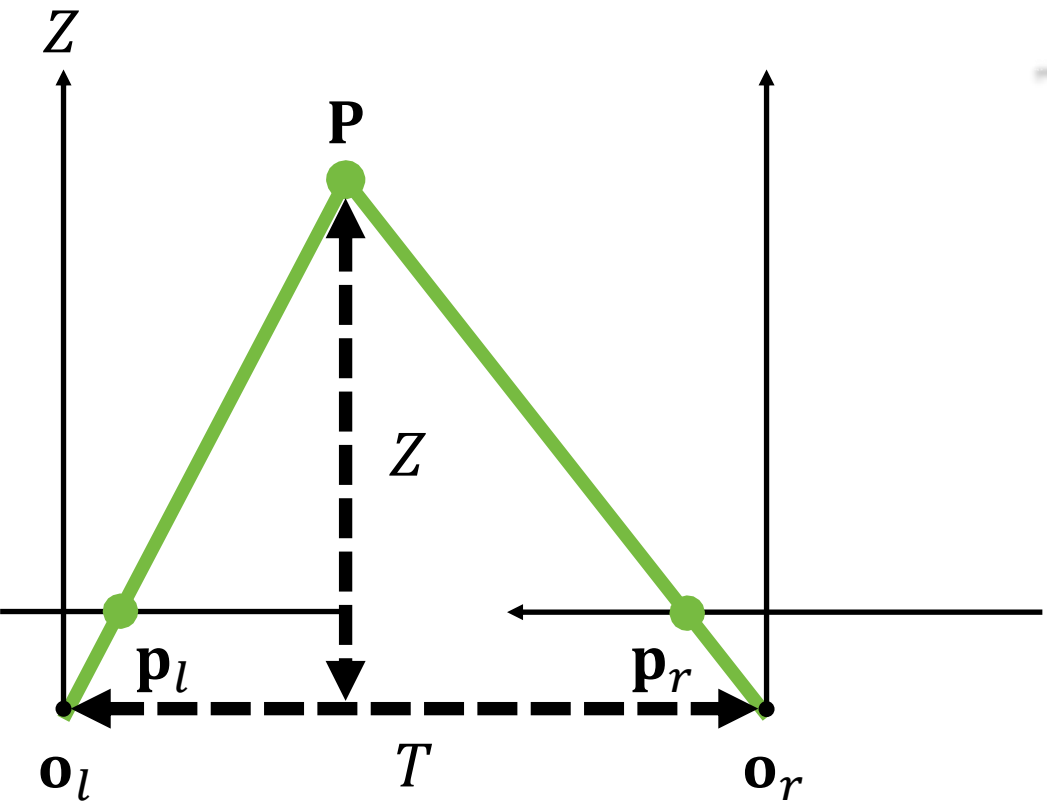


定义

T相机中心距离

基线宽度

三角测量推导

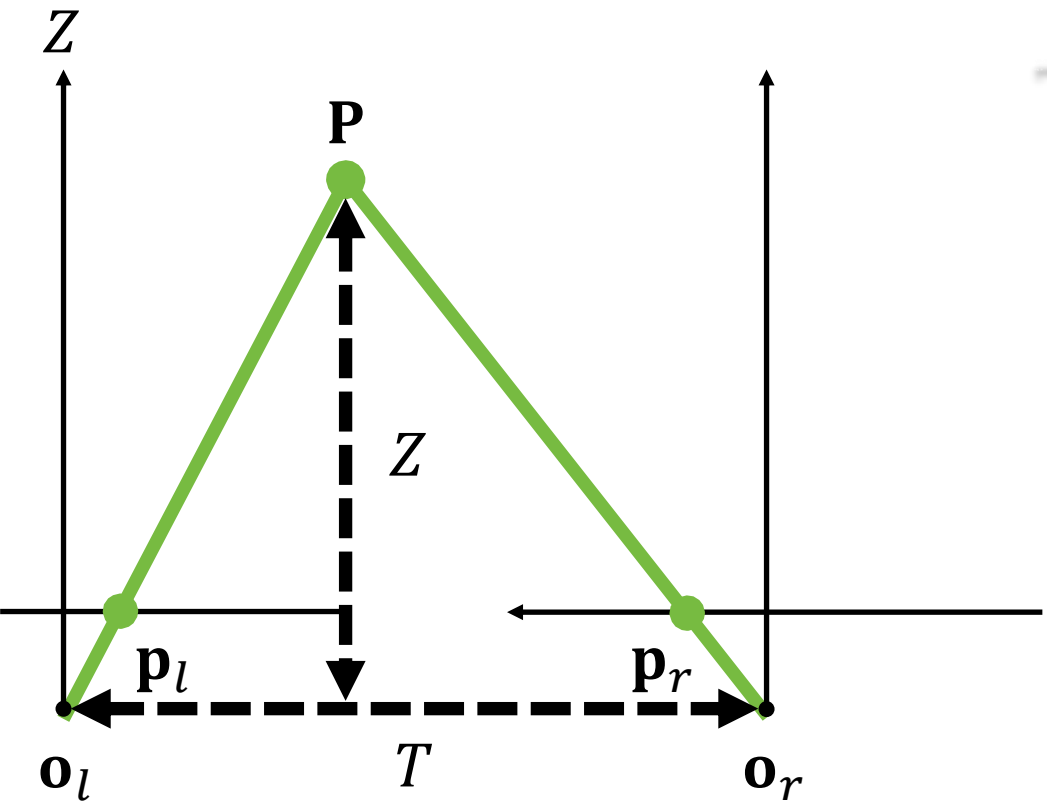


定义

T相机中心距离

Z到基线的距离

三角测量推导



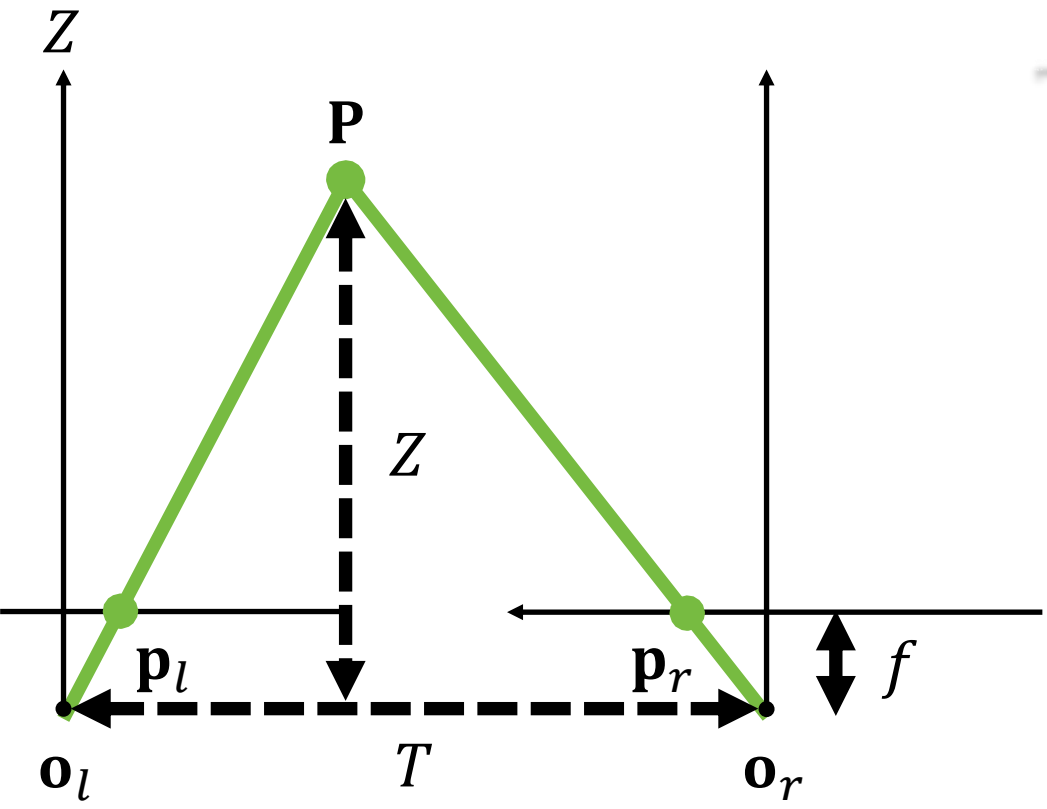
定义

T相机中心距离

Z到基线的距离

深度

三角测量推导



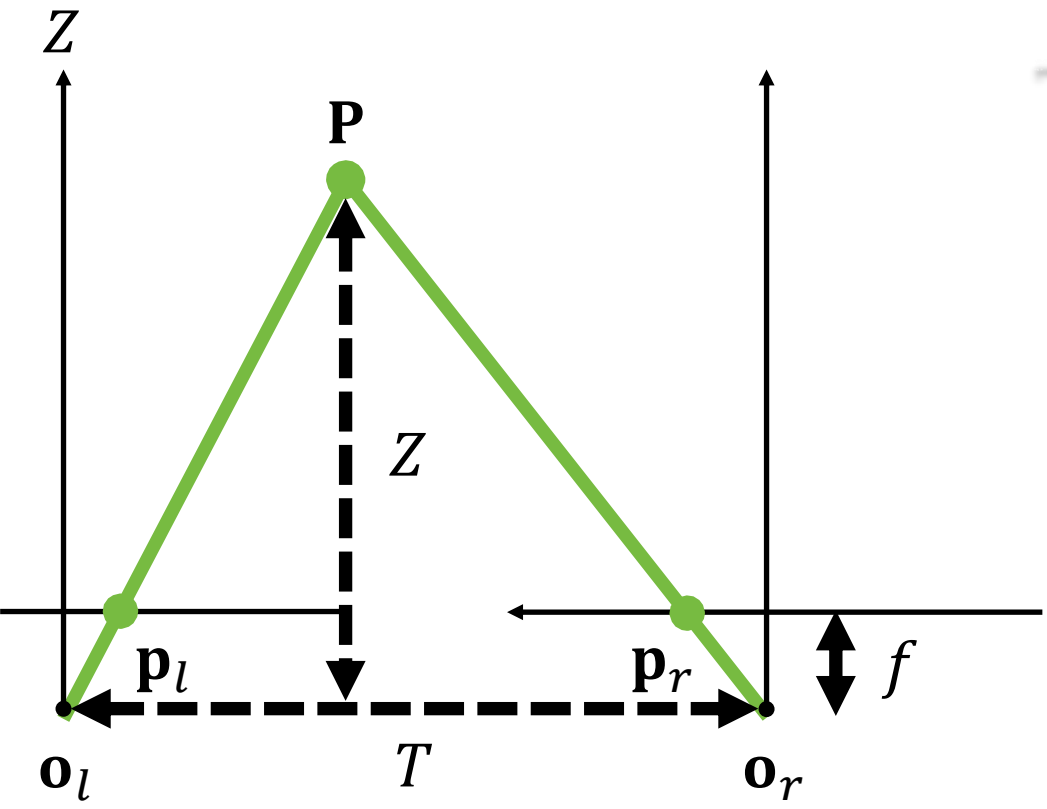
定义

T相机中心距离

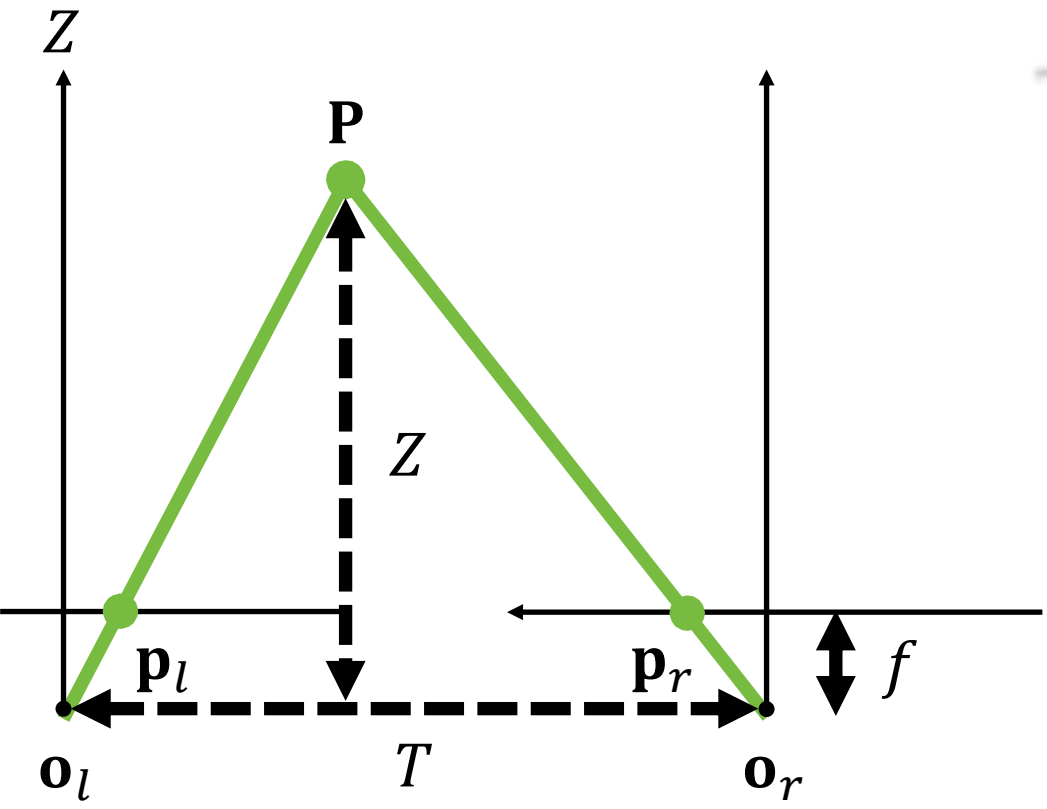
Z到基线的距离

f共同焦距

三角测量推导

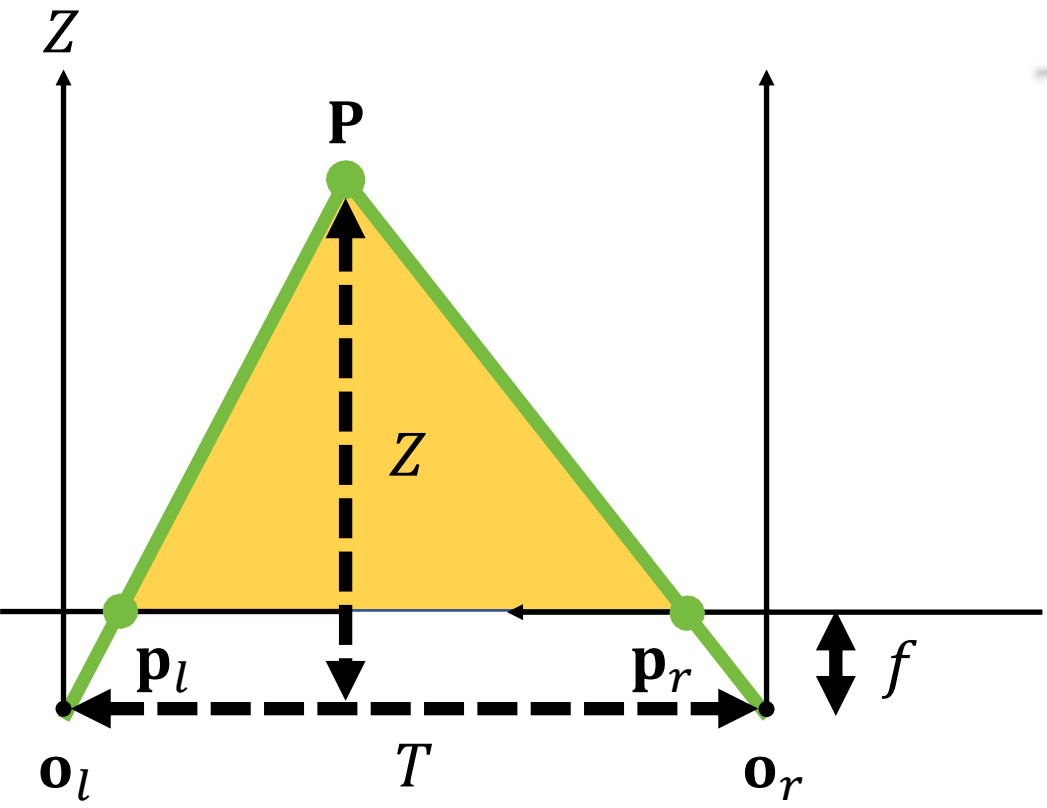


三角测量推导



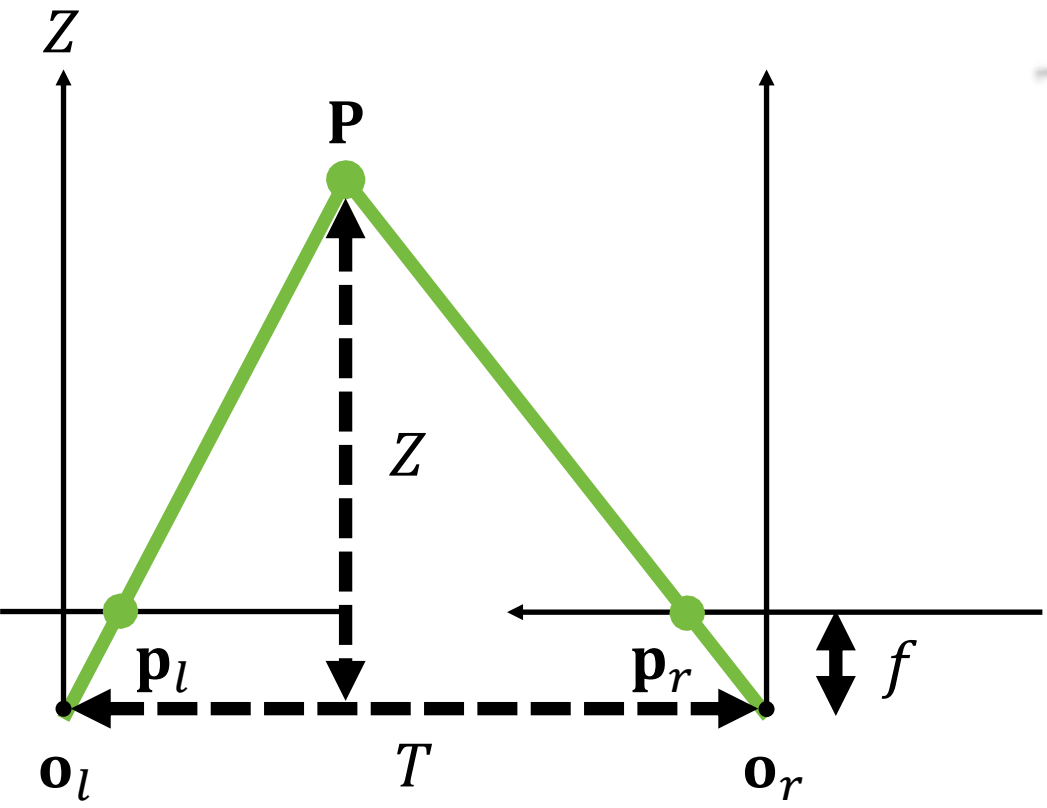
$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

三角测量推导



$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

三角测量推导

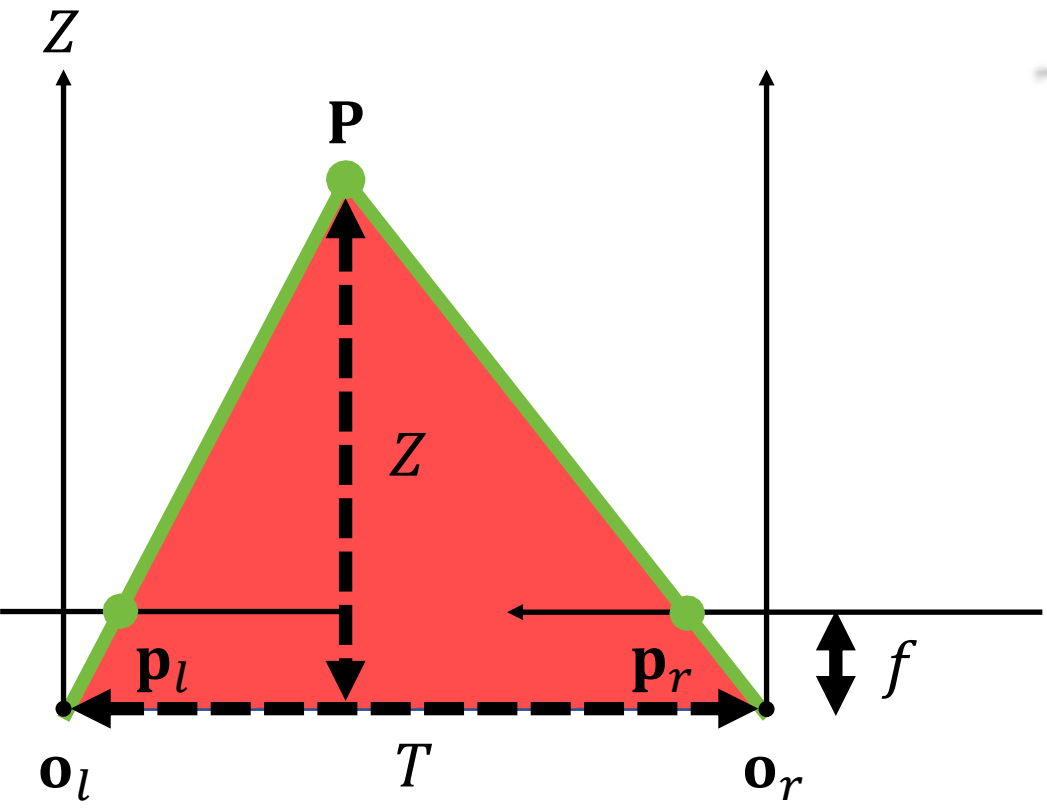


$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

和

$\Delta \mathbf{o}_l \mathbf{P} \mathbf{o}_r$

三角测量推导

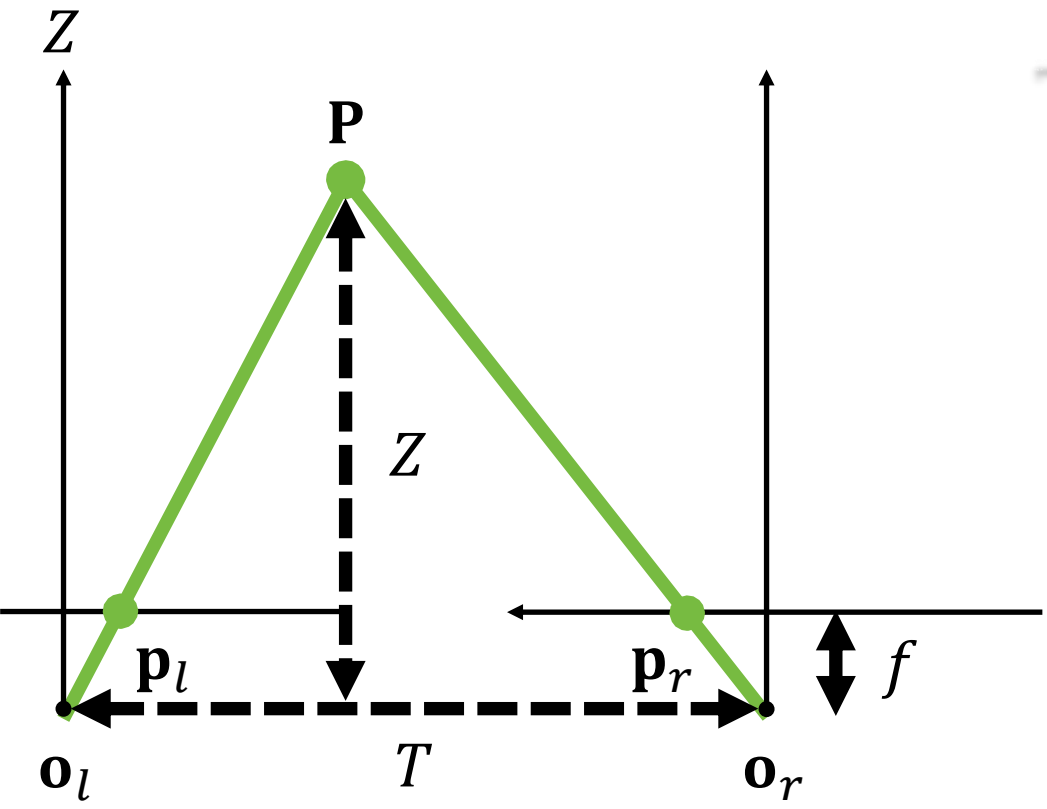


$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

和

$\Delta \mathbf{o}_l \mathbf{P} \mathbf{o}_r$

三角测量推导



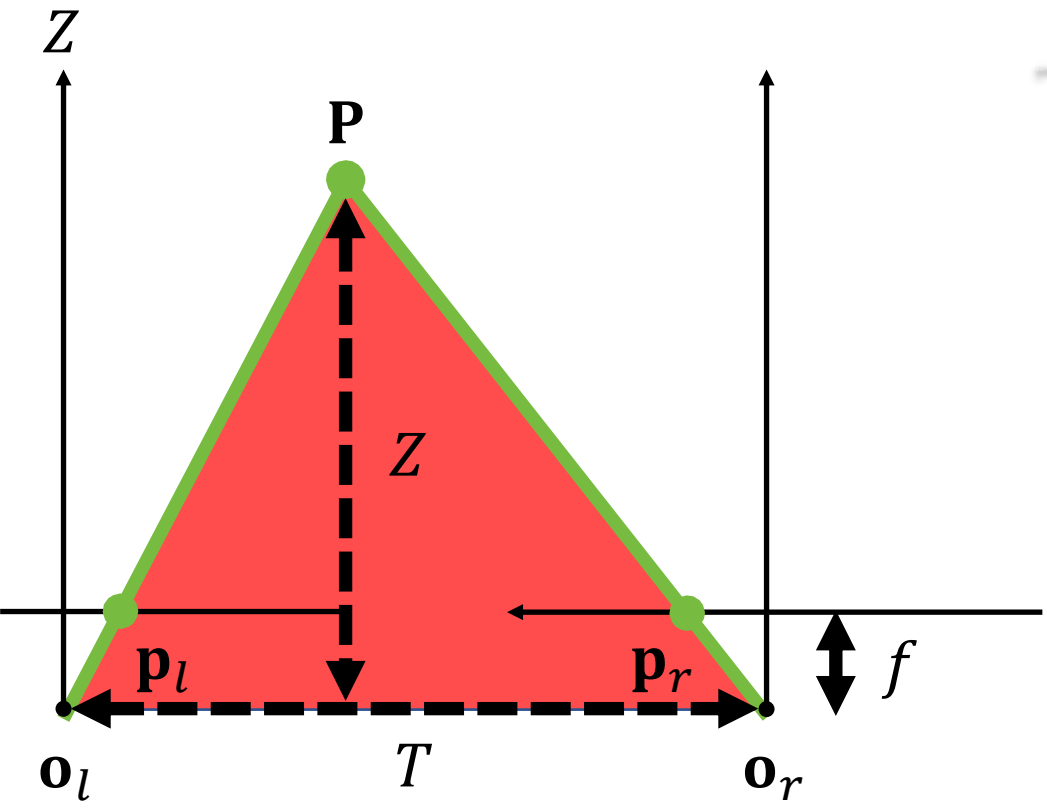
$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

和

$\Delta \mathbf{o}_l \mathbf{P} \mathbf{o}_r$

通过相似三角形

三角测量推导



$\Delta \mathbf{p}_l \mathbf{P} \mathbf{p}_r$

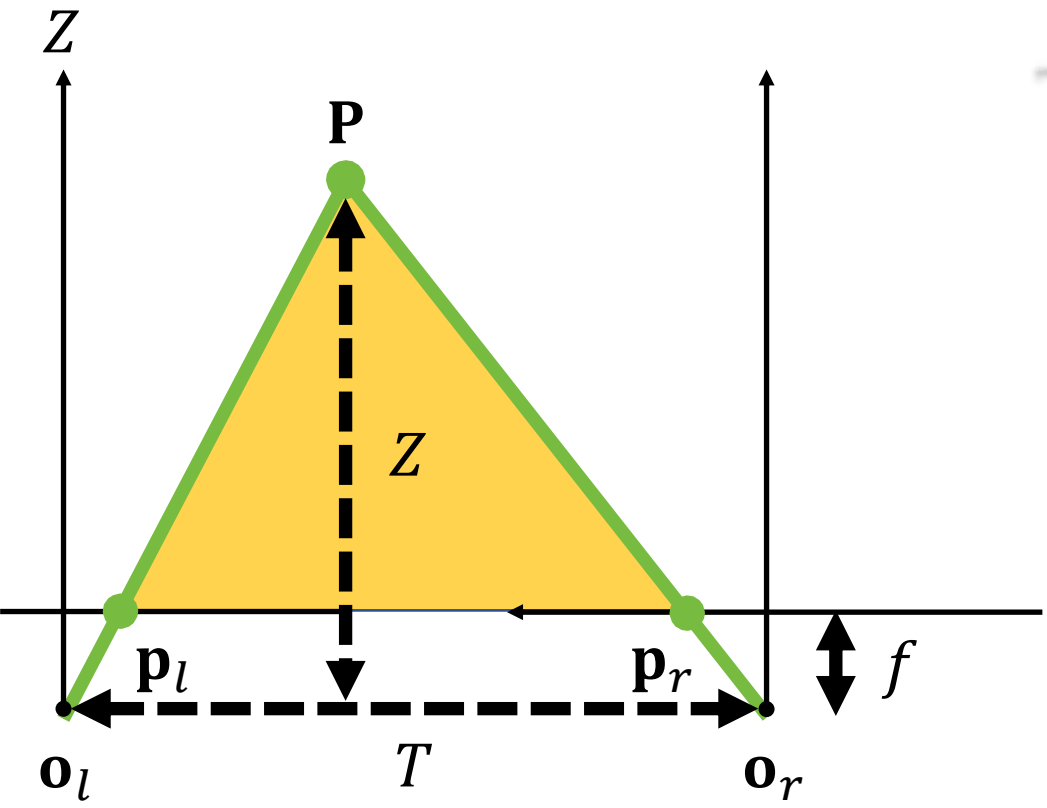
和

$\Delta \mathbf{o}_l \mathbf{P} \mathbf{o}_r$

通过相似三角形

$\frac{T}{Z}$

三角测量推导



$\Delta p_l P p_r$

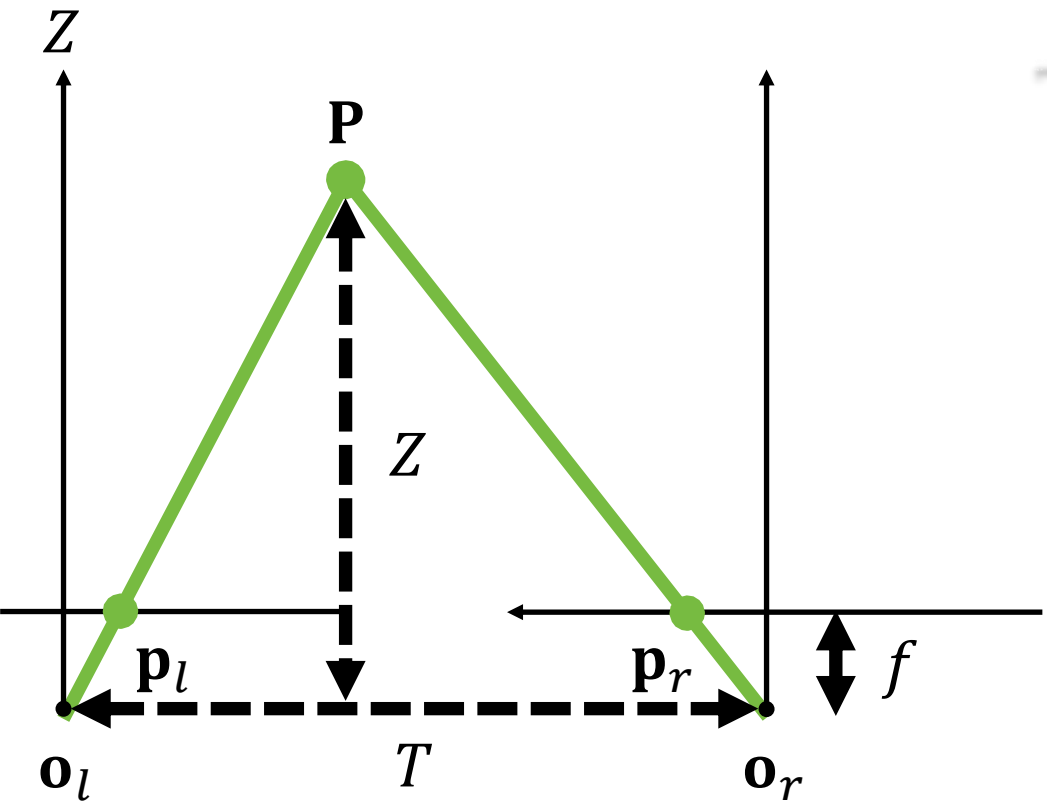
和

$\Delta o_l P o_r$

通过相似三角形

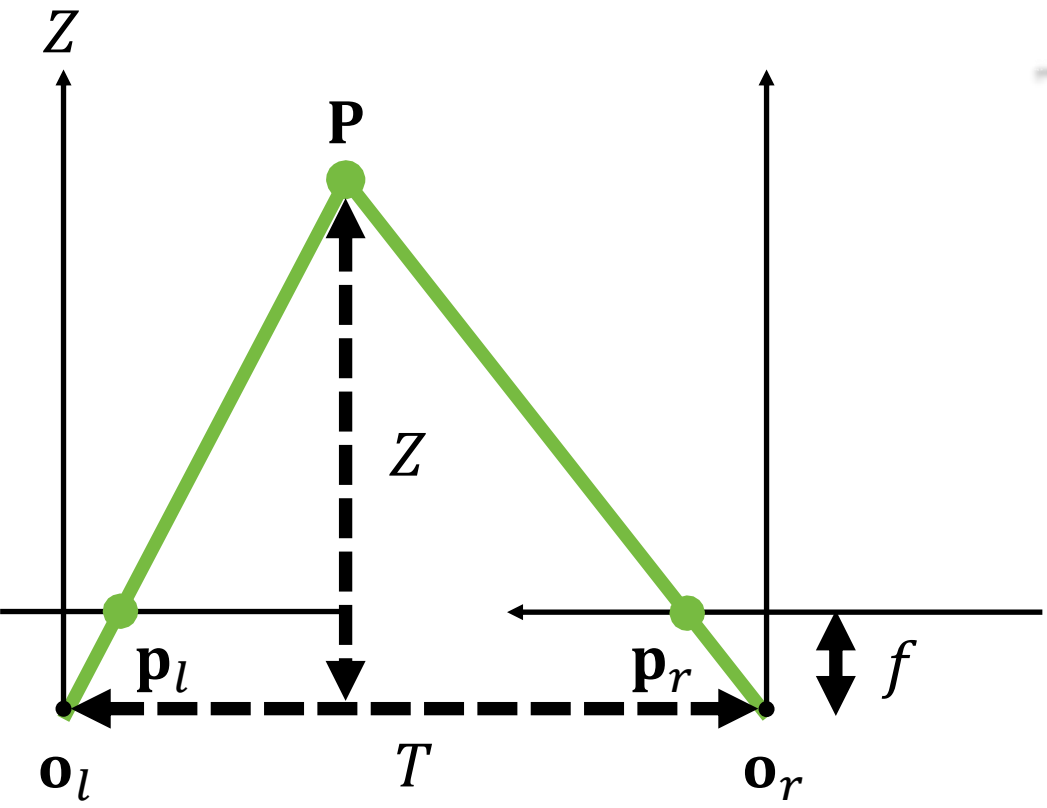
$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

三角测量推导



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

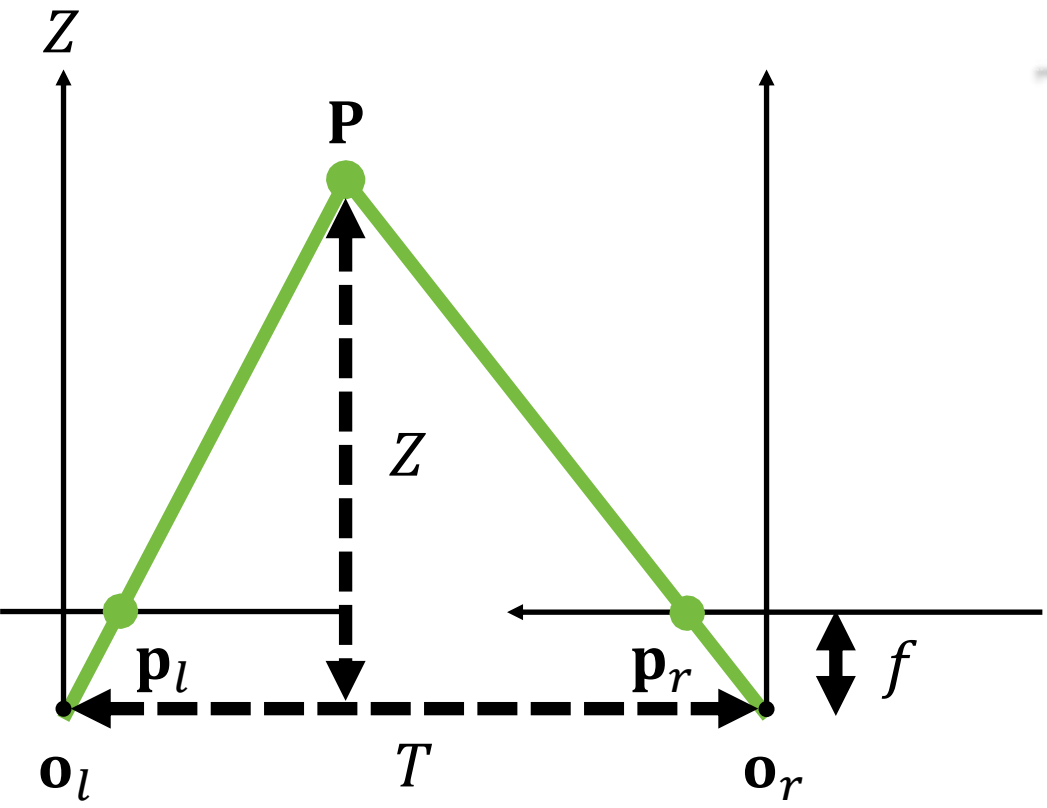
三角测量推导



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

令$d = x_l - x_r$为视差

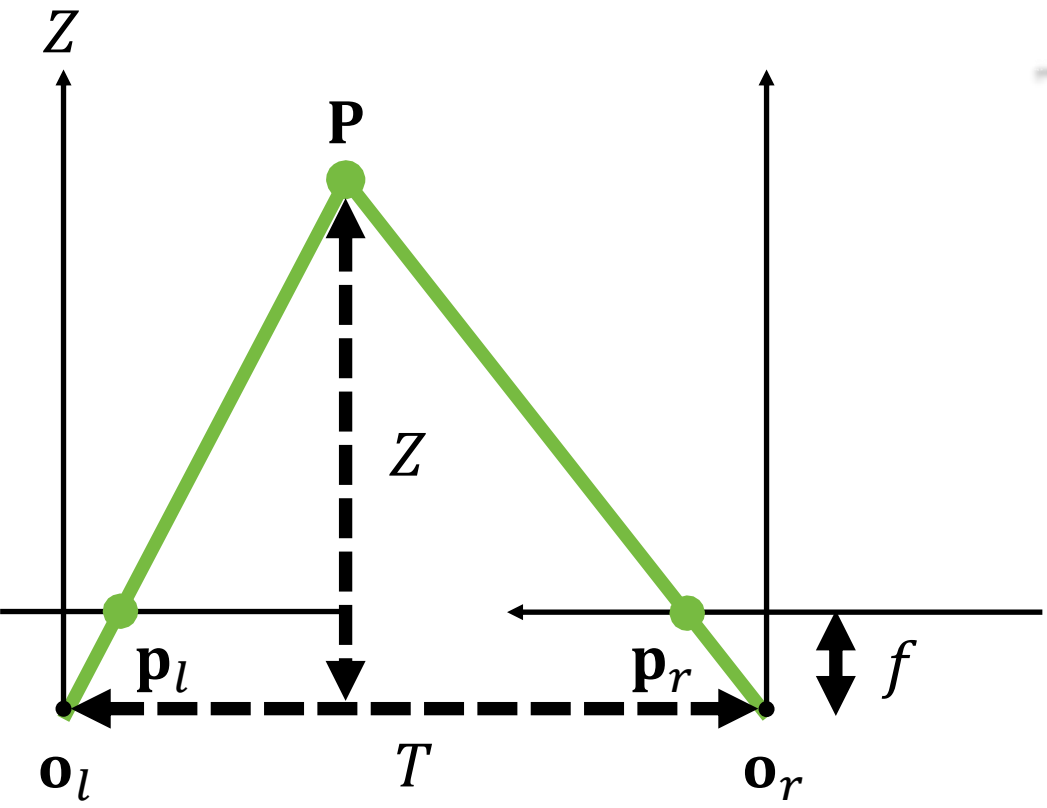
三角测量推导



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

令$d = x_l - x_r$为视差

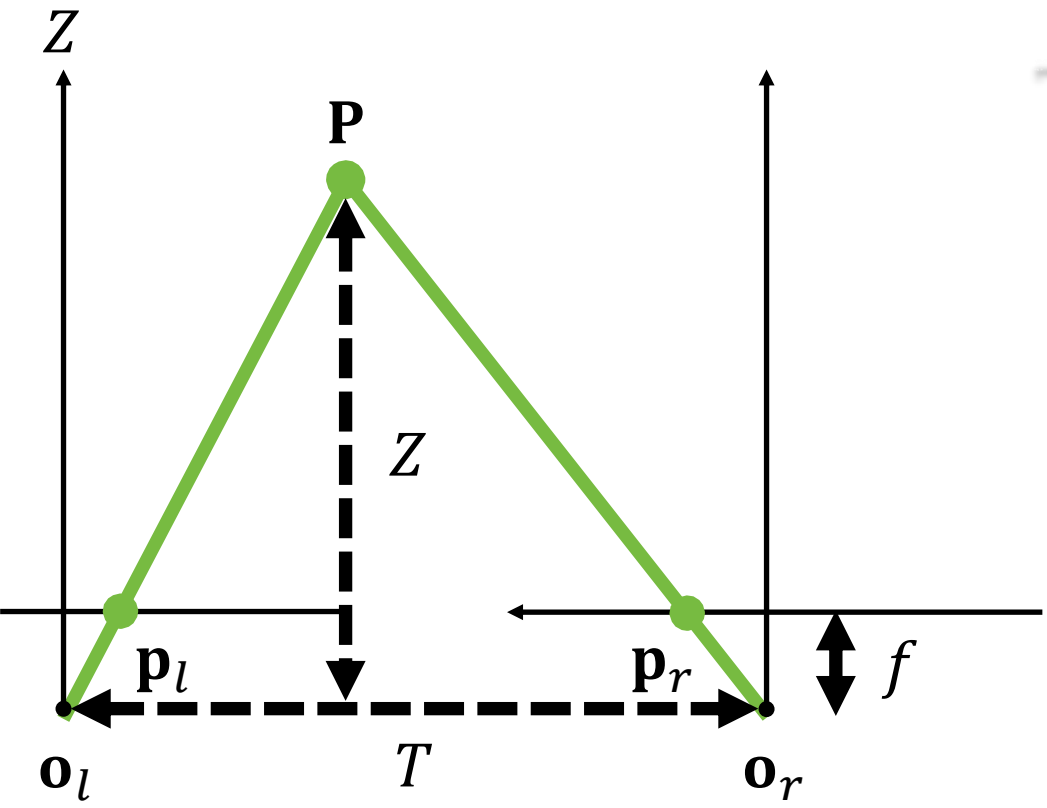
三角测量推导



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

令$d = x_l - x_r$为视差

$$\frac{T + d}{Z - f} = \frac{T}{Z}$$



三角测量推导

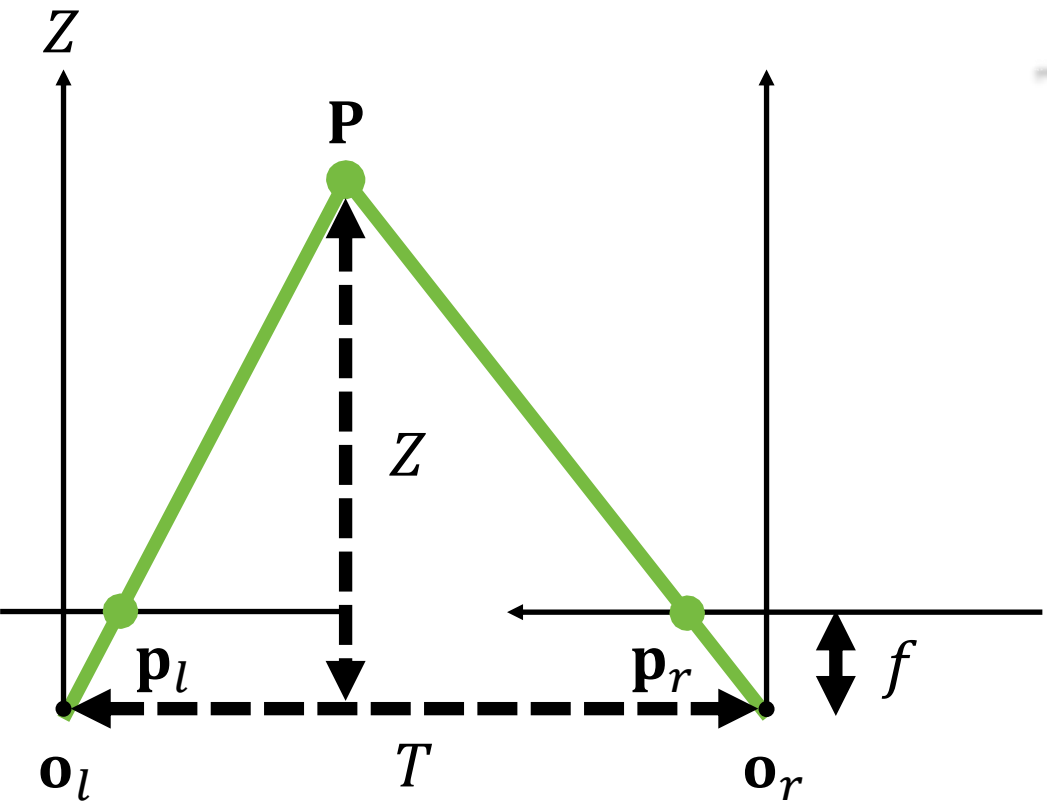
$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

令$d = x_l - x_r$为视差

$$\frac{T + d}{Z - f} = \frac{T}{Z}$$

整理

三角测量推导



$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

令$d = x_l - x_r$为视差

$$\frac{T + d}{Z - f} = \frac{T}{Z}$$

整理

视差方程：$Z = f\frac{T}{d}$

视差方程：$Z = f\frac{T}{d}$

备注：

视差方程：$Z = f\frac{T}{d}$

备注：

假设立体相机是完全平行的

视差方程：$Z = f\frac{T}{d}$

备注：

假设立体相机是完全平行的

假设对应点完全匹配

视差方程：$Z = f\frac{T}{d}$

备注：

假设立体相机是完全平行的

假设对应点完全匹配

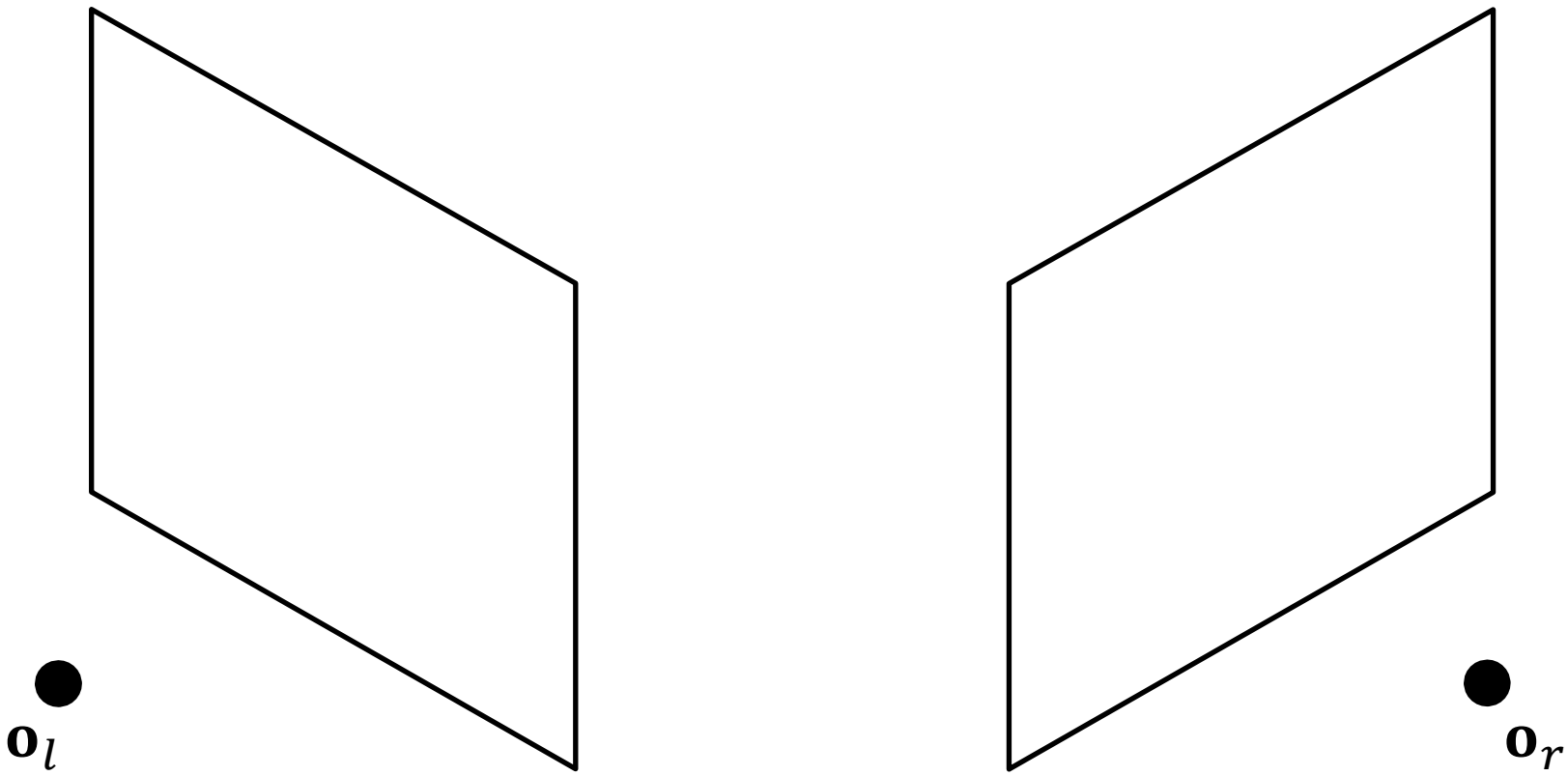
深度与视差成反比

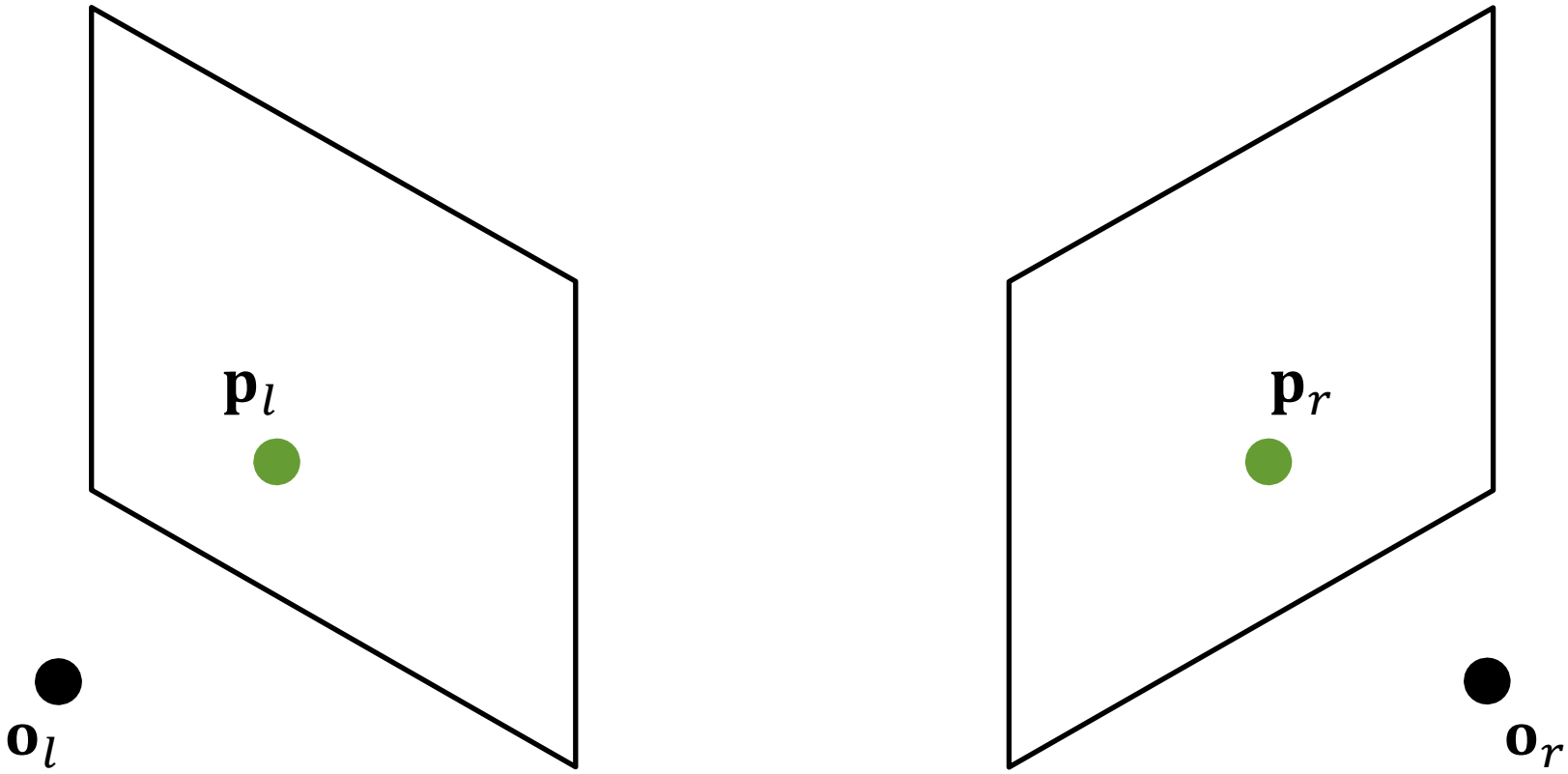
假设我们有一个完全标定的立体视觉装置

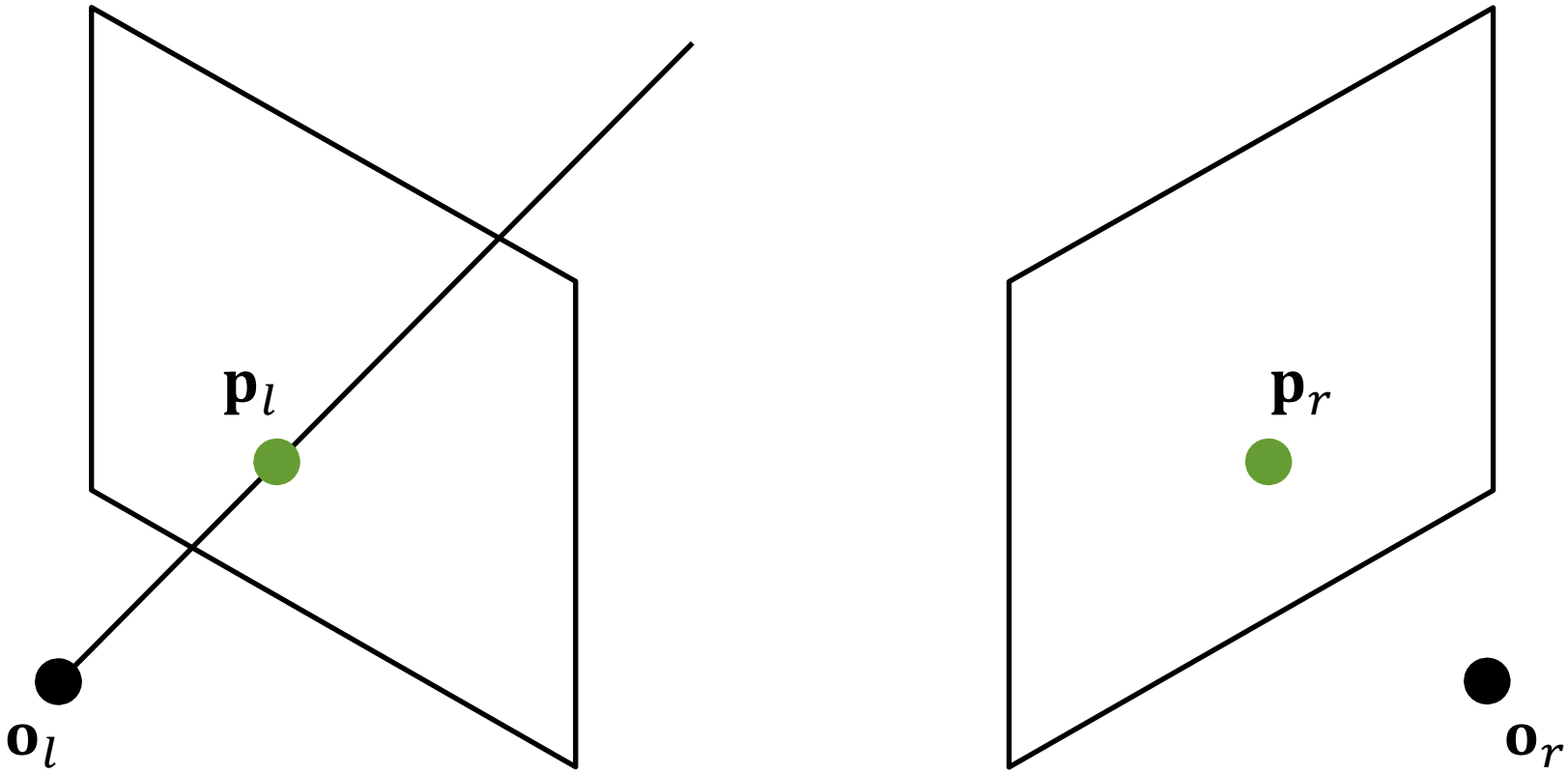
假设我们有一个完全标定的立体视觉装置

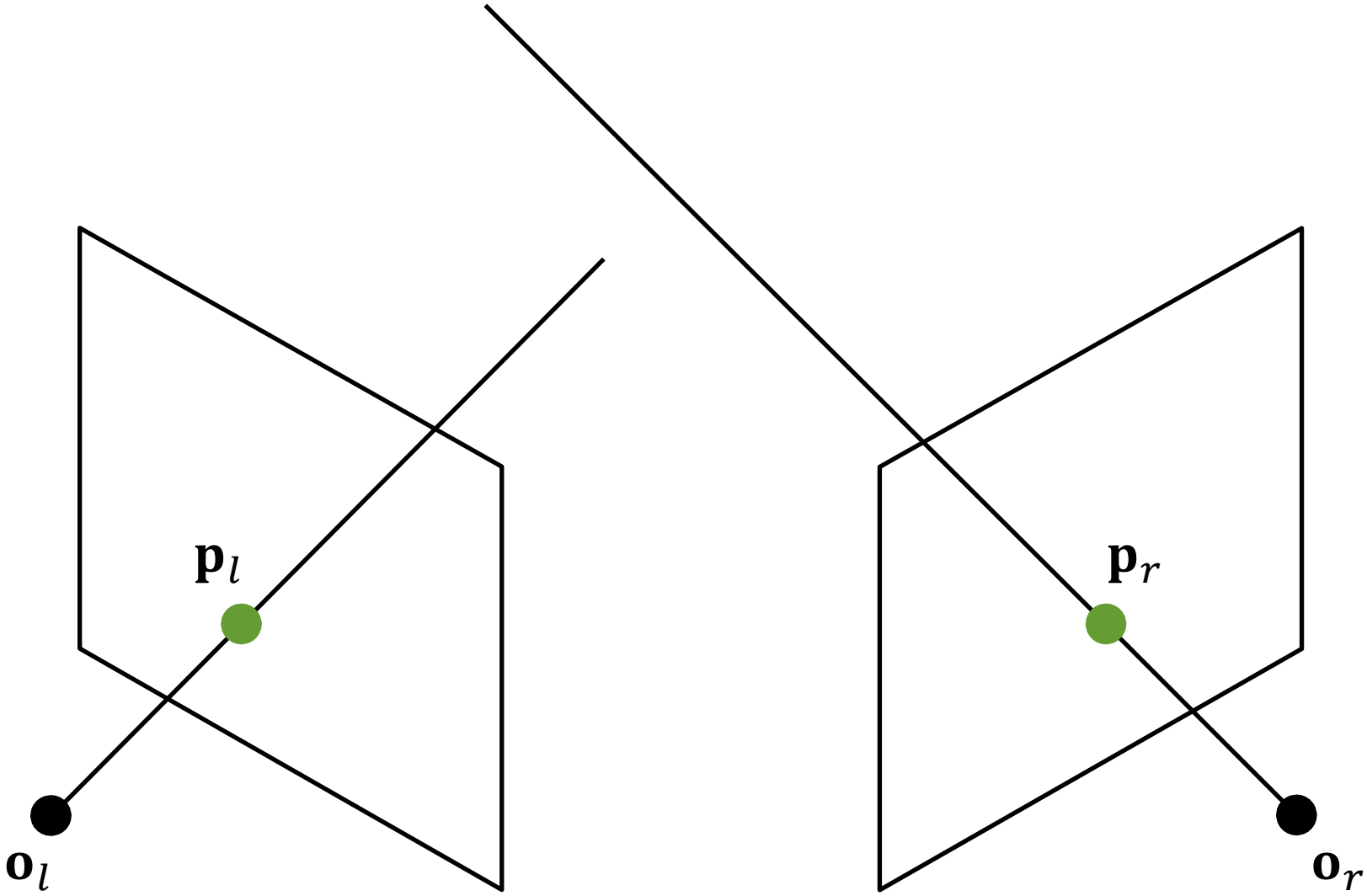
假设我们有一个完全标定的立体视觉装置

摄像机的内参和外参是已知的

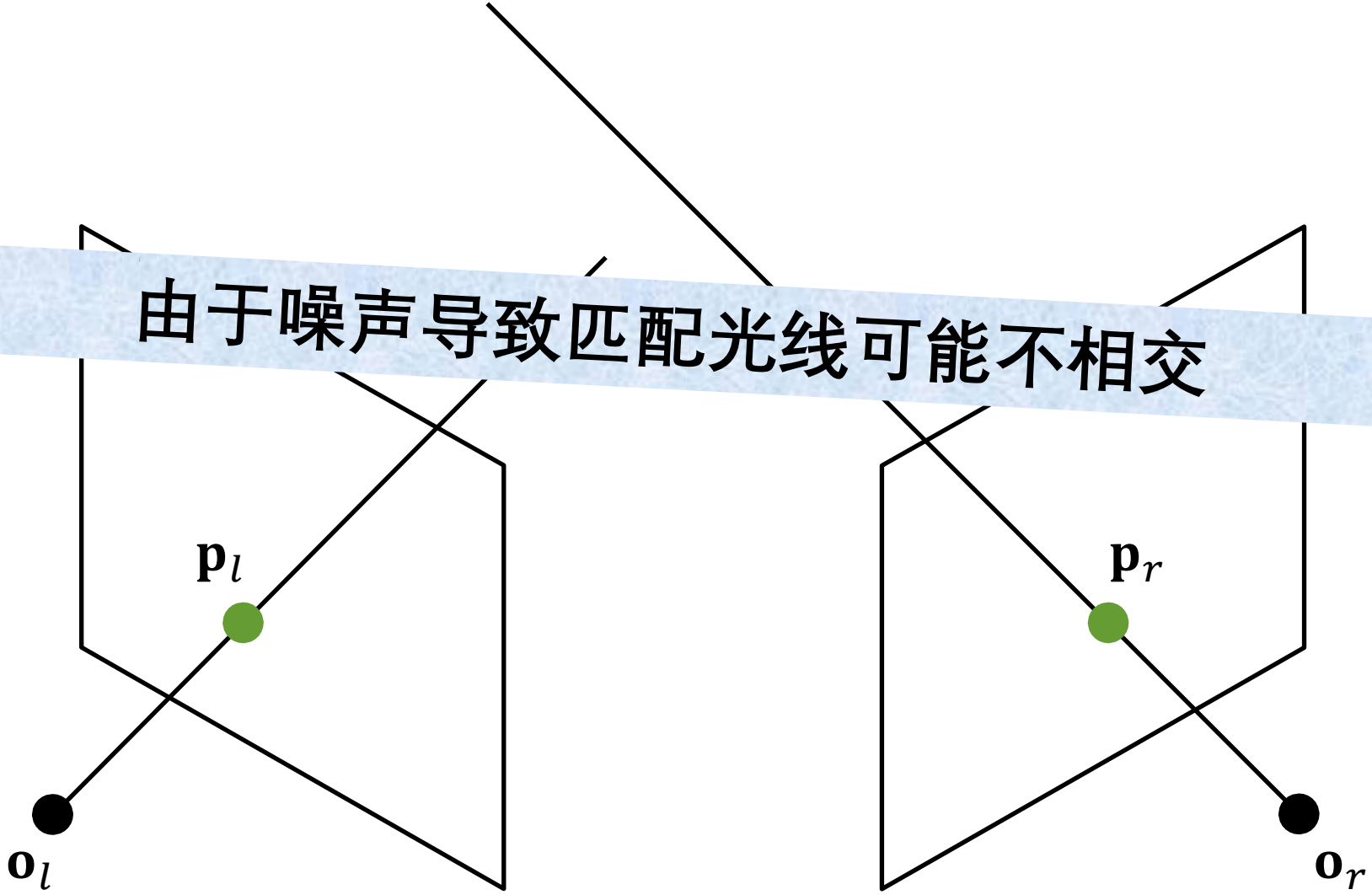

$\mathbf{o}_l$
$\mathbf{o}_r$


$\mathbf{p}_l$
$\mathbf{o}_l$
$\mathbf{p}_r$
$\mathbf{o}_r$

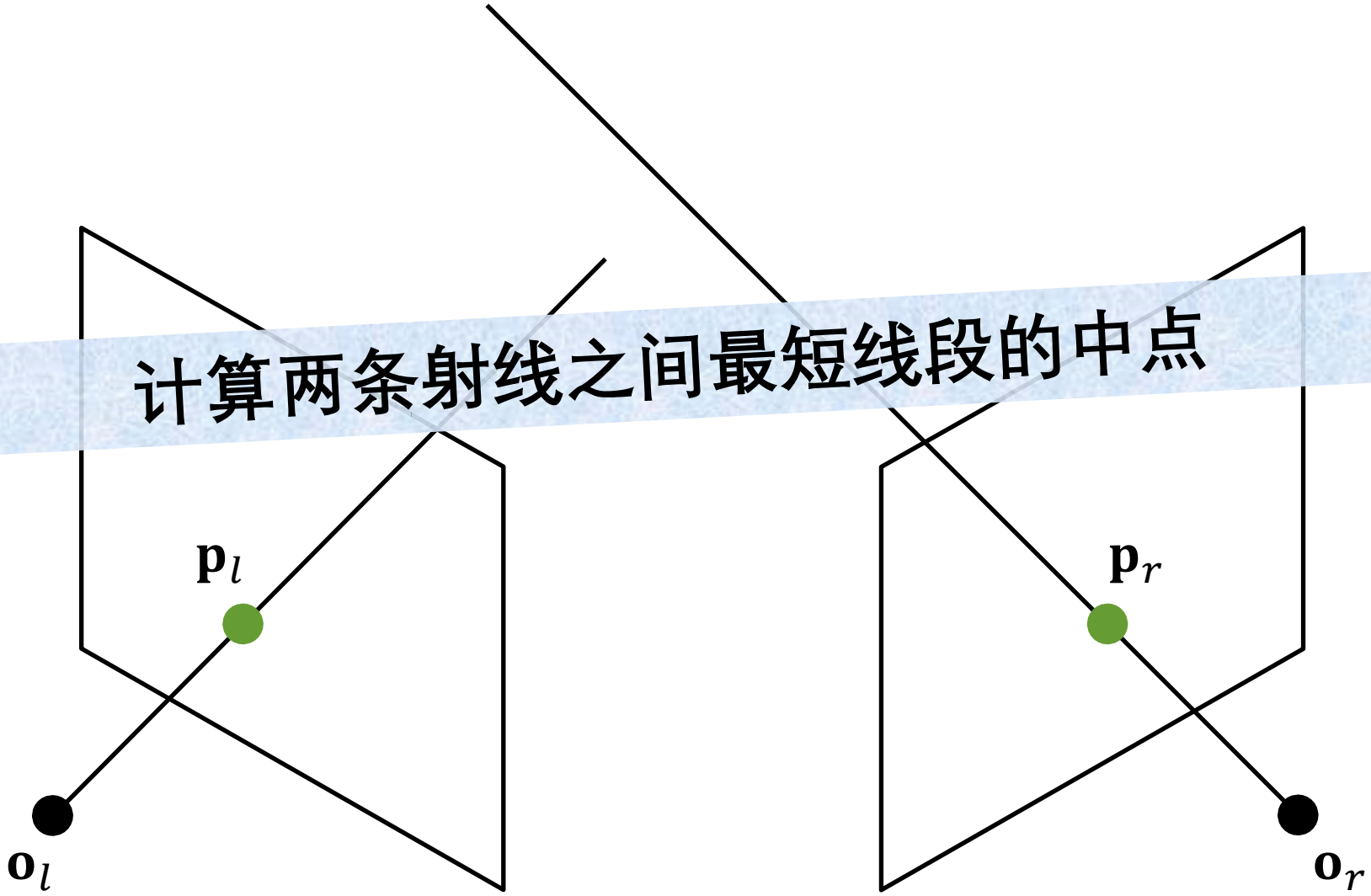

$\mathbf{p}_l$
$\mathbf{o}_l$
$\mathbf{p}_r$
$\mathbf{o}_r$

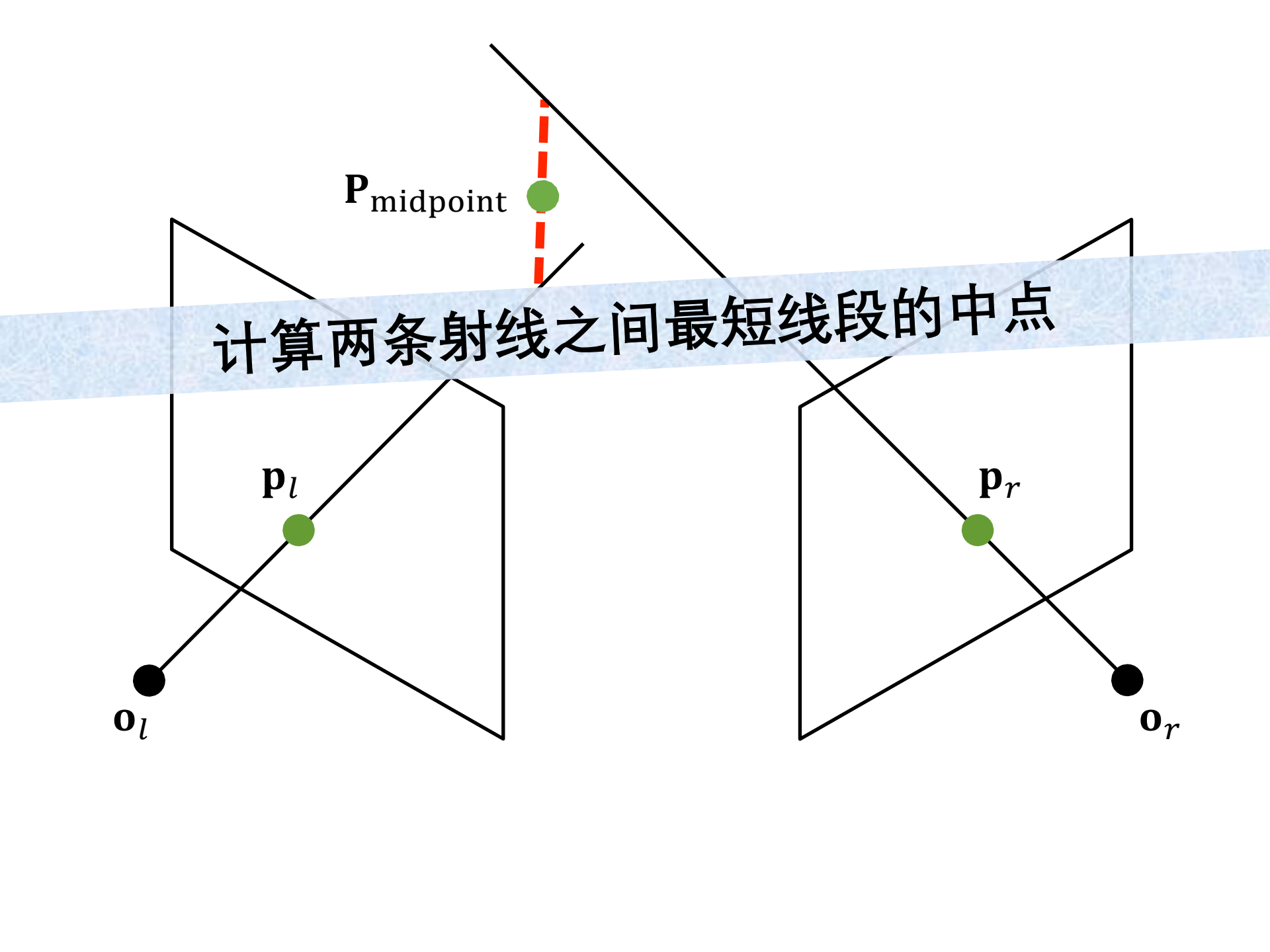

$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$

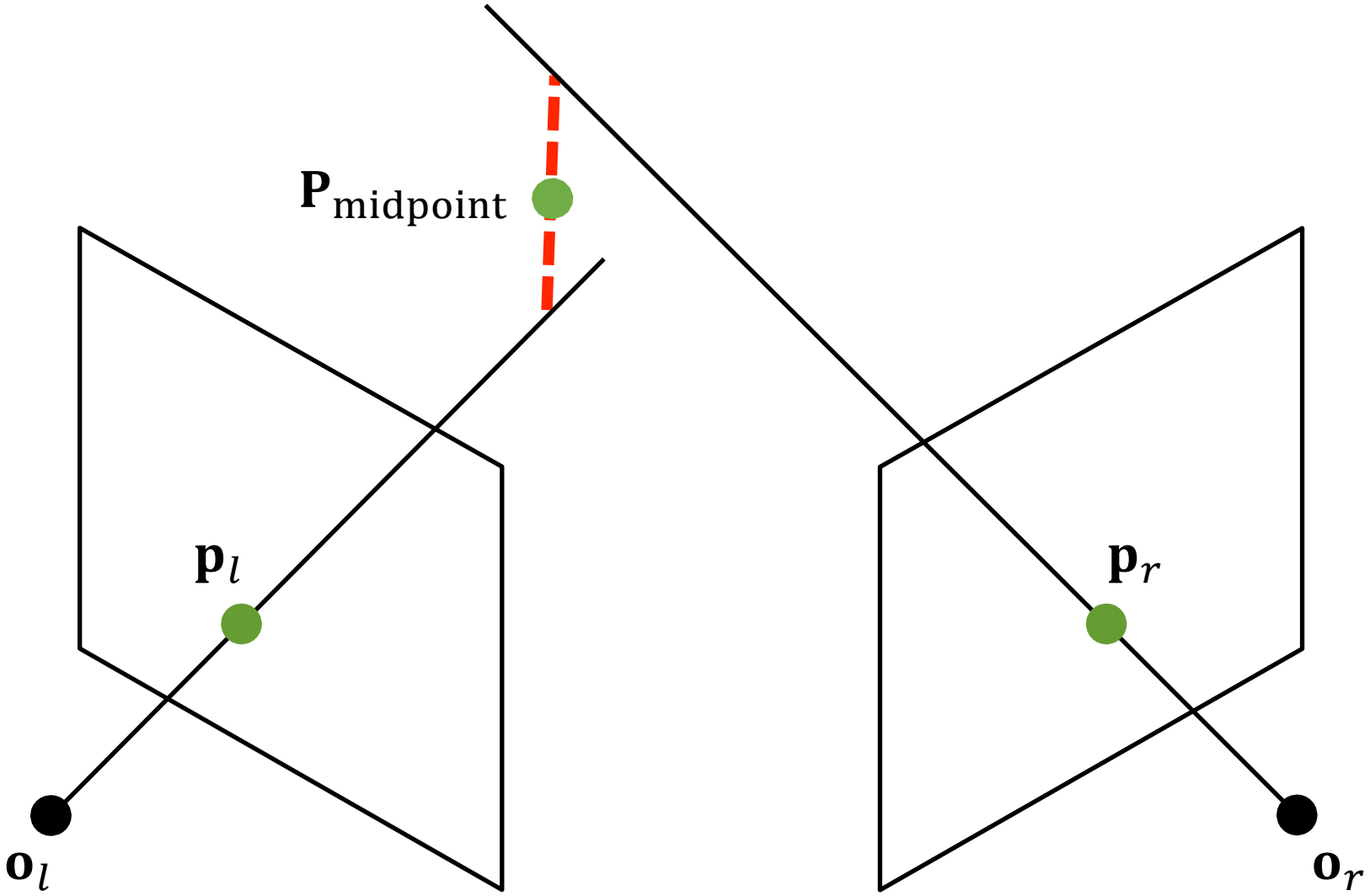
由于噪声导致匹配光线可能不相交

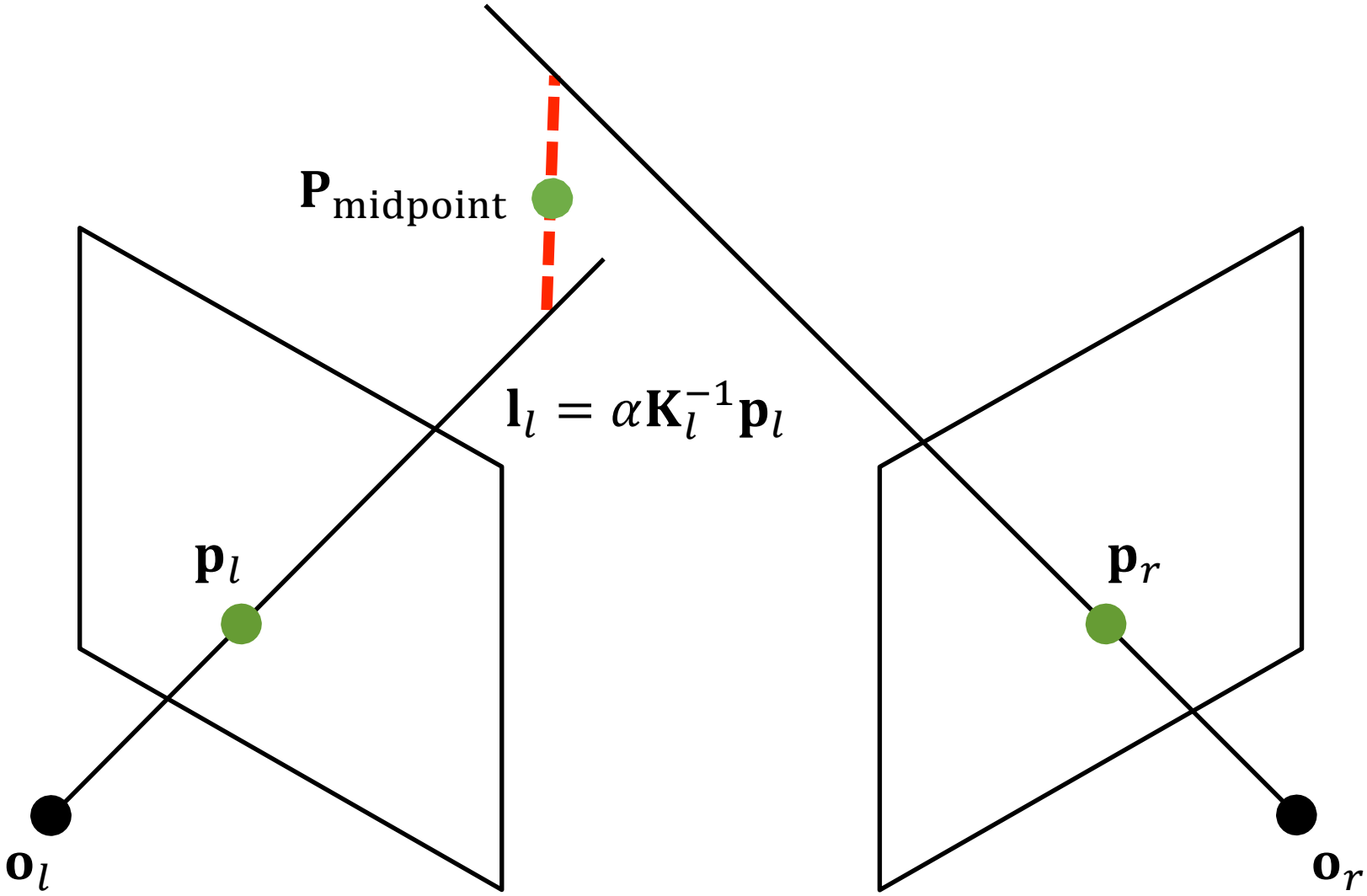


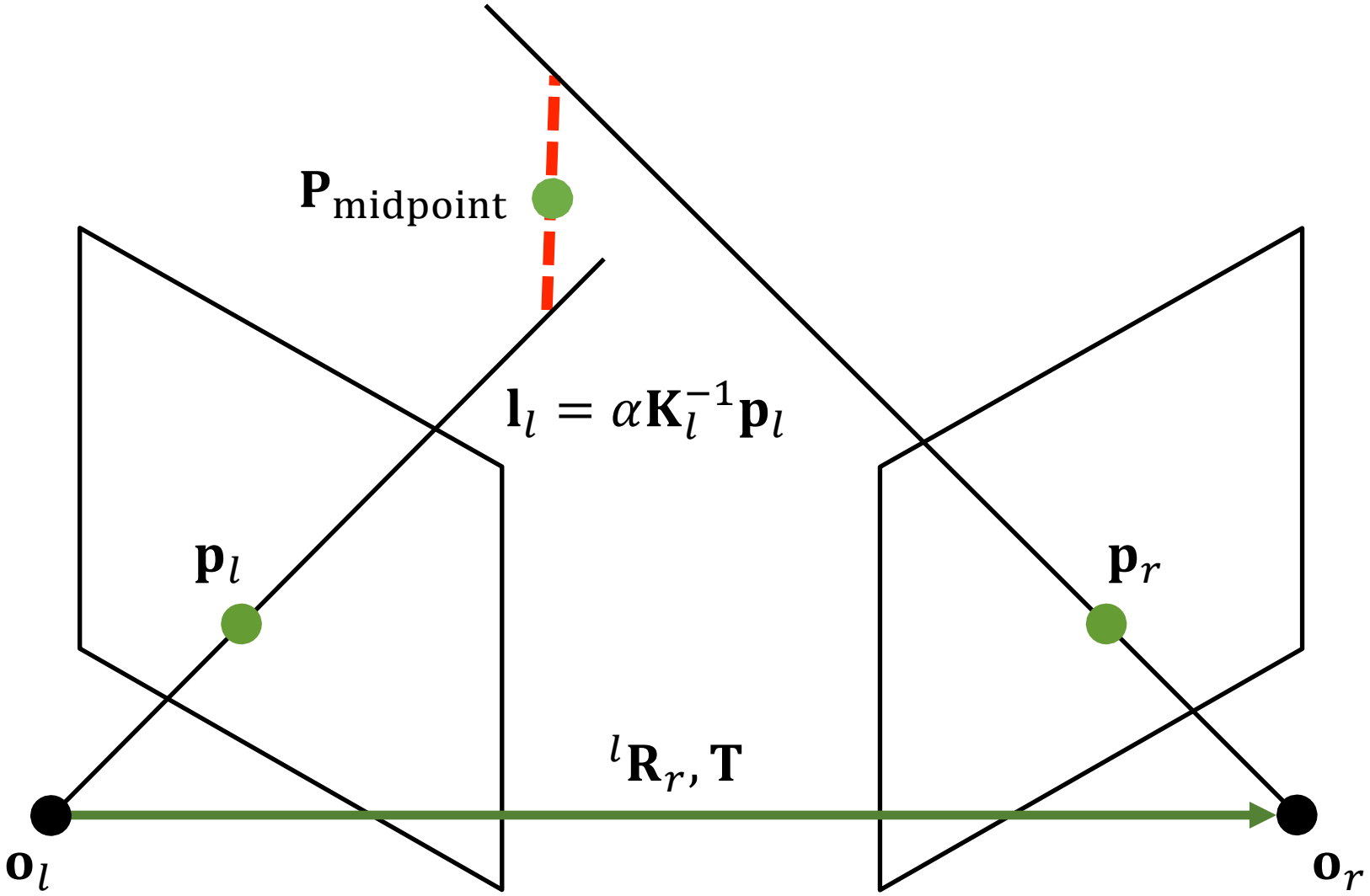
计算两条射线之间最短线段的中点

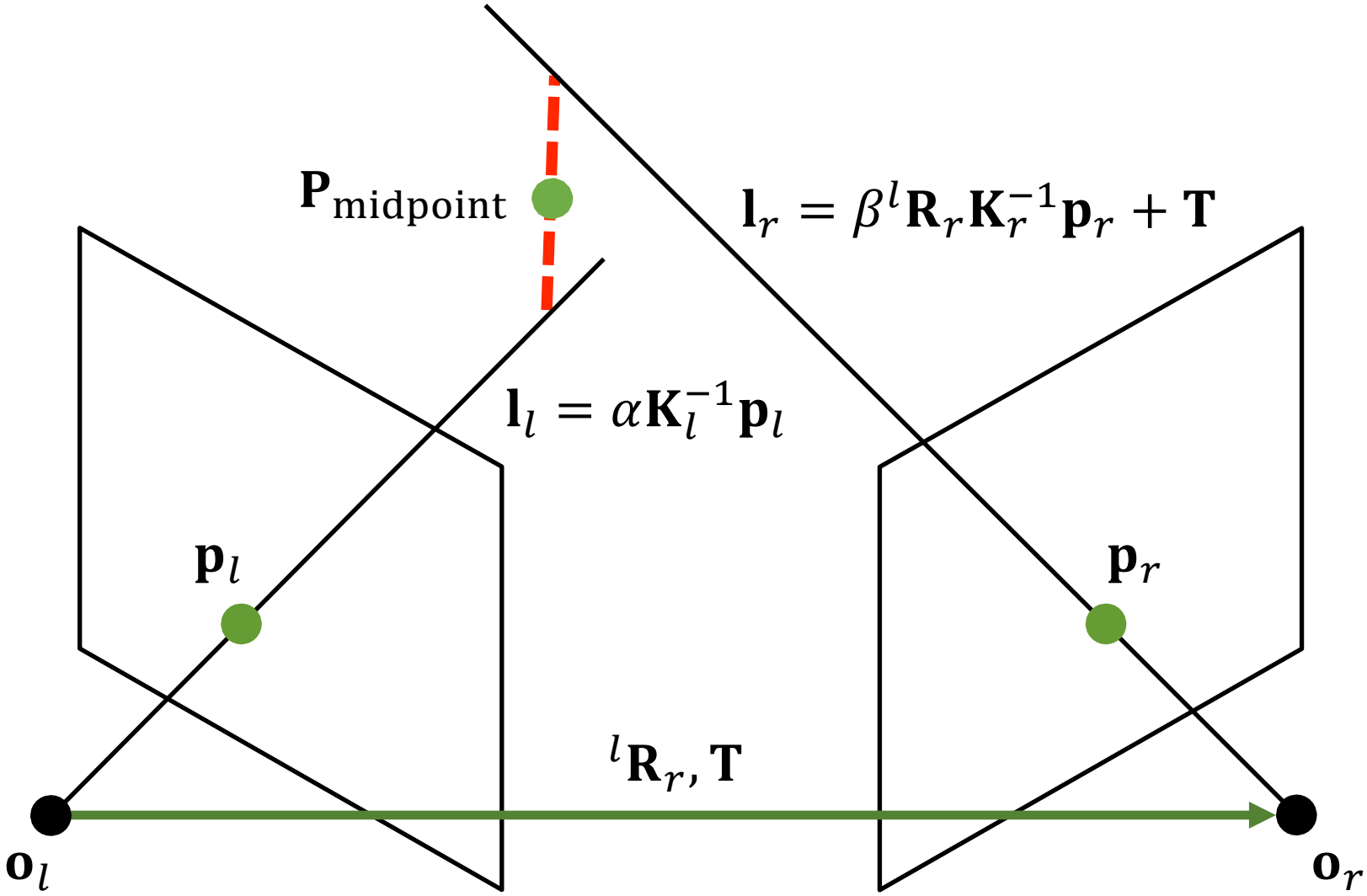


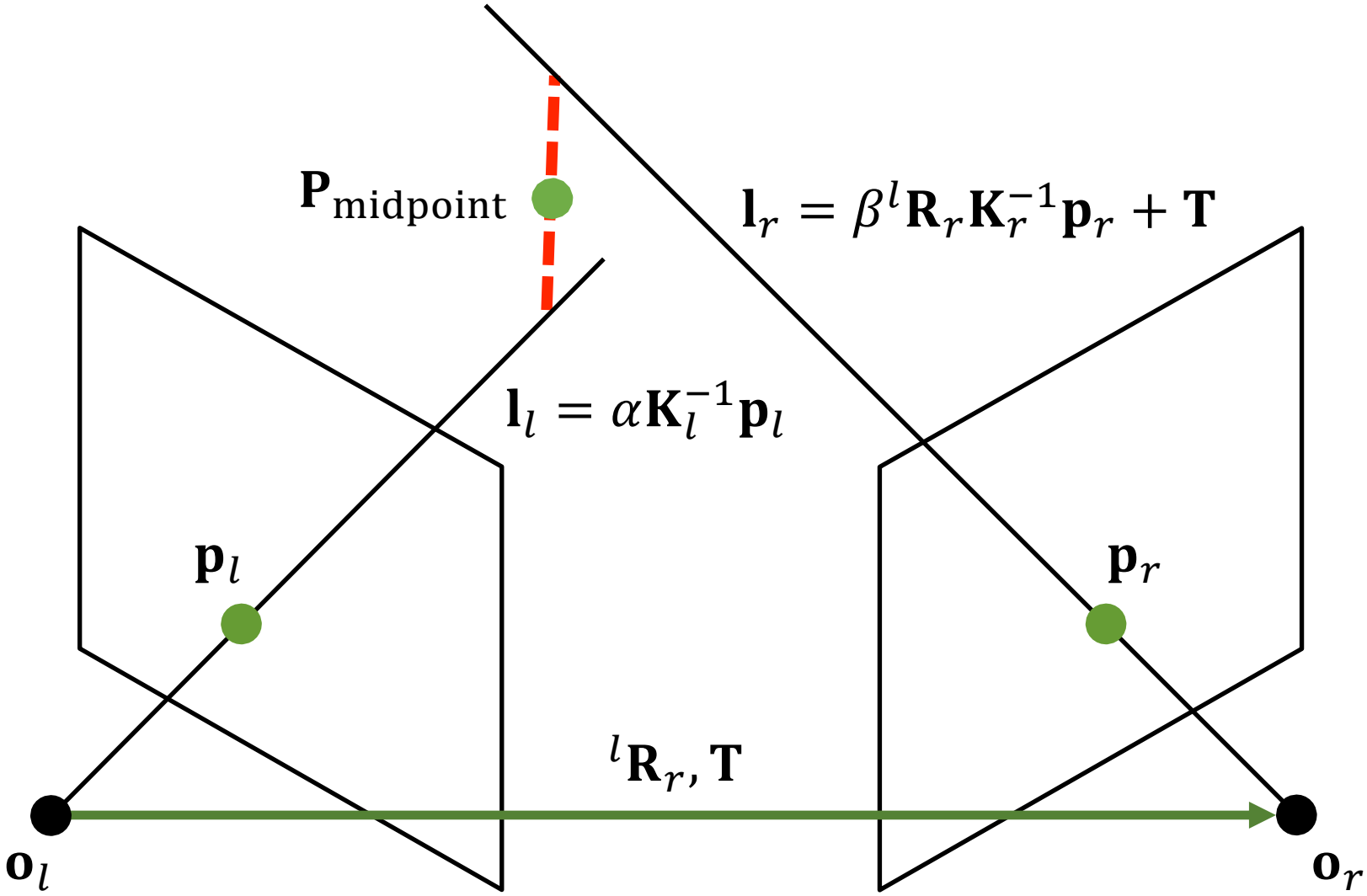

$\mathbf{P}_{\text{midpoint}}$
计算两条射线之间最短线段的中点
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$


$\mathbf{P}_{\text{midpoint}}$
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$


$\mathbf{P}_{\text{midpoint}}$
$\mathbf{l}_l = \alpha \mathbf{K}_l^{-1} \mathbf{p}_l$
$\mathbf{p}_l$
$\mathbf{p}_r$
$\mathbf{o}_l$
$\mathbf{o}_r$


$\mathbf{P}_{\text{midpoint}}$
$\mathbf{l}_l = \alpha \mathbf{K}_l^{-1} \mathbf{p}_l$
$\mathbf{p}_l$
$\mathbf{p}_r$
${}^l\mathbf{R}_r, \mathbf{T}$
$\mathbf{o}_l$
$\mathbf{o}_r$


$\mathbf{P}_{\text{midpoint}}$
$\mathbf{l}_r = \beta^l \mathbf{R}_r \mathbf{K}_r^{-1} \mathbf{p}_r + \mathbf{T}$
$\mathbf{l}_l = \alpha \mathbf{K}_l^{-1} \mathbf{p}_l$
$\mathbf{p}_l$
$\mathbf{p}_r$
${}^l\mathbf{R}_r, \mathbf{T}$
$\mathbf{o}_l$
$\mathbf{o}_r$



$$\arg\min_{\alpha,\beta} \left\| \alpha \mathbf{K}_l^{-1} \mathbf{p}_l - \left(\beta^l \mathbf{R}_r \mathbf{K}_r^{-1} \mathbf{p}_r + \mathbf{T} \right) \right\|$$

立体匹配

基于块的方法

左视图

右视图

左视图

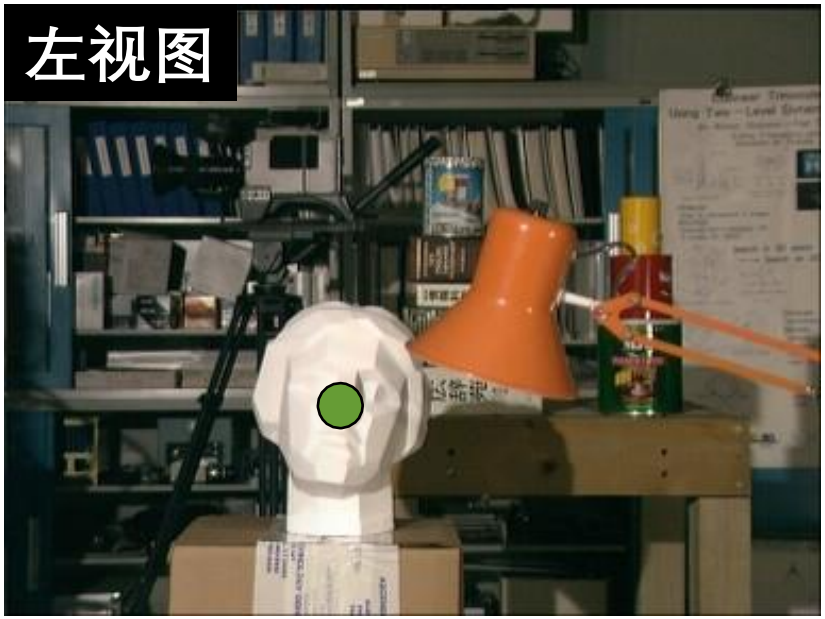
右视图

假设两台相机是平行的

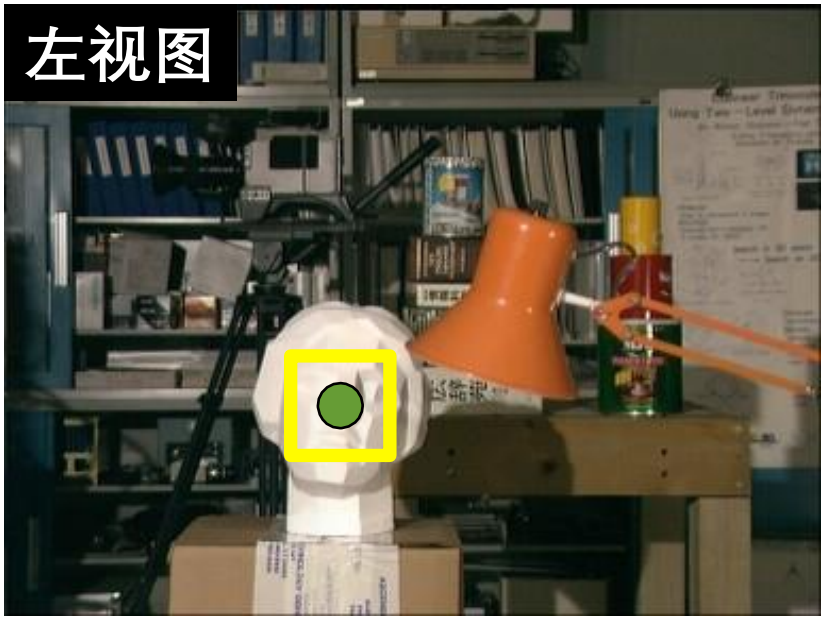
左视图

右视图

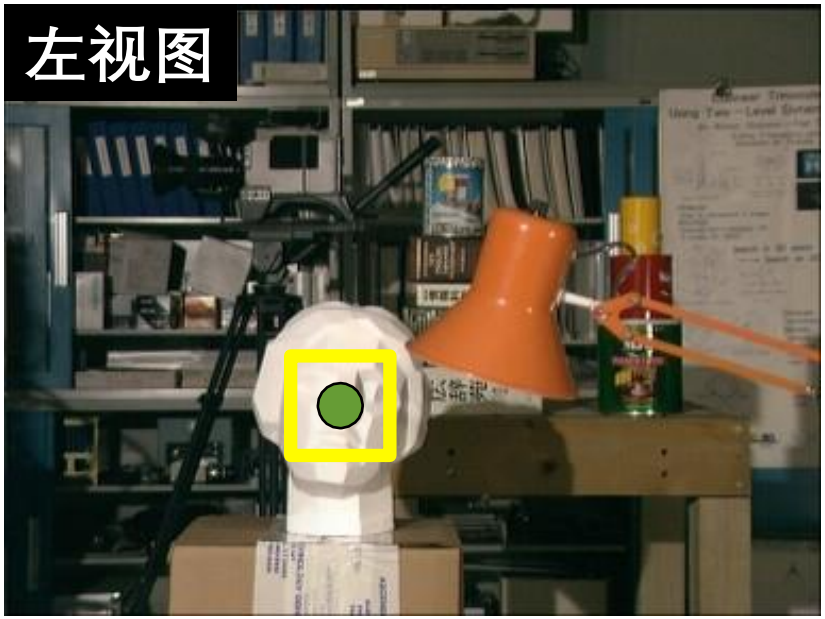
对于左视图中的每个点，
找到右视图中“最佳”匹配块



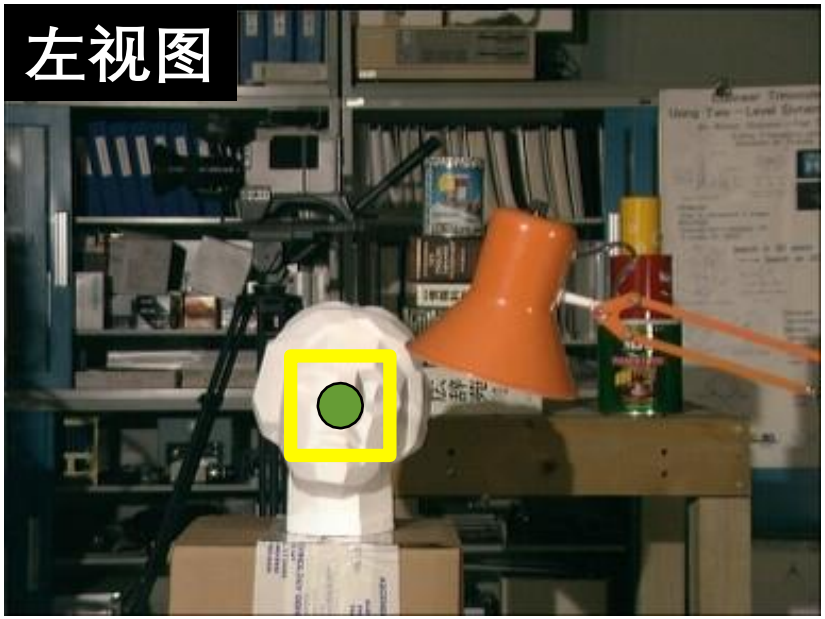
对于左视图中的每个点，
找到右视图中“最佳”匹配块



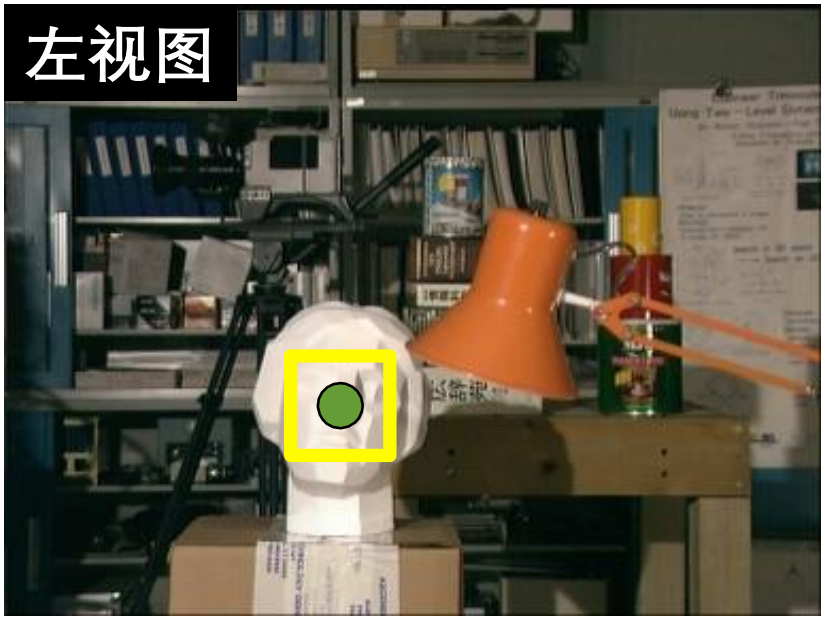
对于左视图中的每个点，
找到右视图中“最佳”匹配块



对于左视图中的每个点，
找到右视图中“最佳”匹配块

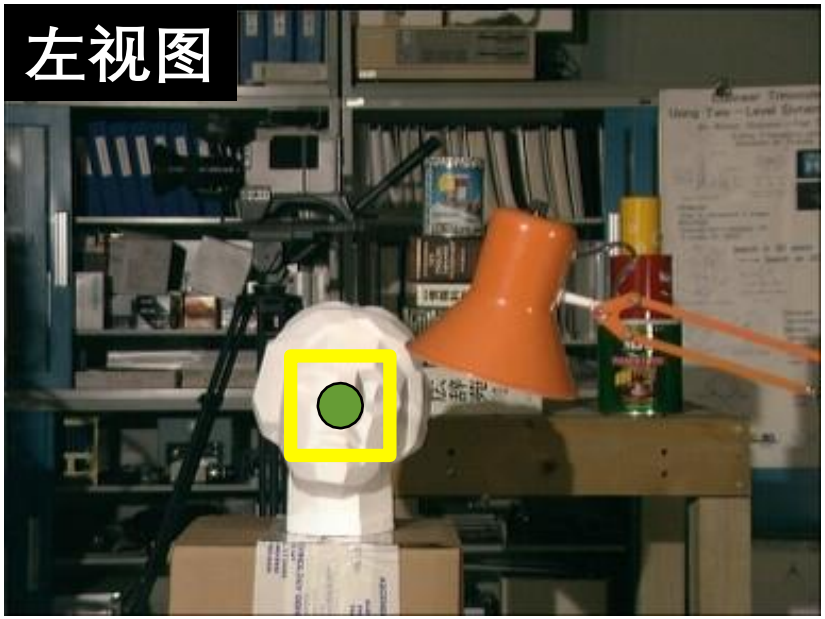

左视图


右视图



匹配代价
(SAD)

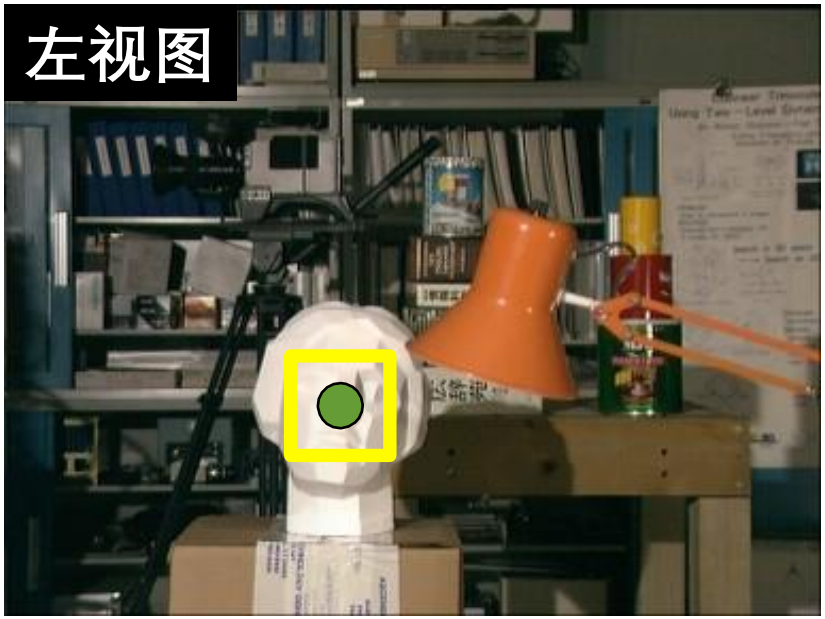
视差



匹配代价
(SAD)

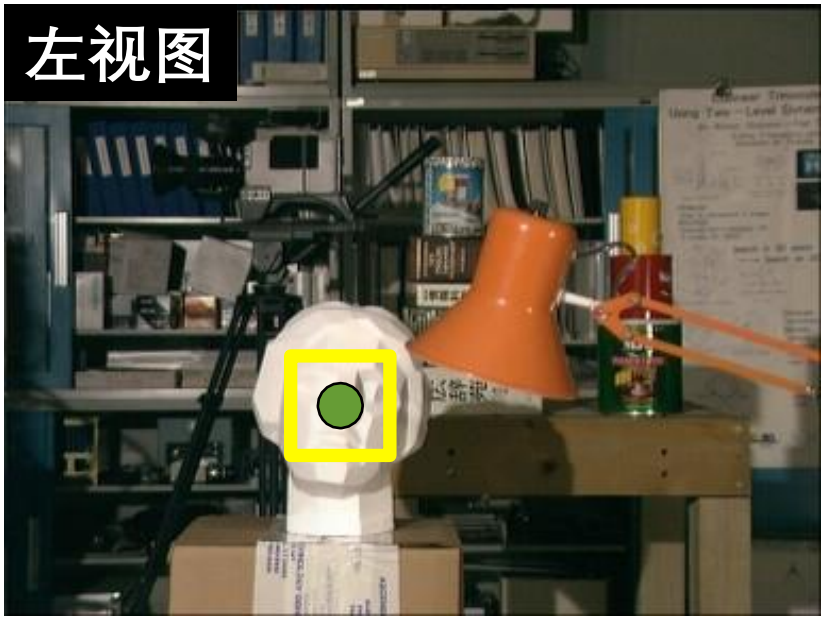
绝对误差和

视差

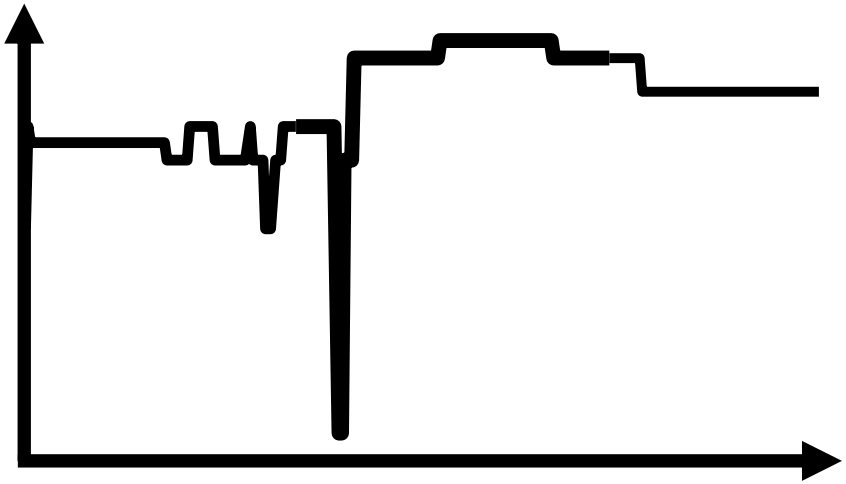


匹配代价
(SAD)

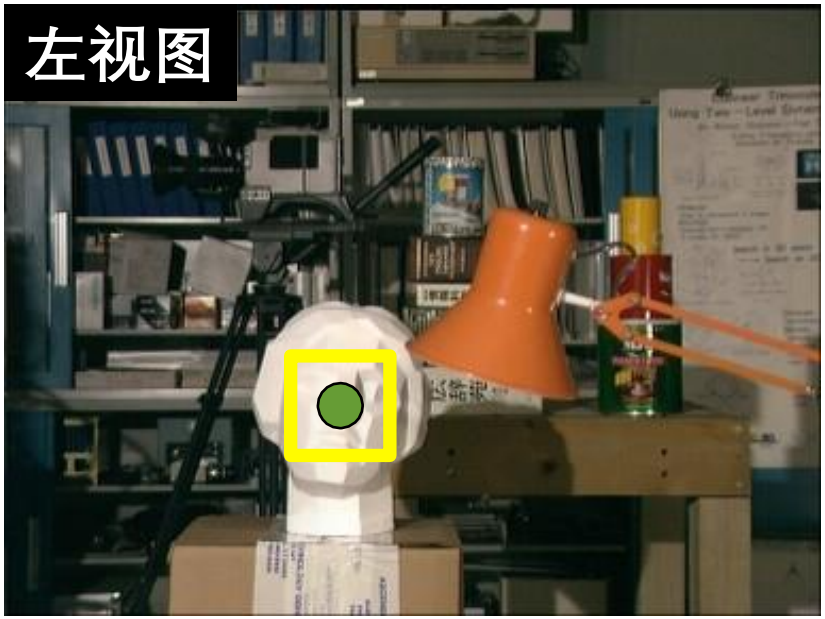
视差



匹配代价
(SAD)

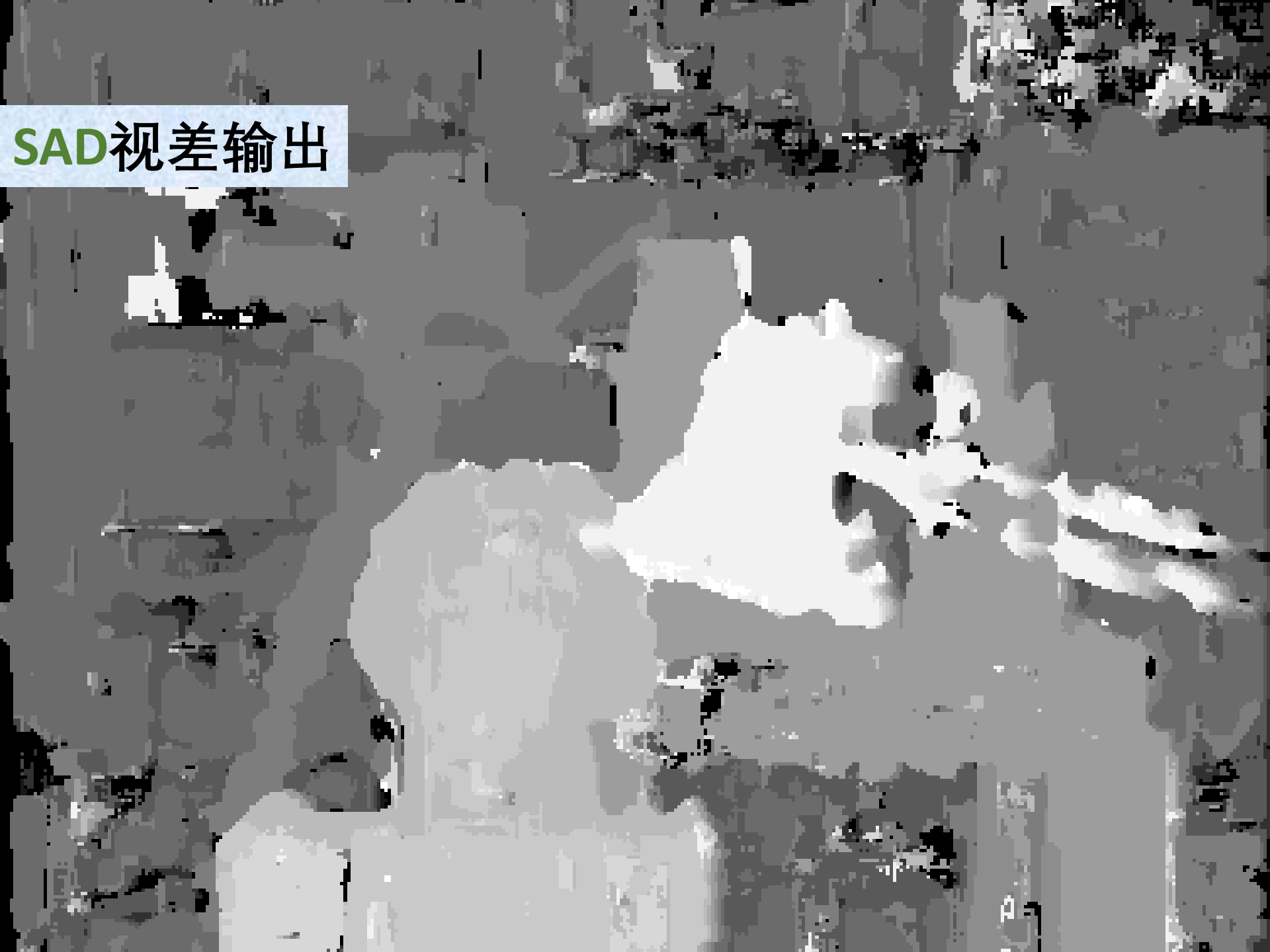


视差



匹配代价
(SAD)

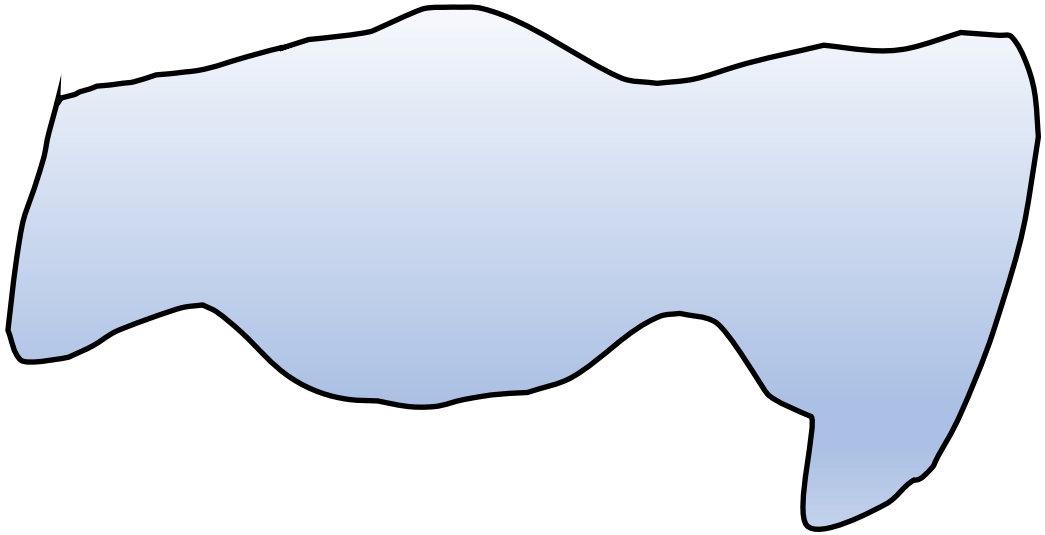
视差

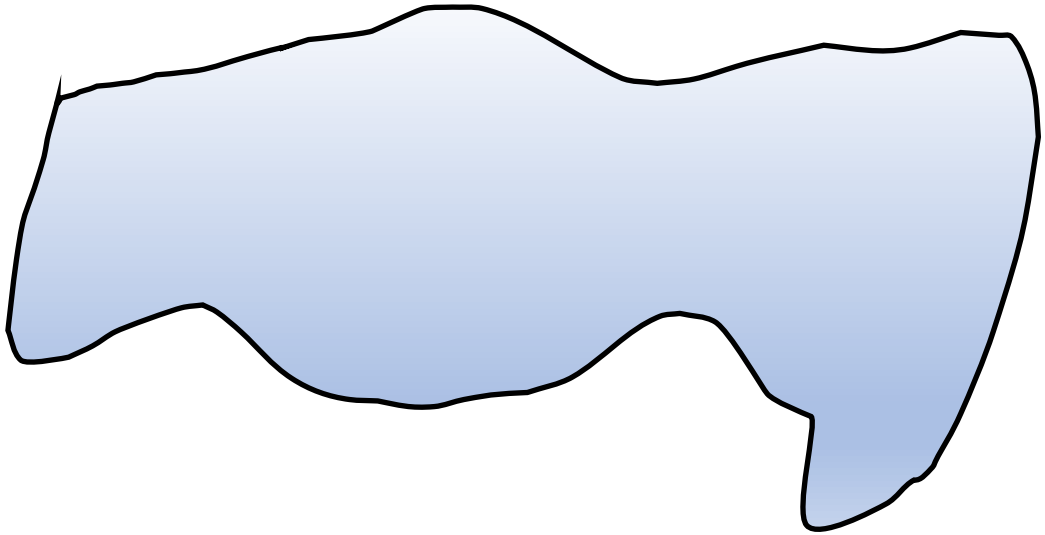

SAD视差输出

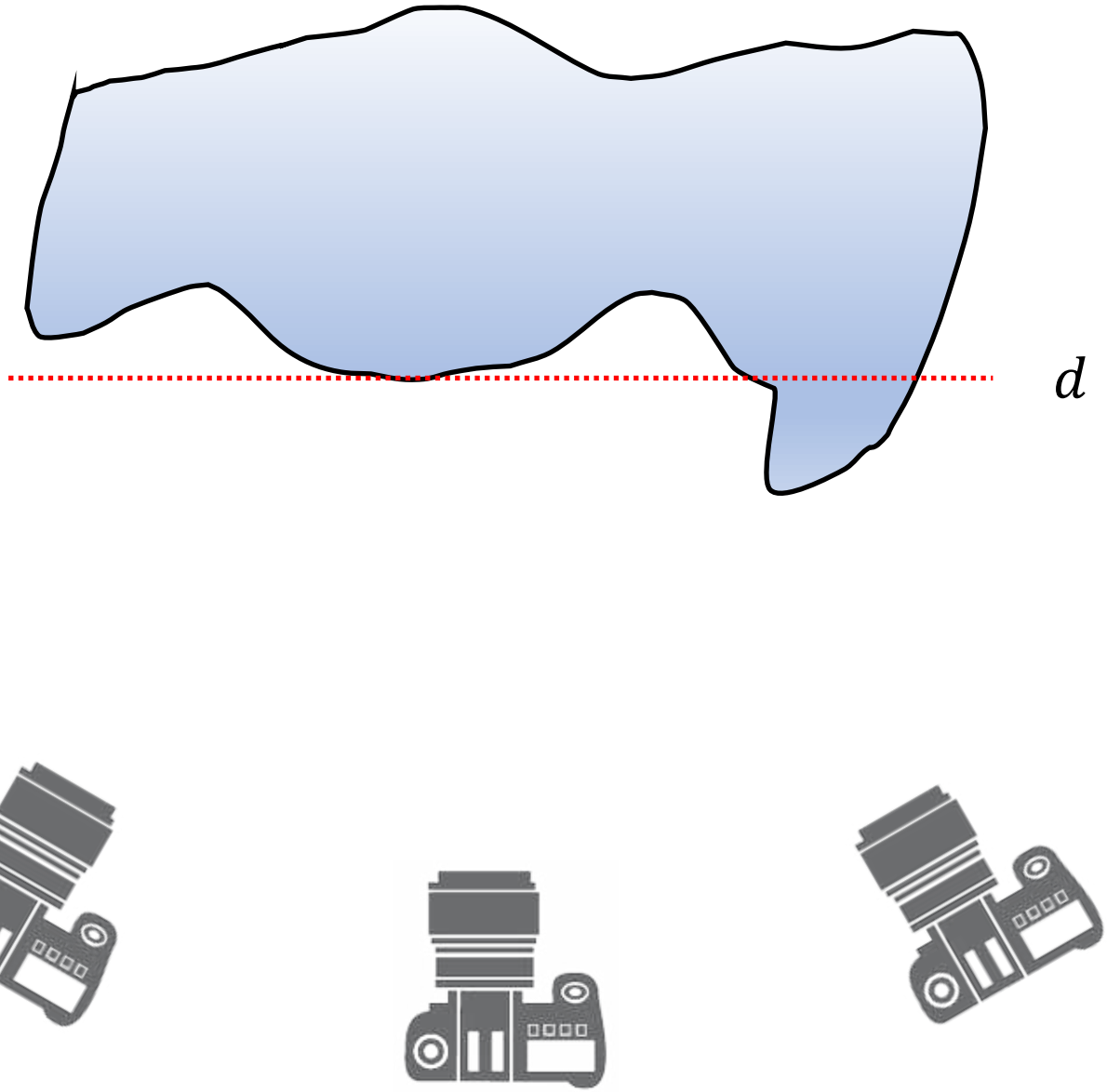

图割视差输出

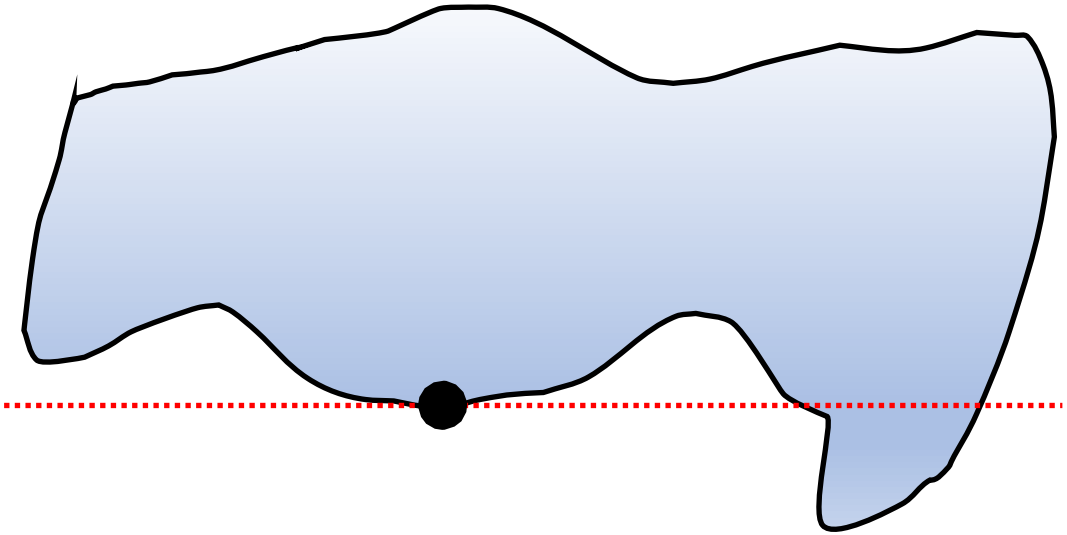
平面扫描

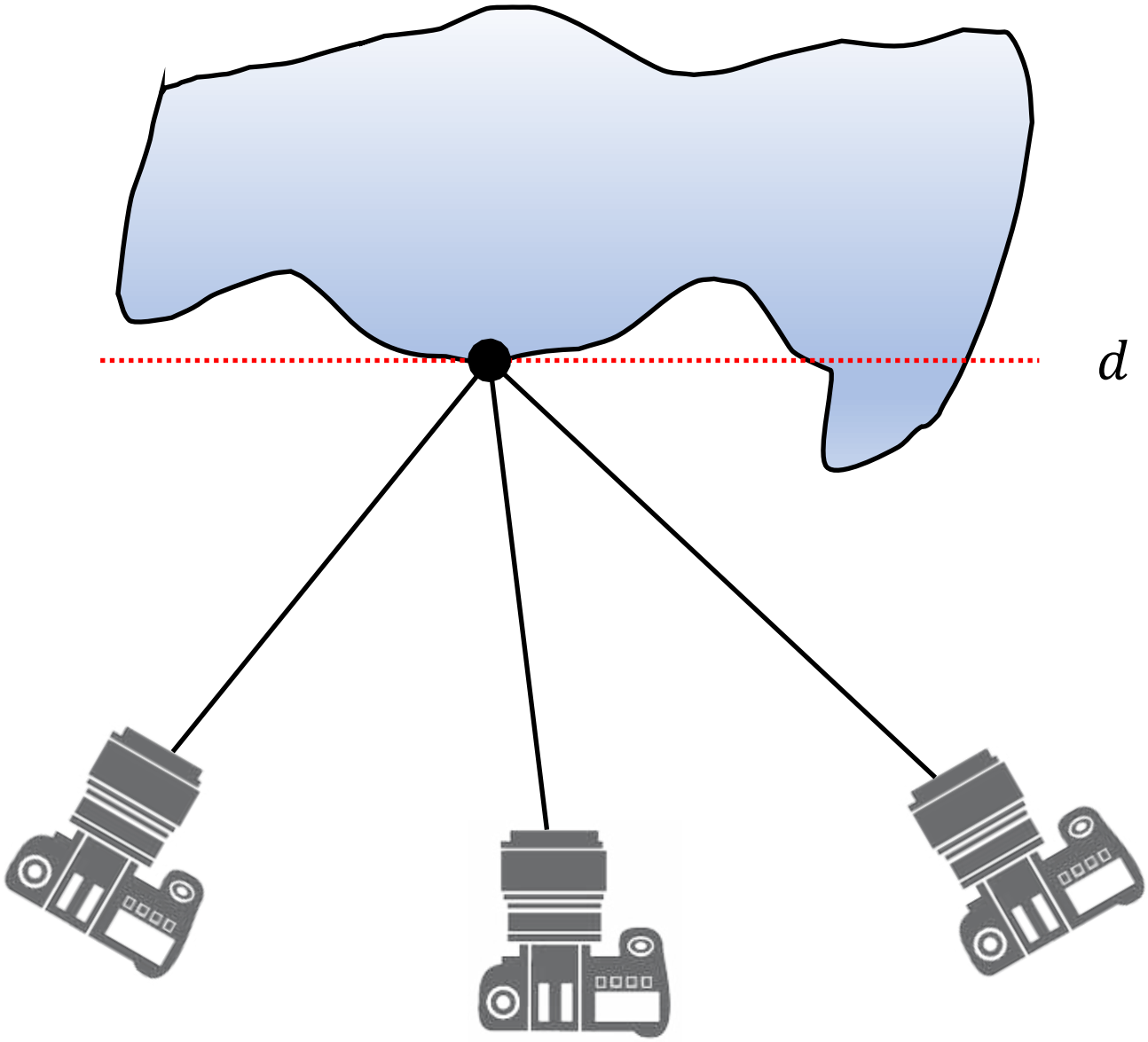
基于单应变换的方法

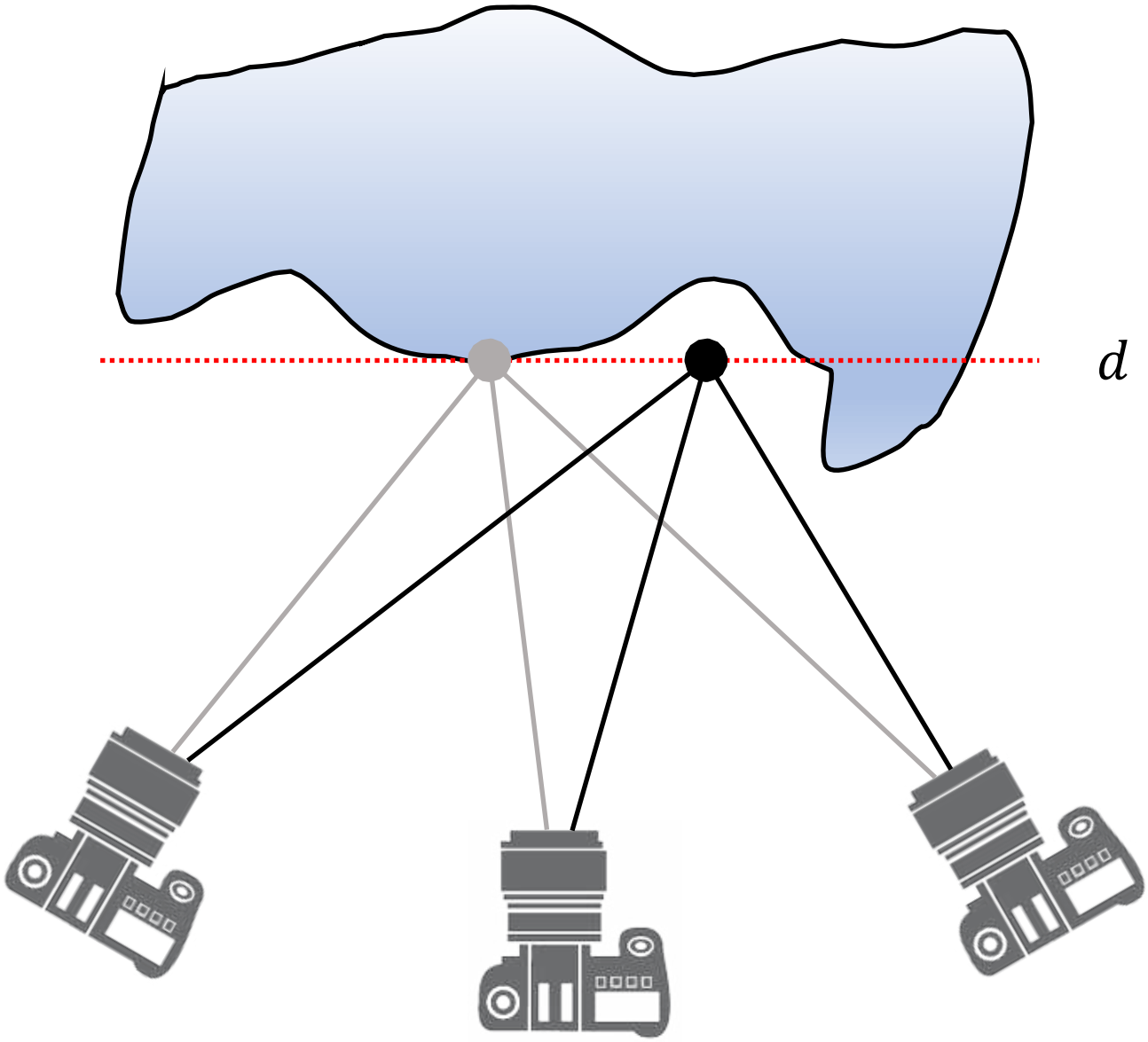


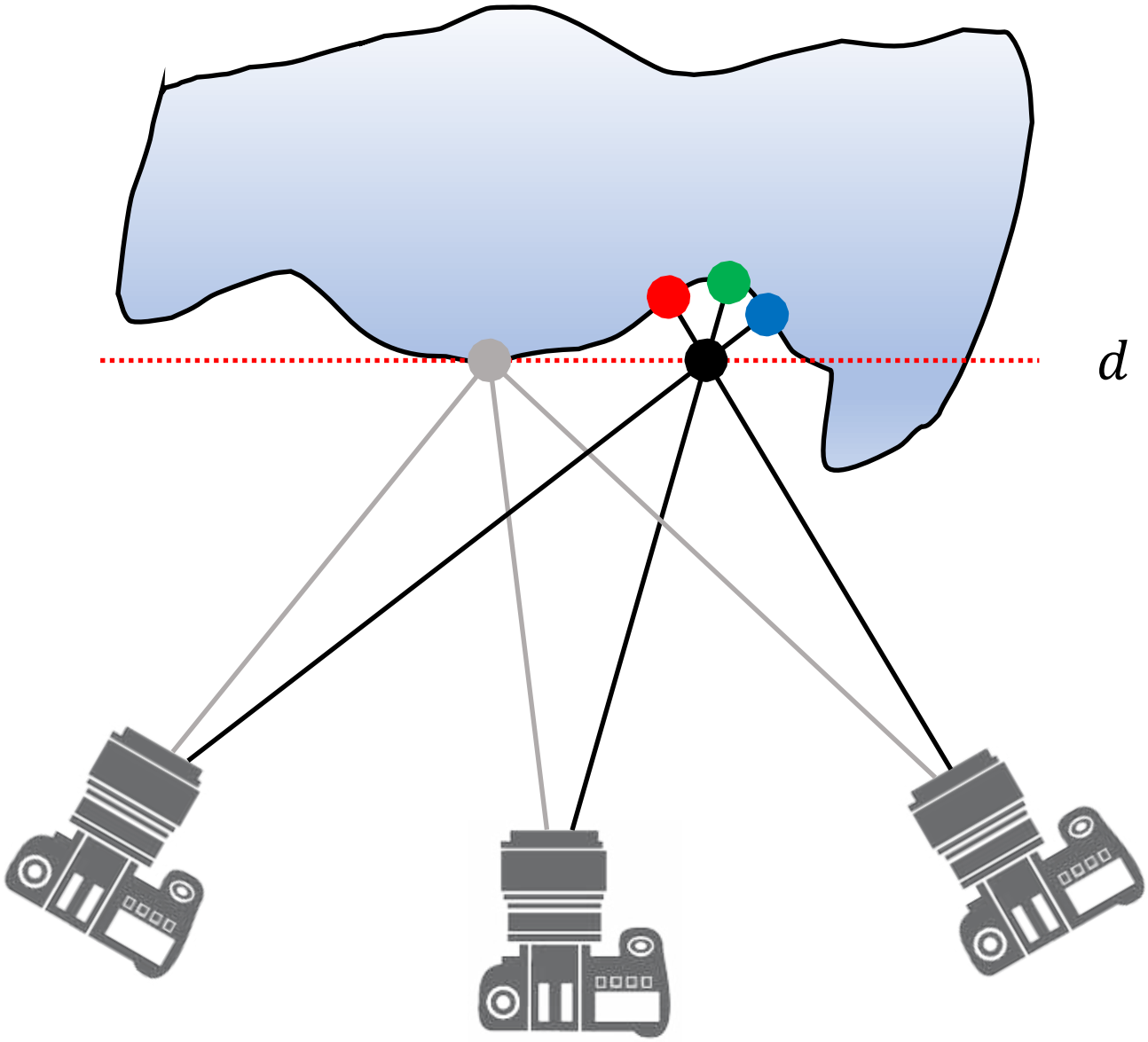


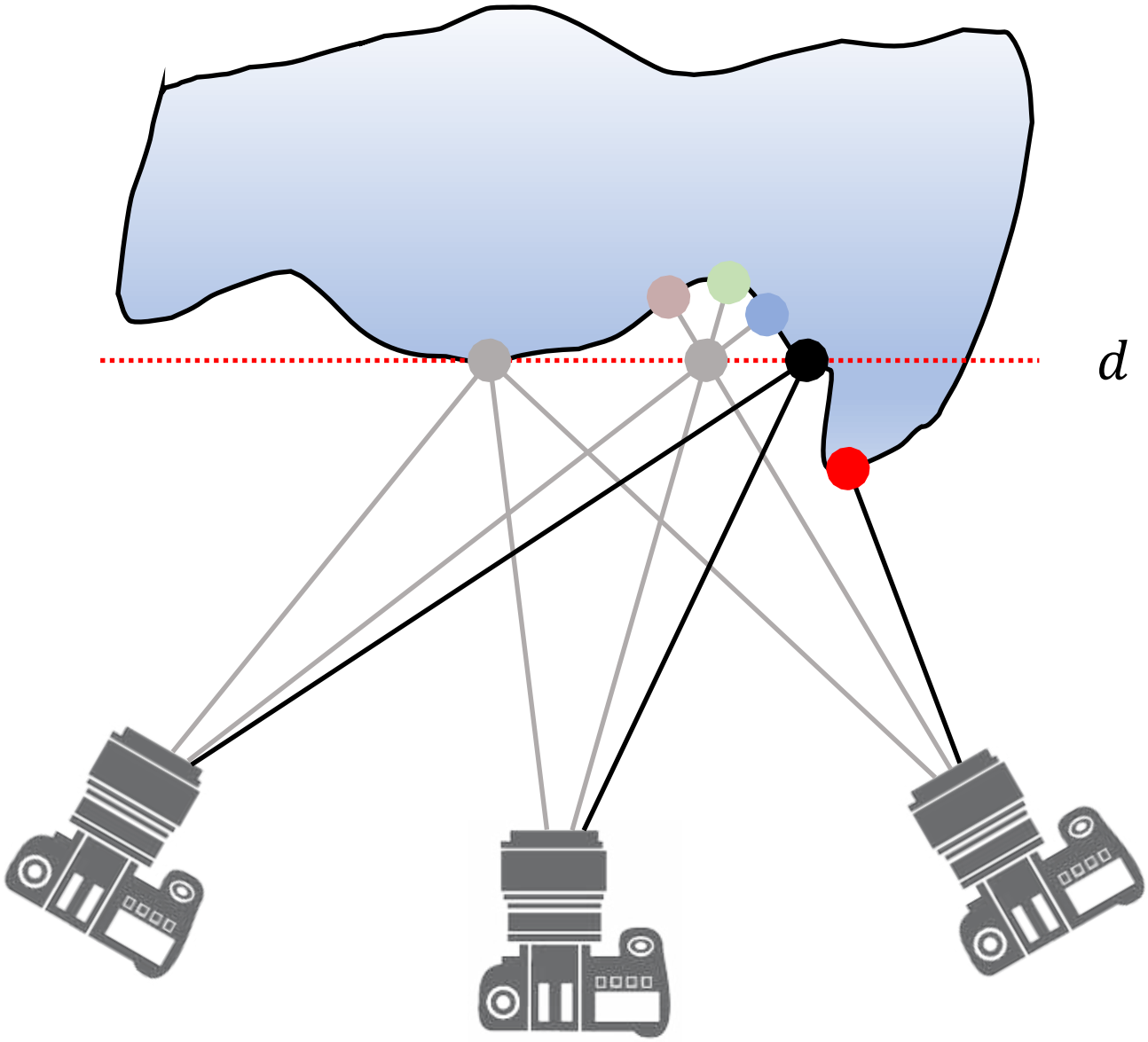

d

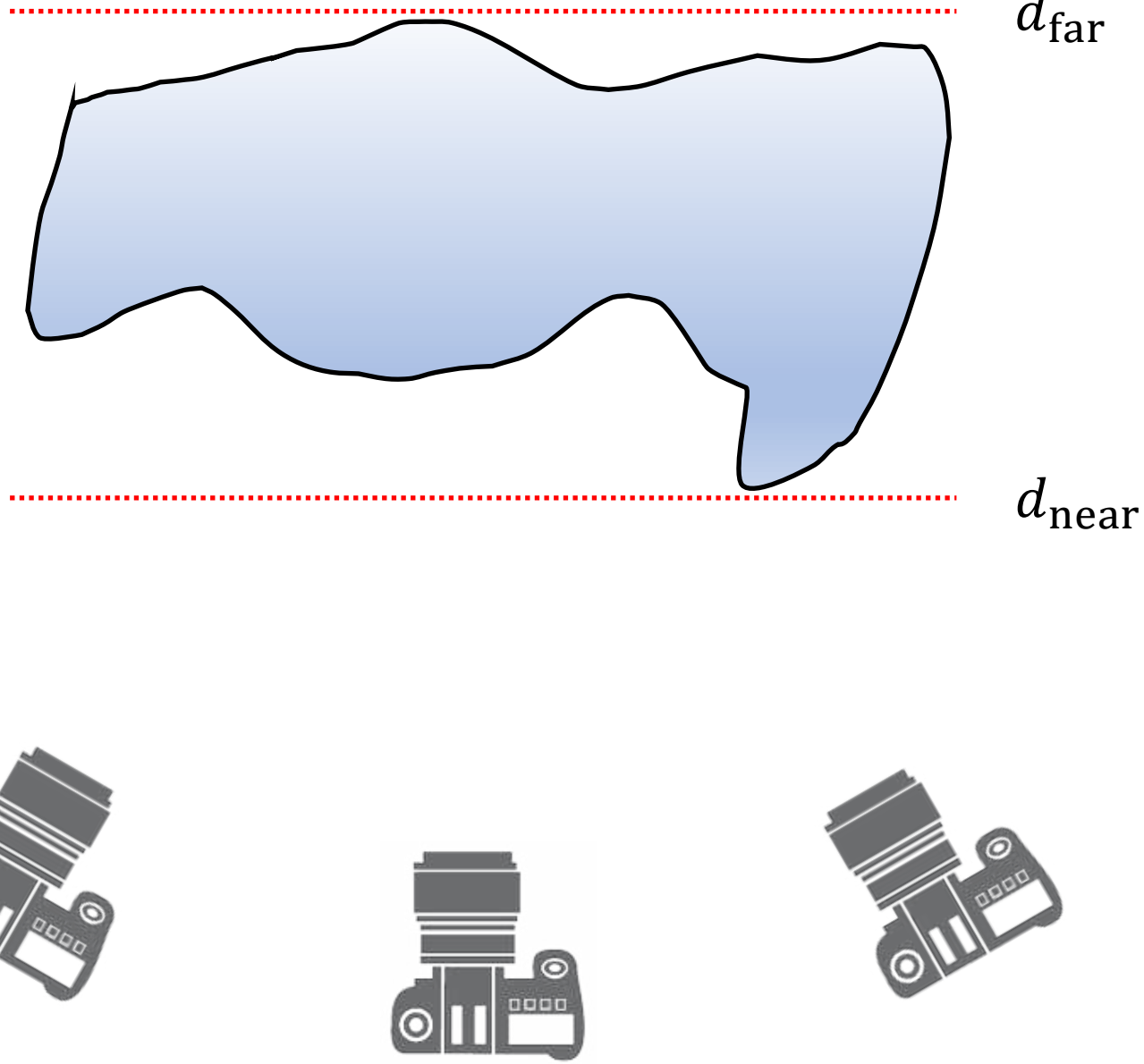

d


d

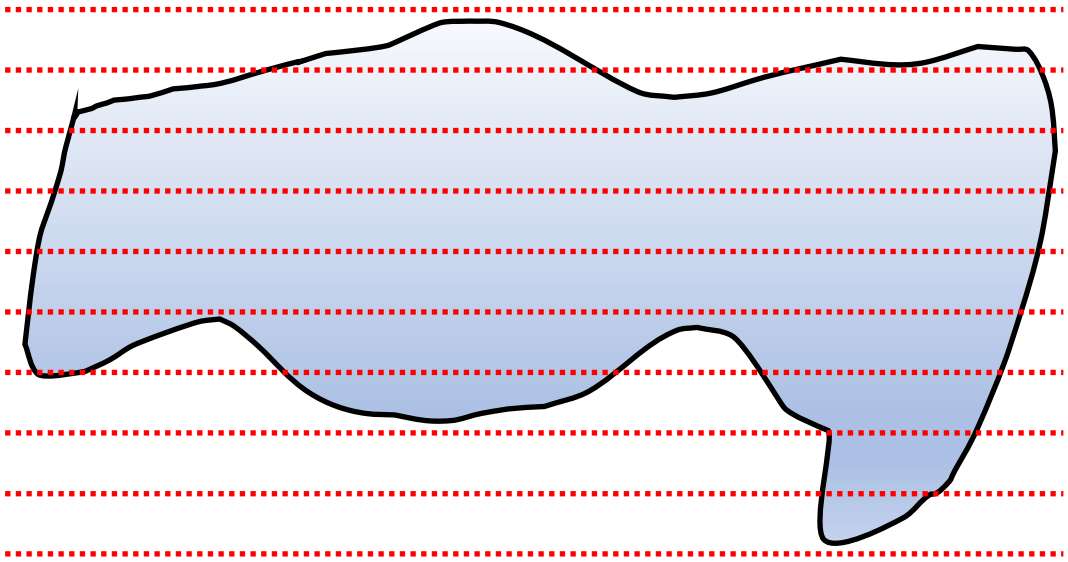

d


d


d


d_{far}
d_{near}

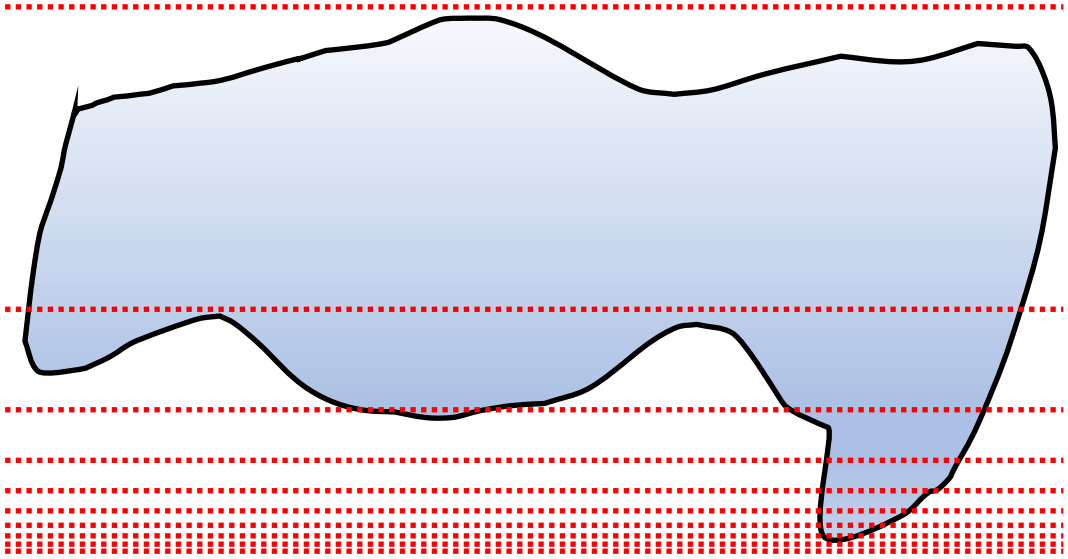
等深度

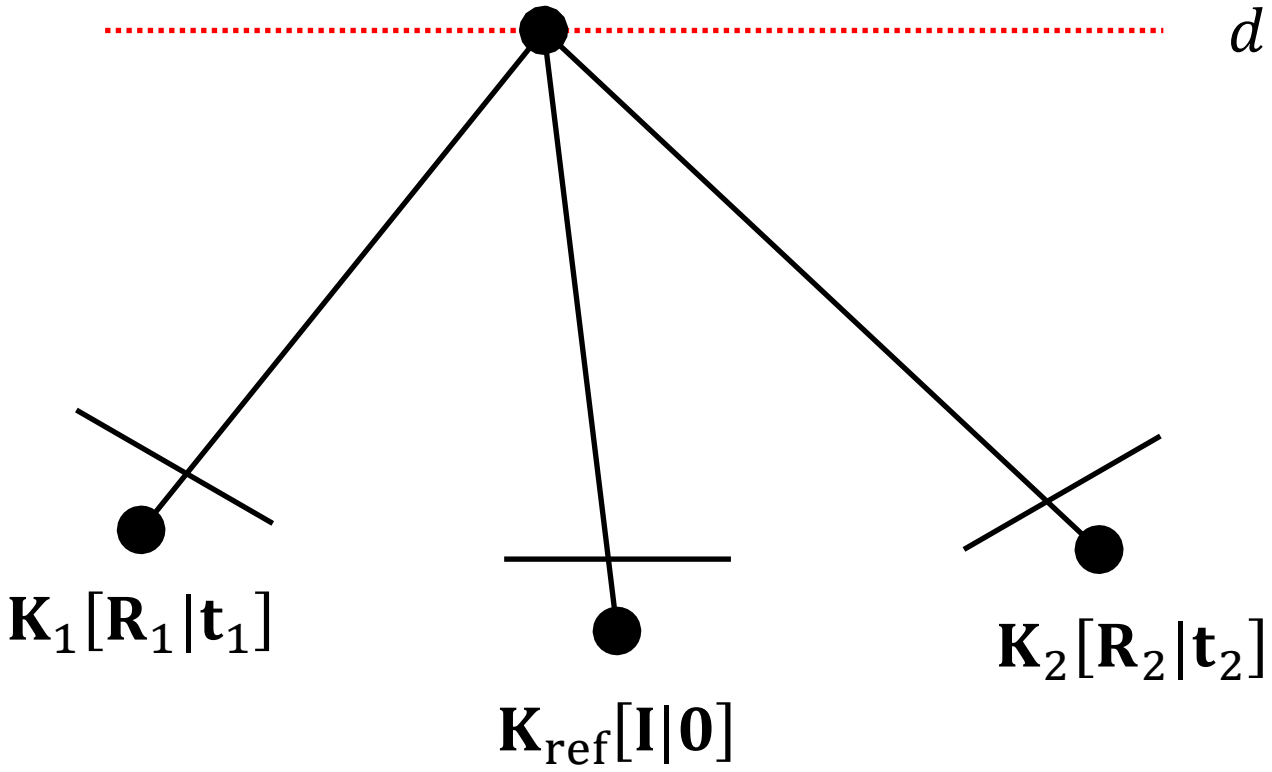


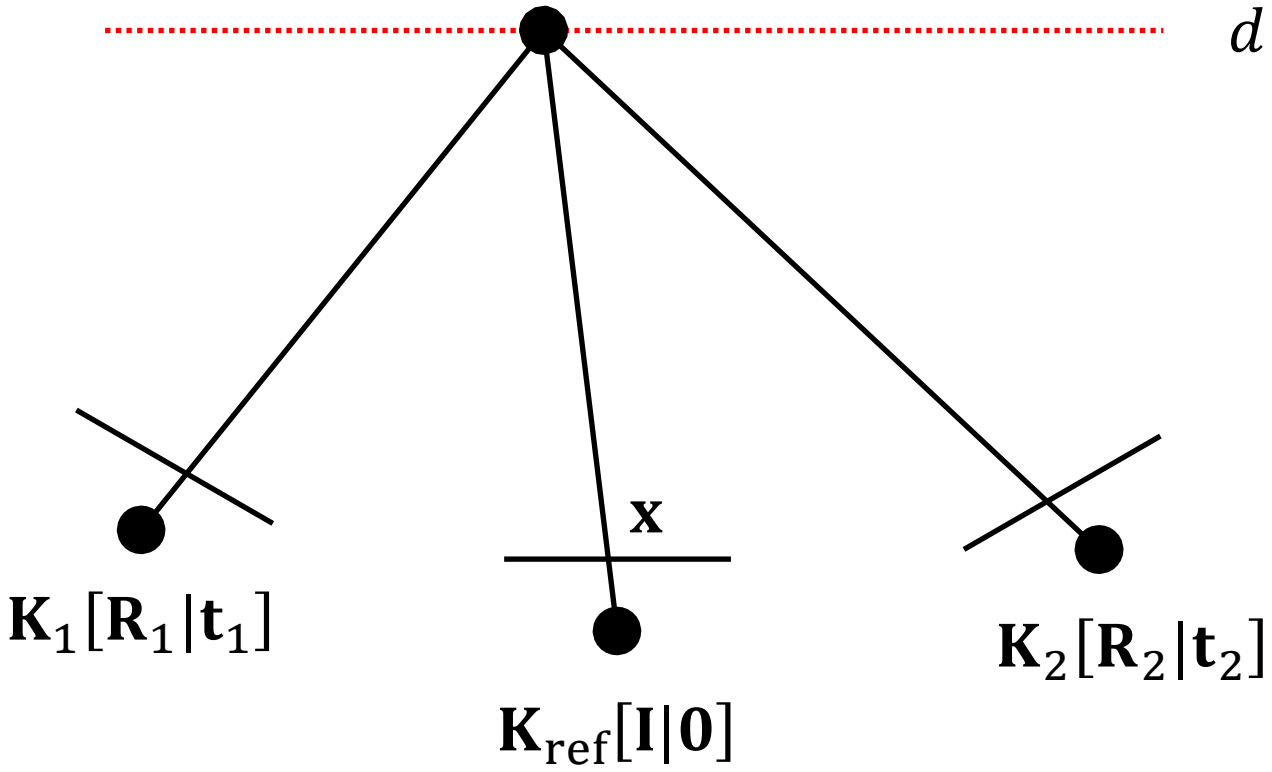
d_{far}

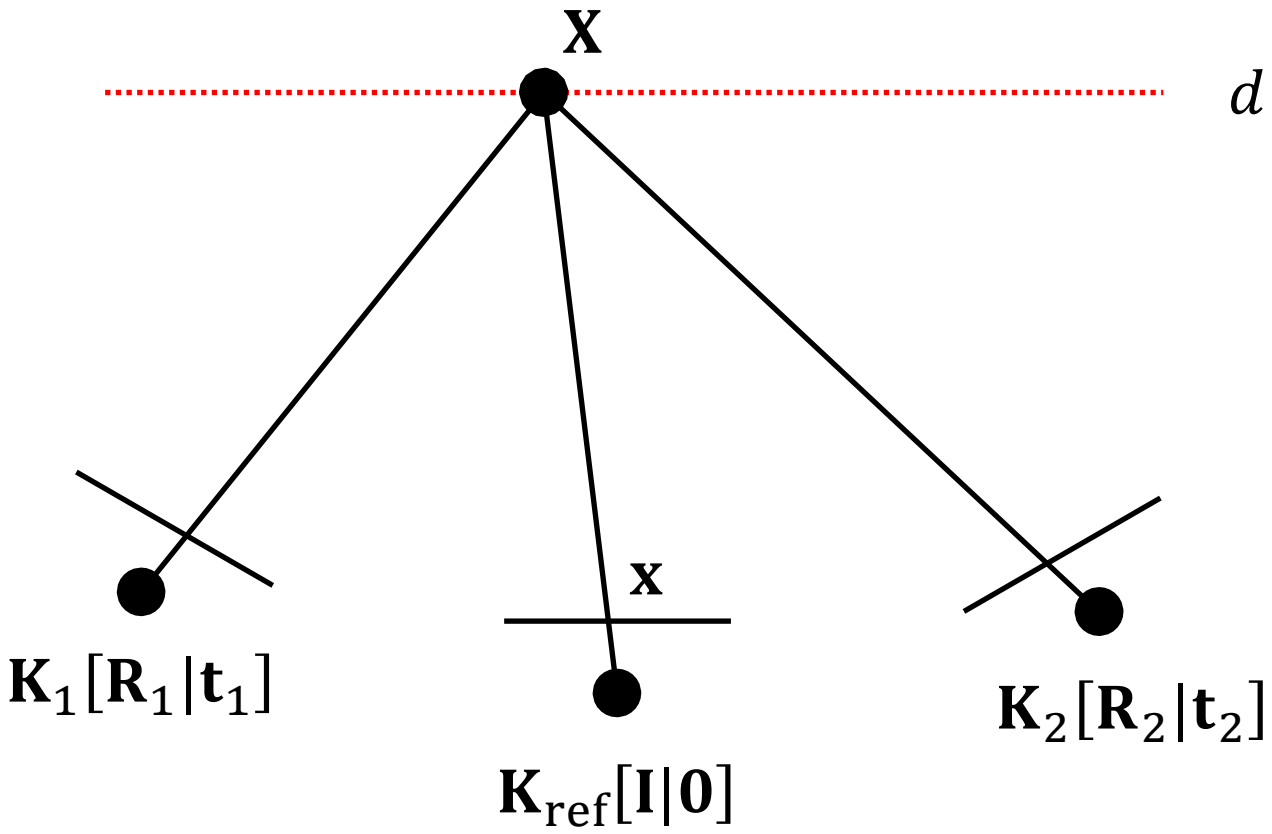
d_{near}

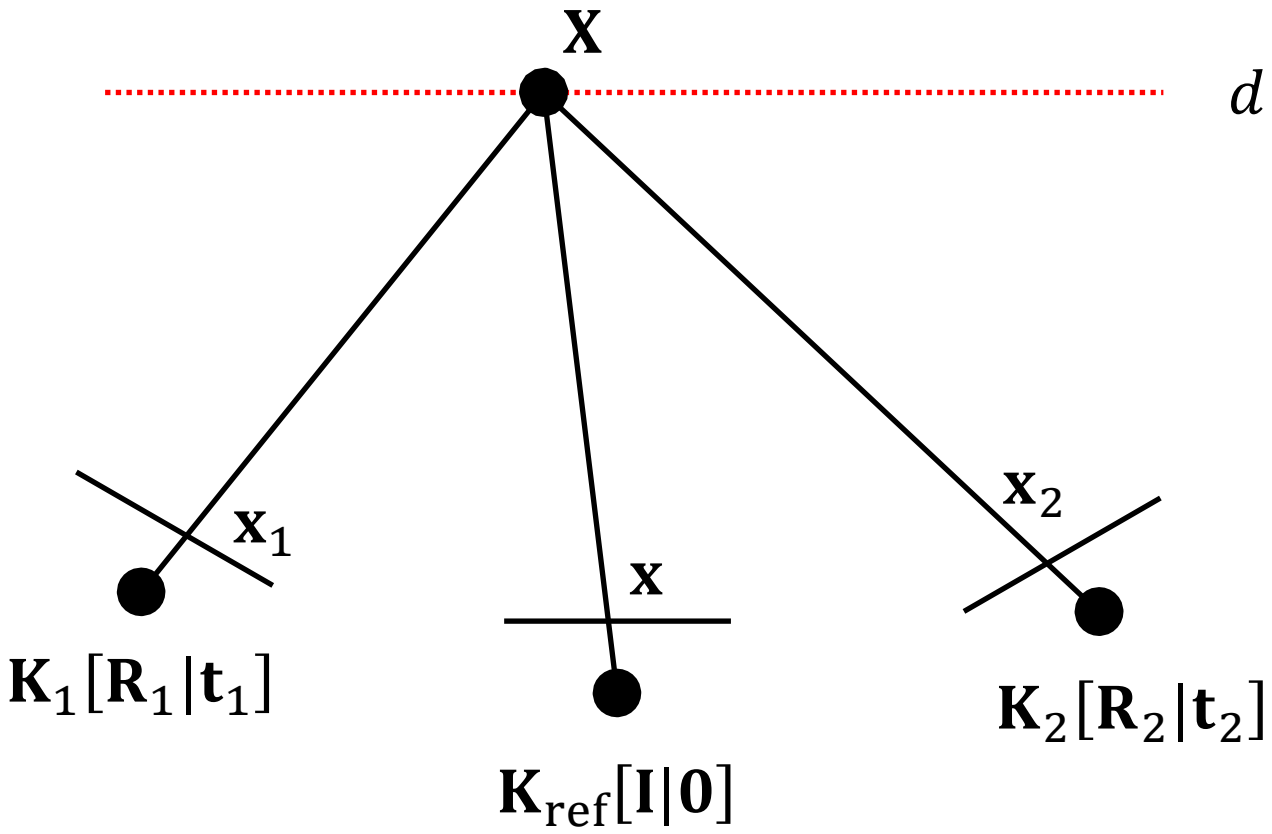
等视差




d
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\text{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$

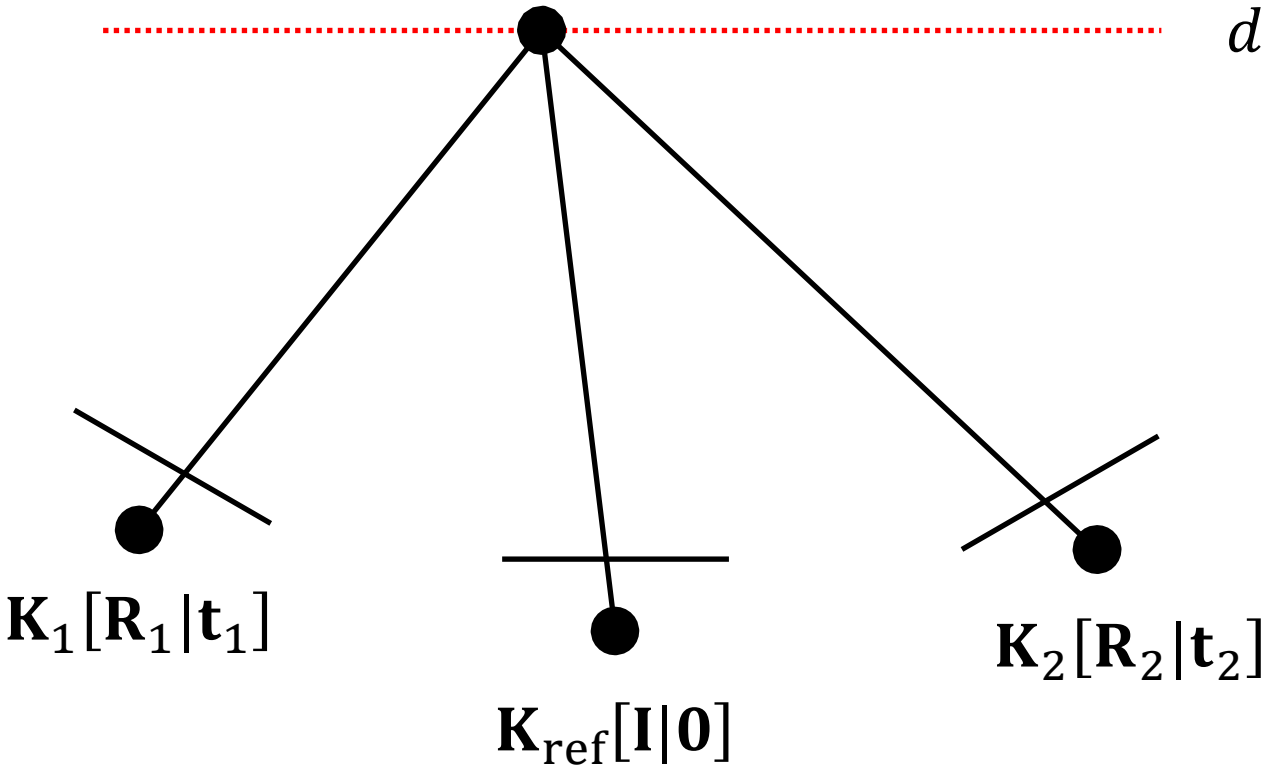

d
$\mathbf{x}$
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\text{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$


X
d
x
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\mathrm{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$

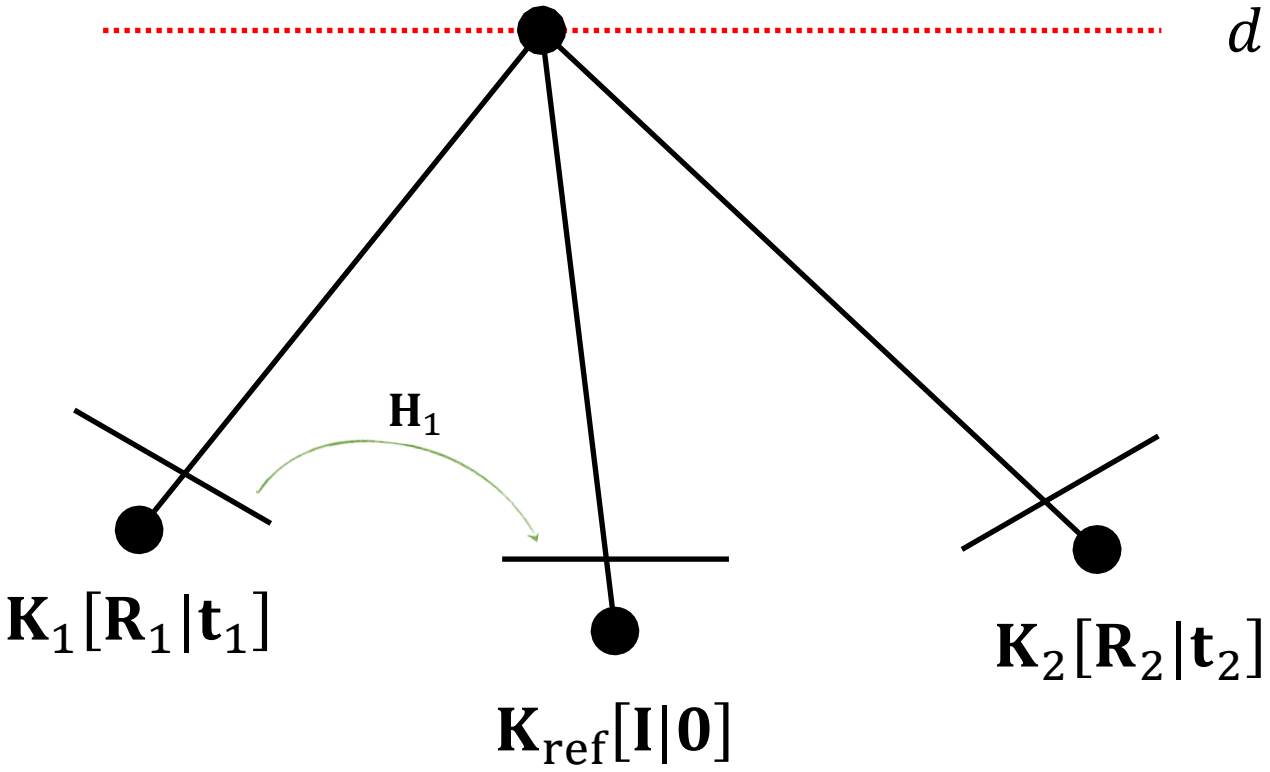

X
d
x1
x
x2
K1[R1|t1]
Kref[I|0]
K2[R2|t2]

有没有更好的办法?

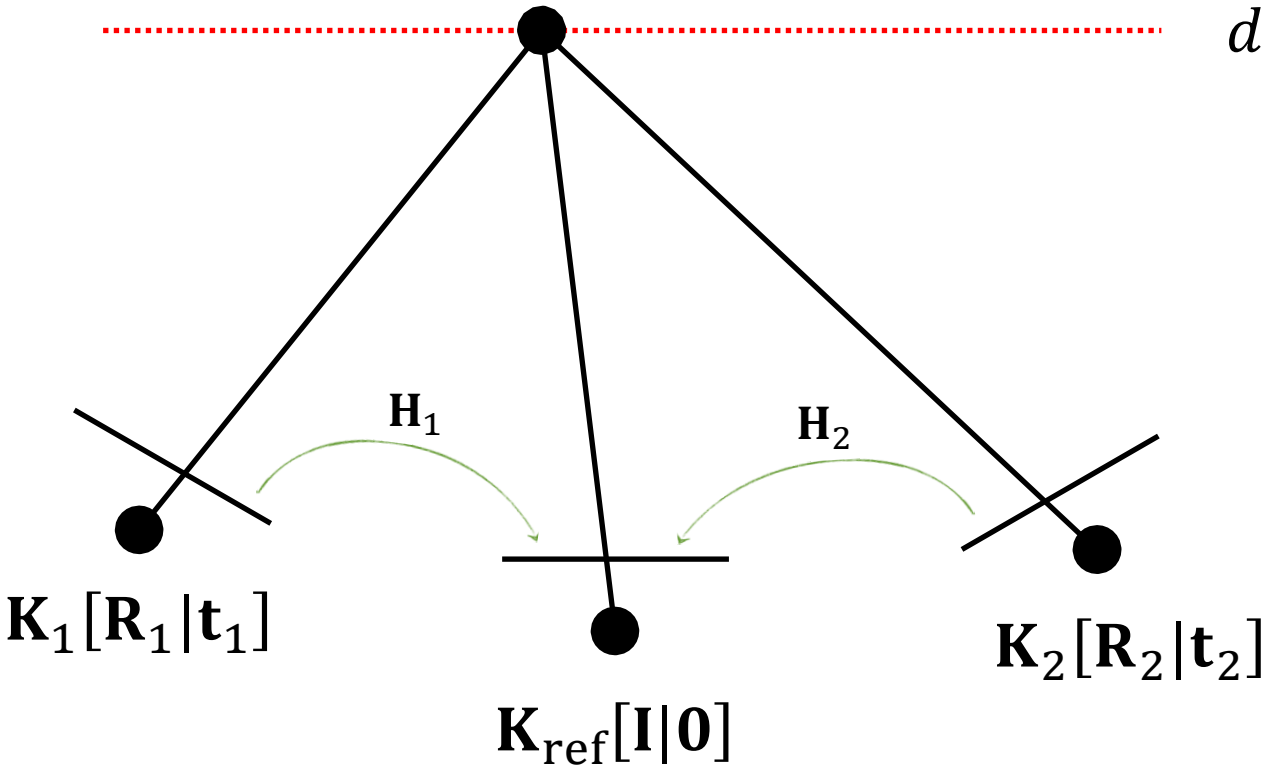
平面扫描


d
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\mathrm{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$

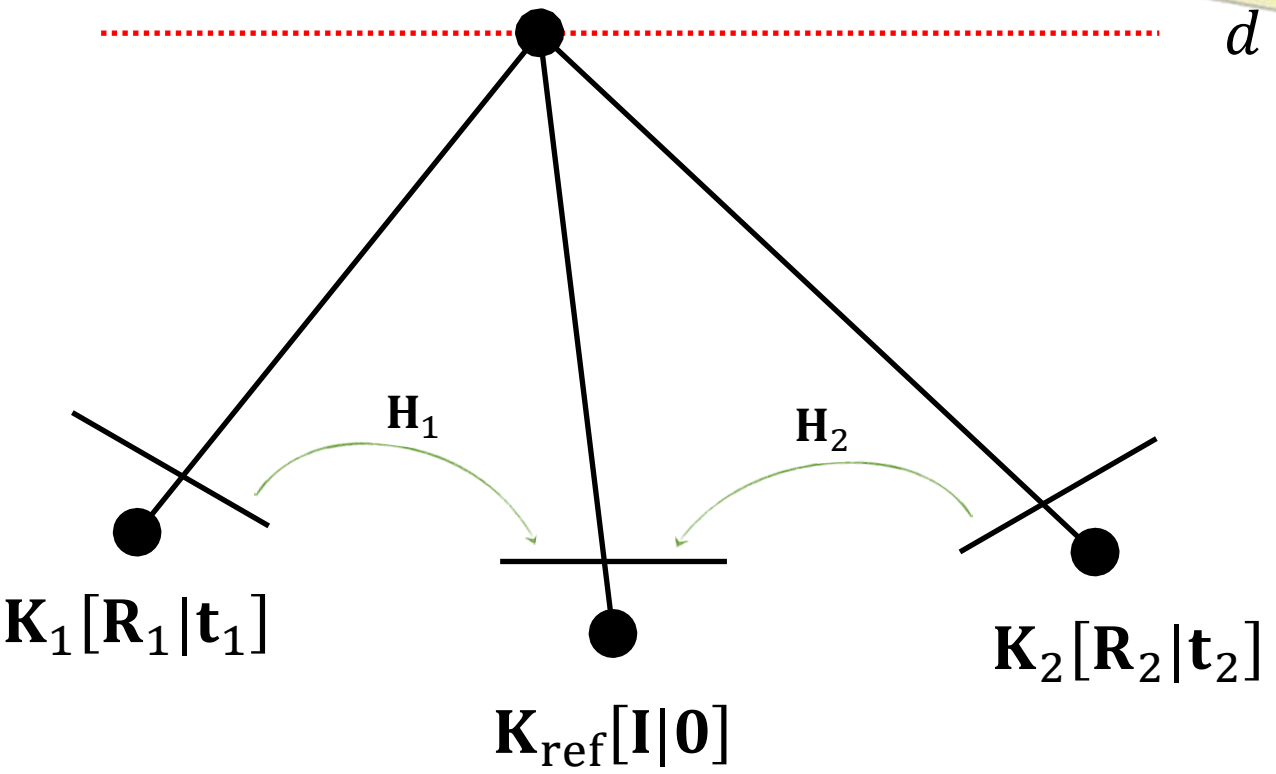
平面扫描


d
$\mathbf{H}_1$
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\mathrm{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$

平面扫描

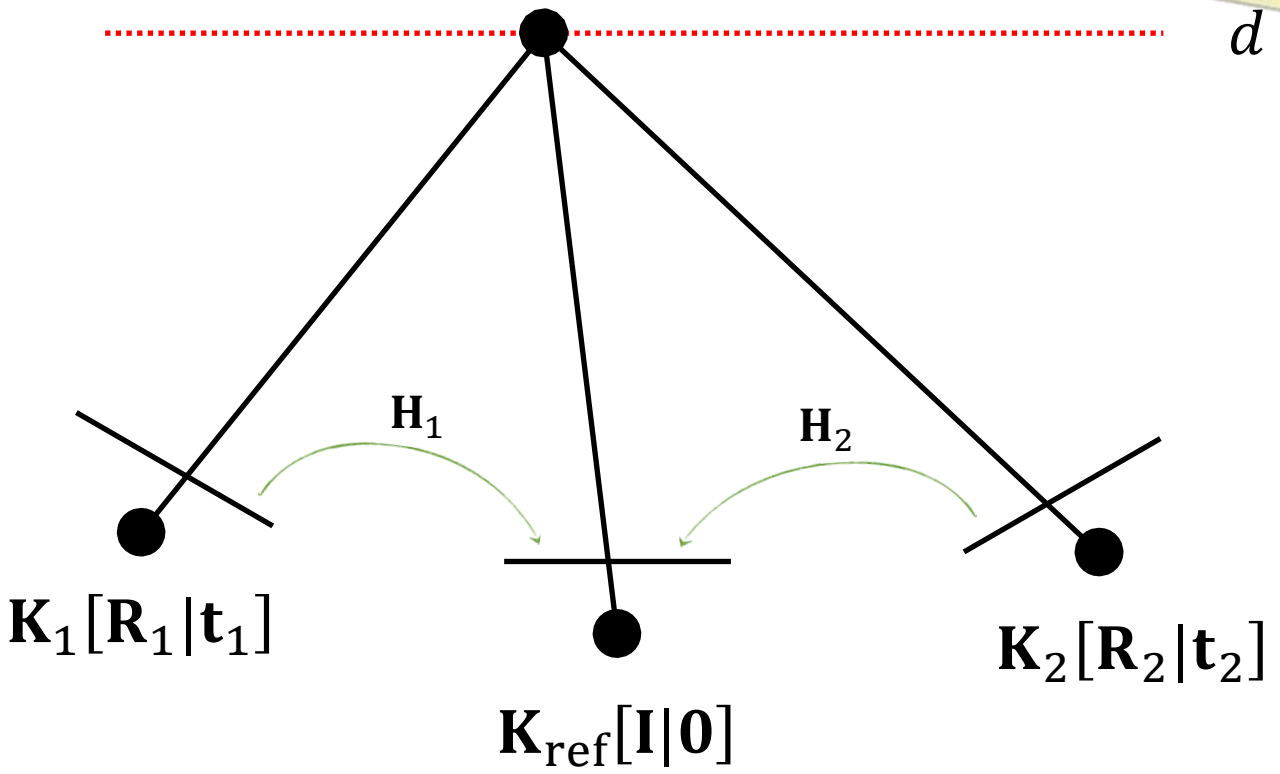

d
$\mathbf{H}_1$
$\mathbf{H}_2$
$\mathbf{K}_1[\mathbf{R}_1|\mathbf{t}_1]$
$\mathbf{K}_{\mathrm{ref}}[\mathbf{I}|\mathbf{0}]$
$\mathbf{K}_2[\mathbf{R}_2|\mathbf{t}_2]$

平面扫描



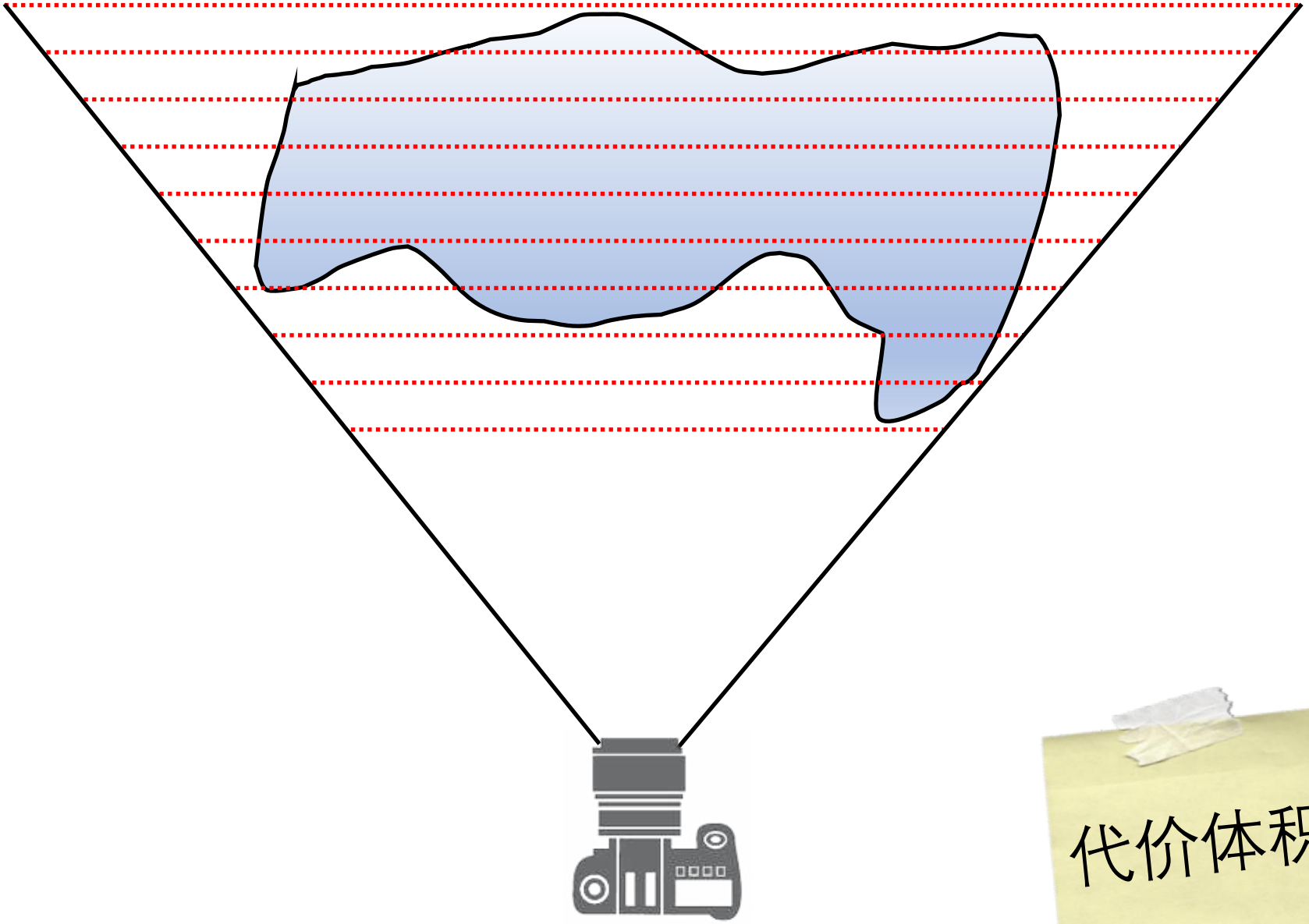
深度d平面的单应变换：$\mathbf{H}_k = \mathbf{K}_k \left(\mathbf{R}_k^{\mathrm{T}} + \frac{\mathbf{R}_k^{\mathrm{T}} \mathbf{t}_k \mathbf{n}^{\mathrm{T}}}{d} \right) \mathbf{K}_{\mathrm{ref}}^{-1}$

平面扫描



深度d平面的单应变换：$\mathbf{H}_k = \mathbf{K}_k \left(\mathbf{R}_k^{\mathrm{T}} + \frac{\mathbf{R}_k^{\mathrm{T}} \mathbf{t}_k \mathbf{n}^{\mathrm{T}}}{d} \right) \mathbf{K}_{\mathrm{ref}}^{-1}$

代价体积：$C(x, y, d) = \sum_{k=0}^{N-1} \sum_{i,j \in W} \left| I_{\mathrm{ref}}(x - i, y - j) - \beta_k^{\mathrm{ref}} I_k^d(x_k - i, y_k - j) \right|$


代价体积

MVSNet: Depth Inference for Unstructured Multi-view Stereo

Yao Yao[1⋆[0000−0001−9866−4291]], Zixin Luo[1⋆[0000−0001−6946−2826]], Shiwei Li[1⋆[0000−0003−0712−0059]], Tian Fang[2[0000−0002−5871−3455]], and Long Quan[1[0000−0001−8148−1771]]

[1] The Hong Kong University of Science and Technology,
{yyaoag, zluoag, slibc, quan}@cse.ust.hk
[2] Shenzhen Zhuke Innovation Technology (Altizure),
fangtian@altizure.com

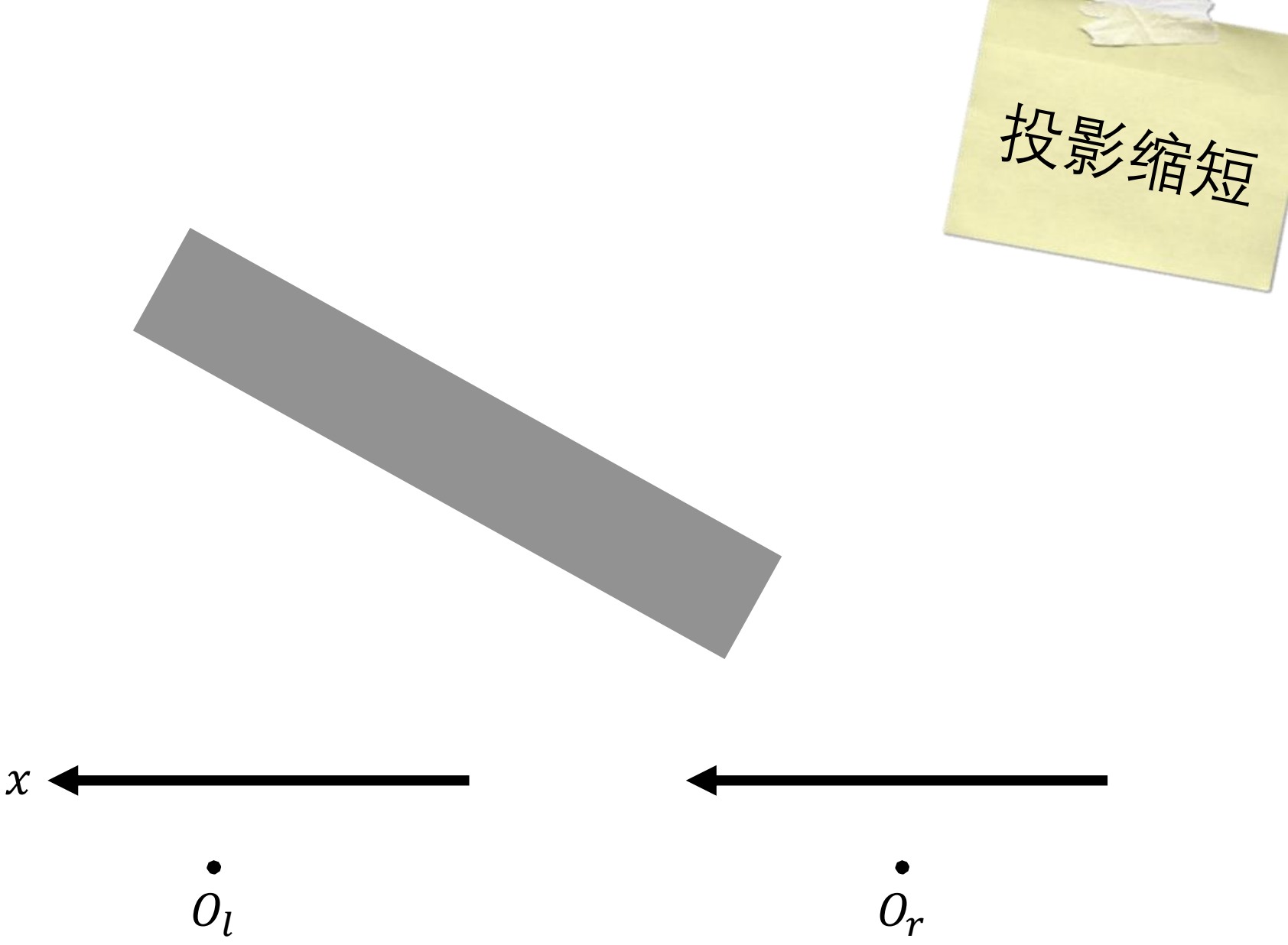
Abstract. We present an end-to-end deep learning architecture for depth map inference from multi-view images. In the network, we first extract deep visual image features, and then build the 3D cost volume upon the reference camera frustum via the differentiable homography warping. Next, we apply 3D convolutions to regularize and regress the initial depth map, which is then refined with the reference image to generate the final output. Our framework flexibly adapts arbitrary N-view inputs using a variance-based cost metric that maps multiple features into one cost feature. The proposed MVSNet is demonstrated on the large-scale indoor *DTU* dataset. With simple post-processing, our method not only significantly outperforms previous state-of-the-arts, but also is several times faster in runtime. We also evaluate MVSNet on the complex outdoor *Tanks and Temples* dataset, where our method ranks first before April 18, 2018 without any fine-tuning, showing the strong generalization ability of MVSNet.

Keywords: Multi-view Stereo, Depth Map, Deep Learning

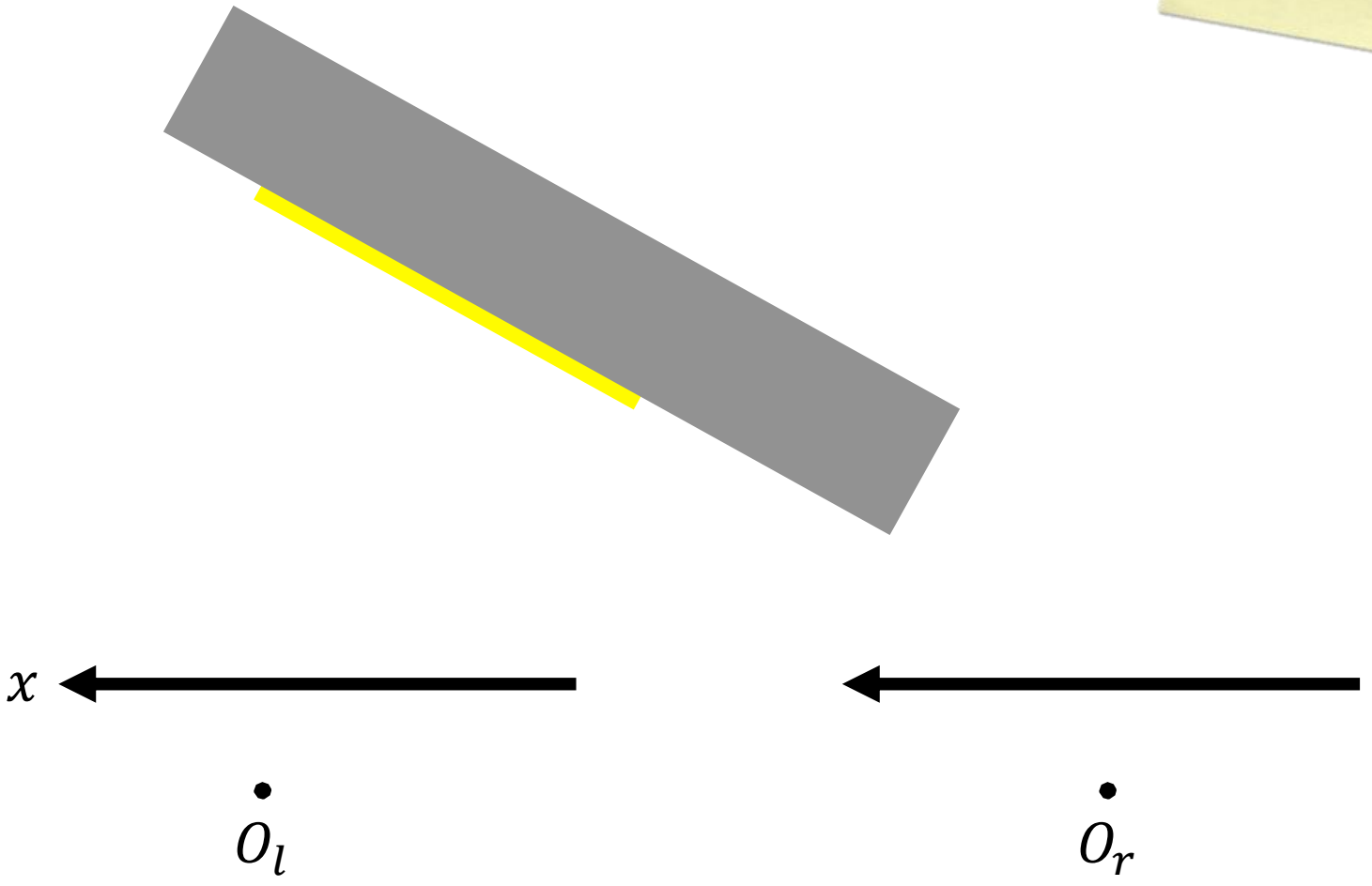
ECCV, 2018

为什么图像对应问题

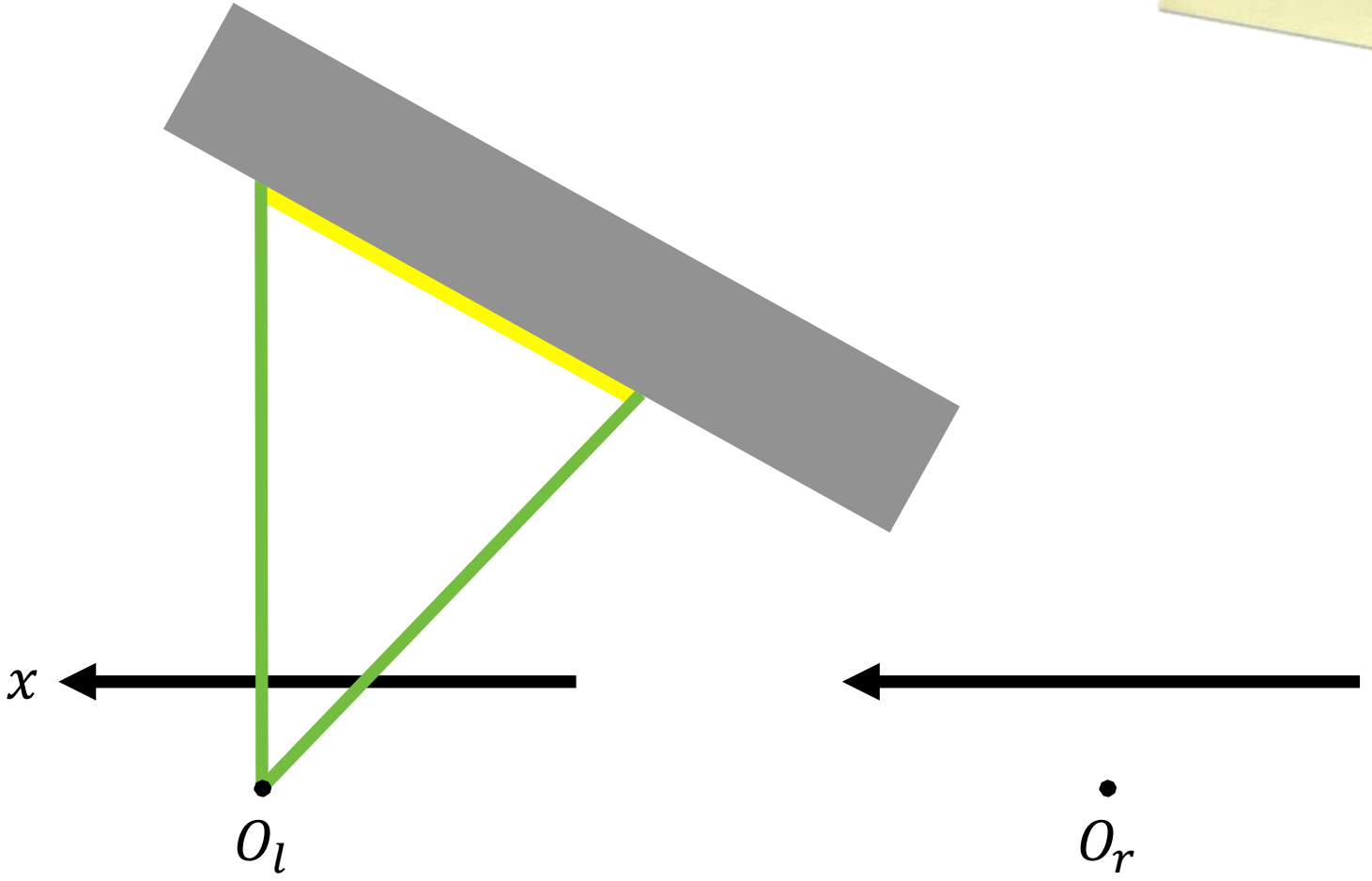
困难？


投影缩短
x
O_l
O_r

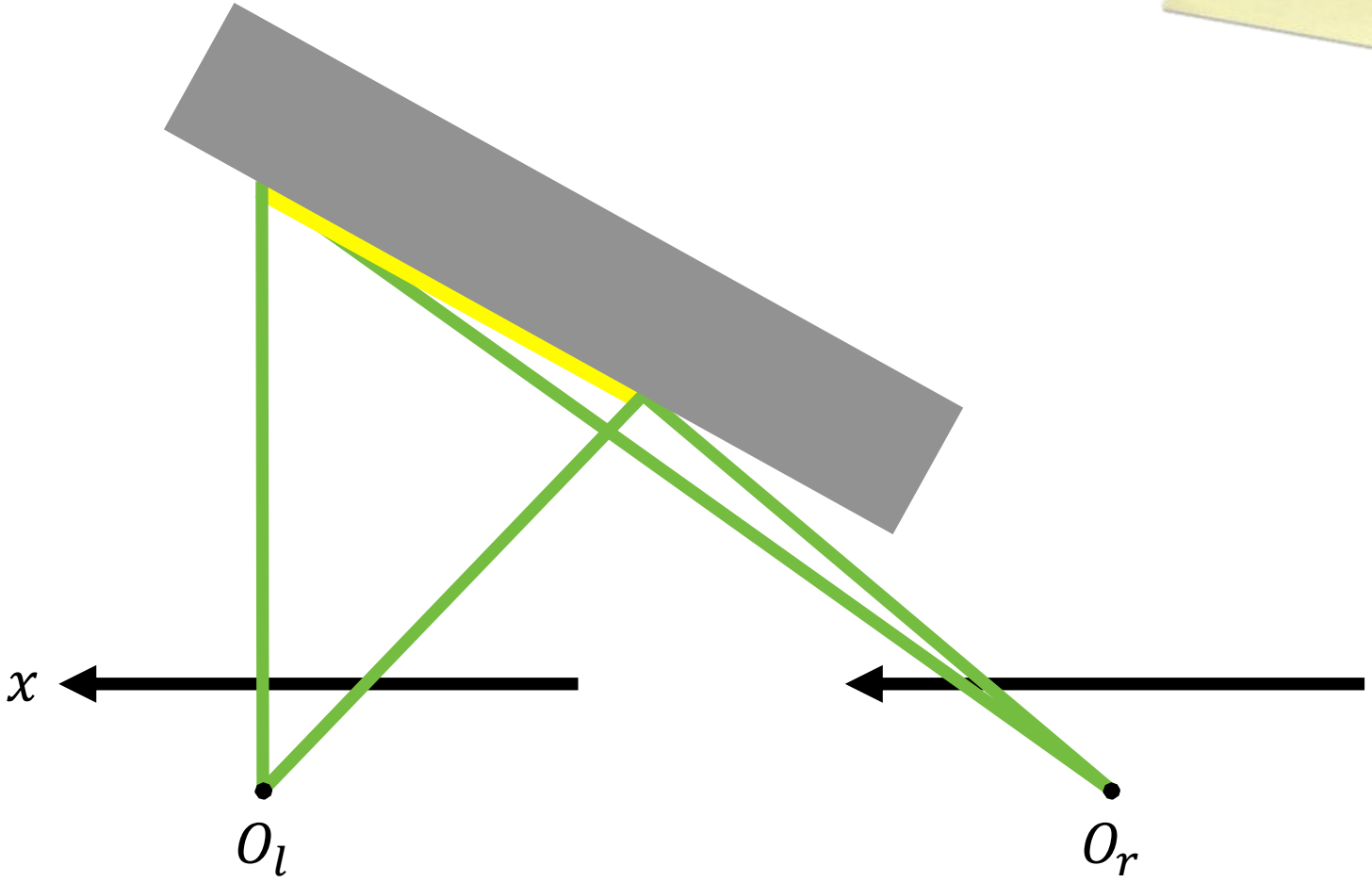
投影缩短



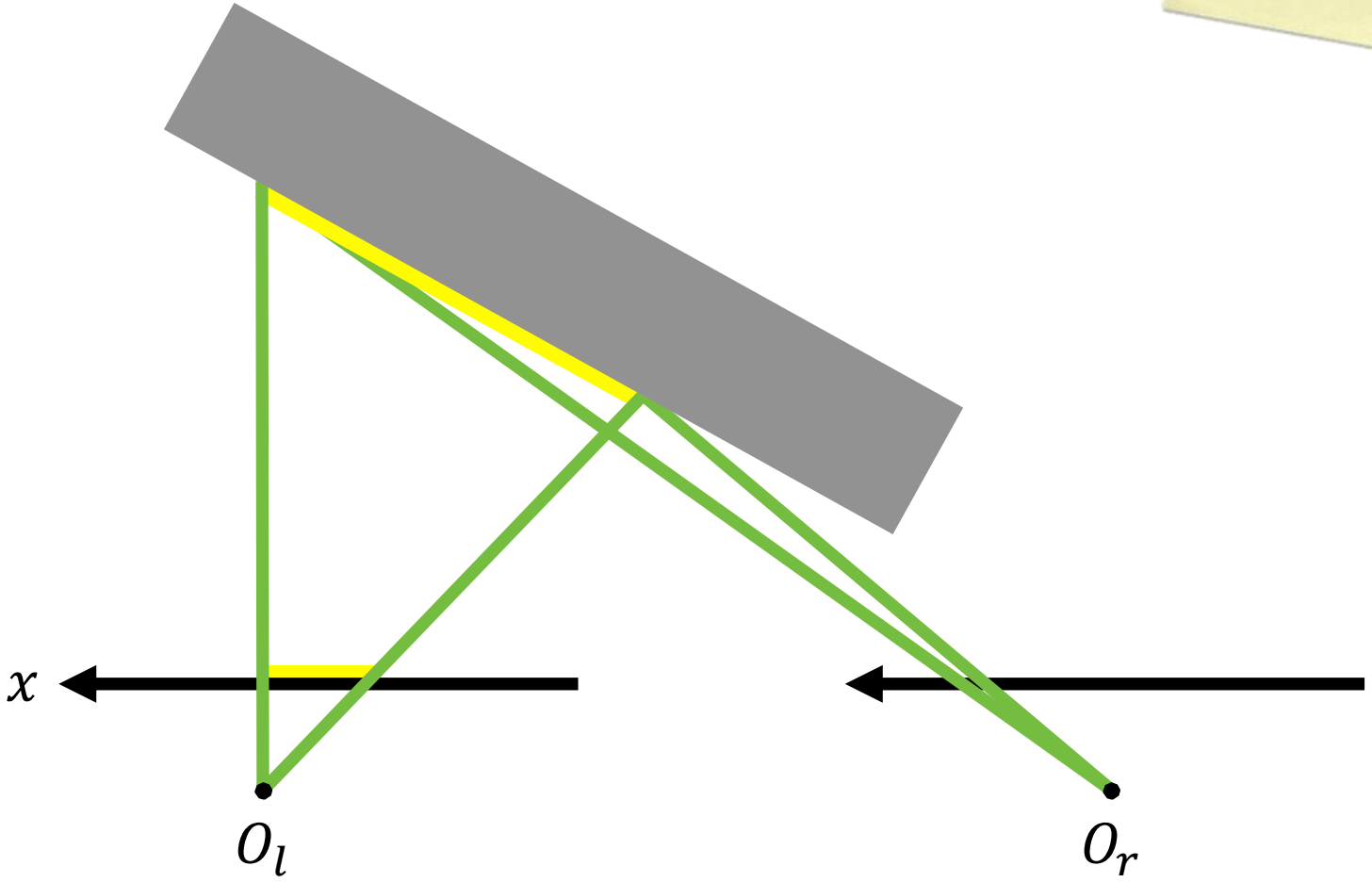
投影缩短



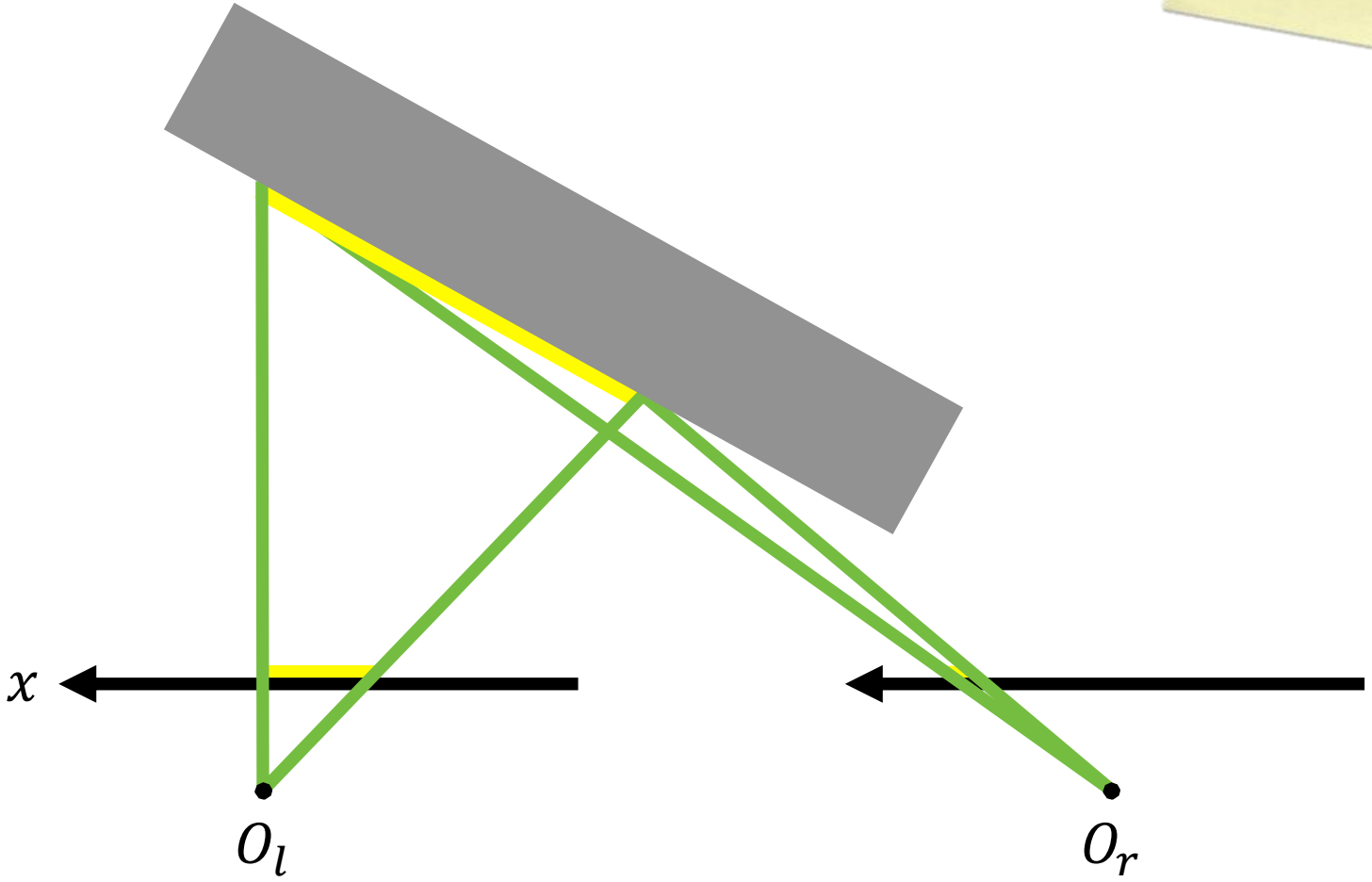
投影缩短



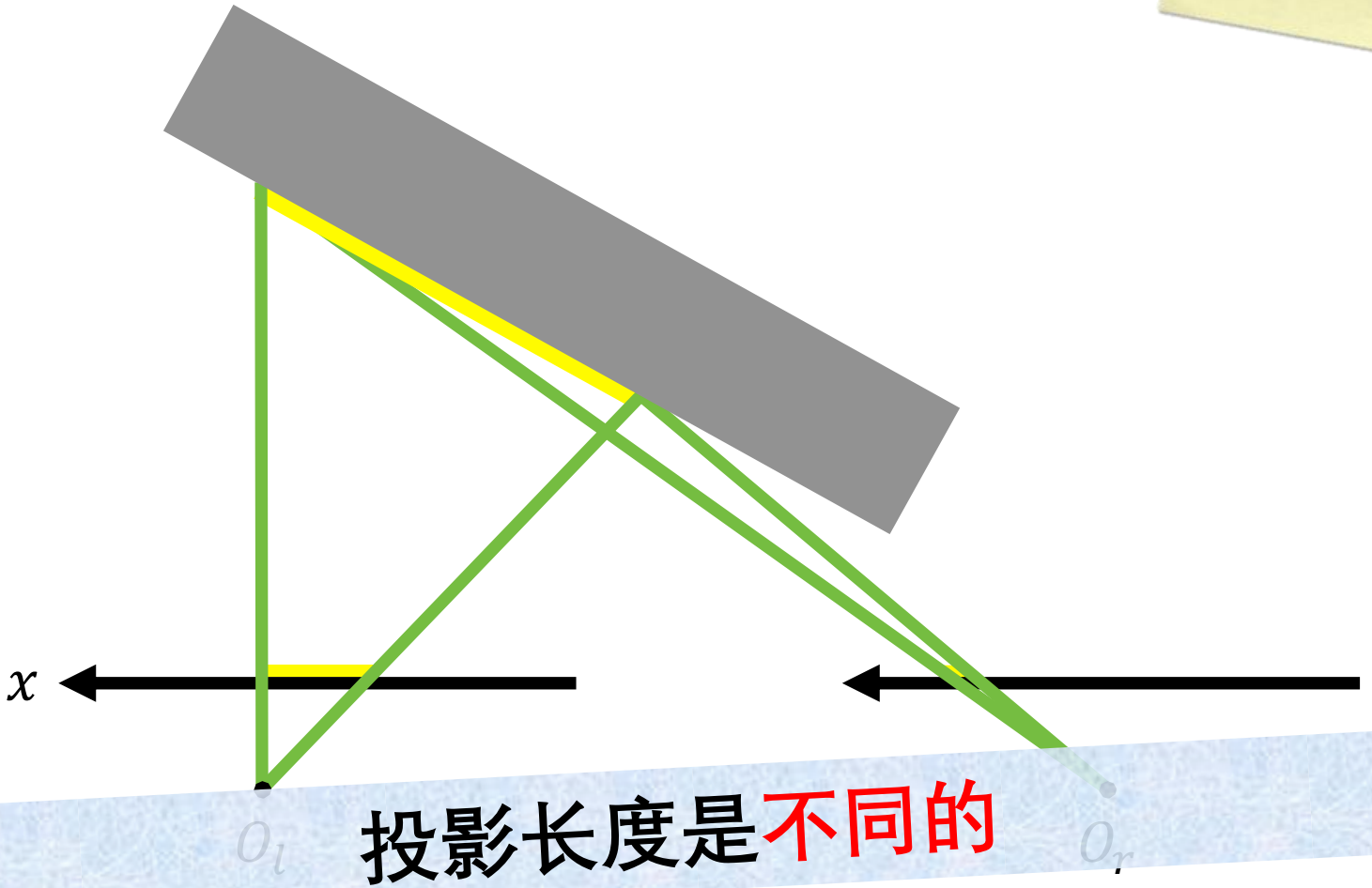
投影缩短



投影缩短



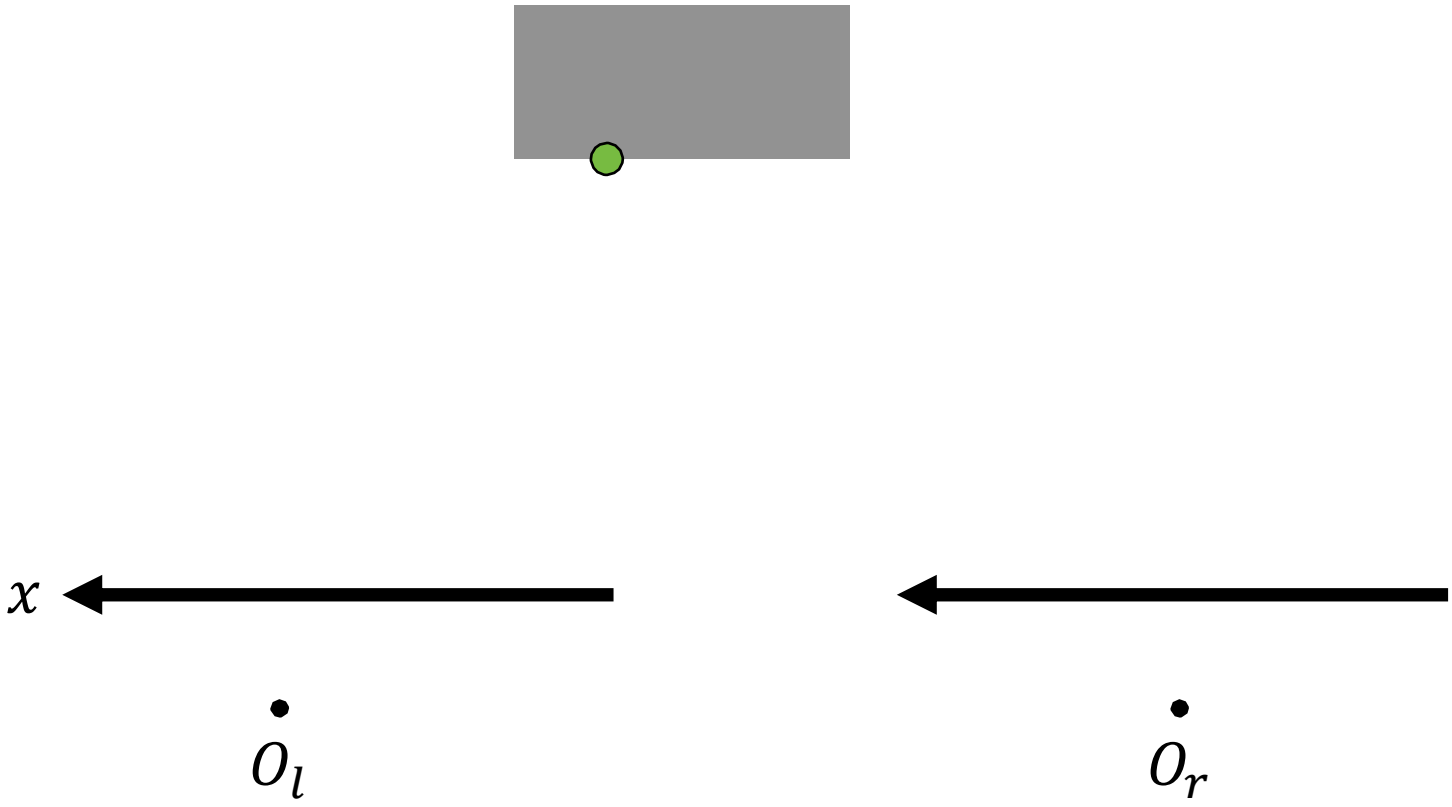
投影缩短



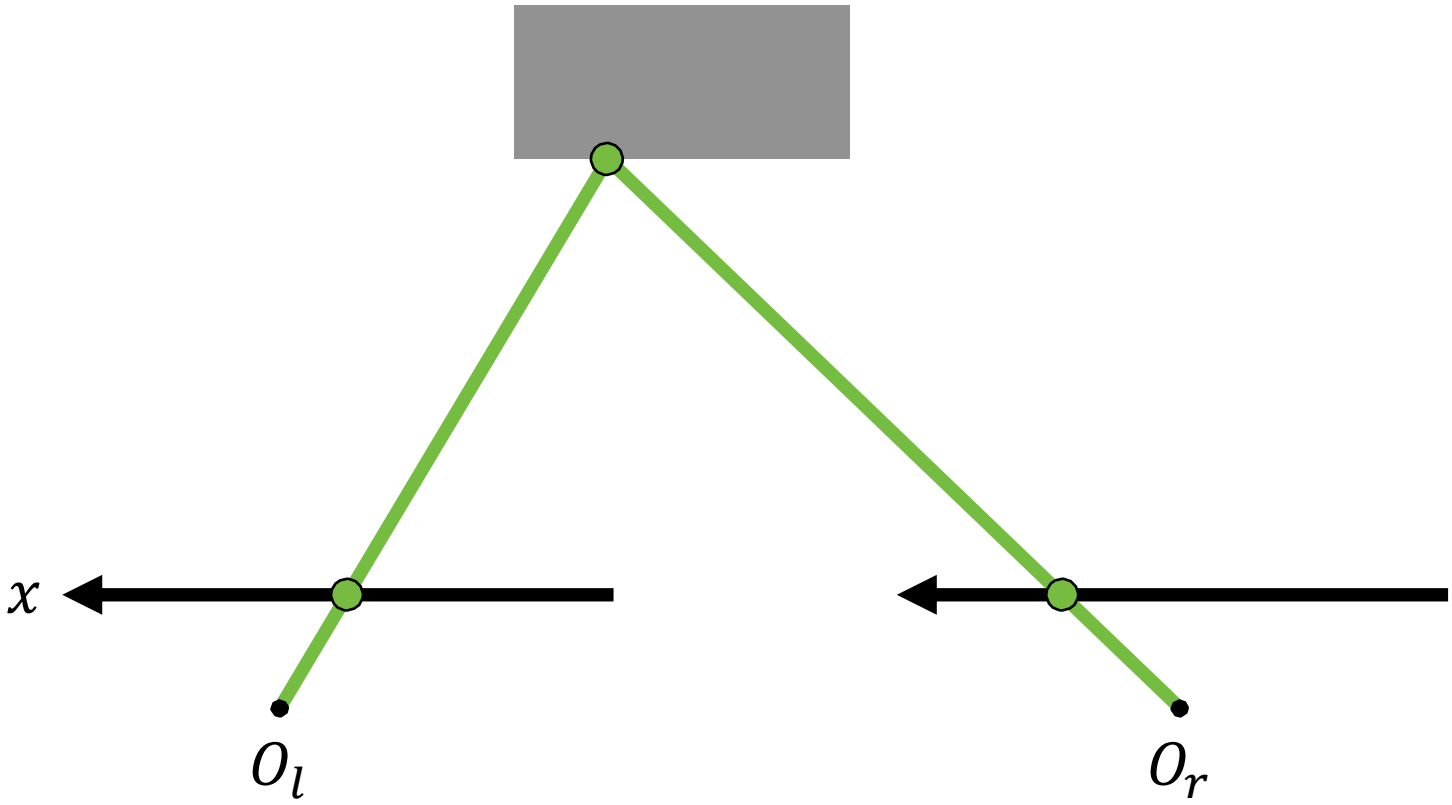
投影长度是不同的

双目
半遮挡

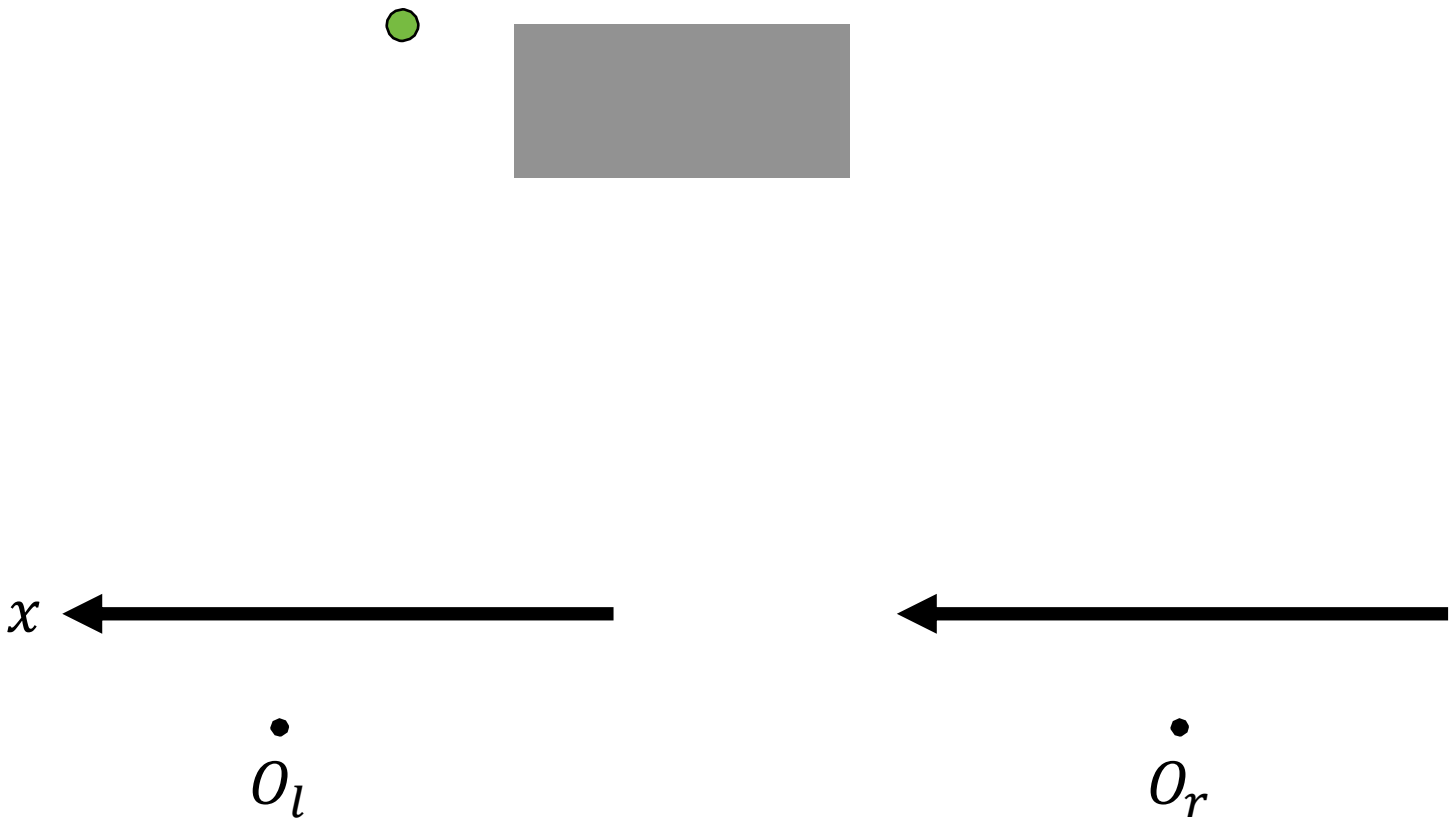
双目
半遮挡



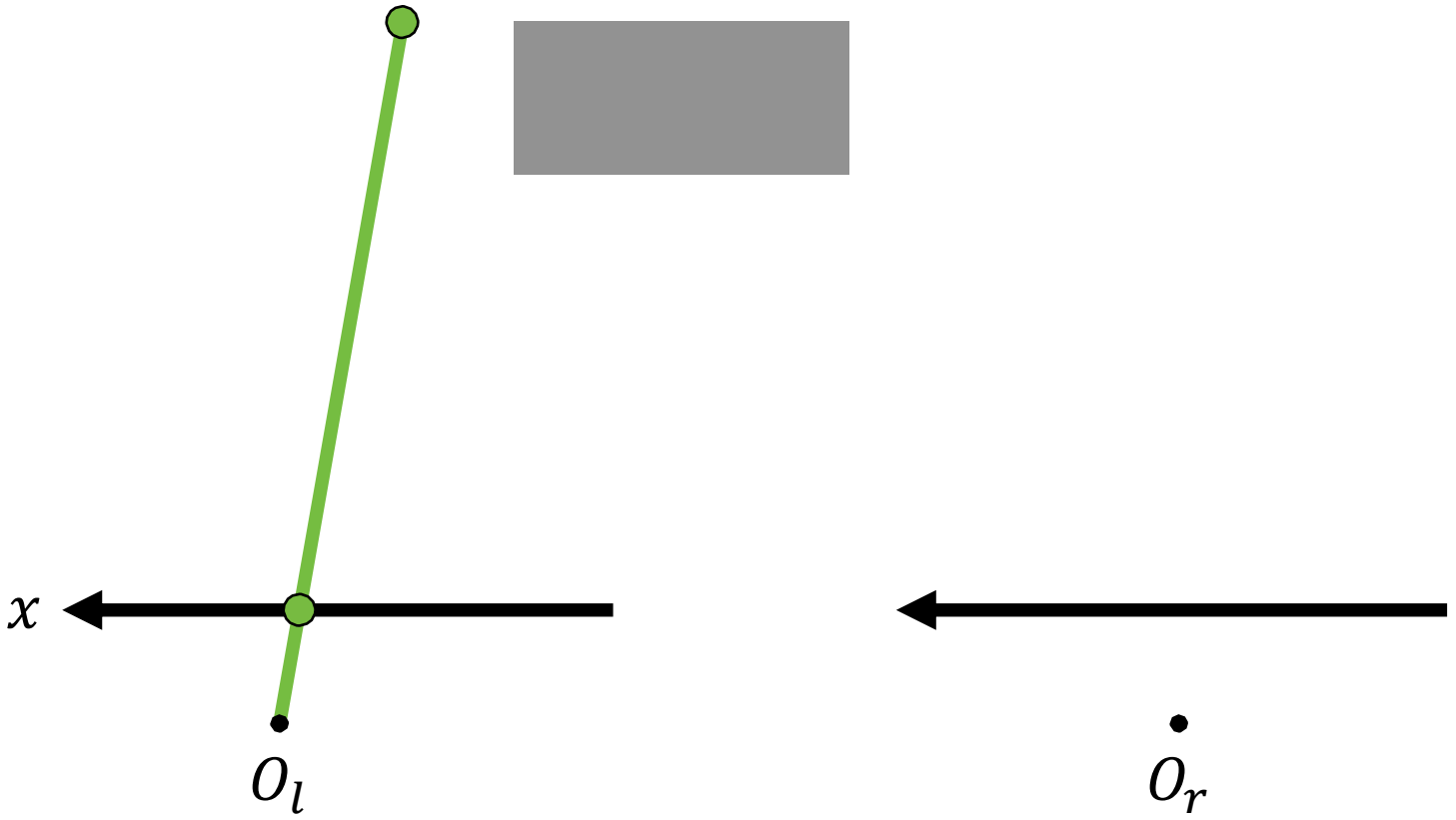
双目
半遮挡



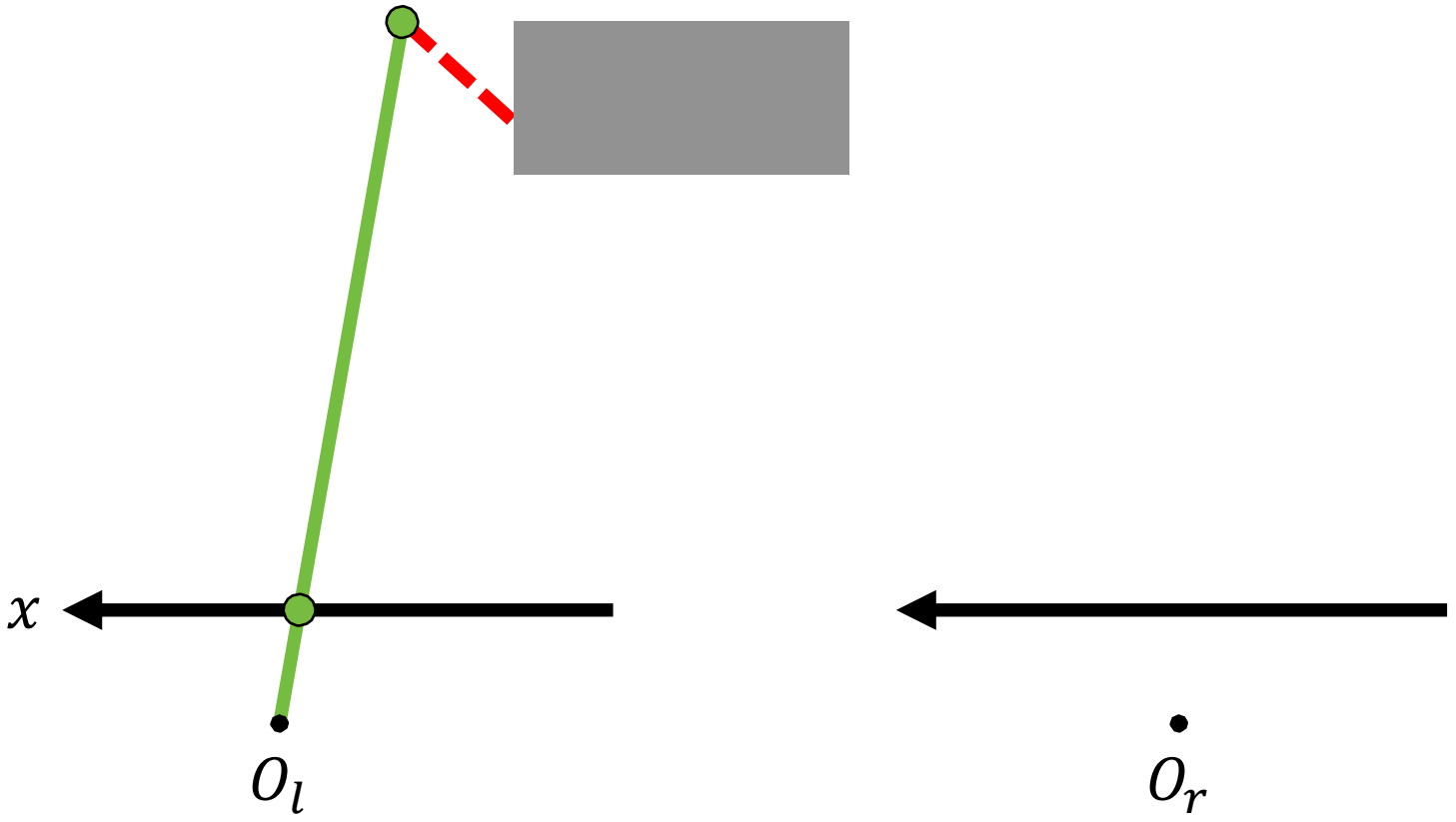
双目
半遮挡



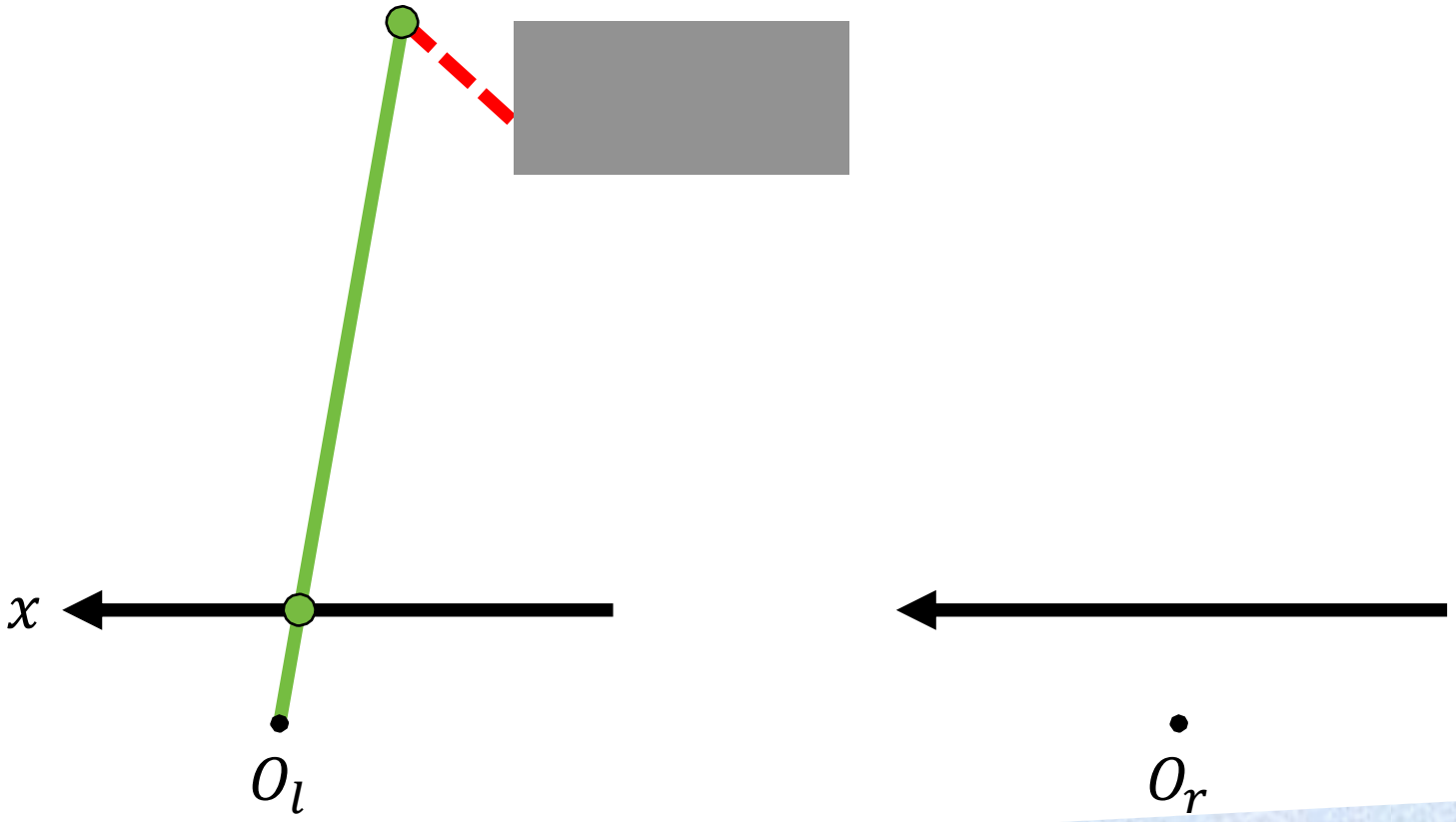
双目
半遮挡



双目
半遮挡



双目
半遮挡



所有点不能同时在两个视图中看到

重复性
结构

重复性
结构

重复性
结构

重复性
结构

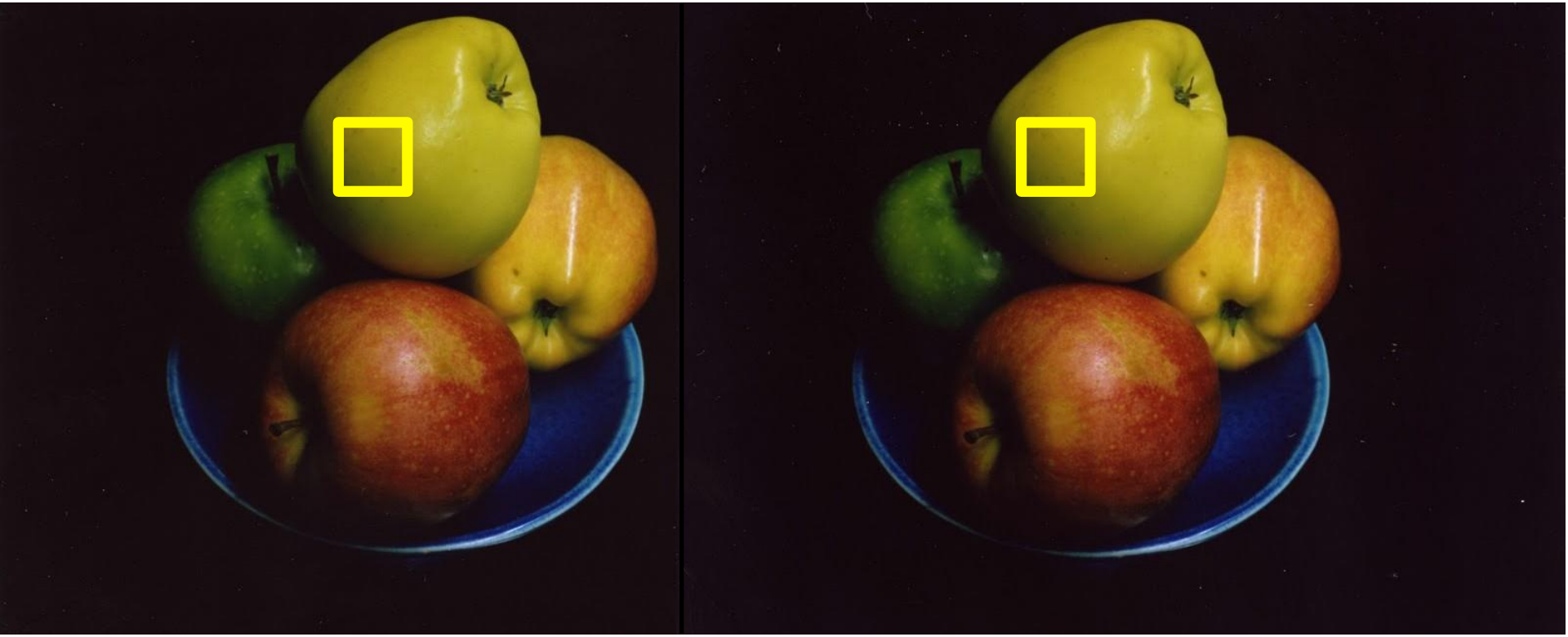
重复性结构

块匹配是有歧义的

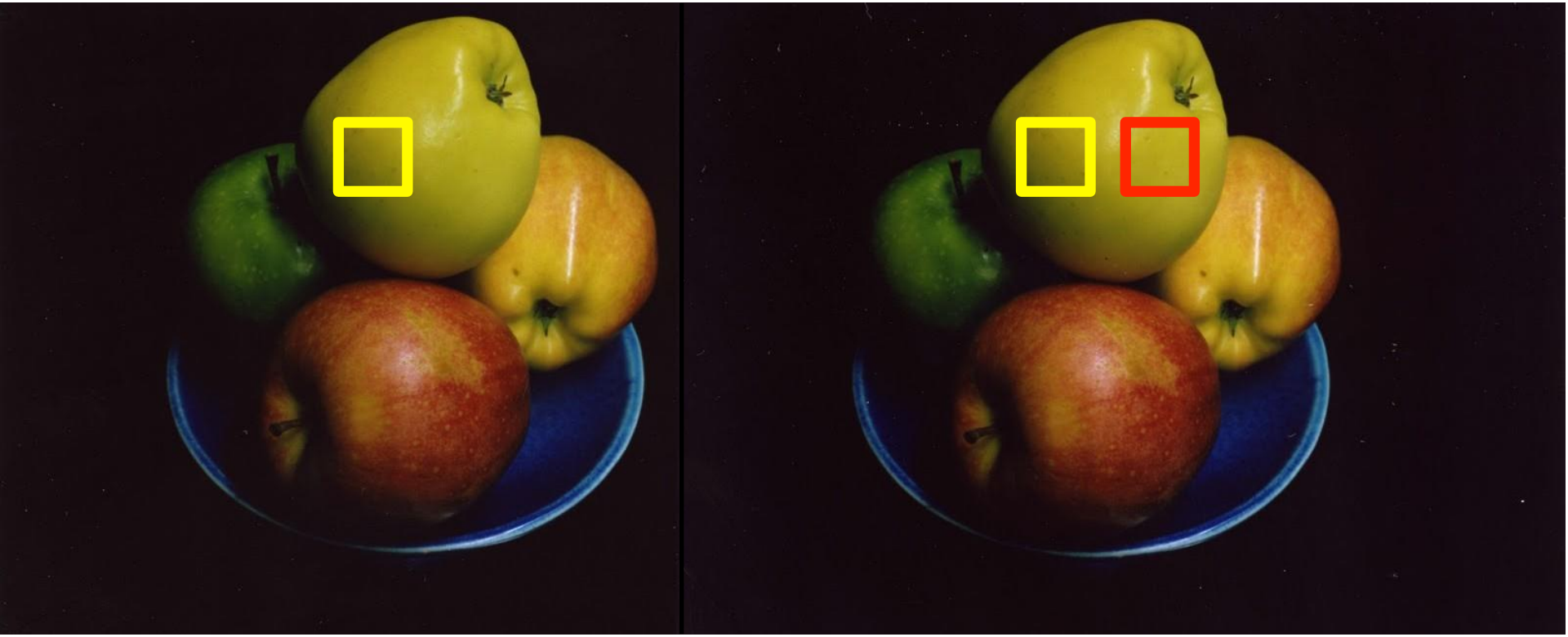
无纹理
表面

无纹理
表面

无纹理
表面



无纹理
表面



Stereo

Daniel Scharstein • Richard Szeliski

Welcome to the Middlebury Stereo Vision Page, formerly located at www.middlebury.edu/stereo. This website accompanies our taxonomy and comparison of two-frame stereo correspondence algorithms [1]. It contains:

- An on-line evaluation of current algorithms
- Many stereo datasets with ground-truth disparities
- Our stereo correspondence software
- An on-line submission script that allows you to evaluate your stereo algorithm in our framework

How to cite the materials on this website:
We grant permission to use and publish all images and numerical results on this website. If you report performance results, we request that you cite our paper [1]. Instructions on how to cite our datasets are listed on the datasets page. If you want to cite this website, please use the URL "**vision.middlebury.edu/stereo/**".

References:

[1] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7-42, April-June 2002.
Microsoft Research Technical Report MSR-TR-2001-81, November 2001.

Other online stereo benchmarks and datasets:

- KITTI vision benchmark
- HCI robust vision challenge

Support for this work was provided in part by NSF CAREER grant 9984485 and NSF grant IIS-0413169. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Journal of Machine Learning Research 17 (2016) 1-32 Submitted 10/15; Revised 4/16; Published 4/16

Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches

Jure Žbontar* JURE.ZBONTAR@FRI.UNI-LJ.SI
Faculty of Computer and Information Science
University of Ljubljana
Večna pot 113, SI-1001 Ljubljana, Slovenia

Yann LeCun† YANN@CS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
715 Broadway, New York, NY 10003, USA

Editor: Zhuowen Tu

Abstract

We present a method for extracting depth information from a rectified image pair. Our approach focuses on the first stage of many stereo algorithms: the matching cost computation. We approach the problem by learning a similarity measure on small image patches using a convolutional neural network. Training is carried out in a supervised manner by constructing a binary classification data set with examples of similar and dissimilar pairs of patches. We examine two network architectures for this task: one tuned for speed, the other for accuracy. The output of the convolutional neural network is used to initialize the

Journal of Machine Learning Research, 2016

data sets.

Keywords: stereo, matching cost, similarity learning, supervised learning, convolutional neural networks

1. Introduction

A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation

Nikolaus Mayer*[1], Eddy Ilg*[1], Philip Häusser*[2], Philipp Fischer*[1†]
[1]University of Freiburg [2]Technical University of Munich
[1]{mayern,ilg,fischer}@cs.uni-freiburg.de [2]haeusser@cs.tum.edu

Daniel Cremers
Technical University of Munich
cremers@tum.de

Alexey Dosovitskiy, Thomas Brox
University of Freiburg
{dosovits,brox}@cs.uni-freiburg.de

Abstract

Recent work has shown that optical flow estimation can be formulated as a supervised learning task and can be successfully solved with convolutional networks. Training of the so-called FlowNet was enabled by a large synthetically generated dataset. The present paper extends the concept of optical flow estimation via convolutional networks to disparity and scene flow estimation. To this end, we propose three synthetic stereo video datasets with sufficient realism, variation, and size to successfully train large networks. Our datasets are the first large-scale datasets to enable training and evaluation of scene flow methods. Besides the datasets,

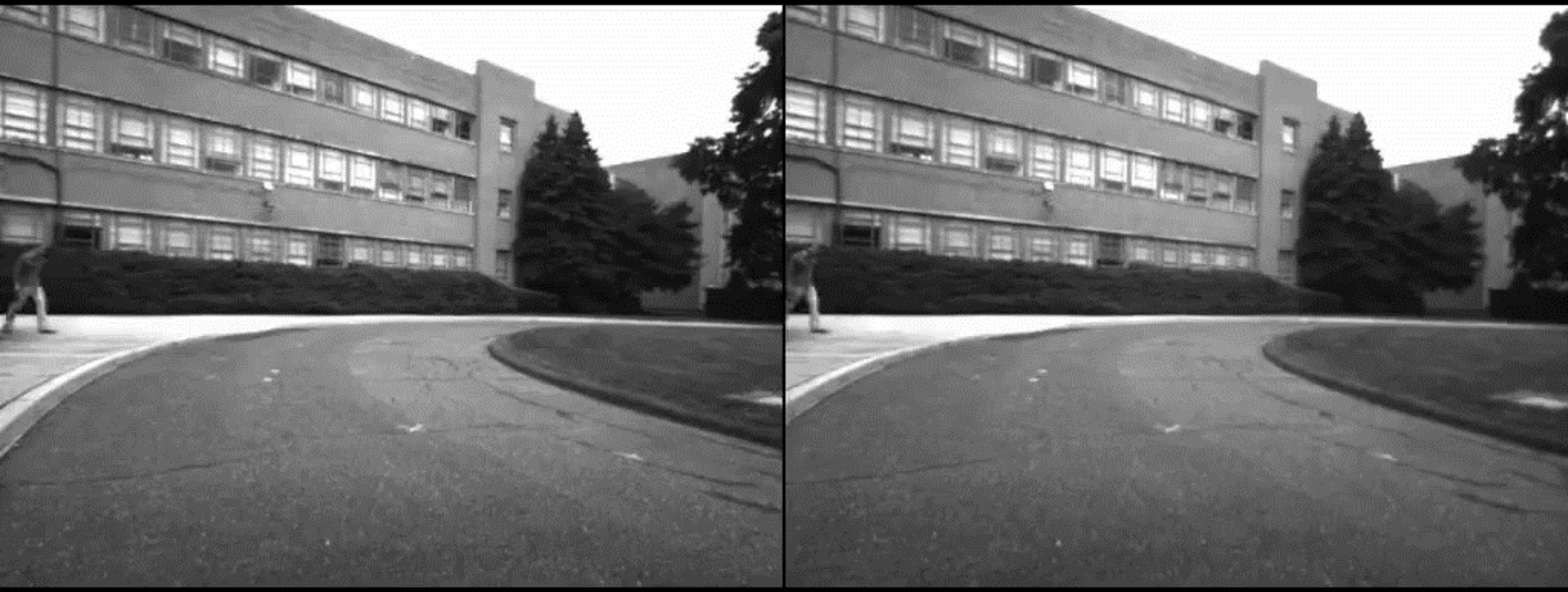
CVPR, 2016

a convolutional network.

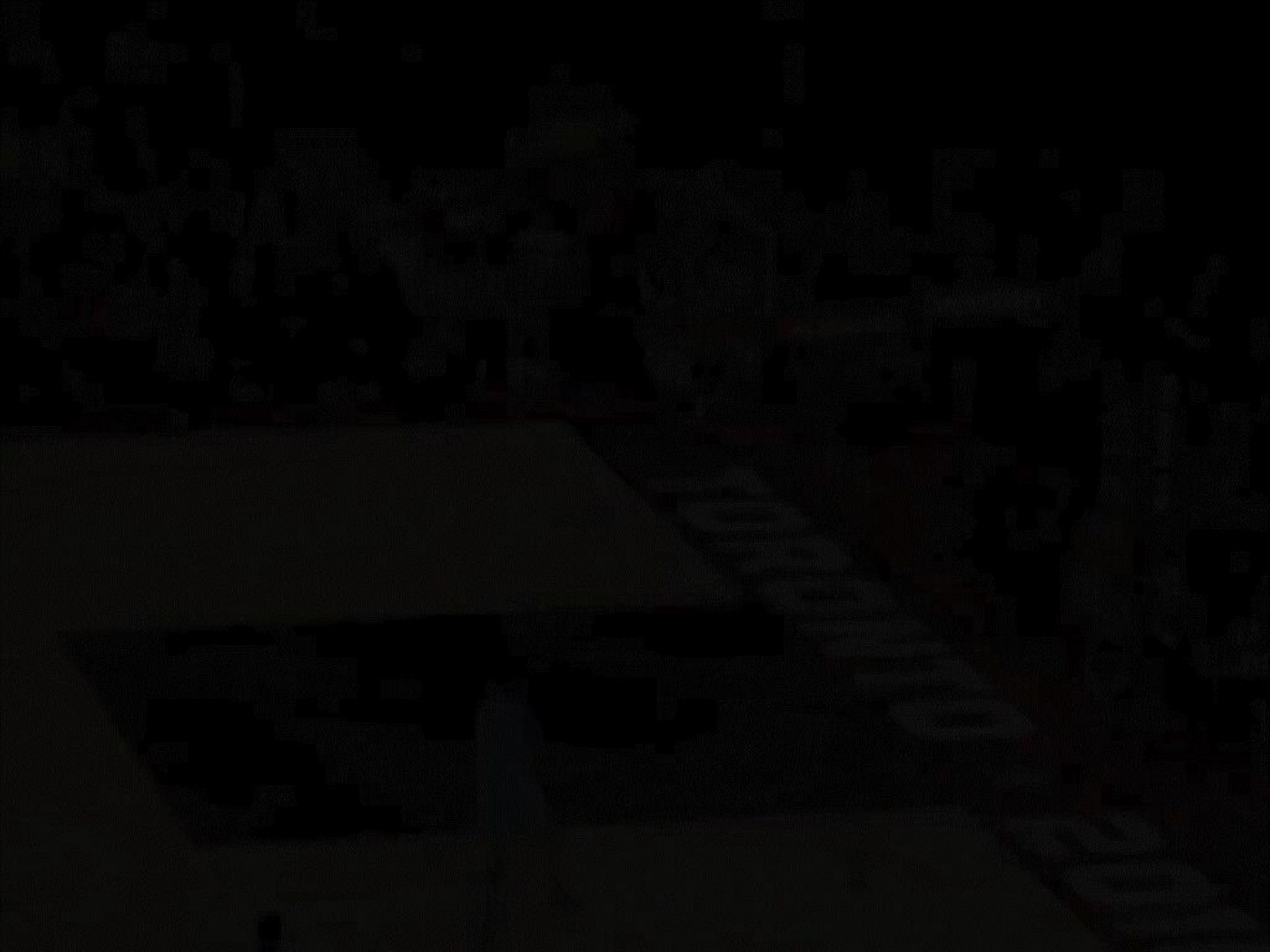
dense ground truth for *optical flow*, *disparity* and *disparity change*, as well as other data such as object segmentation.

1. Introduction

with regard to both efficiency and accuracy. One reason for



M. Sizintsev, et al.
"GPU Accelerated Realtime Stereo for Augmented Reality", 3DPVT, 2010

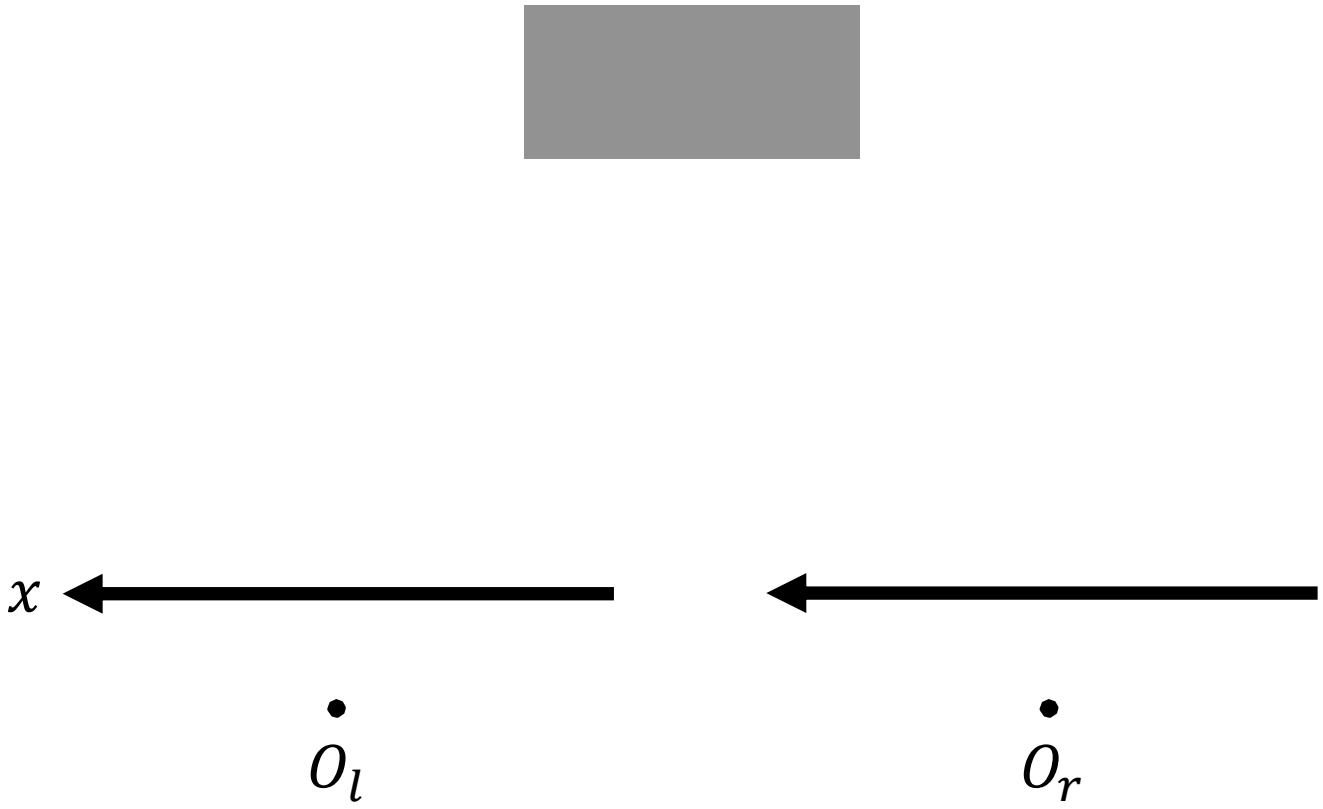


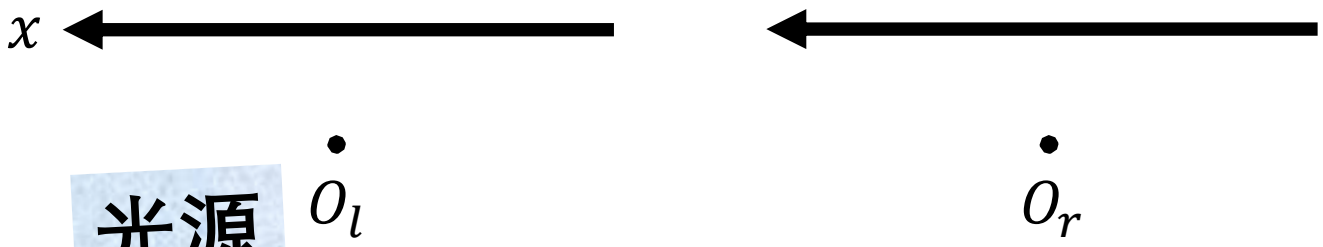
KINECT™
for XBOX 360®

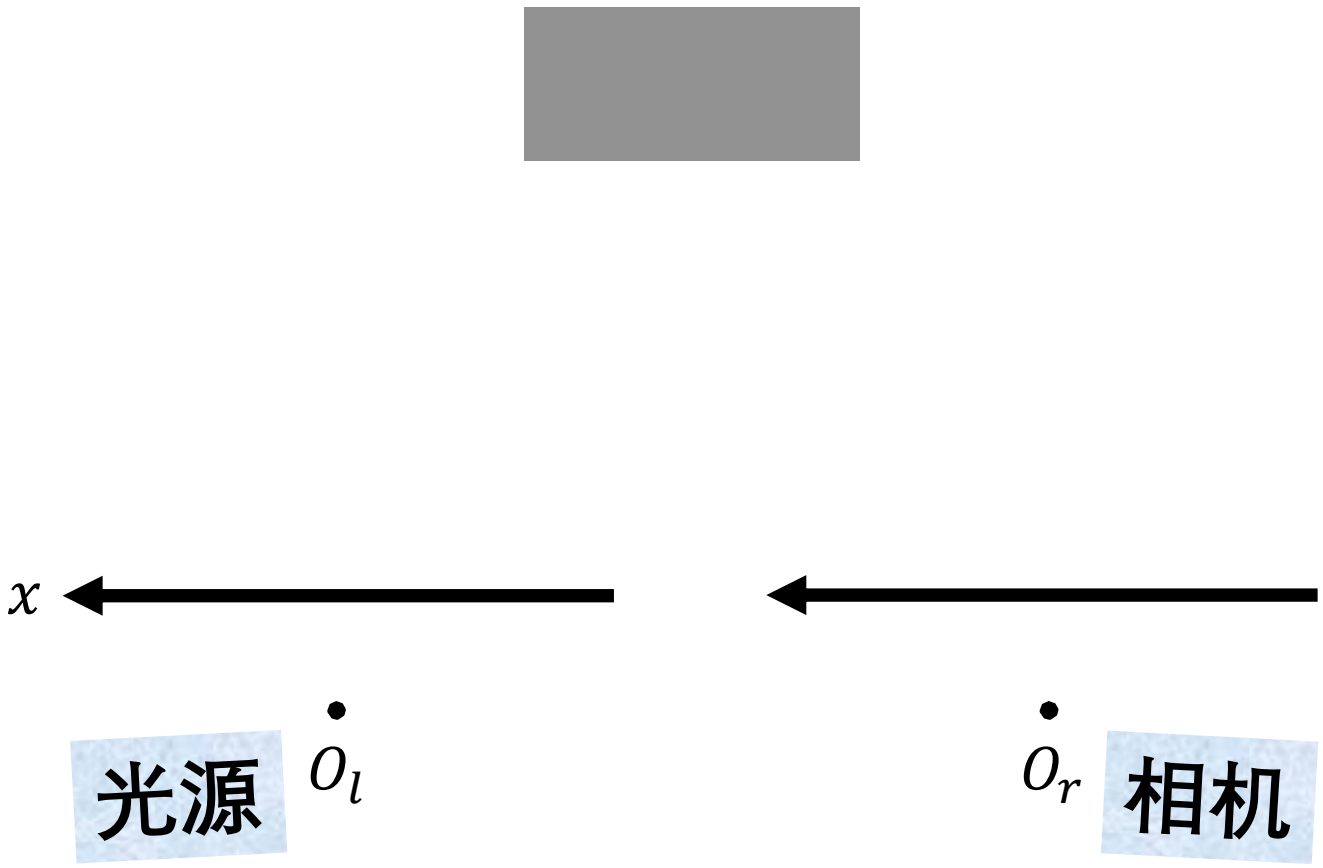
红外光源

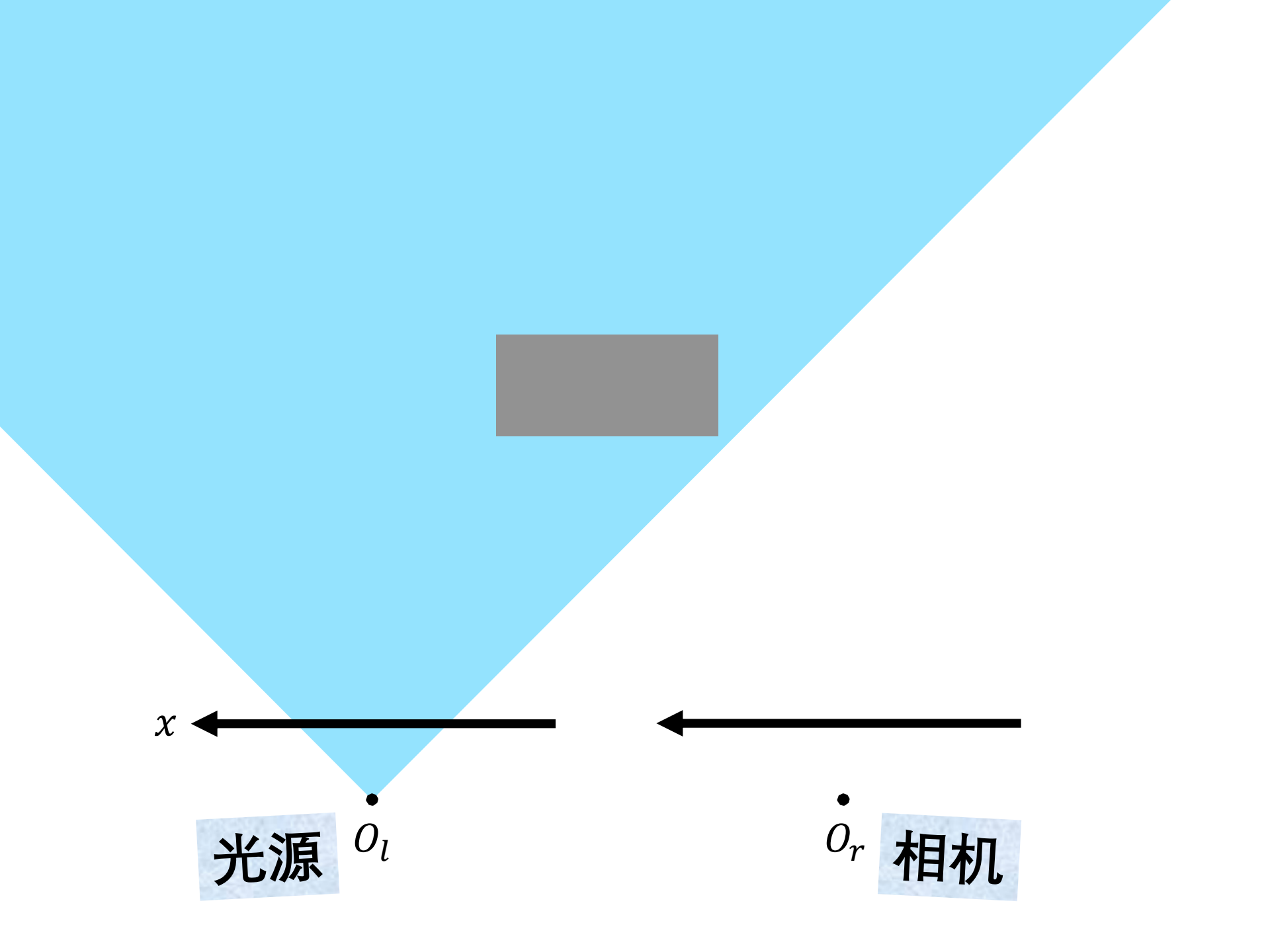
KINECT™
for XBOX 360®

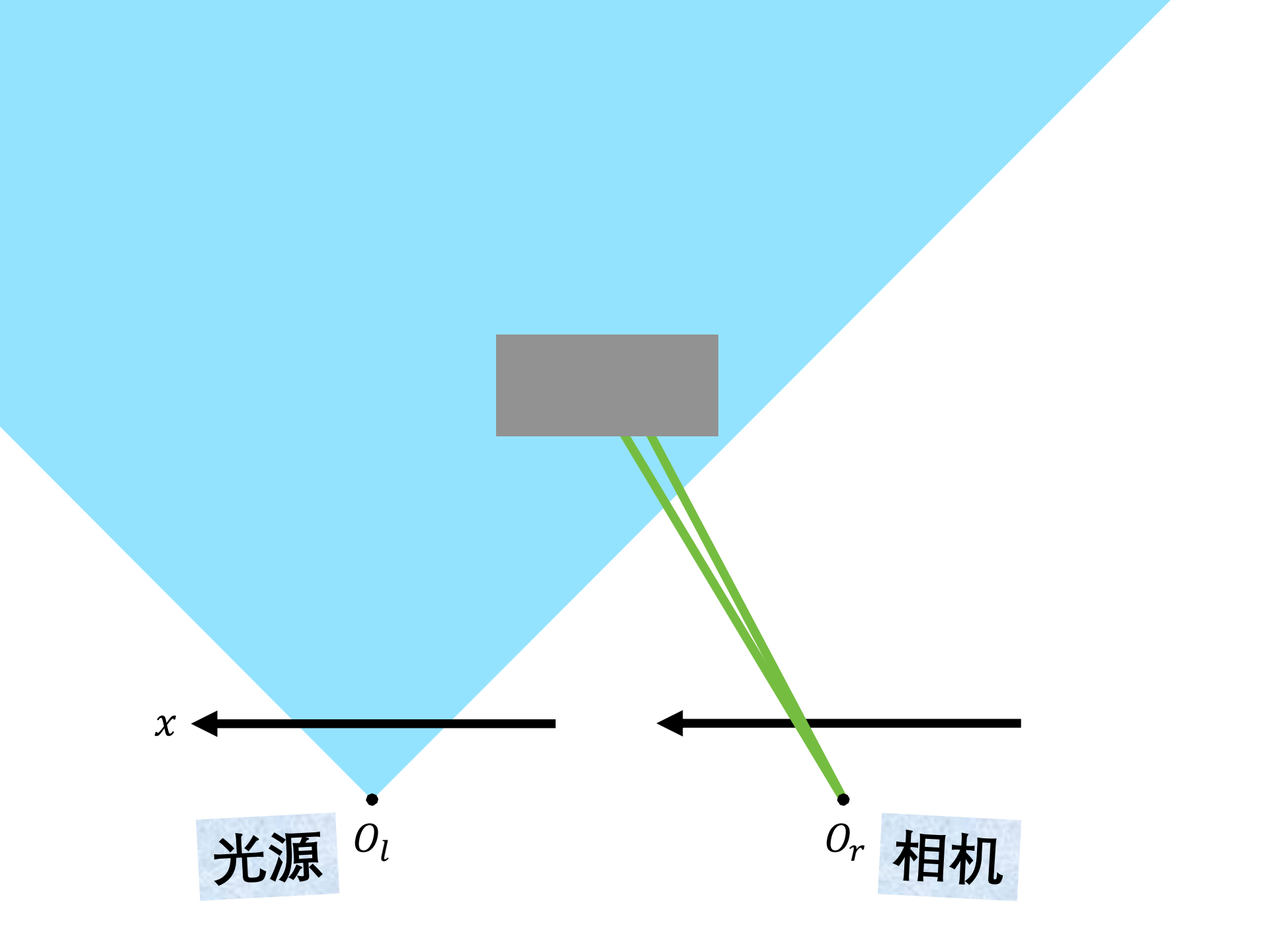
KINECT™
for XBOX 360®


x
O_l
O_r

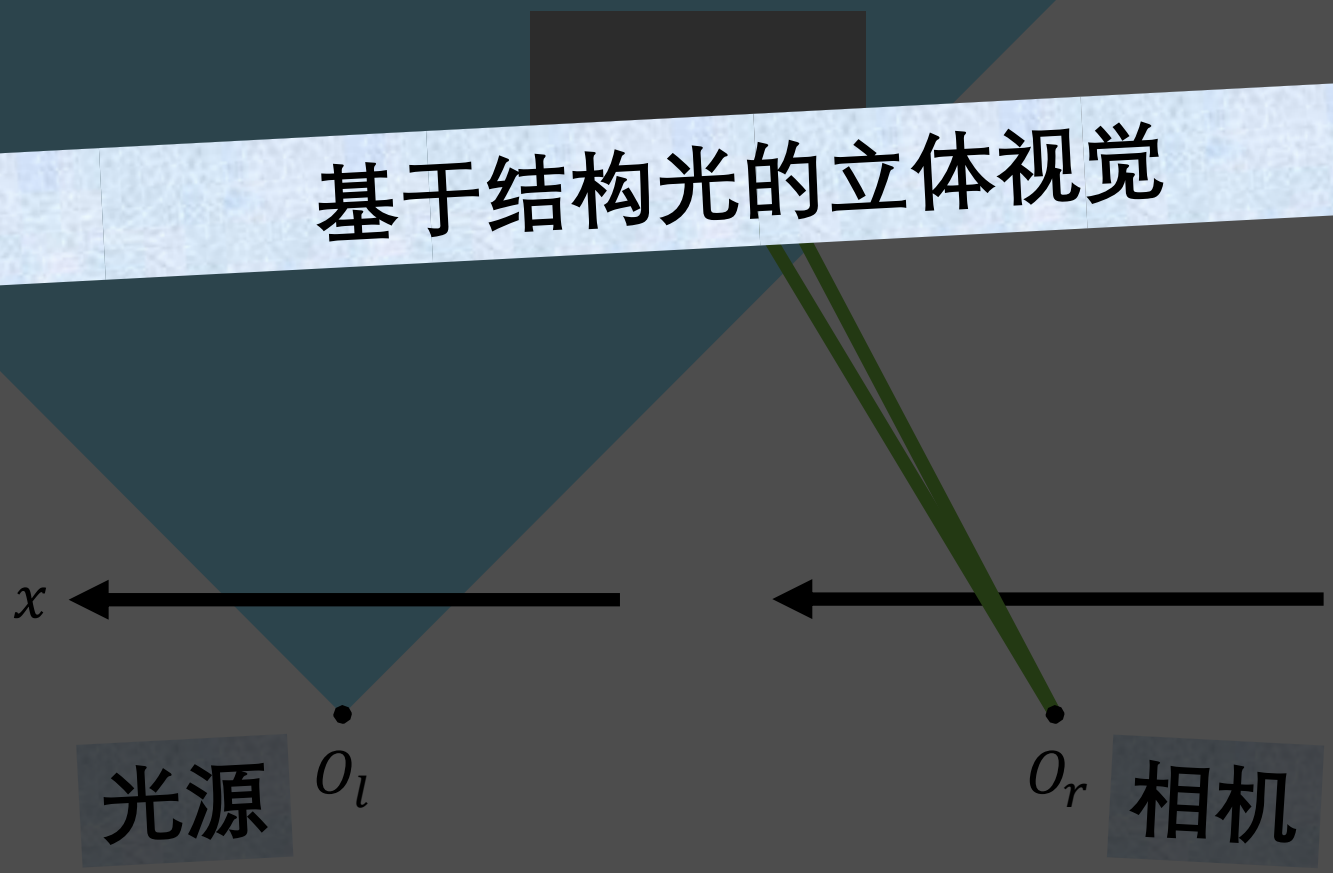

x
光源
O_l
O_r


x
光源
O_l
O_r
相机


x
光源
O_l
O_r
相机

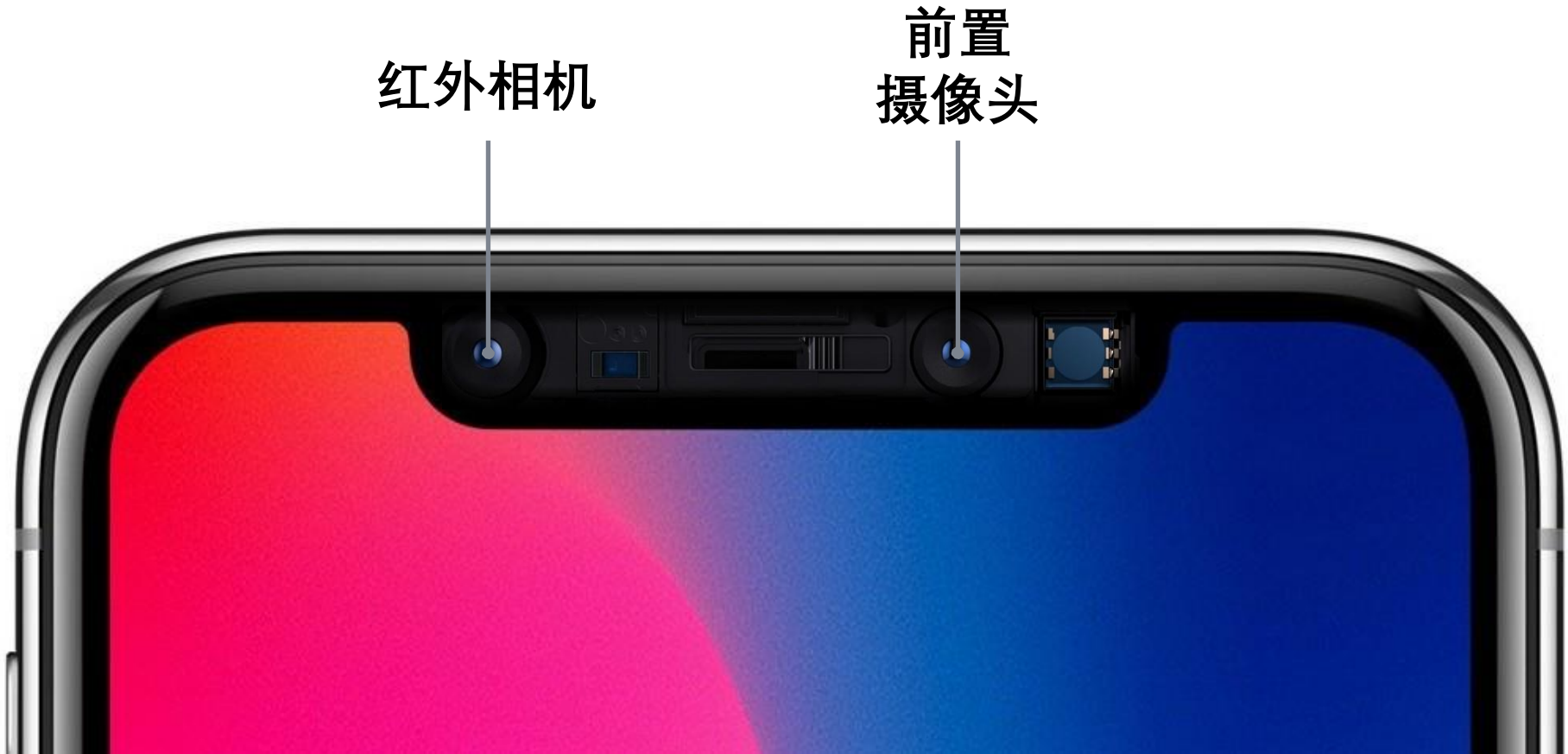

x
光源
O_l
O_r
相机

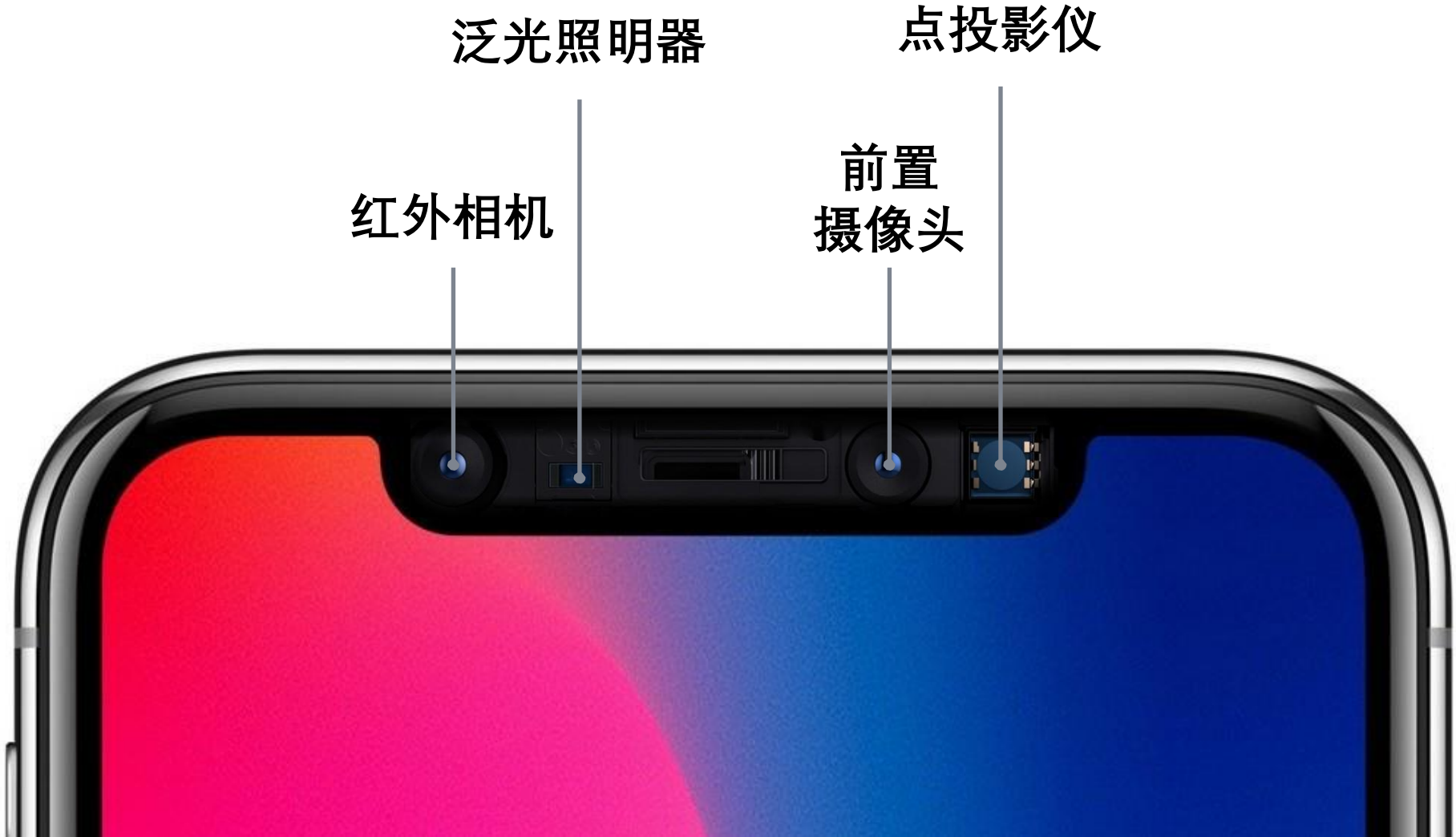
基于结构光的立体视觉

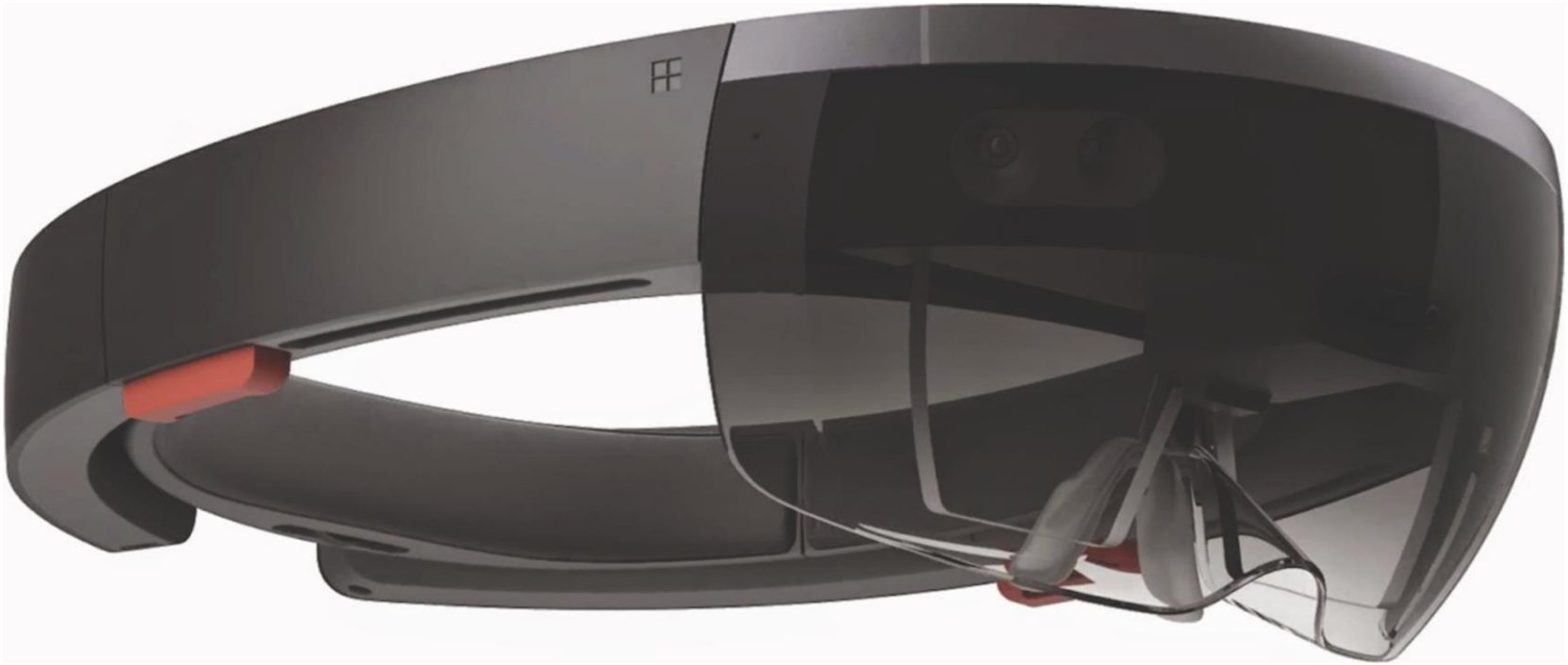


Kinect的红外视图

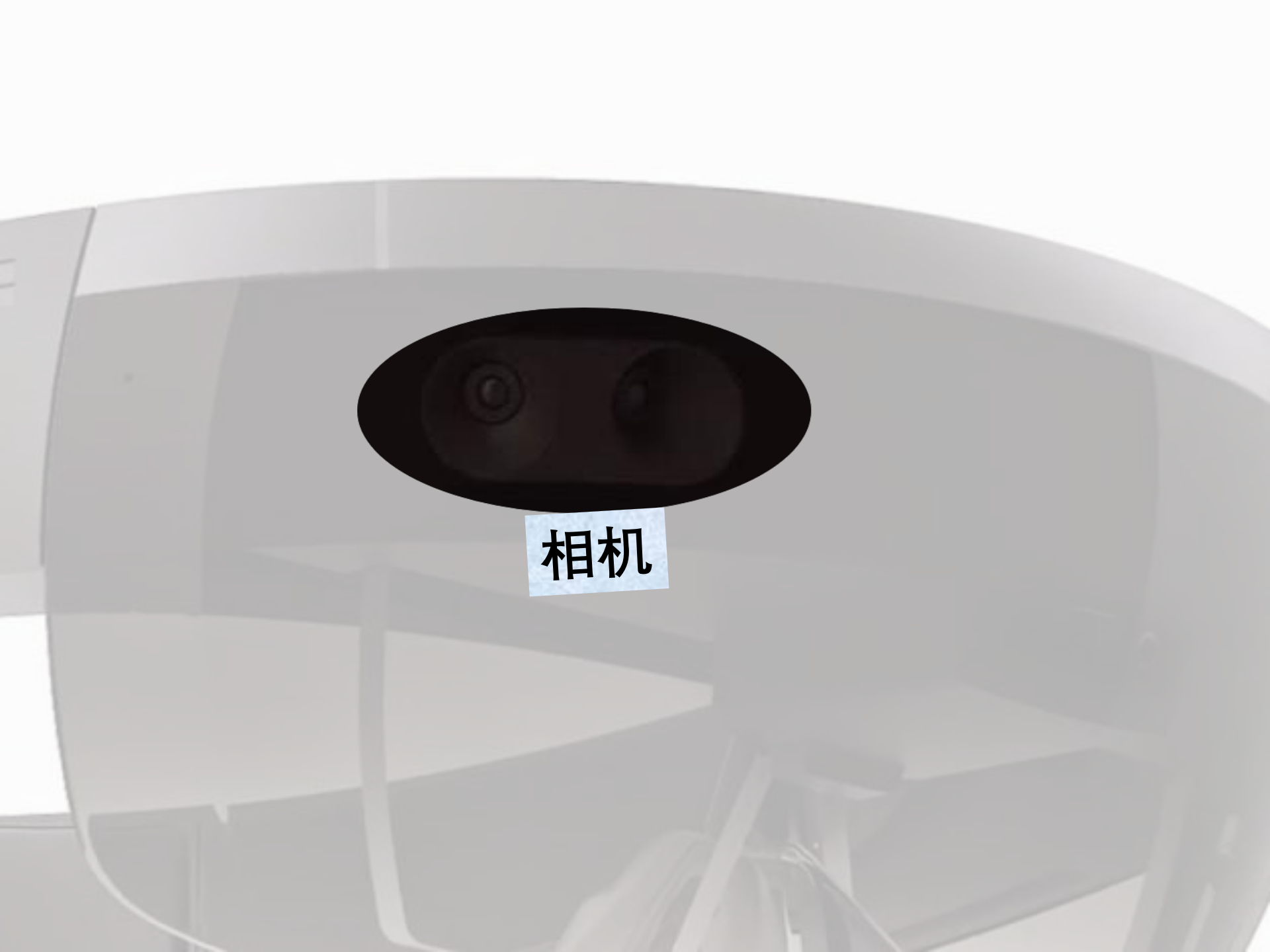
iPhone X


红外相机
前置
摄像头


泛光照明器
点投影仪
红外相机
前置
摄像头



Microsoft
HoloLens


相机

友情听译：译制组
科技时光
微博关注 http://weibo.com/u/5561546480
听译：昀雨

holoportation

http://research.microsoft.com/holoportation

Interactive 3D Technologies

http://research.microsoft.com/groups/i3d

Microsoft Research