

# 第 6 讲 前端 Frontend

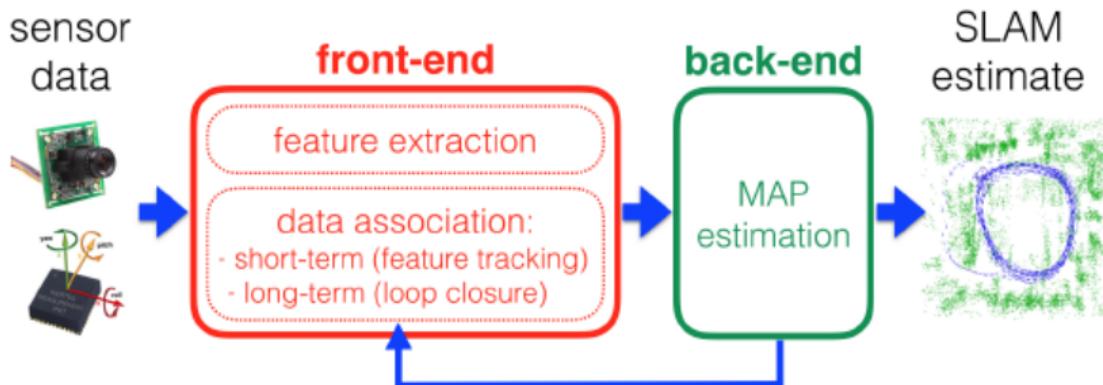
贺一家，高翔，崔华坤

# 前端的工作

## SLAM 的框架

通常的 SLAM 框架由前后端共同构成

- ① 前端：提取特征点，追踪相机 Pose，定位相机
- ② 后端：提供全局优化或滑动窗口优化



# 前端的工作

## 一些现实的问题

- ① 前端的步骤? 初始化、正常追踪、丢失处理
- ② 相机和路标的建模和参数化

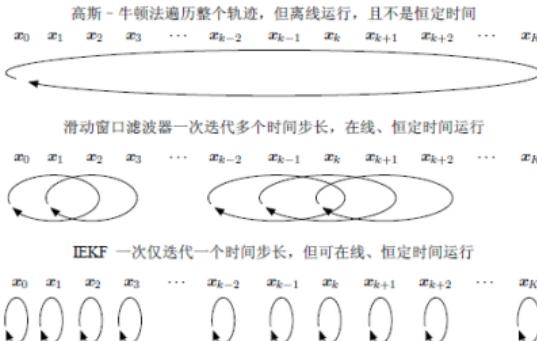
$$\mathbf{x}_{\text{camera}} = [\mathbf{R}, \mathbf{t}], \mathbf{x}_{\text{landmark}} = ? \quad (1)$$

可能的路标: 绝对坐标  $xyz$ , 逆深度  $\rho$ , 灰度值 (灰度 Pattern), 等等

- ③ 关键帧?
  - 需不需要关键帧?
  - 怎么选关键帧?
  - 控制关键帧数量?
  - 仅在关键帧中补充新特征点? 还是对所有帧提取新特征点?
  - 何时进行三角化?

# 前端的工作

- 实际上，前端非常能体现一个 SLAM 的追踪效果。
- 在实现中，尽管后端存在明显的理论差异，但很难直观体验在最终精度上。
- 问：假设给定相同的 Pose 和 Landmark，给定噪声范围，各种方法是否有明显差异？还是仅仅有理论性质上的差异？



# 前端的工作

在某些理想情况下，利用仿真数据可以给出一定的结果：

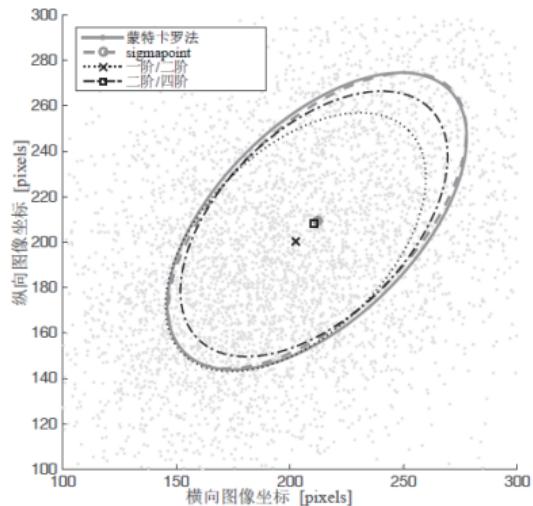


图 7-12 立体相机的实验结果：在给定相机姿态与路标点坐标及对应的不确定性的情况下，四种方法在立体相机中左图部分对应的均值与单位标准偏差椭圆。这种情况对应于图 7-11 中  $\alpha = 1$  的情况

这个实验（双目 Pose/Landmark 估计）中，UKF(SPKF) 给出最接近真值的结果。

# 前端的工作

但是还可以问

- ① 实际情况离理论假设有多远？（高斯噪声）
- ② 拿到的数据究竟有多好？

- 在很多实际场合，很难回答某种后端算法是否明确优于另一种。
- 例如：ORB-SLAM2 使用 Covisibility-Graph，DSO 使用带边缘化的滑动窗口，Tango 使用 MSCKF，等等。
- 实验效果：ORB-SLAM2 具有较好的全局精度，但无回环时 DSO 具有漂移较少。Tango 计算量明显少于其他算法。
- **算法的结果和数据集关系很大。** 例如：KITTI 属于比较简单的（视野开阔，动态物体少，标定准确），EUROC 一般（人工设定场景，纹理丰富，但曝光有变化），TUM-Mono 比较难（场景多样，主要为真实场景）

我们更多考量计算和精度的 trade-off。

# 前端的工作

相比来说，前端的差异就比较明显：

- 追踪算法是否很容易丢失？
- 算法对干扰的鲁棒性如何（光照、遮挡、动态物体）？

前端更多是范式（Paradigm）之间的比较，而非范式之内的比较。

## 好的前端

- 追踪效果好，不容易丢
- 计算速度快
- 累计误差小

## 不好的前端

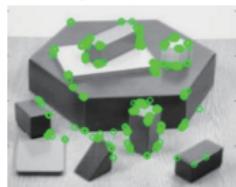
- 追踪效果差
- 计算耗时
- 容易出现累计误差

# 前端的工作

## VIO 方法的定性比较：



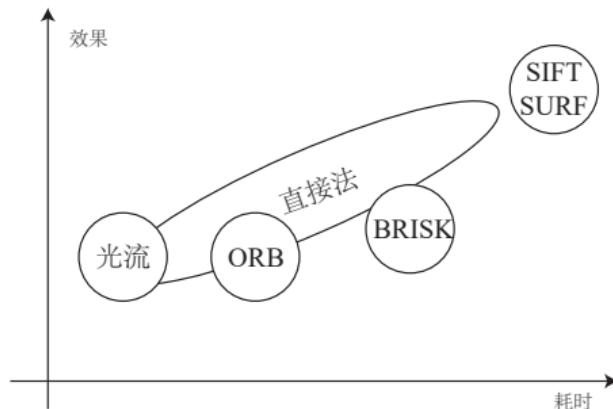
Optical Flow



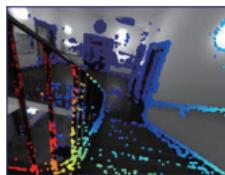
ORB Feature



SIFT Feature



BRISK Feature



Direct method

# 前端的工作

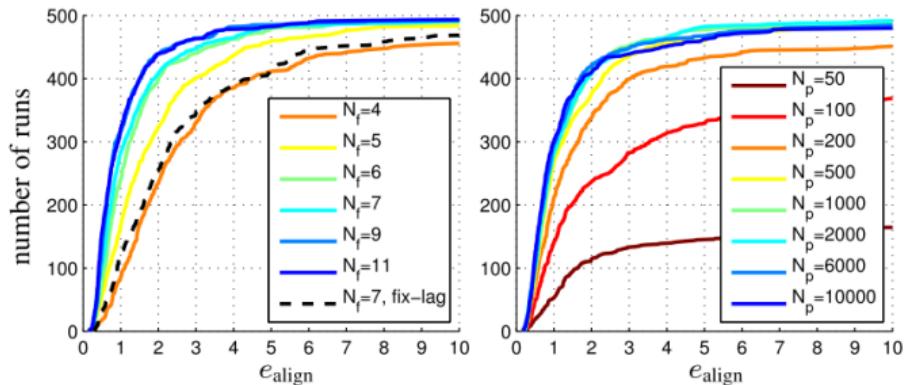
## 几个要点

- 光流法最早，最成熟，缺点也明显（抗光照干扰弱，依赖角点）
- FAST+ 光流是比较实用的快速算法/GFTT+ 光流效果更好，也具备实时性
- 特征匹配需要提的特征具有旋转平移缩放不变性，SIFT/SURF 是最好的一类，BRISK 等次之，ORB 算比较差的
- 特征匹配和光流都非常依赖角点，日常生活场景中角点不明显的很多

# 前端的工作

## 几个要点

直接法不依赖角点，但实现效果根据选点数量变化较大。



DSO 在不同关键帧数量/不同选点数量下的表现。曲线越靠左上侧越好。

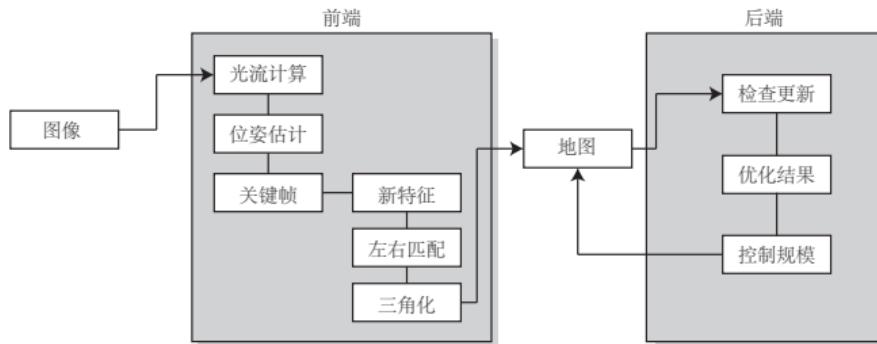
## Section 2

# 特征点提取、匹配和光流

# 特征点提取

我们后文以传统光流为主来展开前端的介绍。

一个传统的双目光流前端流程（正常追踪流程）：



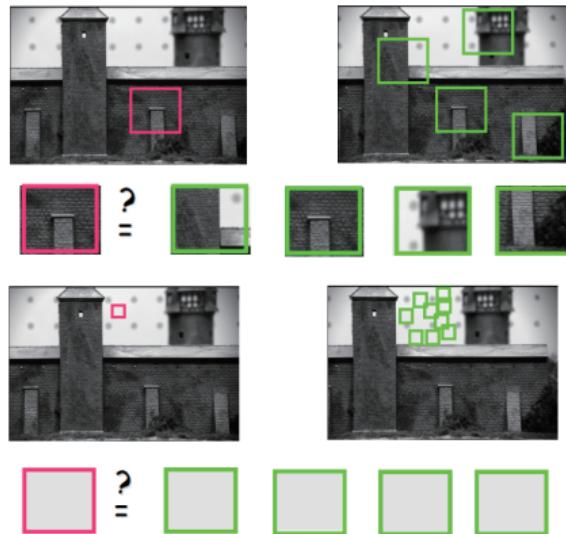
除了正常追踪流程以外，还需要考虑初始化、丢失恢复的情况。

VIO 初始化比常规 VO 多一些步骤（主要为 IMU 参数的估计），我们留到下讲介绍。

思考：为什么要有这样的框架？

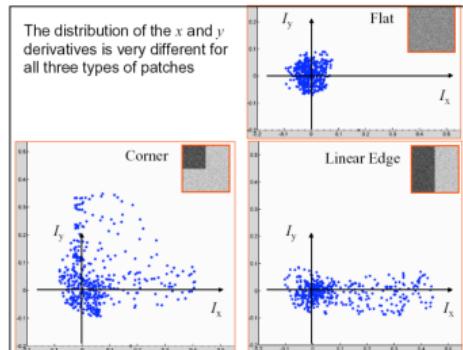
# 特征点提取

在光流中，我们通常选择角点来追踪。  
为什么需要角点？



# 特征点提取

角点的梯度在两个方向都有分布：



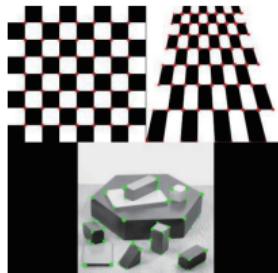
利用角点附近块的两个特征值大小，可以判断该区是否为角点。

$$S_{\text{Harris}} = \det(\mathbf{M}) - k \text{tr}(\mathbf{M})^2. \quad k = 0.04 - 0.06 \quad (2)$$

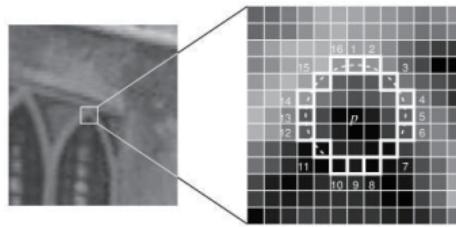
$$\det(\mathbf{M}) = \lambda_1 \lambda_2, \text{tr}(\mathbf{M}) = \lambda_1 + \lambda_2, \quad \mathbf{M} = \sum_{x,y} \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix} \quad (3)$$

# 特征点提取

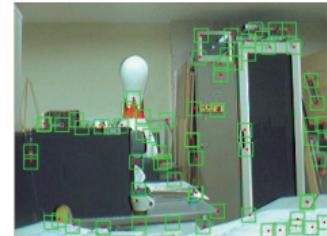
FAST/GFTT 角点：



Harris Corner



FAST Corner



GFTT Corner

- FAST：仅含像素亮度、不含计算的快速角点提取方式；
- GFTT：在 Harris 基础改进：Shi-tomasi 分数，增加固定选点数，等等

$$S_{\text{Shi-Tomasi}} = \min(\lambda_1, \lambda_2) \quad (4)$$

# 光流

光流可以追踪一个时刻的角点在下个时刻的图像位置

## 灰度不变假设

灰度不变假设：

$$\mathbf{I}(x + dx, y + dy, t + dt) = \mathbf{I}(x, y, t) \quad (5)$$

一阶 Taylor 展开：

$$\mathbf{I}(x + dx, y + dy, t + dt) \approx \mathbf{I}(x, y, t) + \frac{\partial \mathbf{I}}{\partial x} dx + \frac{\partial \mathbf{I}}{\partial y} dy + \frac{\partial \mathbf{I}}{\partial t} dt. \quad (6)$$

得到：

$$\frac{\partial \mathbf{I}}{\partial x} \frac{dx}{dt} + \frac{\partial \mathbf{I}}{\partial y} \frac{dy}{dt} = -\frac{\partial \mathbf{I}}{\partial t}. \quad (7)$$

# 光流

带 Warp function 的光流：



$$\mathbf{I}(x, y, t) = \mathbf{I}(\mathbf{W}(x + dx, y + dy), t + dt) \quad (8)$$

其中  $\mathbf{W}$  为 Warp Function，通常取仿射变换<sup>1</sup>：

$$\mathbf{W}(x + dx, y + dy) = \begin{bmatrix} 1 + p_1 & p_3 & p_5 \\ p_2 & 1 + p_4 & p_6 \end{bmatrix} \begin{bmatrix} x + dx \\ y + dy \\ 1 \end{bmatrix} \quad (9)$$

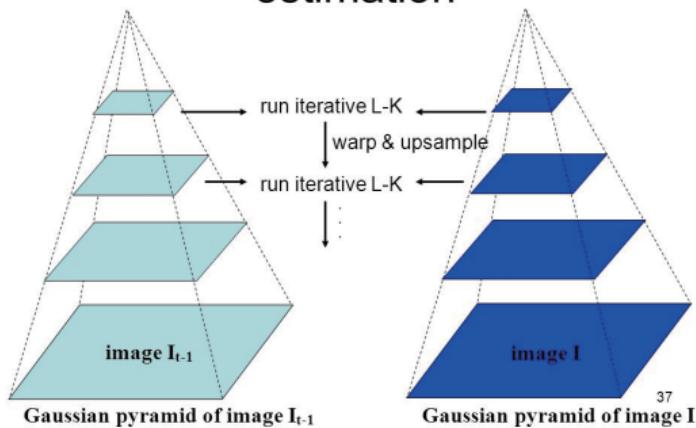
其中  $p_1 - p_6$  为  $\mathbf{W}$  的参数，需要在线估计。

<sup>1</sup> 对应于认为每个 patch 为小平面块

# 光流

金字塔式光流：

## Coarse-to-fine optical flow estimation



Coarse-to-Fine：从顶层最模糊的图像开始计算光流，一直往下迭代到最高分辨率

# 光流

## 光流在 SLAM 中的运用

- ① 光流可以追踪上一帧的角点
- ② 可以一直追踪该角点，直到超出图像范围或被遮挡
- ③ 在单目 SLAM 中，新提出的角点没有 3D 信息，因此可通过追踪角点在各图像位置，进行三角化

## 光流的局限性

- ① 容易受光照变化影响
- ② 只适合连续图像中的短距离追踪，不适合更长距离
- ③ 图像外观发生明显变化时不适用（例：远处的角点凑近看之后不为角点了）
- ④ 对角点强依赖，对 Edge 类型点表现较差
- ⑤ 稀疏光流不约束各点光流的方向统一，可能出现一些 outlier。

## Section 3

# 关键帧与三角化

# 关键帧

## 为什么需要关键帧

- ① 后端通常实时性较差，不适合处理所有帧；
- ② 如果相机停止，可能给后端留下无用的优化，甚至导致后端问题退化<sup>a</sup>

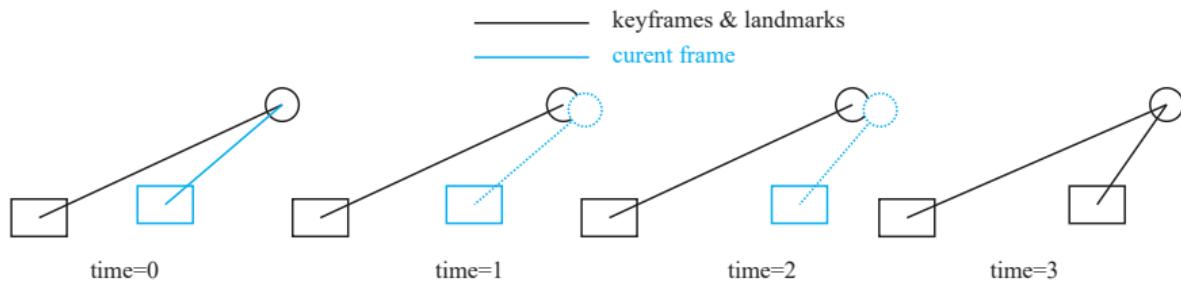
<sup>a</sup>想象我们对一个停留在原地的窗口做优化会发生什么。

## 如何选择关键帧

- ① 关键帧之间不必太近（退化或三角化问题）
- ② 关键帧之间不能太远（共视点太少）
- ③ VIO 中，定期选择关键帧（假设  $b_g, b_a$  在关键帧期间不变）

# 关键帧

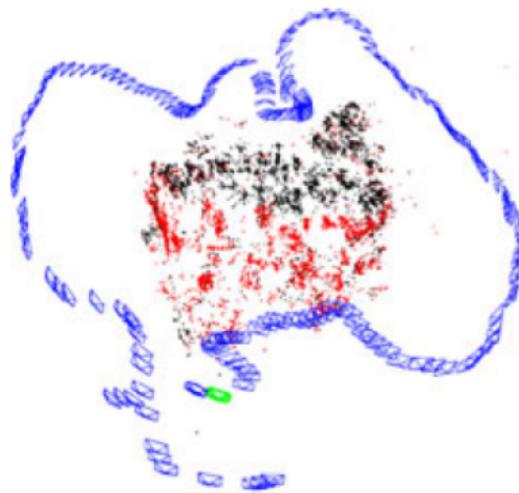
- 对于非关键帧，只执行前端算法，不参与后端优化
- 因此，对于非关键帧，它的误差会逐渐累积。直接将该帧作为关键帧插入后端，BA 才会保证窗口内的一致性



总结关键帧选取的策略：在计算量允许范围内，且不引起退化时，**应尽可能多地插入关键帧**。

# 关键帧策略

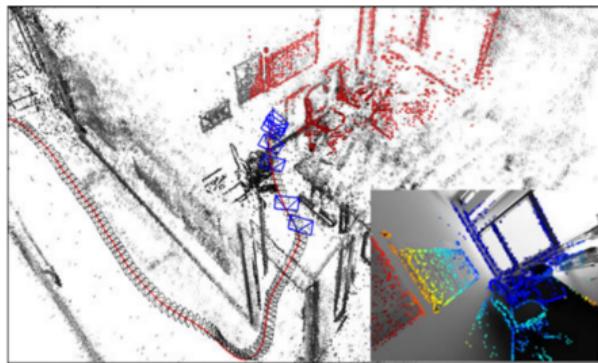
例子：ORB-SLAM2



ORB-SLAM2 使用非常宽松的关键帧策略（大多数时候只要后端线程 Idle 就会插入关键帧），然后在后端剔除冗余的关键帧。

# 关键帧策略

例子：DSO



DSO 利用光度误差插入关键帧（插入比较频繁）。然后在后端计算每个关键帧的 Active Landmarks, Marg 对窗口贡献最低的<sup>2</sup>。因此，DSO 的关键帧窗口通常有一个很远的和两三个很近的，其他几个分布在中间。

<sup>2</sup>同时，每个关键帧有一个最小寿命，防止它刚插进来就被剔除。

# 三角化

在单目 SLAM 中，通常在插入关键帧时计算新路标点的三角化。

- 有的 SLAM 系统在关键帧时提取新 Feature (DSO, SVO)，也有 的方案对每个帧都提新 Feature (VINS, ORB)。
- 前者节省计算量 (非关键帧无需提点，节省 5-10ms 左右)；后者 效果好 (**在单目里需要防止三角化 Landmark 数量不够**)。

# 三角化

## 三角化的数学描述

- 考虑某路标点  $\mathbf{y}$  在若干个关键帧  $k = 1, \dots, n$  中看到。
- $\mathbf{y} \in \mathbb{R}^4$ , 取齐次坐标。每次观测为  $\mathbf{x}_k = [u_k, v_k, 1]^\top$ , 取归一化平面坐标 (这样可以忽略掉内参)。
- 记投影矩阵  $\mathbf{P}_k = [\mathbf{R}_k, \mathbf{t}_k] \in \mathbb{R}^{3 \times 4}$ , 为 World 系到 Camera 系
- 投影关系:

$$\forall k, \lambda_k \mathbf{x}_k = \mathbf{P}_k \mathbf{y}. \quad (10)$$

其中  $\lambda_k$  为观测点的深度值 (未知)

- 根据上式第三行:

$$\lambda_k = \mathbf{P}_{k,3}^\top \mathbf{y} \quad (11)$$

其中  $\mathbf{P}_{k,3}^\top$  为  $\mathbf{P}_k$  的第 3 行。

# 三角化

## 三角化的数学描述

- 将(11)代入(10)的前两行：

$$\begin{aligned} u_k \mathbf{P}_{k,3}^\top \mathbf{y} &= \mathbf{P}_{k,1}^\top \mathbf{y} \\ v_k \mathbf{P}_{k,3}^\top \mathbf{y} &= \mathbf{P}_{k,2}^\top \mathbf{y} \end{aligned} \quad (12)$$

- 每次观测将提供两个这样的方程，视  $\mathbf{y}$  为未知量，并将  $\mathbf{y}$  移到等式一侧：

$$\left[ \begin{array}{c} u_1 \mathbf{P}_{1,3}^\top - \mathbf{P}_{1,1}^\top \\ v_1 \mathbf{P}_{1,3}^\top - \mathbf{P}_{1,2}^\top \\ \vdots \\ u_n \mathbf{P}_{n,3}^\top - \mathbf{P}_{n,1}^\top \\ v_n \mathbf{P}_{n,3}^\top - \mathbf{P}_{n,2}^\top \end{array} \right] \mathbf{y} = \mathbf{0} \rightarrow \mathbf{Dy} = \mathbf{0} \quad (13)$$

- 于是， $\mathbf{y}$  为  $\mathbf{D}$  零空间中的一个非零元素。

# 三角化

## 三角化的数学描述

- 由于  $\mathbf{D} \in \mathbb{R}^{2n \times 4}$ , 在观测次于大于等于两次时, 很可能  $\mathbf{D}$  满秩, 无零空间。
- 寻找最小二乘解:

$$\min_{\mathbf{y}} \|\mathbf{D}\mathbf{y}\|_2^2, \quad s.t. \|\mathbf{y}\| = 1 \quad (14)$$

解法: 对  $\mathbf{D}^\top \mathbf{D}$  进行 SVD:

$$\mathbf{D}^\top \mathbf{D} = \sum_{i=1}^4 \sigma_i^2 \mathbf{u}_i \mathbf{u}_j^\top \quad (15)$$

其中  $\sigma_i$  为奇异值, 且由大到小排列,  $\mathbf{u}_i, \mathbf{u}_j$  正交。

# 三角化

## 三角化的数学描述

- 此时，取  $y = u_4$  (为什么?)，那么该问题的目标函数值为  $\sigma_4$ 。
- 判断该解有效性的条件： $\sigma_4 \ll \sigma_3$ 。若该条件成立，认为三角化有效，否则认为三角化无效。
- Rescale：在某些场合（比如我们使用 UTM 坐标的平移），D 的数值大小差异明显，导致解在数值上不稳定。此时对 D 做一个缩放：

$$Dy = \underbrace{DS}_{\tilde{D}} \underbrace{S^{-1}y}_{\tilde{y}} = 0 \quad (16)$$

其中 S 为一个对角阵，取 D 最大元素之逆即可。

- 实用当中，还需要加上 y 投影正负号的判定作为依据

以上，我们推导了对任意次观测的三角化算法，可作为通用的依据。