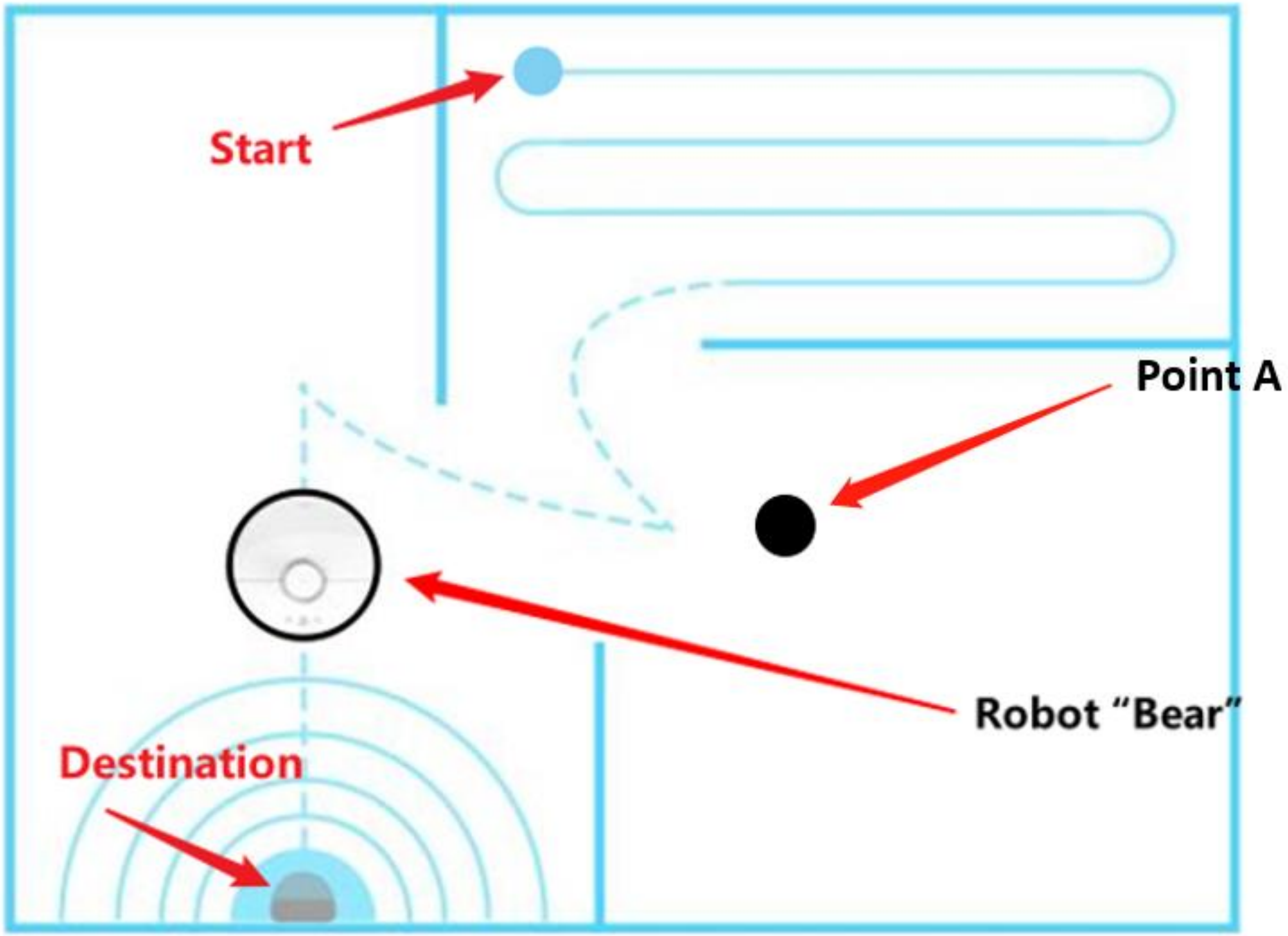
什么是Fundamental Matrix?

- **基础矩阵**
- 本质矩阵
- 单应矩阵

运动示意图



运动描述

- 机器人“Bear”从起点运动到终点
- 通过搭载的摄像头观测一点**路标点A**

运动分析

- 机器人“Bear”转弯：**Rotation**
- 机器人“Bear”移动：**Translation**

F矩阵含义

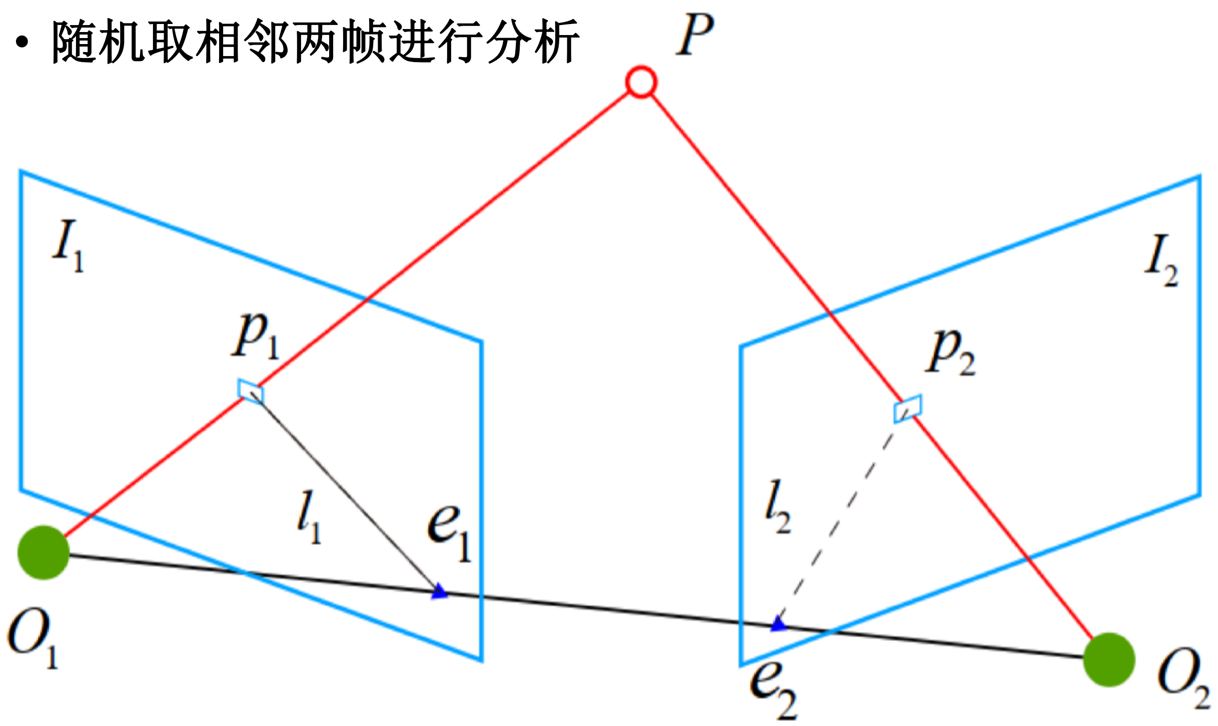
- 相邻两帧一组对应像素点的约束关系
- 像素点在下一帧极线上的位置

推导Fundamental Matrix表达式

基础矩阵
本质矩阵
单应矩阵

运动示意图

- 随机取相邻两帧进行分析



I_1、I_2代表归一化平面

O_1、O_2代表相机光心位置

$$\overrightarrow{O_1O_2} = t \quad \overrightarrow{O_1P} = x_1 \quad \overrightarrow{O_2P} = x_2$$

约束关系

- **O_1、O_2、P**三点共面

$$\boxed{\begin{gathered}\overrightarrow{O_2P} \cdot (\overrightarrow{O_1O_2} \times \overrightarrow{O_1P}) = 0 \\ x_1' = Kx_1 \quad x_2' = Kx_2 \quad x_2 = Rx_1 + t\end{gathered}}$$

- **Fundamental Matrix表达式**

$$x_2'^T \boxed{K^{-T} t_\times R K^{-1}} x_1' = 0$$

$$F = K^{-T} t_\times R K^{-1}$$

极线位置

- **点在线上**

$$\boxed{x_2'^T F x_1' = 0 = x_2'^T l_2}$$

根据匹配点信息求解Fundamental Matrix

八点法前的归一化

- 改进前：八点法各个坐标间相差数量级大，影响估计结果
- 改进方法：引入归一化校正矩阵

$$x' = \frac{x - \bar{x}}{\sigma_x} \Rightarrow x' = \sqrt{2} \cdot \frac{x - \bar{x}}{\sigma_x}$$

- 校正矩阵T的变换关系

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/\sigma_x & 0 & -\sqrt{2}\bar{x}/\sigma_x \\ 0 & \sqrt{2}/\sigma_y & -\sqrt{2}\bar{y}/\sigma_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

八点法的讨论

- 可行性分析

$$X_2'^{\mathrm{T}} \begin{bmatrix} f_{11} & \cdots & f_{13} \\ \vdots & \ddots & \vdots \\ f_{31} & \cdots & f_{33} \end{bmatrix} X_1' = 0$$

$$\Downarrow$$

$$\begin{bmatrix} x_2'^1 x_1'^1 & \cdots & 1 \\ \vdots & \cdots & \vdots \\ x_2'^8 x_1'^8 & \cdots & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ \vdots \\ f_{33} \end{bmatrix} = A \cdot f = 0$$

- **Fundamental Matrix的自由度 = 7**

① **F约束关系属于平面：尺度缩放**

② **Det(F) = 0**

根据匹配点信息求解Fundamental Matrix

基础矩阵

本质矩阵

单应矩阵

最小二乘求解(≥8点)

- 前提结论：

对于行数多于列数的矩阵A，若**Ax = 0，且||x|| = 1**，通过SVD分解得到$A = U\Sigma V^T$，则**x为V矩阵的最后一列**。

- SVD求解过程：

$$A_{[K,9]} \xrightarrow{SVD} U_{[K,K]}\Sigma_{[K,9]}V_{[9,9]}^T$$

$$f = V_{[:,9]} \rightarrow F$$

$$F \xrightarrow{SVD} M\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}N^T \xrightarrow{if\ \ \sigma_3\ \ \min} M\begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}N^T$$

最小二乘求解的讨论

- 可行性分析

$$F = K^{-T}\boxed{t_\times}RK^{-1}$$

$$t_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

- **Fundamental Matrix的秩**

① **Rank(F) = Rank($t_\times$) = 2**

② **Fundamental Matrix仅有两个特征值**

根据匹配点信息求解Fundamental Matrix

基础矩阵

本质矩阵

单应矩阵

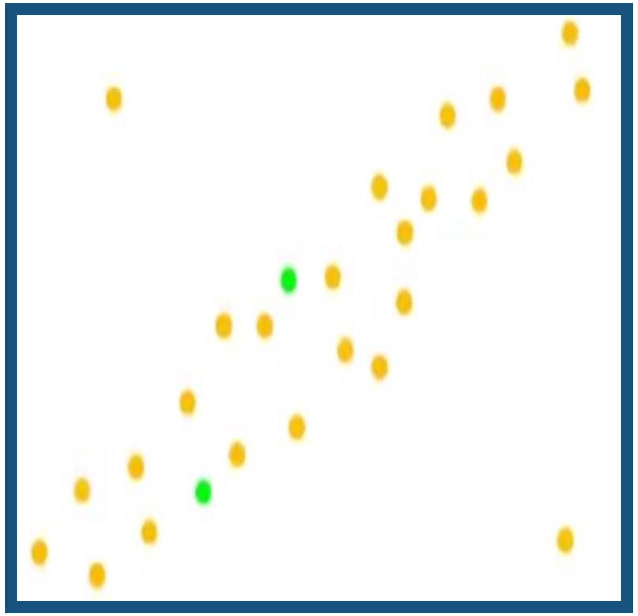
无敌的RANSAC(≥8点)

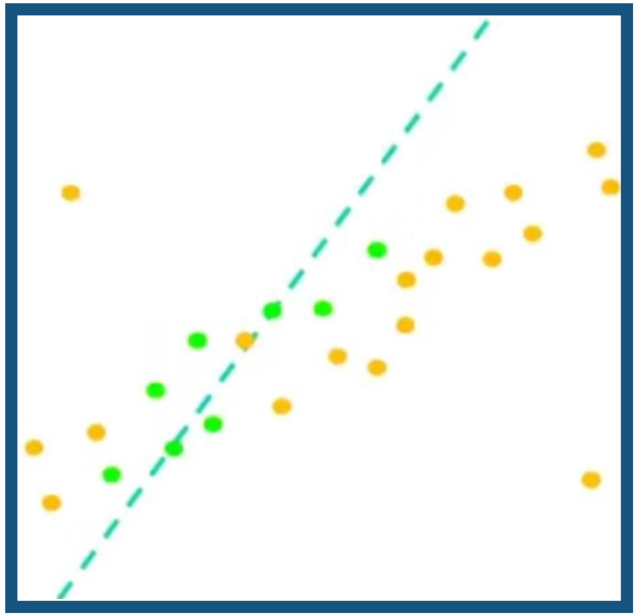
- 如何求解：基于归一化8点法与最小二乘法改进(以直线拟合为例说明RANSAC)

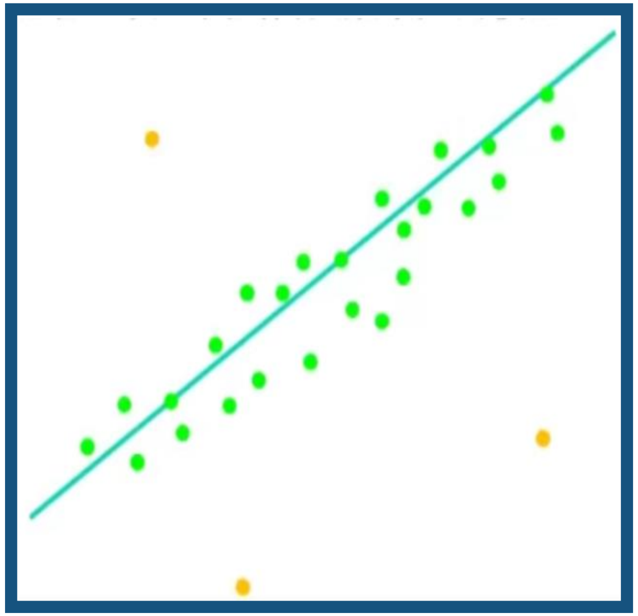
1. 随机选取数据中的两(N)个点

2. 拟合直线
根据阈值统计内外点

3. 搜索包含内点最多模型
基于内点重新拟合







根据匹配点信息求解Fundamental Matrix

RANSAC的讨论

- 阈值设定 and 内点/外点判断

① 阈值：点到拟合直线的距离

② 内点(外点)：距离不超过(超过)阈值的数据点

- 确定采样次数M

前提说明：共有N个数据点，一次采样K个点，采样内点的概率为P

① 一次采样全部是内点：P^K

② 一次采样不全都是内点(采样失败)：$1-P^K$

③ M次采样全部失败：$(1-P^K)^M$

④ M次采样至少有一次采到所有内点：$P_s=1-(1-P^K)^M$

移项然后两侧取对数

- 确定采样次数M

$$M=\frac{\log(1-P_s)}{\log(1-P^K)}$$

根据匹配点信息求解Fundamental Matrix

基础矩阵

本质矩阵

单应矩阵

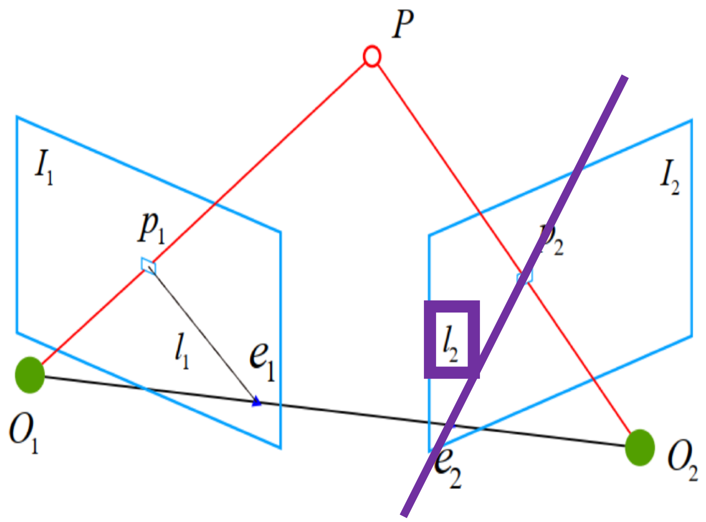
RANSAC的讨论

- 基于RANSAC的Fundamental Matrix求解规则

① 参数(假设)：有N = 500个点，一次采样K = 8个点，采样到内点概率P = 0.9(我随便编的)

若想要$P_s \geq 0.99$，则M应为？

① 如何判断内点/外点：设定点到直线距离的阈值，若x_2'到极线l_2距离不超过阈值则记为内点



$$x_2'^T F x_1' = 0 = x_2'^T l_2$$

$$\Downarrow$$

$$F x_1' = l_2$$

根据匹配点信息求解Fundamental Matrix

RANSAC的讨论

- 基于RANSAC的Fundamental Matrix求解步骤

① 随机选取8对匹配点，应用归一化八点法求解F_init

② 计算其余点到极线的距离，不超出距离阈值的点记为内点，反之记为外点

③ 统计此次内点总数，开始下一次随机采样与求解

④ 重复步骤1~3共M次(或者某次内点统计总数超过95%)，选取内点总数最多的模型

⑤ 应用最小二乘方法对上述内点处理，估计得到最终F矩阵

根据匹配点信息求解Fundamental Matrix

基础矩阵

本质矩阵

单应矩阵

OpenCV中的函数

findFundamentalMat() [1/3]

```
Mat cv::findFundamentalMat ( InputArray   points1,
                             InputArray   points2,
                             int          method,
                             double       ransacReprojThreshold,
                             double       confidence,
                             int          maxIters,
                             OutputArray  mask = noArray()
                           )
```

- method → 选用方法:当然是RANSAC
- ransacReprojThreshold → RANSAC的距离阈值
- confidence → RANSAC的Ps
- maxIters → RANSAC的迭代次数
- mask → RANSAC的内点索引结果

findFundamentalMat() [2/3]

```
Mat cv::findFundamentalMat ( InputArray   points1,
                             InputArray   points2,
                             int          method = FM_RANSAC,
                             double       ransacReprojThreshold = 3.,
                             double       confidence = 0.99,
                             OutputArray  mask = noArray()
                           )
```

→ 完全使用RANSAC

由Fundamental得到Essential Matrix

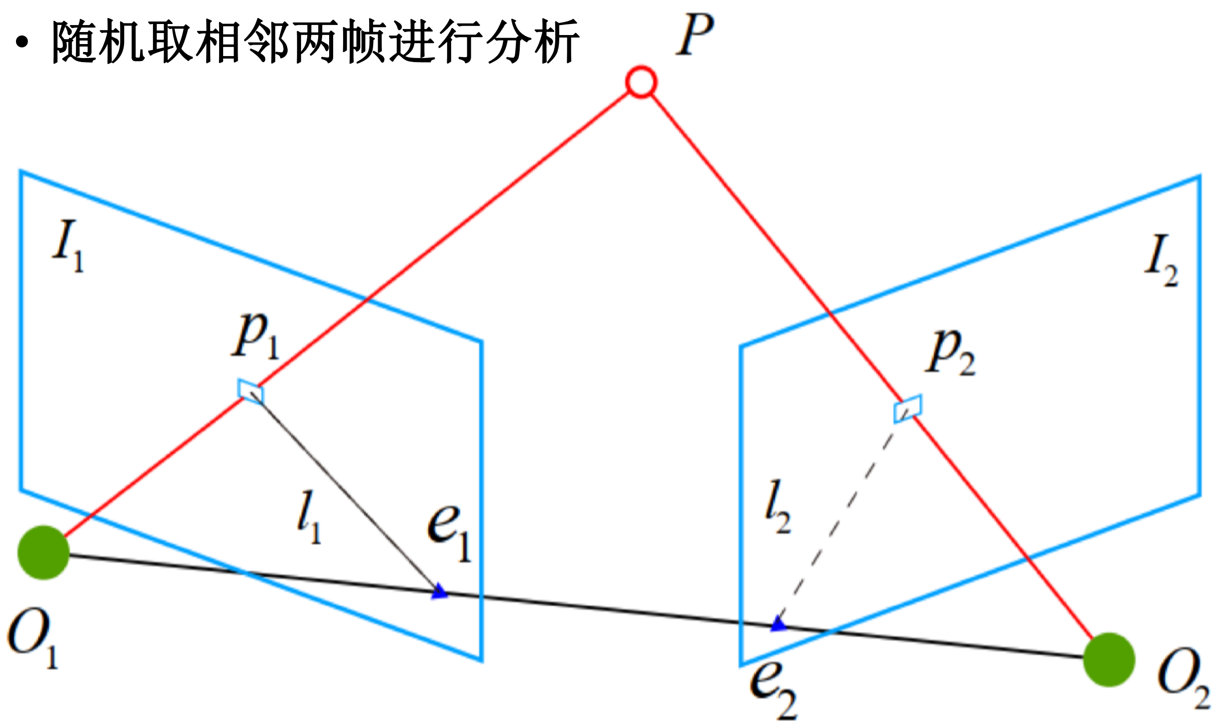
基础矩阵

本质矩阵

单应矩阵

运动示意图

- 随机取相邻两帧进行分析



I_1、I_2代表归一化平面

O_1、O_2代表相机光心位置

$$\overrightarrow{O_1O_2} = t \quad \overrightarrow{O_1P} = x_1 \quad \overrightarrow{O_2P} = x_2$$

约束关系

- O_1、O_2、P三点共面

$$\boxed{\begin{gathered}\overrightarrow{O_2P} \cdot (\overrightarrow{O_1O_2} \times \overrightarrow{O_1P}) = 0 \\ x_1' = Kx_1 \quad x_2' = Kx_2 \quad x_2 = Rx_1 + t\end{gathered}}$$

- Essential Matrix表达式

$$x_2^T \boxed{t_\times R} x_1 = 0$$

$$E = t_\times R$$

点面关系

- 点在面上

$$\boxed{x_2^T E x_1 = 0 = x_2{}^T \pi_{O_1O_2P}}$$

根据匹配点信息求解Essential Matrix

基础矩阵
本质矩阵
单应矩阵

具体方法(不展开)

- 八点法：同Fundamental Matrix(无需归一化)

解释：通过内参矩阵K，所有坐标统一在像素坐标系下

- 最小二乘法(≥8对点)：同Fundamental Matrix

- RANSAC大法：同Fundamental Matrix

注意有一点不同：

定义距离阈值时，求解E时需要设定点到平面O_1O_2P的距离

关于E矩阵的讨论

- Essential Matrix的自由度

① 仅考虑矩阵元素：8自由度(尺度缩放)

② 从定义式考虑：5自由度(3 + 3 - 1)

- Essential Matrix的秩

① $\mathrm{Rank}(E) = \mathrm{Rank}(t_\times) = 2$

② Essential Matrix仅有两个特征值

根据匹配点信息求解Essential Matrix

基础矩阵

本质矩阵

单应矩阵

OpenCV中的函数

findEssentialMat() [1/2]

```
Mat cv::findEssentialMat ( InputArray   points1,
                           InputArray   points2,
                           InputArray   cameraMatrix,
                           int          method = RANSAC,
                           double       prob = 0.999,
                           double       threshold = 1.0,
                           OutputArray  mask = noArray()
                         )
```

- cameraMatrix → 相机内参
- method = RANSAC → 选用方法:当然是RANSAC
- prob = 0.999 → RANSAC的Ps
- threshold = 1.0 → RANSAC的距离阈值
- mask = noArray() → RANSAC的内点索引结果

findEssentialMat() [2/2]

```
Mat cv::findEssentialMat ( InputArray   points1,
                           InputArray   points2,
                           double       focal = 1.0,
                           Point2d      pp = Point2d(0, 0),
                           int          method = RANSAC,
                           double       prob = 0.999,
                           double       threshold = 1.0,
                           OutputArray  mask = noArray()
                         )
```

- double focal = 1.0 → 相机内参中的fx fy
- Point2d pp = Point2d(0, 0) → 相机内参中的cx cy

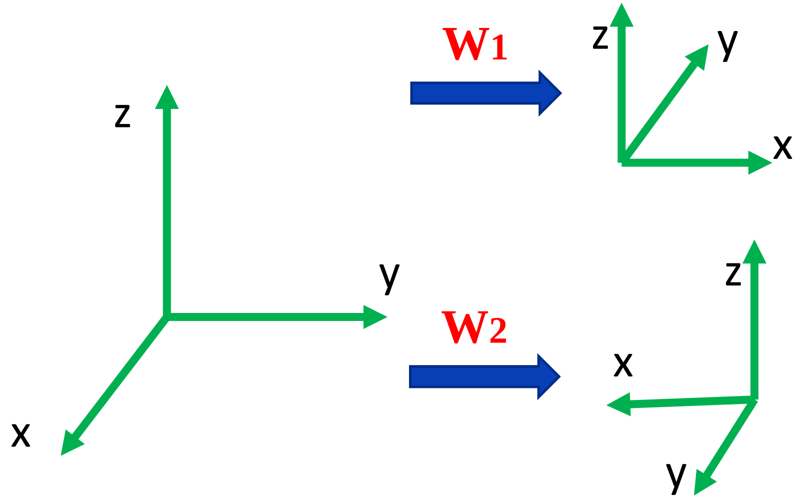
根据Essential Matrix恢复位姿信息R与t

Rotation Matrix

- 前提理论：

$$t_\times = UBU^T, B = \begin{bmatrix} 0 & a & 0 \\ -a & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- 实际推导：



$$B = Diag\left(|a|,|a|,0\right)\cdot W = Diag\left(|a|,|a|,0\right) \boxed{\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}} / \boxed{\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}$$

W1 W2

$$\left.\begin{aligned} E &= t_\times R = \underline{U} \cdot \underline{Diag(|a|,|a|,0)} \cdot \underline{WU^T} R \\ E &\xrightarrow{SVD} U \cdot \Sigma \cdot V^T \end{aligned}\right\} \quad R = UW^T V^T$$

根据Essential Matrix恢复位姿信息R与t

Translation Vector

- **前提理论：**

对于行数多于列数的矩阵A，若Ax = 0，且||x|| = 1，通过SVD分解得到A = UΣV^T，则x为V矩阵的最后一列。

- **实际推导：**

$$B = Diag\left(|a|,|a|,0\right)\cdot W = Diag\left(|a|,|a|,0\right)\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

W1 **W2**

$$t_\times t = UBU^T \cdot t = M \cdot t = 0, \quad \|t\| = 1$$

$$\left.\begin{aligned} M &= UBU^T = \underline{U} \cdot \underline{Diag(|a|,|a|,0)} \cdot \underline{W U^T} \\ M &\xrightarrow{SVD} U \cdot \Sigma \cdot V^T \end{aligned}\right\} \quad t = UW^T_{[:,3]}$$

根据Essential Matrix恢复位姿信息R与t

基础矩阵

本质矩阵

单应矩阵

OpenCV中的函数

◆ recoverPose() [1/3]

```
int cv::recoverPose ( InputArray        E,
                      InputArray        points1,
                      InputArray        points2,
                      InputArray        cameraMatrix,
                      OutputArray       R,
                      OutputArray       t,
                      InputOutputArray  mask = noArray()
                    )
```

cameraMatrix → 相机内参

mask = noArray() → 非空则代表启用内点

◆ recoverPose() [3/3]

```
int cv::recoverPose ( InputArray        E,
                      InputArray        points1,
                      InputArray        points2,
                      InputArray        cameraMatrix,
                      OutputArray       R,
                      OutputArray       t,
                      double            distanceThresh,
                      InputOutputArray  mask = noArray(),
                      OutputArray       triangulatedPoints = noArray()
                    )
```

distanceThresh → 滤除超出距离阈值的点

triangulatedPoints = noArray() → 自动返回三角化结果

根据Essential Matrix估计深度(三角化)

基础矩阵

本质矩阵

单应矩阵

Triangulate

- 最小二乘无处不在：

$$s_2 x_2 = s_1 R x_1 + t$$

左侧叉乘 x_2

$$s_2 (x_2 \times x_2) = \boxed{0} = s_1 x_2 \times R x_1 + x_2 \times t$$

考虑噪声干扰，这里不一定为**0**

$$\boxed{\arg\min \quad t - \left(s_2 x_2 - s_1 R x_1\right)}$$

◆ triangulatePoints()

```
void cv::triangulatePoints ( InputArray   projMatr1,
                             InputArray   projMatr2,
                             InputArray   projPoints1,
                             InputArray   projPoints2,
                             OutputArray  points4D
                           )
```

- projMatr1 → 第一批点 相机与世界坐标系的转换矩阵
- projMatr2 → 第二批点 相机与世界坐标系的转换矩阵
- projPoints1 → 第一批点 归一化坐标系
- projPoints2 → 第二批点 归一化坐标系
- points4D → 三角化结果

Homography Matrix的意义

基础矩阵

本质矩阵

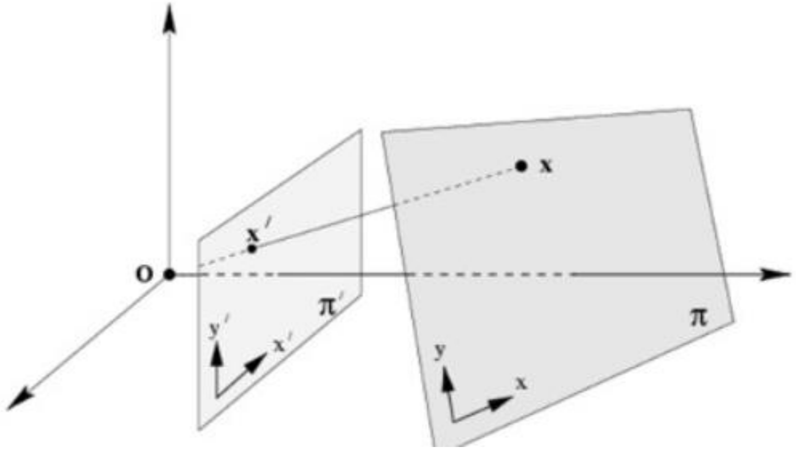
单应矩阵

意义一

- 描述不同平面上一对点的对应关系

(或者特征点都落在同一平面上)

$$\boxed{p' = Hp}$$

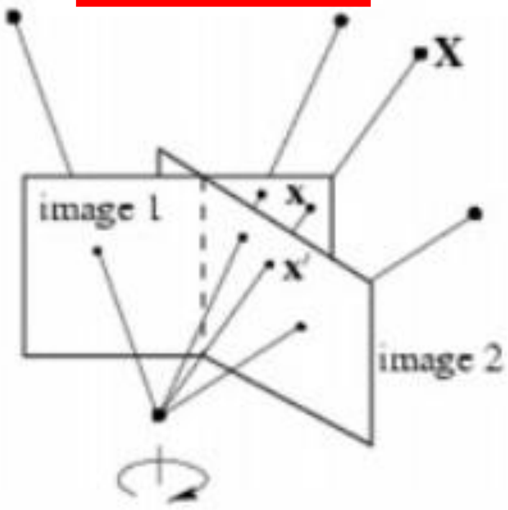


应用：贴图

意义二

- 相机纯旋转问题

$$\boxed{p' = Hp}$$



应用：图像拼接

根据Essential Matrix恢复位姿信息R与t

基础矩阵

本质矩阵

单应矩阵

OpenCV中的函数

findHomography() [1/2]

```
Mat cv::findHomography ( InputArray   srcPoints,
                         InputArray   dstPoints,
                         int          method = 0,
                         double       ransacReprojThreshold = 3,
                         OutputArray  mask = noArray(),
                         const int    maxIters = 2000,
                         const double confidence = 0.995
                       )
```

findHomography() [2/2]

```
Mat cv::findHomography ( InputArray   srcPoints,
                         InputArray   dstPoints,
                         OutputArray  mask,
                         int          method = 0,
                         double       ransacReprojThreshold = 3
                       )
```