深蓝学院
shenlanxueyuan.com

# 基于图像的三维模型重建

## 第2节 帧间运动估计

# 课程内容

深蓝学院
shenlanxueyuan.com

- ✓ **三角化(Triangulation)**
  - ✓ 直接线性变换法
  - ✓ RANSAC鲁棒估计
- ✓ **3D-2D: PnP问题**
  - ✓ 直接线性变换法
  - ✓ P3P/EPnP
- ✓ **捆绑调整Bundle Adjustment**

# 三角量测

已知**相机参数**和**匹配点**，恢复三维点的坐标

第 $i$个相机投影矩阵：

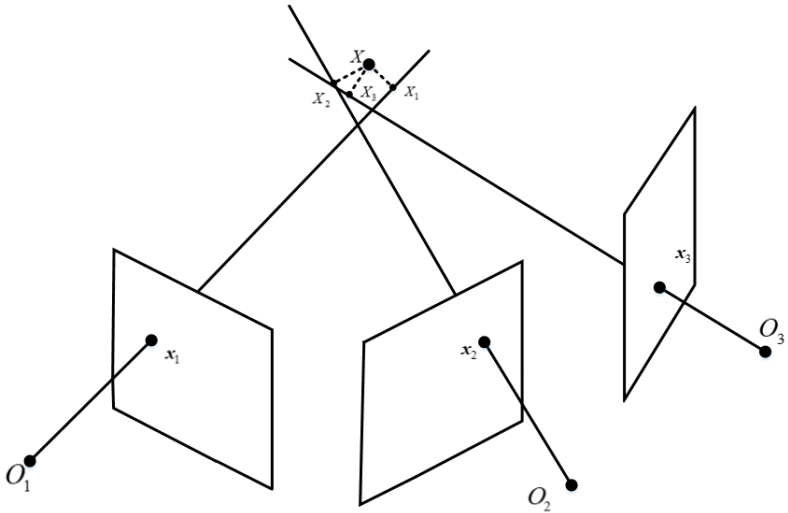
$$\boldsymbol{P}_{\mathrm{i}} = \boldsymbol{K}_i[\boldsymbol{R}_i, \boldsymbol{t}_i] = \begin{bmatrix} \boldsymbol{P}_{i1} \\ \boldsymbol{P}_{i2} \\ \boldsymbol{P}_{i3} \end{bmatrix}$$

空间三维点坐标：

$$\boldsymbol{X} = [x, y, z, 1]^T$$

对应在第 $i$个视角中投影的图像坐标为：

$$\boldsymbol{x}_i = [x_i, y_i, 1]^T$$



# 三角量测



已知**相机参数**和**匹配点**，恢复三维点的坐标

根据投影方程可以得到：

$$d_i \boldsymbol{x}_i = \boldsymbol{P}_i \boldsymbol{X},$$

上述等式两侧同时叉乘 $\boldsymbol{x}_i$：

$$\boldsymbol{x}_i \times (\boldsymbol{P}_i \boldsymbol{X}) = \boldsymbol{0}$$

其中，

$$\boldsymbol{x}_i = [x_i, y_i, 1]^T$$

$$\boldsymbol{P}_i \boldsymbol{X} = \begin{bmatrix} \boldsymbol{P}_{i1} \\ \boldsymbol{P}_{i2} \\ \boldsymbol{P}_{i3} \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} \boldsymbol{P}_{i1}\boldsymbol{X} \\ \boldsymbol{P}_{i2}\boldsymbol{X} \\ \boldsymbol{P}_{i3}\boldsymbol{X} \end{bmatrix}$$ 3行1列

根据向量叉积的运算，展开得

$$\begin{aligned} x_i(\boldsymbol{P}_{i3}\boldsymbol{X}) - \boldsymbol{P}_{i1}\boldsymbol{X} &= \boldsymbol{0} \\ y_i(\boldsymbol{P}_{i3}\boldsymbol{X}) - \boldsymbol{P}_{i2}\boldsymbol{X} &= \boldsymbol{0} \\ x_i(\boldsymbol{P}_{i2}\boldsymbol{X}) - y_i(\boldsymbol{P}_{i1}\boldsymbol{X}) &= \boldsymbol{0} \end{aligned}$$

$$\begin{bmatrix} x_i\boldsymbol{P}_{i3} - \boldsymbol{P}_{i1} \\ y_i\boldsymbol{P}_{i3} - \boldsymbol{P}_{i2} \end{bmatrix} \boldsymbol{X} = \boldsymbol{0}$$

注意：第3个方程与前两个方程线性相关。具体地，第三个等式等于第二个等式乘以$x_i$减第一个等式乘以$y_i$。

# 三角量测



已知**相机参数**和**匹配点**，恢复三维点的坐标

1个观测点提供2个约束，$\boldsymbol{X}$有3个自由度，求解该方程至少2对点（从几何意义上解释，两条射线相交可确定空间中的一个三维点）。

$$\boldsymbol{AX} = \boldsymbol{0},$$

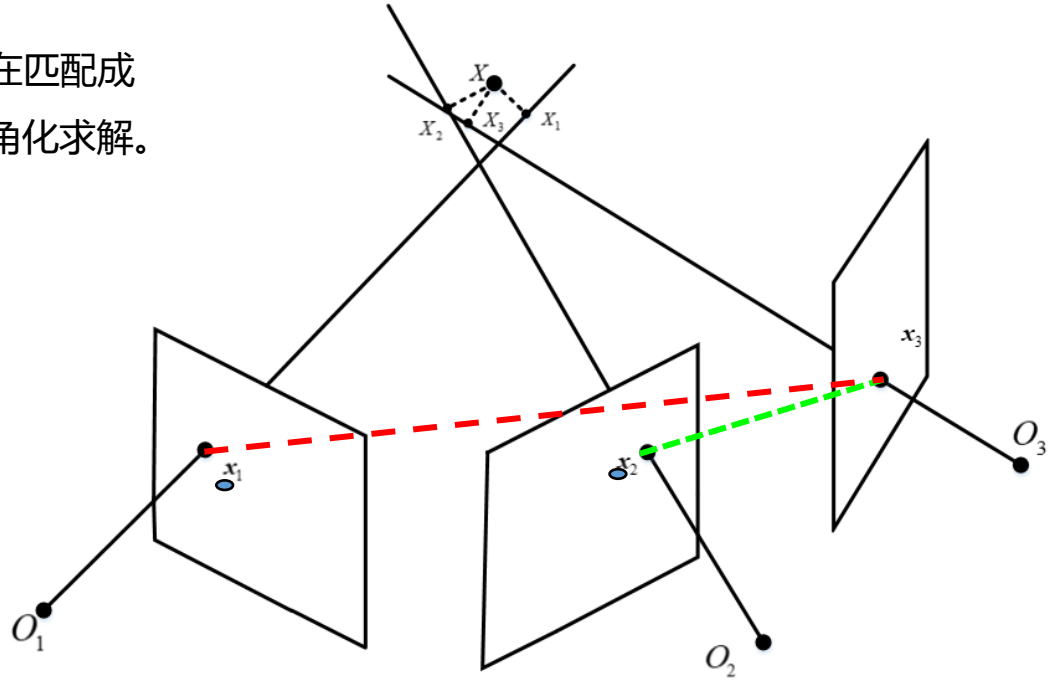
$$\boldsymbol{A}=\begin{bmatrix} x_1\boldsymbol{P}_{13} - \boldsymbol{P}_{11} \\ y_1\boldsymbol{P}_{13} - \boldsymbol{P}_{12} \\ \cdots \\ x_i\boldsymbol{P}_{i3} - \boldsymbol{P}_{i1} \\ y_i\boldsymbol{P}_{i3} - \boldsymbol{P}_{i2} \\ \cdots \\ x_N\boldsymbol{P}_{N3} - \boldsymbol{P}_{N1} \\ y_N\boldsymbol{P}_{N3} - \boldsymbol{P}_{N2} \end{bmatrix}, \boldsymbol{N} \geq \boldsymbol{2}$$

# 三角量测



已知**相机参数**和**匹配点**，恢复三维点的坐标

宽基线（相机间的距离）求解更加稳定，在匹配成功的点对中选择宽基线中的匹配对进行三角化求解。



# 三角量测



## 思考

如果存在外点（匹配点或姿态）的情况下，如何得到准确的的三维点坐标?

**基于RANSAC的三角化算法流程**

1）计算RANSAC采样次数，设置内点阈值（重投影误差）；

2）随机采样一对视角，计算三维点坐标；

3）计算每个视角中的重投影误差，统计内点个数；

4）重复2), 3)直到满足采样次数，选择内点数最多的视角；

5）利用所有内点重新计算三维点坐标。

# 三角量测

深蓝学院
shenlanxueyuan.com

## Coding

完成task-2/task2-1_test_triangulation.cc中利用直接线性变化法求三维点坐标。

# 课程内容

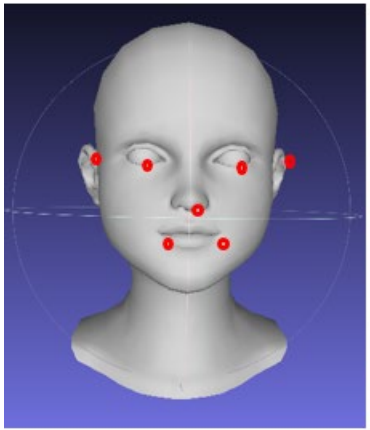
深蓝学院
shenlanxueyuan.com

- ✓三角化(Triangulation)
  - ✓直接线性变换法
  - ✓RANSAC鲁棒估计
- ✓ **3D-2D: PnP问题**
  - ✓直接线性变换法
  - ✓P3P/EPnP
- ✓ 捆绑调整Bundle Adjustment

# 3D-2D: PnP问题-PnP

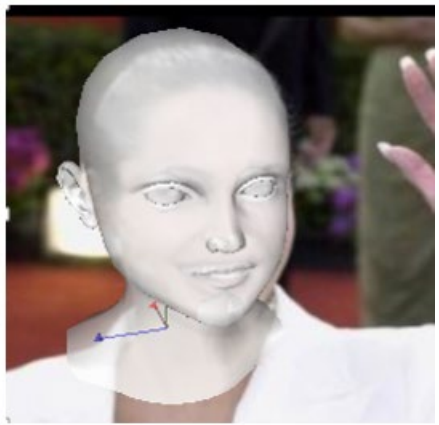


已知**三维点**和**对应二维点**，求解相机内外参数

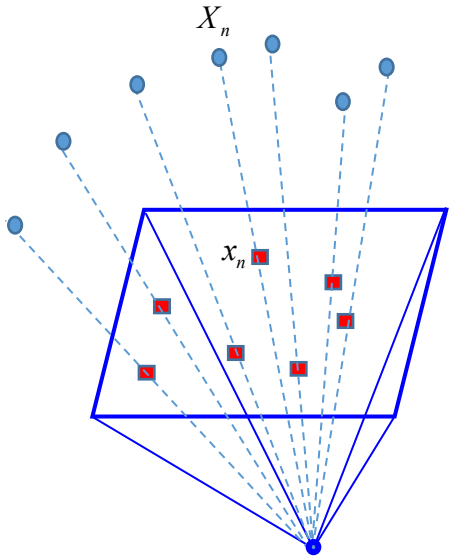


已知3D点坐标

对应的2D点坐标（紫色）
以及重投影点（红色）



优化后的人脸姿态实现的AR效果




给定内参矩阵 $K$

外参矩阵 $R,t$?

# 3D-2D: PnP问题-直接线性变换法

已知**三维点**和**对应二维点**，求解相机内外参数

$$s\begin{pmatrix}u\\v\\1\end{pmatrix}=\boldsymbol{K}\left[\boldsymbol{R},\boldsymbol{t}\right]\boldsymbol{X}=\boldsymbol{K}\left[\boldsymbol{R},\boldsymbol{t}\right]\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{bmatrix}T_{11}, & T_{12}, & T_{13}, & T_{14}\\T_{21}, & T_{22}, & T_{23}, & T_{24}\\T_{31}, & T_{32}, & T_{33}, & T_{34}\end{bmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}\boldsymbol{r}_1^T\\\boldsymbol{r}_2^T\\\boldsymbol{r}_3^T\end{pmatrix}\boldsymbol{X}$$

$$u=\frac{T_{11}X+T_{12}Y+T_{13}Z+T_{14}}{T_{31}X+T_{32}Y+T_{33}Z+T_{34}}=\frac{\boldsymbol{r}_1^T\boldsymbol{X}}{\boldsymbol{r}_3^T\boldsymbol{X}}=\frac{\boldsymbol{X}^T\boldsymbol{r}_1}{\boldsymbol{X}^T\boldsymbol{r}_3}$$

标量

$$v=\frac{T_{21}X+T_{22}Y+T_{23}Z+T_{24}}{T_{31}X+T_{32}Y+T_{33}Z+T_{34}}=\frac{\boldsymbol{r}_2^T\boldsymbol{X}}{\boldsymbol{r}_3^T\boldsymbol{X}}=\frac{\boldsymbol{X}^T\boldsymbol{r}_2}{\boldsymbol{X}^T\boldsymbol{r}_3}$$

$$\boldsymbol{X}^T\boldsymbol{r}_1-\boldsymbol{X}^T\boldsymbol{r}_3u=0$$

$$\boldsymbol{X}^T\boldsymbol{r}_2-\boldsymbol{X}^T\boldsymbol{r}_3v=0$$

# 3D-2D: PnP问题-直接线性变换法



已知**三维点**和**对应二维点**，求解相机内外参数

$$\begin{pmatrix} \boldsymbol{X}_1^T & 0 & -u\boldsymbol{X}_1^T \\ 0 & \boldsymbol{X}_1^T & -v\boldsymbol{X}_1^T \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^T & 0 & -u\boldsymbol{X}_N^T \\ 0 & \boldsymbol{X}_N^T & -v\boldsymbol{X}_N^T \end{pmatrix} \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \boldsymbol{r}_3 \end{pmatrix} = \mathbf{0}$$

一共12个未知参数，每对3D-2D对应点提供2个线性约束，因此共需要至少**6对3D-2D对应点**，才可求得$\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3$，即求得矩阵 $\boldsymbol{T} = [\boldsymbol{KR}, \boldsymbol{Kt}]$

**求解内参及旋转**

已知 $\boldsymbol{T}(:, 1:3) = \boldsymbol{KR}$，且内参矩阵$\boldsymbol{K}$为上三角矩阵，$\boldsymbol{R}$为正交矩阵。

对矩阵$\boldsymbol{T}(:, 1:3)^{-1}$进行QR分解，分别得到$\boldsymbol{R}^{-1}$以及$\boldsymbol{K}^{-1}$，进而得到内参矩阵$\boldsymbol{K}$以及旋转矩阵$\boldsymbol{R}$。

**求解平移**

已知 $\boldsymbol{T}(:, 4) = \boldsymbol{Kt}$，以及内参矩阵$\boldsymbol{K}$，可得平移向量$\boldsymbol{t}$。

# 3D-2D: PnP问题-Kneip算法



**A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation**

Laurent Kneip laurent.kneip@mavt.ethz.ch
Davide Scaramuzza davide.scaramuzza@mavt.ethz.ch
Roland Siegwart rsiegwart@ethz.ch
Autonomous Systems Lab, ETH Zurich

- P3p: 从3对3D-2D的对应点中确定相机的朝向和位置
- 通常会产生4对解，需要用第4对匹配关系确定
- 一般的求解思路是首先计算点在相机中的3D坐标，然后通过ICP的方式计算相机姿态
- Kneip是一种close-form的P3p求解方式，主要思想是引入相机和世界坐标系的中间坐标系

计算它们之间的相对姿态和位置来得到相机的姿态，Kneip的优势是求解稳定、速度快

# 3D-2D: PnP问题-Kneip算法

## 创建新的相机坐标系

世界坐标系 $(O, X, Y, Z)$

世界坐标系中的3个点 $P_1, P_2, P_3$

相机坐标系$\nu$ 中的单位向量$\vec{f}_1$ $\vec{f}_2$ $\vec{f}_3$

创建新的相机坐标系：$\tau = (C, \vec{t}_x, \vec{t}_y, \vec{t}_z)$

$$\begin{aligned} \vec{t}_x &= \vec{f}_1 \\ \vec{t}_z &= \frac{\vec{f}_1 \times \vec{f}_2}{||\vec{f}_1 \times \vec{f}_2||} \\ \vec{t}_y &= \vec{t}_z \times \vec{t}_x. \end{aligned}$$

通过旋转变换 $T = [\vec{t}_x, \vec{t}_y, \vec{t}_z]^T$ 将单位向量由 $\nu$ 转换到 $\tau$

$$\vec{f}_i^{\tau} = T \cdot \vec{f}_i$$

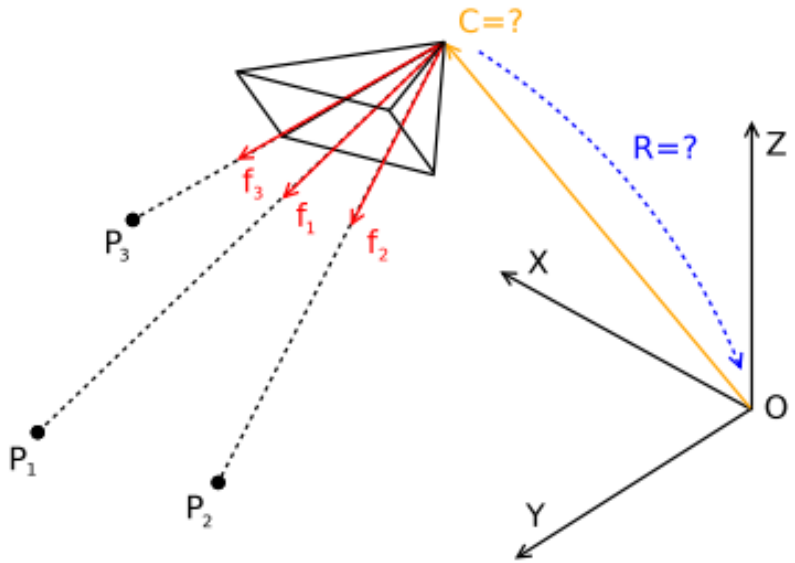



Figure 1. Synopsis of the problem.

# 3D-2D: PnP问题-Kneip算法

深蓝学院 shenlanxueyuan.com

## 创建新的世界坐标系

创建新的世界坐标系 $\eta = (P_1, \vec{n}_x, \vec{n}_y, \vec{n}_z)$

$$
\begin{aligned}
\vec{n}_x &= \frac{\overrightarrow{P_1P_2}}{\| \overrightarrow{P_1P_2} \|} \\
\vec{n}_z &= \frac{\vec{n}_x \times \overrightarrow{P_1P_3}}{\| \vec{n}_x \times \overrightarrow{P_1P_3} \|} \\
\vec{n}_y &= \vec{n}_z \times \vec{n}_x .
\end{aligned}
$$

通过旋转矩阵 $N = [\vec{n}_x, \vec{n}_y, \vec{n}_z]^T$ 将世界坐标系的点转换到 $\eta$ 坐标系下

$$P_i^{\eta} = N \cdot (P_i - P_1).$$

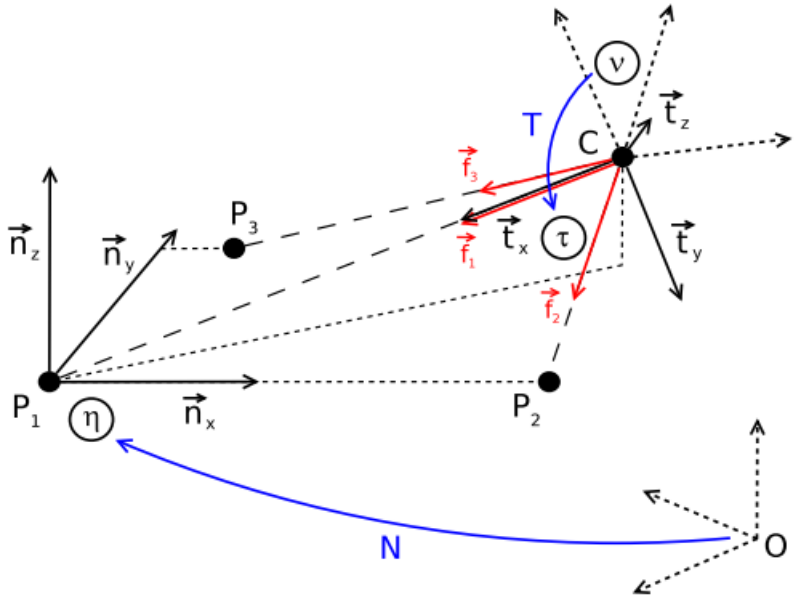



Figure 2. Illustration of the intermediate camera frame $\tau = (C, \vec{t}_x, \vec{t}_y, \vec{t}_z)$ and the intermediate world frame $\eta = (P_1, \vec{n}_x, \vec{n}_y, \vec{n}_z)$.

# 3D-2D: PnP问题-Kneip算法

## 相机中心在平面中的表达

$P_1, P_2, C$ 三个点构成一个平面 $\Pi$　　定义 $\beta = \vec{f}_1 \cdot \vec{f}_2$

根据几何约束 $\frac{\| \overrightarrow{CP_1} \|}{d_{12}} = \frac{\sin(\pi - \alpha - \beta)}{\sin \beta}$

相机中心 $C$ 在平面 $\Pi$ 中可以表达为

$$
\begin{aligned}
C^{\Pi}(\alpha) &= \begin{pmatrix} \cos\alpha \cdot \| \overrightarrow{CP_1} \| \\ \sin\alpha \cdot \| \overrightarrow{CP_1} \| \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} d_{12} \cos\alpha \sin(\alpha+\beta) \sin^{-1}\beta \\ d_{12} \sin\alpha \sin(\alpha+\beta) \sin^{-1}\beta \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} d_{12} \cos\alpha (\sin\alpha \cot\beta + \cos\alpha) \\ d_{12} \sin\alpha (\sin\alpha \cot\beta + \cos\alpha) \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\Rightarrow C^{\Pi}(\alpha) = \begin{pmatrix} d_{12} \cos\alpha (\sin\alpha \cdot b + \cos\alpha) \\ d_{12} \sin\alpha (\sin\alpha \cdot b + \cos\alpha) \\ 0 \end{pmatrix}
$$

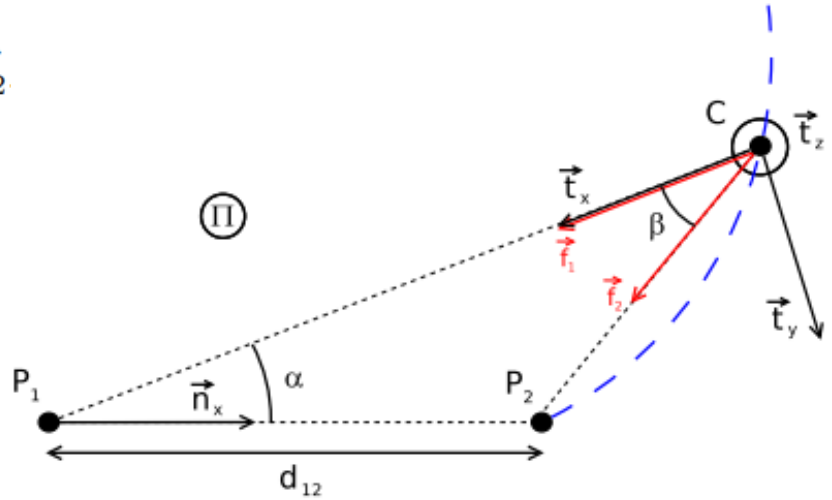



Figure 3. Semi-plane $\Pi$ containing the triangle $(P_1, P_2, C)$. The blue trajectory indicates the possible locations of the camera centre $C$ depending on the free parameter $\alpha$, and the fixed parameters $d_{12}$ and $\beta$.

# 3D-2D: PnP问题-Kneip算法

## 相机中心和姿态在新建世界坐标系中表达

新建相机坐标系 $\tau$ 的基向量在平面中 $\Pi$ 的表达为

$$\begin{aligned}\vec{t}_x^{\Pi} &= (-\cos\alpha, -\sin\alpha, 0)^T, \\ \vec{t}_y^{\Pi} &= (\sin\alpha, -\cos\alpha, 0)^T \\ \vec{t}_z^{\Pi} &= (0,0,1)^T\end{aligned}$$

绕 $\vec{n}_x$ 的旋转矩阵可以表达为

$$R_\theta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

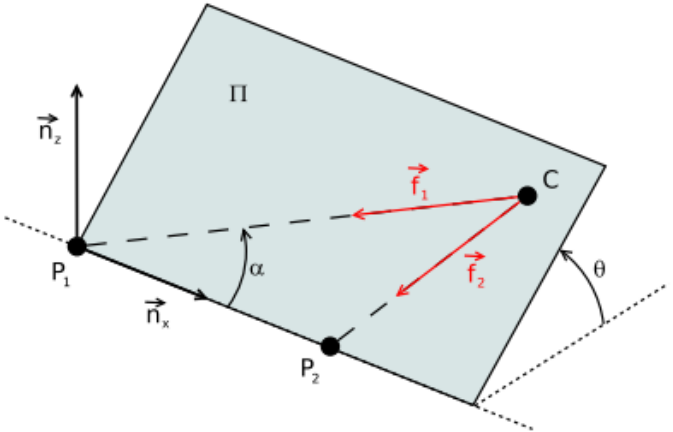



Figure 4. Rotation of the plane Π around $\vec{n}_x$ by the angle $\theta$.

相机在中心$C$在坐标系 $\eta$ 的表达为

$$\begin{aligned} C^{\eta}(\alpha,\theta) &= R_\theta \cdot C^{\Pi} \\ &= \begin{pmatrix} d_{12}\cos\alpha(\sin\alpha\cdot b+\cos\alpha) \\ d_{12}\sin\alpha\cos\theta(\sin\alpha\cdot b+\cos\alpha) \\ d_{12}\sin\alpha\sin\theta(\sin\alpha\cdot b+\cos\alpha) \end{pmatrix} \end{aligned}$$

$\eta$ 到 $\tau$ 的旋转表达为

$$\begin{aligned} Q(\alpha,\theta) &= \left[R_\theta\cdot(\vec{t}_x^{\Pi}\vec{t}_y^{\Pi}\vec{t}_z^{\Pi})\right]^T \\ &= \begin{pmatrix} -\cos\alpha & -\sin\alpha\cos\theta & -\sin\alpha\sin\theta \\ \sin\alpha & -\cos\alpha\cos\theta & -\cos\alpha\sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \end{aligned}$$

# 3D-2D: PnP问题-Kneip算法



## 将第3个点从新建世界坐标系 $\eta$ 转换到新建相机坐标系 $\tau$ 中

已知 $P_3^\eta = (p_1, p_2, 0)^T$

$$P_3^\tau = Q(\alpha,\theta)\cdot(P_3^\eta - C^\eta(\alpha,\theta))$$

$$= \begin{pmatrix} -\cos\alpha\cdot p_1 - \sin\alpha\cos\theta\cdot p_2 + d_{12}(\sin\alpha\cdot b + \cos\alpha) \\ \sin\alpha\cdot p_1 - \cos\alpha\cos\theta\cdot p_2 \\ -\sin\theta\cdot p_2 \end{pmatrix}$$

定义 $\phi_1 = \dfrac{f_{3,x}^\tau}{f_{3,z}^\tau} \quad \phi_2 = \dfrac{f_{3,y}^\tau}{f_{3,z}^\tau}$

$$\begin{cases} \phi_1 = \frac{P_{3,x}^\tau}{P_{3,z}^\tau} \\ \phi_2 = \frac{P_{3,y}^\tau}{P_{3,z}^\tau} \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} \phi_1 = \frac{-\cos\alpha\cdot p_1 - \sin\alpha\cos\theta\cdot p_2 + d_{12}(\sin\alpha\cdot b+\cos\alpha)}{-\sin\theta\cdot p_2} \\ \phi_2 = \frac{\sin\alpha\cdot p_1 - \cos\alpha\cos\theta\cdot p_2}{-\sin\theta\cdot p_2} \end{cases}$$

$$\Leftrightarrow \quad \begin{cases} \frac{\sin\theta}{\sin\alpha}p_2 = \frac{-\cot\alpha\cdot p_1 - \cos\theta\cdot p_2 + d_{12}(b+\cot\alpha)}{-\phi_1} \\ \frac{\sin\theta}{\sin\alpha}p_2 = \frac{p_1 - \cot\alpha\cos\theta\cdot p_2}{-\phi_2} \end{cases}$$

$$\Rightarrow \quad \cot\alpha = \frac{\frac{\phi_1}{\phi_2}p_1 + \cos\theta\cdot p_2 - d_{12}\cdot b}{\frac{\phi_1}{\phi_2}\cos\theta\cdot p_2 - p_1 + d_{12}}.$$

可以得到

$$\phi_2 = \frac{P_{3,y}^\tau}{P_{3,z}^\tau}$$

$$\Leftrightarrow \quad \sin^2\theta\cdot f_2^2 p_2^2 = \sin^2\alpha(p_1 - \cot\alpha\cos\theta\cdot p_2)^2$$

$$\Leftrightarrow \quad (1-\cos^2\theta)(1+\cot^2\alpha)f_2^2p_2^2 = p_1^2 - 2\cot\alpha\cos\theta\cdot p_1p_2 + \cot^2\alpha\cos^2\theta\cdot p_2^2$$

# 3D-2D: PnP问题-Kneip算法

深蓝学院 shenlanxueyuan.com

## 构建四次方程

最终可以得到一个关于$\cos\theta$的四次方程

$$a_4 \cdot \cos^4\theta + a_3 \cdot \cos^3\theta + a_2 \cdot \cos^2\theta + a_1 \cdot \cos\theta + a_0 = 0$$

其中

$$\begin{aligned}
a_4 &= -\phi_2^2 p_2^4 - \phi_1^2 p_2^4 - p_2^4 \\
a_3 &= 2p_2^3 d_{12} b + 2\phi_2^2 p_2^3 d_{12} b - 2\phi_1 \phi_2 p_2^3 d_{12} \\
a_2 &= -\phi_2^2 p_1^2 p_2^2 - \phi_2^2 p_2^2 d_{12}^2 b^2 - \phi_2^2 p_2^2 d_{12}^2 + \phi_2^2 p_2^4 \\
&\quad + \phi_1^2 p_2^4 + 2p_1 p_2^2 d_{12} + 2\phi_1 \phi_2 p_1 p_2^2 d_{12} b \\
&\quad - \phi_1^2 p_1^2 p_2^2 + 2\phi_2^2 p_1 p_2^2 d_{12} - p_2^2 d_{12}^2 b^2 - 2p_1^2 p_2^2 \\
a_1 &= 2p_1^2 p_2 d_{12} b + 2\phi_1 \phi_2 p_2^3 d_{12} \\
&\quad - 2\phi_2^2 p_2^3 d_{12} b - 2p_1 p_2 d_{12}^2 b \\
a_0 &= -2\phi_1 \phi_2 p_1 p_2^2 d_{12} b + \phi_2^2 p_2^2 d_{12}^2 + 2p_1^3 d_{12} \\
&\quad - p_1^2 d_{12}^2 + \phi_2^2 p_1^2 p_2^2 - p_1^4 - 2\phi_2^2 p_1 p_2^2 d_{12} \\
&\quad + \phi_1^2 p_1^2 p_2^2 + \phi_2^2 p_2^2 d_{12}^2 b^2 .
\end{aligned}$$

每一个$\cos\theta$对应一个 $\cot\alpha$ 一共可以得到4组解

$$C = P_1 + N^T \cdot C^\eta$$

$$R = N^T \cdot Q^T \cdot T$$

通过第4个点可以选择出正确的解

# 3D-2D: PnP问题-基于RANSAC的Kneip算法



**基于RANSAC的Kneip算法流程**

1）计算RANSAC采样次数，设置内点阈值（重投影误差）；

2）随机采样三对3D-2D对应点，计算相机的姿态；

3）计算每个视角中的重投影误差，统计内点个数；

4）重复2), 3)直到满足采样次数，选择内点数最多的相机姿态；

# 3D-2D: PnP问题-其他常用方法

深蓝学院 shenlanxueyuan.com

P3P法：需要4对不共面的点 求出2D点在当前相机坐标系中的3D点，然后进行3D-3D的姿态求解

参考资料1：https://www.cnblogs.com/mafuqiang/p/8302663.html

Complete Solution Classification for the Perspective-Three-Point Problem

参考资料2：A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation

参考资料3：《视觉SLAM十四讲》第7.7.2节

ePnP法：需要(>=4不共面或者3对共面) 点进行求解

参考资料1：https://www.cnblogs.com/jian-li/p/5689122.html

参考资料2：EPnP: Efficient Perspective-n-Point Camera Pose Estimation

# 三角量测

深蓝学院
shenlanxueyuan.com

## Coding

完成task-2/task2-2_test_p3p_kneip.cc代码，掌握算法流程。

完成task-2/task2-2_test_p3p_ransac.cc代码，掌握算法流程。

# 课程内容



- ✓三角化(Triangulation)
  - ✓直接线性变换法
  - ✓RANSAC鲁棒估计
- ✓ 3D-2D: PnP问题
  - ✓直接线性变换法
  - ✓P3P/EPnP
- ✓ **捆绑调整Bundle Adjustment**

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

**问题阐述：**同时对三维点位置和相机参数进行非线性优化。

$\chi_{ij}=1$表示第$i$个点在第$j$个相机中可见；

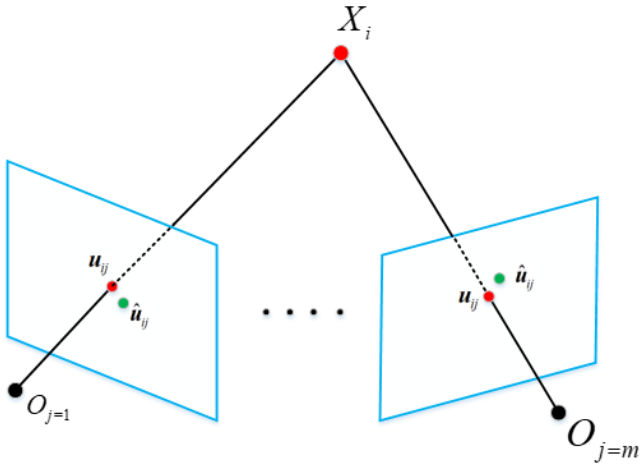
$\chi_{ij}=0$表示第$i$个点在第$j$个相机中不可见；

$\boldsymbol{X}_i=(X_i,Y_i,Z_i)^T$表示三维点世界坐标系中的坐标；

$\boldsymbol{C}_j$表示第$j$个相机的内参数以及世界坐标系与第$j$个相机的外参数

$\hat{\boldsymbol{u}}_{ij}$表示第$i$个点在第$j$个相机中的观测点；

$\boldsymbol{u}_{ij}$表示第$i$个点在第$j$个相机中的投影点；




$$g(\boldsymbol{\theta})=\frac{1}{2}\sum_{i}^{n}\sum_{j}^{m}\chi_{ij}\left\|\hat{\boldsymbol{u}}_{ij}-\boldsymbol{u}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)\right\|^2=\frac{1}{2}\sum_{i}^{n}\sum_{j}^{m}\chi_{ij}e_{ij}$$

其中，$\boldsymbol{\theta}=\left(\boldsymbol{C}_1,\ldots,\ \boldsymbol{C}_m,\boldsymbol{X}_1,\ldots\boldsymbol{X}_n\right)$为待优化的量，$\boldsymbol{\theta}\in\boldsymbol{R}^{9m\times 3n}$高维空间的非线性优化。

# 捆绑调整Bundle Adjustment



无约束非线性最小优化问题

$$min\ g(\boldsymbol{\theta}) = min\ \frac{1}{2}\|\boldsymbol{x} - f(\boldsymbol{\theta})\|^2,\ \boldsymbol{\theta} \in \boldsymbol{R}^n$$

上述问题的最优解通常是指它的**局部最优解**，对初始值较为敏感，因此需要一个较好的初始值。

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

最速下降法--假设函数一阶可微

假设g($\boldsymbol{\theta}$)在$\boldsymbol{\theta}_t$处可微，则它在$\boldsymbol{\theta}_t$处有Taylor展开式(略去高阶不计)：

$$g(\boldsymbol{\theta}) = g(\boldsymbol{\theta}_t) + (\nabla g(\boldsymbol{\theta}_t))^T \delta\boldsymbol{\theta},$$

其中 $\boldsymbol{\theta} = \boldsymbol{\theta}_t + \delta\boldsymbol{\theta}$

- 当 $(\nabla g(\boldsymbol{\theta}_t))^T \ \delta\boldsymbol{\theta} < 0$时可保证g($\boldsymbol{\theta}$)的值是在下降；
- 当$\delta\boldsymbol{\theta}$方向取梯度反方向时，可达到最快的下降速度。

为什么当$\delta\boldsymbol{\theta}$方向取梯度反方向时，达到最快的下降速度?

$$\Delta g = g(\boldsymbol{\theta}) - g(\boldsymbol{\theta}_t) = (\nabla g(\boldsymbol{\theta}_t))^T \ \delta\boldsymbol{\theta},$$

可见，函数变化量$\Delta g$为向量$\nabla g(\boldsymbol{\theta}_t)$与$\delta\boldsymbol{\theta}$的内积，根据向量内积的计算公式，得：

$$\Delta g = \|\nabla g(\boldsymbol{\theta}_t)\| \|\delta\boldsymbol{\theta}\| cos\alpha,$$

因为$cos\alpha \in [-1，1]$，当且仅当$\delta\boldsymbol{\theta}$的方向取梯度反方向时，函数值下降最快。当然这只是一个方向，通常我们还需要指定一个步长$\lambda$。

# 捆绑调整Bundle Adjustment

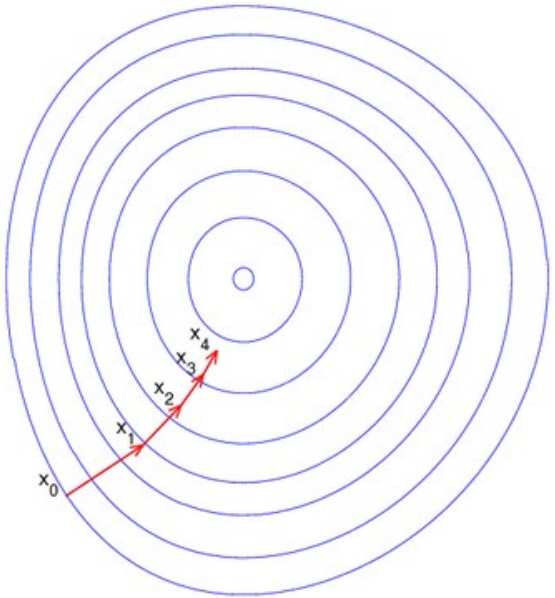
深蓝学院 shenlanxueyuan.com

最速下降法--算法流程

1）给定初始点$\boldsymbol{\theta}_0$，终止控制函数$\epsilon > 0$和步长$\lambda$，令$t = 0$；

2）计算$\nabla g(\boldsymbol{\theta}_t)$，若$\|\nabla g(\boldsymbol{\theta}_t)\| \leq \epsilon$，停止迭代，输出$\boldsymbol{\theta}_t$，
否则进行下一步；

3）取$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \lambda \nabla g(\boldsymbol{\theta}_t), t = t + 1$ 转第2)步。




越接近极值速度越慢

# 捆绑调整Bundle Adjustment



牛顿法--假设函数二阶可微

假设$g(\boldsymbol{\theta})$在$\boldsymbol{\theta}_t$处二阶可微，且假定二阶导数$\boldsymbol{\nabla}^2 g(\boldsymbol{\theta})$总是正定的，则它在$\boldsymbol{\theta}_t$处以$g(\boldsymbol{\theta})$的二阶近似函数$Q(\boldsymbol{\theta})$的极小值点作为下一次迭代点$\boldsymbol{\theta}_{t+1}$。

$$Q(\boldsymbol{\theta}) = g(\boldsymbol{\theta}_t) + (\boldsymbol{\nabla} g(\boldsymbol{\theta}_t))^T \delta\boldsymbol{\theta} + \frac{\mathbf{1}}{\mathbf{2}} \delta^T \boldsymbol{\theta} \boldsymbol{\nabla}^2 g(\boldsymbol{\theta}_t) \delta\boldsymbol{\theta}, \qquad \boldsymbol{\theta} = \boldsymbol{\theta}_t + \delta\boldsymbol{\theta}$$

上式对$\delta\boldsymbol{\theta}$求梯度并令其等于0，可以得到：

$$\boldsymbol{\nabla} Q(\boldsymbol{\theta}) = \boldsymbol{\nabla}\, g(\boldsymbol{\theta}_t) + \boldsymbol{\nabla}^2 g(\boldsymbol{\theta}_t) \delta\boldsymbol{\theta} = 0$$

$$\delta\boldsymbol{\theta} = -(\boldsymbol{\nabla}^2 g(\boldsymbol{\theta}_t))^{-1} \boldsymbol{\nabla} g(\boldsymbol{\theta}_t)$$

# 捆绑调整Bundle Adjustment

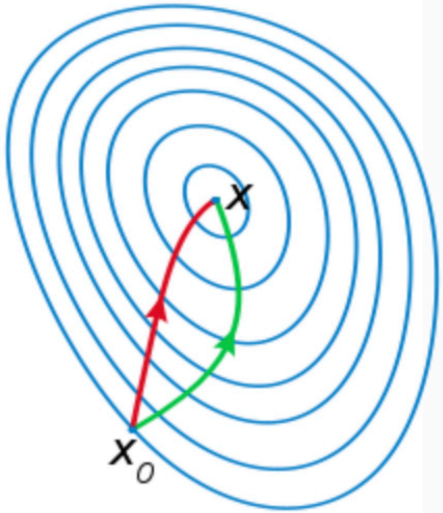
深蓝学院 shenlanxueyuan.com

## 牛顿法--算法流程

1）给定初始点$\boldsymbol{\theta}_0$，终止控制函数$\epsilon > 0$和步长$\lambda = 1$，令$t = 0$；

2）计算$\nabla g(\boldsymbol{\theta}_t)$，若$\|\nabla g(\boldsymbol{\theta}_t)\| \leq \epsilon$，停止迭代，输出$\boldsymbol{\theta}_t$，否则进行下一步；

3）取$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - (\nabla^2 g(\boldsymbol{\theta}_t))^{-1} \nabla g(\boldsymbol{\theta}_t)$，$t = t + 1$ 转第2)步。



# 捆绑调整Bundle Adjustment

深蓝学院
shenlanxueyuan.com

## 牛顿法—优点与缺点

1）速度快
最速下降法是局部平面拟合，牛顿法是局部二次曲面拟合；

2）计算量大
需要计算和保存二阶Hessian矩阵的逆矩阵；

3）要求初始点离最优点较近，否则无法保证收敛，甚至无法保证下降性。

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

Levenberg-Marquardt法--原理与优势

原理：是一种“信赖阈”的方法，当收敛速度较快时，增大信赖域，使算法趋向于牛顿法；当收敛速度较慢时，减小信赖域，使算法趋向于最速下降法。

优势：

- 速度快，只用到一阶矩阵；
- 对初始值有一定鲁棒性，可以在距离初始值较远处得到最优解。

# 捆绑调整Bundle Adjustment



Levenberg-Marquardt法--实现

$$g(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{x} - f(\boldsymbol{\theta})\|^2,\ \boldsymbol{\theta} \in \boldsymbol{R}^n$$

可得$\nabla g(\boldsymbol{\theta}_t) = -\boldsymbol{J}^T(\boldsymbol{\theta}_t)(\boldsymbol{x} - f(\boldsymbol{\theta}_t))$，其中$\boldsymbol{J}(\boldsymbol{\theta}_t) = \nabla f(\boldsymbol{\theta}_t)$。

$$\nabla^2 g(\boldsymbol{\theta}_t)\delta\boldsymbol{\theta} = -\nabla g(\boldsymbol{\theta}_t)$$

将上述公式中的$\nabla^2 g(\boldsymbol{\theta}_t)$替换成$\boldsymbol{J}^T(\boldsymbol{\theta}_t)\boldsymbol{J}(\boldsymbol{\theta}_t) + \frac{1}{\lambda}\boldsymbol{I}$（$\lambda$为信赖域半径），可得

$$(\boldsymbol{J}^T(\boldsymbol{\theta}_t)\boldsymbol{J}(\boldsymbol{\theta}_t) + \frac{1}{\lambda}\boldsymbol{I})\delta\boldsymbol{\theta} = \boldsymbol{J}^T(\boldsymbol{\theta}_t)(\boldsymbol{x} - f(\boldsymbol{\theta}))$$

增量正规方程

当$\lambda$趋向于无穷大时，$\boldsymbol{J}^T(\boldsymbol{\theta}_t)\boldsymbol{J}(\boldsymbol{\theta}_t)\delta\boldsymbol{\theta} = -\nabla g(\boldsymbol{\theta}_t)$；

当$\lambda$趋向于零时，$\delta\boldsymbol{\theta} = -\nabla g(\boldsymbol{\theta}_t)$。

# 捆绑调整Bundle Adjustment



Levenberg-Marquardt法--算法流程

1. $t=0$ 时, 选取初始点$\boldsymbol{\theta}_0$ , 终止控制常数$\varepsilon$, 令$e^0=\left\|\mathbf{x}-f\left(\boldsymbol{\theta}_0\right)\right\|^2$ , $\lambda_0=10^{-3}$

2.计算 $J^T\left(\boldsymbol{\theta}_t\right)$

3.构造增量正规方程$(\boldsymbol{J}^T(\boldsymbol{\theta}_t)\boldsymbol{J}(\boldsymbol{\theta}_t)+\frac{1}{\lambda}\boldsymbol{I})\delta\boldsymbol{\theta}=\boldsymbol{J}^T(\boldsymbol{\theta}_t)(\mathbf{x}-f(\boldsymbol{\theta}))$

4.通过求解增量正规方程, 得到 $\delta\left(\boldsymbol{\theta}\right)$

如果$\left\|\mathbf{x}-f\left(\boldsymbol{\theta}_t+\delta\boldsymbol{\theta}\right)\right\|^2<e^k$, 令 $\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\delta\left(\boldsymbol{\theta}\right)$,

如果$\|\delta(\boldsymbol{\theta})\|<\varepsilon$, 终止迭代;

否则令$\lambda_{t+1}=10\lambda_t \quad t=t+1$, 执行第2步

否则$\left\|\mathbf{x}-f\left(\boldsymbol{\theta}_t+\delta\boldsymbol{\theta}\right)\right\|^2\geq e^k$, 令$\lambda_{t+1}=0.1\lambda_t$ , 执行第3步。

# 捆绑调整Bundle Adjustment

深蓝学院
shenlanxueyuan.com

## Coding

阅读task-2/task2-3_test_lm_optimization.cc的代码流程，归纳LM算法的流程，并写成伪代码。

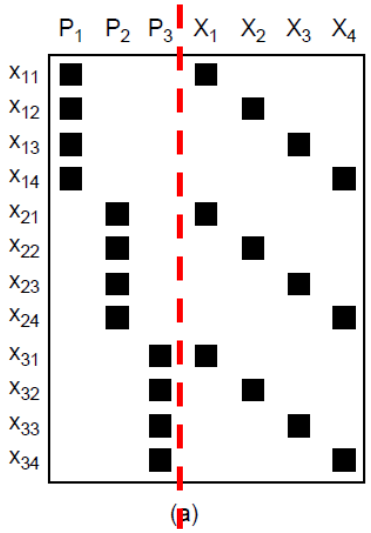
# 捆绑调整Bundle Adjustment

Levenberg-Marquardt法--增量规方程的求解

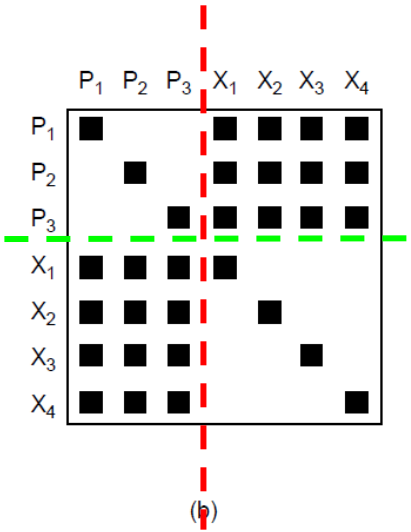
对称、稀疏

$$\left(J^T(\theta_t)J(\theta_t)+\frac{1}{\lambda}I\right)\delta\theta = J^T(\theta_t)(x-f(\theta))$$

$$J(\theta_t) = [J_C\ J_X]$$




$J^T(\theta)$




$J^T(\theta)J(\theta)$

$$J^T(\theta_t)\,J(\theta_t) = \begin{bmatrix} J_C^T J_C & J_C^T J_X \\ J_X^T J_C & J_X^T J_X \end{bmatrix} = \begin{bmatrix} J_{CC} & J_{CX} \\ J_{XC} & J_{XX} \end{bmatrix}$$

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

Levenberg-Marquardt法--正规方程的求解

$$\left(J^T(\theta_t)J(\theta_t)+\frac{1}{\lambda}I\right)\delta\theta = J^T(\theta_t)(\mathbf{x}-f(\theta))$$

左乘 $\begin{bmatrix} I & -J_{CX}\tilde{J}_{XX}^{-1} \\ 0 & I \end{bmatrix}$

$$\tilde{J}_{CC} = J_{CC}+\frac{1}{\lambda}I_{CC}$$
$$\tilde{J}_{XX} = J_{XX}+\frac{1}{\lambda}I_{XX}$$

Schur补

$$\begin{bmatrix} \tilde{J}_{CC}-J_{CX}\tilde{J}_{XX}^{-1}J_{XC} & 0 \\ J_{XC} & \tilde{J}_{XX} \end{bmatrix}\begin{bmatrix}\delta_C \\ \delta_X\end{bmatrix} = \begin{bmatrix} b_C-J_{CX}\tilde{J}_{XX}^{-1}b_X \\ b_X \end{bmatrix}$$

$$\begin{bmatrix} \tilde{J}_{CC} & J_{CX} \\ J_{XC} & \tilde{J}_{XX} \end{bmatrix}\begin{bmatrix}\delta_C \\ \delta_X\end{bmatrix} = \begin{bmatrix} b_C \\ b_X \end{bmatrix}$$

$$\begin{cases} \left(\tilde{J}_{CC}-J_{CX}\tilde{J}_{XX}^{-1}J_{XC}\right)\delta_C = b_C-J_{CX}\tilde{J}_{XX}^{-1}b_X \\ \tilde{J}_{XX}\delta_X = b_X-J_{XC}\delta_C \end{cases}$$

线性方程，共轭梯度法求解

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

Jacobian矩阵的计算

以第 $i$个三维点在第$j$个相机中的投影为例

$$e_{ij}=\frac{1}{2}\left\|\boldsymbol{e}_{ij}\right\|^2=\frac{1}{2}\left\|\hat{\boldsymbol{u}}_{ij}-\boldsymbol{u}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)\right\|^2=\frac{1}{2}\left(\left(\hat{u}_{ij}-u_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)\right)^2+\left(\hat{v}_{ij}-v_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)\right)^2\right)$$

$$\boldsymbol{\xi}_{ij}=\begin{bmatrix}\boldsymbol{C}_j\\ \boldsymbol{X}_i\end{bmatrix},\quad \boldsymbol{C}_j=\left(f_j,k_1,k_2,\boldsymbol{R}_j,\boldsymbol{t}_j\right)$$

详细过程见参考文件BAJacobian矩阵推导.pdf

$$\frac{\partial e_{ij}}{\partial \boldsymbol{\xi}_{ij}^T}=\boldsymbol{e}_{ij}^T\frac{\partial \boldsymbol{e}_{ij}}{\partial \boldsymbol{\xi}_{ij}^T}=-\boldsymbol{e}_{ij}^T\frac{\partial \hat{\boldsymbol{u}}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)}{\partial \boldsymbol{\xi}_{ij}^T}$$

$$\frac{\partial \hat{\boldsymbol{u}}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)}{\partial \boldsymbol{\xi}_{ij}^T}=\begin{bmatrix}\frac{\partial \hat{\boldsymbol{u}}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)}{\partial \boldsymbol{C}_j^T} & \frac{\partial \hat{\boldsymbol{u}}_{ij}\left(\boldsymbol{C}_j,\boldsymbol{X}_i\right)}{\partial \boldsymbol{X}_i^T}\end{bmatrix}=\begin{bmatrix}\frac{\partial u_{ij}(\boldsymbol{C}_i,\boldsymbol{X}_j)}{\partial \boldsymbol{C}_j^T} & \frac{\partial u_{ij}(\boldsymbol{C}_i,\boldsymbol{X}_j)}{\partial \boldsymbol{X}_j^T}\\ \frac{\partial v_{ij}(\boldsymbol{C}_i,\boldsymbol{X}_j)}{\partial \boldsymbol{C}_j^T} & \frac{\partial v_{ij}(\boldsymbol{C}_i,\boldsymbol{X}_j)}{\partial \boldsymbol{X}_j^T}\end{bmatrix}$$

# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

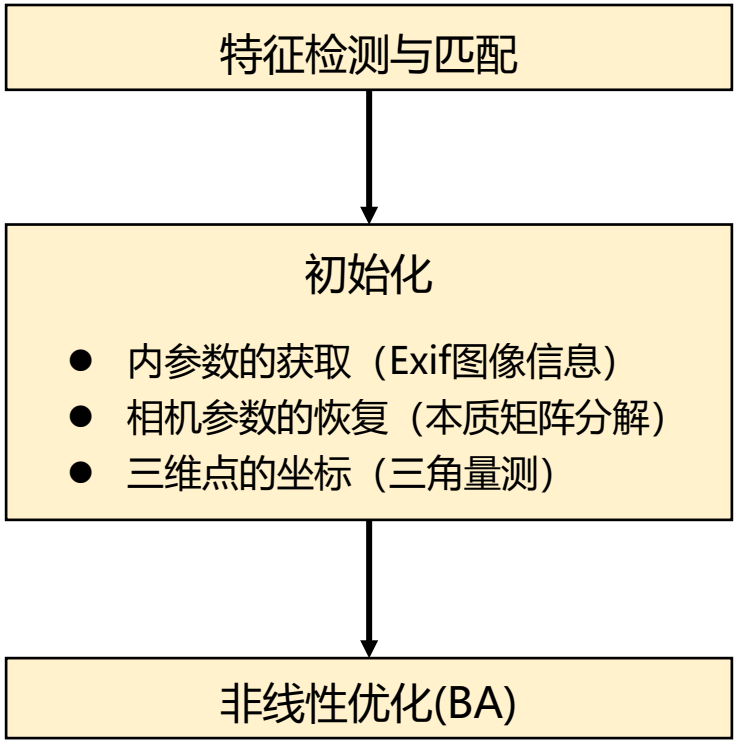
## Coding

仔细推导Jacobian矩阵，并完成task2/task2-4_test_jacobian.cc中Jacobian矩阵的实现。

理解->推导->实现== 真正掌握

# 捆绑调整Bundle Adjustment





# 捆绑调整Bundle Adjustment

深蓝学院 shenlanxueyuan.com

## Coding

阅读task2/task2-5_test_bundle_adjustment.cc中，熟悉两视角的BA原理和流程。

理解->推导->实现== 真正掌握

深蓝学院
shenlanxueyuan.com

# 感谢聆听！

Thanks for Listening