深蓝学院
shenlanxueyuan.com

DSO源码解析
——滑窗优化

主讲人 龚益群

东北大学

预备知识

初始化

跟踪

建图

滑窗优化

总结

滑窗优化

1、滑窗优化

2、零空间的处理

3、DSO边缘化策略

4、FEJ策略

5、DSO中的solver

6、总结

滑窗优化

1、滑窗优化

2、零空间的处理

3、DSO边缘化策略

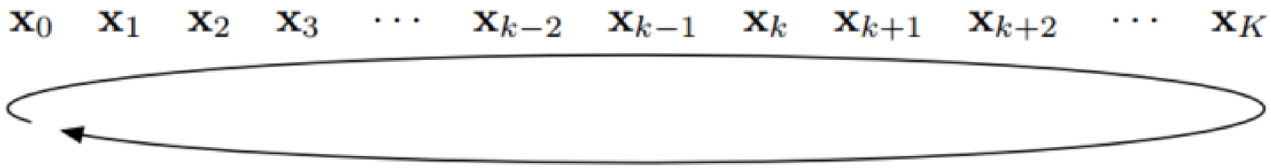
4、FEJ策略

5、DSO中的solver

6、总结

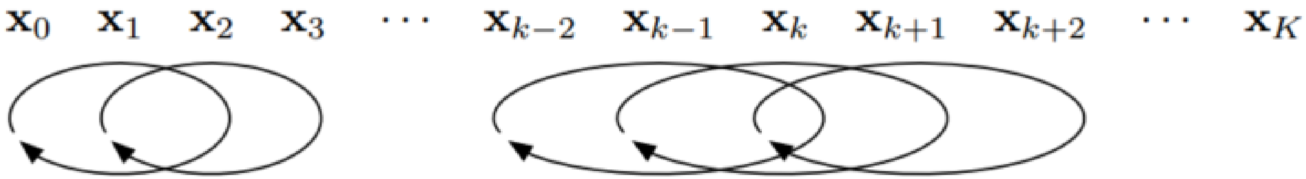
1、滑窗优化

- Batch (Full-Smoother)



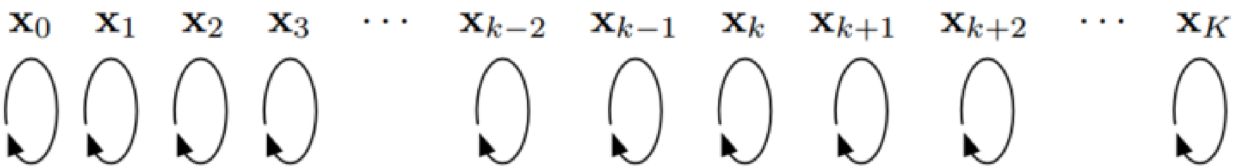
离线
非恒定时间

- Fixed-Lag (Sliding-Window)



在线
恒定时间

- Filter (EKF)



在线
恒定时间

Barfoot T D. State estimation for robotics[M]. Cambridge University Press, 2017. 4.3 Page 142

- 高斯牛顿收敛性
  - 最小二乘的目标函数为：

$$G(x)=\frac{1}{2}\|F(x)\|^2 \ldots\ldots(1)$$

  - 对G(x)进行二次泰勒展开：

$$G(x_{k+1})=G(x_k)+\frac{\partial G}{\partial x}\Delta x+\frac{1}{2}\Delta x^T\frac{\partial^2 G}{\partial x^2}\Delta x$$

$$=G(x_k)+F(x_k)\frac{\partial F}{\partial x}\Delta x+ \qquad \ldots\ldots(2)$$

$$\frac{1}{2}\Delta x^T\left(\frac{\partial F}{\partial x}^T\frac{\partial F}{\partial x}+F(x_k)\frac{\partial^2 F}{\partial x^2}\right)\Delta x$$

  - 在高斯牛顿中，对F(x)进行一阶展开：

$$F(x_{k+1})=F(x_k)+\frac{\partial F}{\partial x}\Delta x \ldots\ldots(3)$$

  - 将公式(3)带入(1)得：

$$G(x_{k+1})$$
$$=\frac{1}{2}F^2(x_k)+F(x_k)\frac{\partial F}{\partial x}\Delta x+\frac{1}{2}\Delta x^T\frac{\partial F}{\partial x}^T\frac{\partial F}{\partial x}\Delta x \ldots\ldots(4)$$

  - 对比公式(2)和(4)的二阶项，相差：

$$S(x^*)=F(x^*)\frac{\partial^2 F}{\partial x^2}\ldots\ldots(5)$$

  - 做如下符号假设：

$$J(x)=\frac{\partial F}{\partial x} \qquad H(x)=\frac{\partial^2 F}{\partial x^2}$$

- 高斯牛顿收敛性

根据公式(5)可知，当$x_k \to x^*$时，GN的收敛速度与最优残差量大小和线性程度相关，具体如下：

- 二阶收敛：当多余的项 $\|S(x^*)\| = 0$，即$F(x^*) = 0$或者$H(x^*) = 0$时，在零残差问题或者线性最小二乘情况具有Newton法的收敛速度。
- 线性收敛：当$\|S(x^*)\| \neq 0$，则具有线性收敛速度，随着$\|S(x^*)\|$增大而变慢。
- $\|S(x^*)\|$很大，则可能不收敛。
- H要求正定，非奇异，要求极值点附近Lipschitz连续。

1、滑窗优化

Jacobian

- **求导的参数包括：**
  - 相对的光度参数
  - 相对位姿
  - 主帧上点的逆深度
  - 相机内参

- **对主帧上逆深度求导**
  - 与初始化逆深度导数相同

$$\frac{\partial r_k}{\partial \delta d_{\mathbf{p}_i}} = \frac{1}{P_z'} d_x f_x \left(t_x - \frac{P_x'}{P_z'} t_z \right) + \frac{1}{P_z'} d_y f_y \left(t_y - \frac{P_y'}{P_z'} t_z \right)$$

- **对相对位姿求导**
  - 与初始化中位姿导数相同

$$\frac{\partial r_k}{\partial \delta \xi} = \begin{pmatrix} d_x f_x & d_y f_y \end{pmatrix} \begin{pmatrix} \frac{d_{\mathbf{p}_i}}{P_z'} & 0 & -\frac{d_{\mathbf{p}_i}}{P_z'} \frac{P_x'}{P_z'} & -\frac{P_x' P_y'}{P_z'} & 1 + \frac{P_x'^2}{P_z'^2} & -\frac{P_y'}{P_z'} \\ 0 & \frac{d_{\mathbf{p}_i}}{P_z'} & -\frac{d_{\mathbf{p}_i}}{P_z'} \frac{P_y'}{P_z'} & -1 - \frac{P_y'^2}{P_z'^2} & \frac{P_x' P_y'}{P_z'^2} & \frac{P_y'}{P_z'} \end{pmatrix}$$

1、滑窗优化 Jacobian

- **对相对的光度参数求导：**

$$r_k = I_j[\mathbf{p}_j] - \frac{t_j e^{a_j}}{t_i e^{a_i}} I_i[\mathbf{p}_i] - \left(b_j - \frac{t_j e^{a_j}}{t_i e^{a_i}} b_i\right)$$

$$r_k = I_j[\mathbf{p}_j] - e^{a_{ji}} I_i[\mathbf{p}_i] - (b_{ji})$$

  - 光度求导的变量有变化

$$\delta_{\text{photo}} = [-e^{a_{ji}}, -b_{ji}]$$

$$\mathbf{J}_{photo} = \frac{\partial r_k}{\partial \delta_{photo}} = \begin{bmatrix} \frac{\partial r_k}{\partial\text{-}e^{a_{ji}}} & \frac{\partial r_k}{\partial\text{-}b_{ji}} \end{bmatrix}$$

$$= \begin{bmatrix} I_i[\mathbf{p}_i] - b_i & 1 \end{bmatrix}$$

- **对相对内参求导：**

$$\mathbf{P}_i' = \pi_C^{-1}(\mathbf{p}_i) = \mathbf{K}^{-1}\begin{pmatrix}\mathbf{p}_i \\ 1\end{pmatrix} \quad ......(6)$$

$$\mathbf{P}_j' = \mathbf{R}\mathbf{P}_i' + \mathbf{t}d_{\mathbf{p}_i} \quad ......(7)$$

$$\begin{pmatrix}\mathbf{p}_j \\ 1\end{pmatrix} = \pi_{\mathbf{c}}\left(\omega(\mathbf{P}_j')\right) \quad ......(8)$$

  - 将公式(8)里的公式进行展开

$$\omega(\mathbf{P}_j') = \begin{bmatrix} \frac{P_{jx}'}{P_{jz}'} \\ \frac{P_{jy}'}{P_{jz}'} \\ 1 \end{bmatrix} = \begin{bmatrix} u_j \\ v_j \\ 1 \end{bmatrix} \quad ......(9)$$

$$\mathbf{p}_{jx} = f_x \cdot \frac{P_{jx}'}{P_{jz}'} + c_x \quad ......(10)$$

$$\mathbf{p}_{jy} = f_y \cdot \frac{P_{jy}'}{P_{jz}'} + c_y \quad ......(11)$$

1、滑窗优化 Jacobian

- **对相对内参求导：**

$$\frac{\partial r_k}{\partial \delta \mathbf{c}} = \mathbf{J}_I \cdot \frac{\partial \mathbf{p}_j}{\partial \delta \mathbf{c}}$$

- 像素坐标针对相机参数求导包含两部分，投影和反投影：

$$\frac{\partial \mathbf{p}_j}{\partial \delta \mathbf{c}} = \overset{\text{投影}}{\boxed{\begin{pmatrix} u_j & 0 & 1 & 0 \\ 0 & v_j & 0 & 1 \end{pmatrix}}} + \overset{\text{反投影}}{\boxed{\begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \frac{\partial \omega(\mathbf{P}'_j)}{\partial \delta \mathbf{c}}}}$$

- 结合公式(9)有：

$$\frac{\partial \omega(\mathbf{P}'_j)}{\partial \delta \mathbf{c}} = \frac{\partial \omega(\mathbf{P}'_j)}{\partial \mathbf{P}'_i} \frac{\partial \mathbf{P}'_i}{\partial \delta \mathbf{c}} = \begin{pmatrix} \frac{\partial u_j}{\partial \delta \mathbf{c}} \\ \frac{\partial v_j}{\partial \delta \mathbf{c}} \end{pmatrix} \text{……(12)}$$

- 结合公式(7)(8)将公式(12)分别展开：

$$\frac{\partial \omega(\mathbf{P}'_j)}{\partial \mathbf{P}'_i} = \frac{\partial \omega(\mathbf{P}'_j)}{\partial \mathbf{P}'_j} \frac{\partial \mathbf{P}'_j}{\partial \mathbf{P}'_i}$$

$$= \underbrace{\frac{1}{P'_{jz}} \begin{pmatrix} 1 & 0 & -u_j \\ 0 & 1 & -v_j \\ 0 & 0 & 0 \end{pmatrix}}_{\text{投影过程}} \underbrace{\begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}}_{\text{旋转}\boldsymbol{R}} \text{……(13)}$$

$$\frac{\partial \mathbf{P}'_i}{\partial \delta \mathbf{c}} = \frac{\partial \mathbf{K}^{-1} \mathbf{p}_i}{\partial \delta \mathbf{c}}$$

$$= \begin{pmatrix} -f_x^{-2}(p_{ix} - c_x) & 0 & -f_x^{-1} & 0 \\ 0 & -f_y^{-2}(p_{iy} - c_y) & 0 & -f_y^{-1} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} -f_x^{-1} P'_{ix} & 0 & -f_x^{-1} & 0 \\ 0 & -f_y^{-1} P'_{iy} & 0 & -f_y^{-1} \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{……(14)}$$

1、滑窗优化 Jacobian

- **对相对内参求导：**
  - 结合公式(12)(13)(14)得到：
    - 针对坐标u_j：

$$\frac{\partial u_j}{\partial \delta \mathbf{c}} = \frac{\partial u_j}{\partial \mathbf{P}_i'}\frac{\partial \mathbf{P}_i'}{\partial \delta \mathbf{c}}$$

$$= \overset{\text{公式(13)}}{\boxed{\frac{1}{P_z'}\begin{pmatrix}1 & 0 & -u_j\end{pmatrix}\mathbf{R}}}\overset{\text{公式(14)}}{\boxed{\begin{pmatrix}-f_x^{-1}P_x & 0 & -f_x^{-1} & 0\\ 0 & -f_y^{-1}P_y & 0 & -f_y^{-1}\\ 0 & 0 & 0 & 0\end{pmatrix}}}$$

$$= \frac{1}{P_z'}\begin{pmatrix}P_x f_x^{-1}(r_{20}u_j - r_{00})\\ P_y f_y^{-1}(r_{21}u_j - r_{01})\\ f_x^{-1}(r_{20}u_j - r_{00})\\ f_y^{-1}(r_{21}u_j - r_{01})\end{pmatrix}^T \quad \dots\dots(15)$$

- 同理针对坐标v_j：

$$\frac{\partial v_j}{\partial \delta \mathbf{c}} = \frac{\partial v_j}{\partial \mathbf{P}_i'}\frac{\partial \mathbf{P}_i'}{\partial \delta \mathbf{c}}$$

$$= \frac{1}{P_z'}\begin{pmatrix}P_x f_x^{-1}(r_{20}v_j - r_{10})\\ P_y f_y^{-1}(r_{21}v_j - r_{11})\\ f_x^{-1}(r_{20}v_j - r_{10})\\ f_y^{-1}(r_{21}v_j - r_{11})\end{pmatrix}^T \quad \dots\dots(16)$$

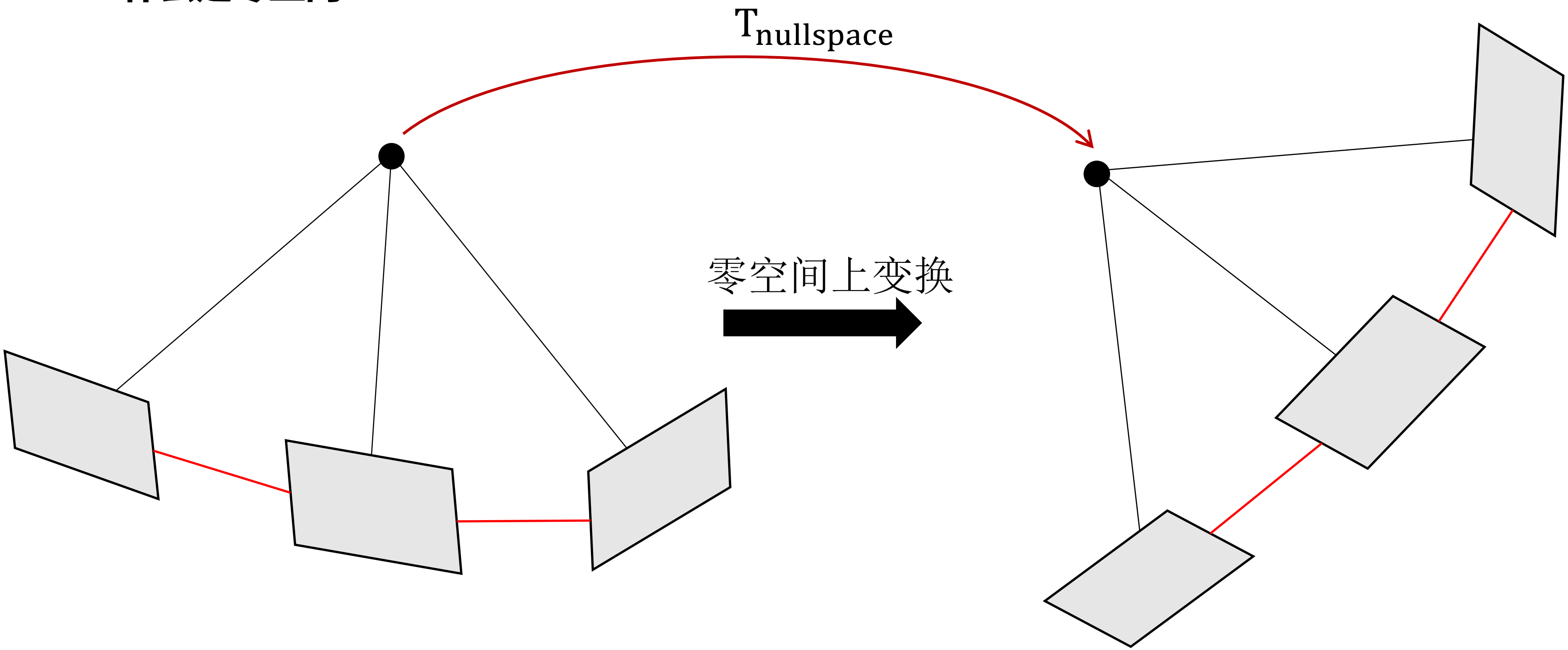
- **对相对内参求导：**
  - 结合公式(12)(15)(16)得到总体导数：

$$
\begin{aligned}
\frac{\partial r_k}{\partial \delta \mathbf{c}} &= \mathbf{J}_I \cdot \frac{\partial \mathbf{p}_j}{\partial \delta \mathbf{c}} \\
&= \mathbf{J}_I \left(\begin{pmatrix} u_j & 0 & 1 & 0 \\ 0 & v_j & 0 & 1 \end{pmatrix} + \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} \frac{\partial u_j}{\partial \delta \mathbf{c}} \\ \frac{\partial v_j}{\partial \delta \mathbf{c}} \end{pmatrix} \right) \\
&= \mathbf{J}_I \begin{pmatrix} u_j & 0 & 1 & 0 \\ 0 & v_j & 0 & 1 \end{pmatrix} + \mathbf{J}_I \frac{1}{P_z'} \begin{pmatrix} P_x(r_{20}u - r_{00}) & P_y \frac{f_x}{f_y}(r_{21}u - r_{01}) & (r_{20}u - r_{00}) & \frac{f_x}{f_y}(r_{21}u - r_{01}) \\ P_x \frac{f_y}{f_x}(r_{20}v - r_{10}) & P_y(r_{21}v - r_{11}) & \frac{f_y}{f_x}(r_{20}v - r_{10}) & (r_{21}v - r_{11}) \end{pmatrix}
\end{aligned}
$$

滑窗优化

1、滑窗优化
2、零空间的处理
3、DSO边缘化策略
4、FEJ策略
5、DSO中的solver
6、总结

- **什么是零空间：**



- **什么是零空间：**

- 对于系统的正规方程有：

$$\mathbf{H}\Delta x = \mathbf{b} \ldots\ldots(17)$$

- 其中海森矩阵的零空间满足：

$$\mathbf{H}\Delta x_{ns} = 0 \ldots\ldots(18)$$

- 结合式(17,18)，一定满足：

$$\mathbf{H}(\Delta x + \Delta x_{ns}) = \mathbf{b} \ldots\ldots(19)$$

- 因此在零空间上的漂移不会违反系统的约束，但是会对结果产生影响。

- 对于零空间上的状态量都无法满足，因此是不可观的。

$$\forall\theta \in \mathbb{R}^d, \{\theta \neq \theta_0\} \Rightarrow \{h(\theta) \neq h(\theta_0)\}$$

- 对于VO，7自由度的状态是都不可观的，都是位于系统零空间上的。

怎么求零空间：

- 对于VO来说，是7个自由度不可观的，即零空间是以这7个状态量为基构成的。需要注意的是零空间的变换是global的，求基方法即增加global小量扰动。
- 然而要求的是每帧位姿，使用伴随性质将global增量变换到local上，具体：
- 伴随性质定义为

$$\mathrm{Exp}\left(\mathrm{Ad}_{\mathbf{T}} \cdot \xi\right) \doteq \mathbf{T}\mathrm{Exp}(\xi)\mathbf{T}^{-1} \;\ldots\ldots(20)$$

- 对于位姿来说，左乘表示全局的扰动，右乘表示局部扰动。假设两个等价变换：

$$\mathbf{T}_{wc}\mathrm{Exp}(\delta\xi_c) = \mathrm{Exp}(\delta\xi_w)\mathbf{T}_{wc} \;\ldots\ldots(21)$$

- 公式（21）结合（20）变换后有：

$$\mathrm{Exp}(\delta\xi_c) = \mathbf{T}_{cw}\mathrm{Exp}(\delta\xi_w)\mathbf{T}_{wc}$$

$$\boxed{\delta\xi_c} = \mathrm{Ad}_{\mathrm{T_{cw}}}\boxed{\delta\xi_w}$$

局部增量　全局增量

$$\mathrm{R_{IC}R_{CW}} = \mathrm{R_{IW}}$$
$$\mathrm{R_{WC}} = \mathrm{R_{WI}R_{IC}}$$

2、零空间的处理

- **怎么求零空间：**

- 举例：以位姿中的一个值为例，求零空间基。

$$\delta\xi_1 = [\epsilon \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

$$\delta\xi_2 = [-\epsilon \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

- 求global扰动导致的local的扰动增量。

$$\mathbf{\Delta T_p} = \mathbf{T}\mathrm{Exp}(\delta\xi_1)\mathbf{T}^{-1}$$

$$\mathbf{\Delta T_n} = \mathbf{T}\mathrm{Exp}(\delta\xi_2)\mathbf{T}^{-1}$$

- 最终求得滑窗内的零空间基。

$$t_{x_{nullspace}} = \frac{\log(\mathbf{\Delta T_p}) - \log(\mathbf{\Delta T_n})}{\delta\xi_1 - \delta\xi_2}$$

- 对于VIO系统零空间有解析解

$$\mathbf{N}_1 = \begin{bmatrix} \mathbf{0}_3 & \mathbf{C}\left({}^I\bar{q}_{G,1}\right){}^G\mathbf{g} \\ \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\lfloor {}^G\mathbf{v}_{I,1} \times \rfloor {}^G\mathbf{g} \\ \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{p}_{I,1} \times \rfloor {}^G\mathbf{g} \\ \mathbf{I}_3 & -\lfloor {}^G\mathbf{f} \times \rfloor {}^G\mathbf{g} \end{bmatrix}$$

$$= [\mathbf{N}_{t,1} \quad | \quad \mathbf{N}_{r,1}]$$

- 横向量对应状态为

$$[\text{rotation, bias\_g, velocity, bias\_a, position, map}]^T$$

Roumeliotis S I, Kottas D G, Guo C, et al. Observability-constrained vision-aided inertial navigation: U.S. Patent 9,243,916[P]. 2016-1-26.

- **如何处理零空间：**

- Gauge fixation：把不可观的状态固定为某一些值，具体的为固定第一帧的位姿，等价于设对应的残差雅克比为0。相当于权重趋近于无穷大。
- Gauge prior：给这些状态设置先验，增加相应的惩罚项。
- Free gauge：让它自由演变，不管，使用伪逆或者增加阻尼（LM）使得问题可解。相当于先验权重为0。

	Size of parameter vec.	Hessian (Normal eqs)
Fixed gauge	$n-4$	inverse, $(n-4)\times(n-4)$
Gauge prior	n	inverse, $n\times n$
Free gauge	n	pseudoinverse, $n\times n$

Zhang Z, Gallego G, Scaramuzza D. On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation[J]. IEEE Robotics and Automation Letters, 2018, 3(3): 2710-2717.

2、零空间的处理

- **如何处理零空间：**

Constructing Orthogonal Projectors

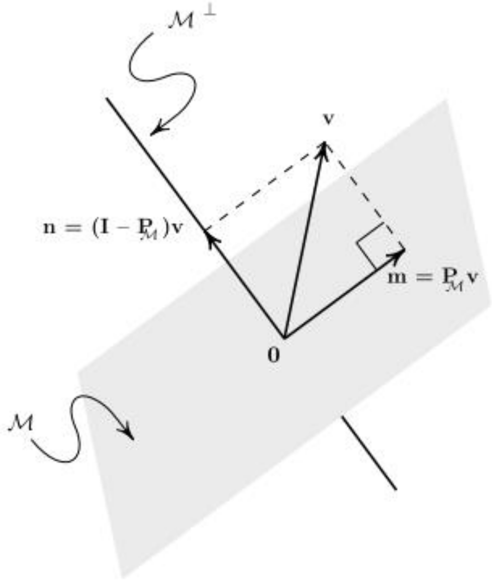
Let $\mathcal{M}$ be an r-dimensional subspace of $\Re^n$, and let the columns of $\mathbf{M}_{n\times r}$ and $\mathbf{N}_{n\times n-r}$ be bases for $\mathcal{M}$ and $\mathcal{M}^\perp$, respectively. The orthogonal projectors onto $\mathcal{M}$ and $\mathcal{M}^\perp$ are

- $\mathbf{P}_{\mathcal{M}} = \mathbf{M}\left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T$ and $\mathbf{P}_{\mathcal{M}^\perp} = \mathbf{N}\left(\mathbf{N}^T\mathbf{N}\right)^{-1}\mathbf{N}^T$. (5.13.3)

If $\mathbf{M}$ and $\mathbf{N}$ contain *orthonormal* bases for $\mathcal{M}$ and $\mathcal{M}^\perp$, then

- $\mathbf{P}_{\mathcal{M}} = \mathbf{M}\mathbf{M}^T$ and $\mathbf{P}_{\mathcal{M}^\perp} = \mathbf{N}\mathbf{N}^T$. (5.13.4)
- $\mathbf{P}_{\mathcal{M}} = \mathbf{U}\begin{pmatrix}\mathbf{I}_r & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\mathbf{U}^T$, where $\mathbf{U} = (\mathbf{M}\,|\,\mathbf{N})$. (5.13.5)
- $\mathbf{P}_{\mathcal{M}^\perp} = \mathbf{I} - \mathbf{P}_{\mathcal{M}}$ in all cases. (5.13.6)

Note: Extensions of (5.13.3) appear on p. 634.

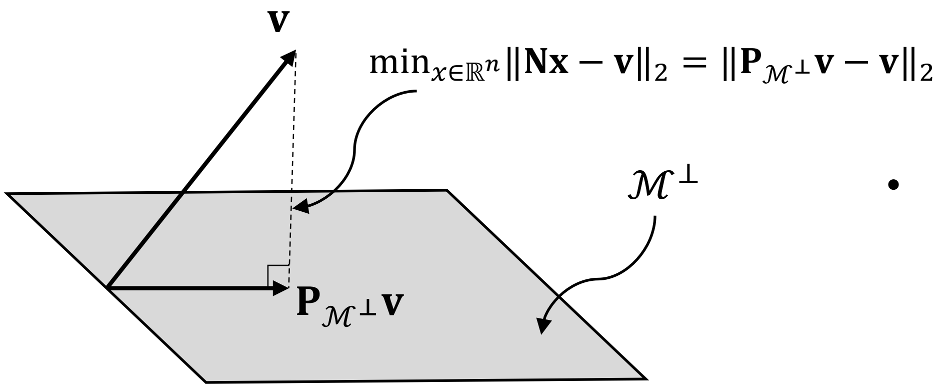


Meyer C D. Matrix analysis and applied linear algebra[M]. Siam, 2000.

2、零空间的处理

- **如何处理零空间：**

  - $\mathbf{N}$是空间$\mathcal{M}^{\perp}$的一组基，也就是$\mathcal{M}$的零空间的一组基。



  - 对于零空间，求解如下最小二乘

$$\mathbf{Nx} = \mathbf{v}$$

$$\mathbf{N}^{\mathbf{T}}(\mathbf{Nx}-\mathbf{v}) = \mathbf{0}$$

$$\mathbf{x} = \boxed{\left(\mathbf{N}^{\mathbf{T}}\mathbf{N}\right)^{-1}\mathbf{N}^{\mathbf{T}}}\mathbf{v}$$

伪逆 $\left\{\begin{array}{l}\text{1. 公式}\\ \text{2. SVD}\\ \text{3. QR}\end{array}\right.$

  - 想求得零空间上的分量

$$\mathbf{Nx} = \mathbf{P}_{\mathcal{M}^{\perp}}\mathbf{v}$$

$$\mathbf{P}_{\mathcal{M}^{\perp}}\mathbf{v} = \mathbf{N}\boxed{\left(\mathbf{N}^{\mathbf{T}}\mathbf{N}\right)^{-1}\mathbf{N}^{\mathbf{T}}}\mathbf{v}$$

伪逆

Meyer C D. Matrix analysis and applied linear algebra[M]. Siam, 2000.

滑窗优化

1、滑窗优化

2、零空间的处理

3、DSO边缘化策略

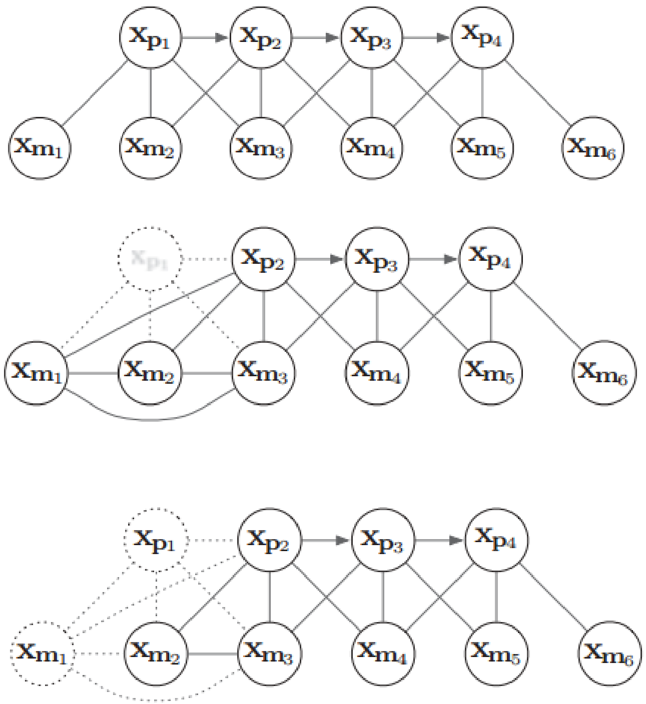
4、FEJ策略

5、DSO中的solver

6、总结

边缘化的方法

- **边缘化的方法**



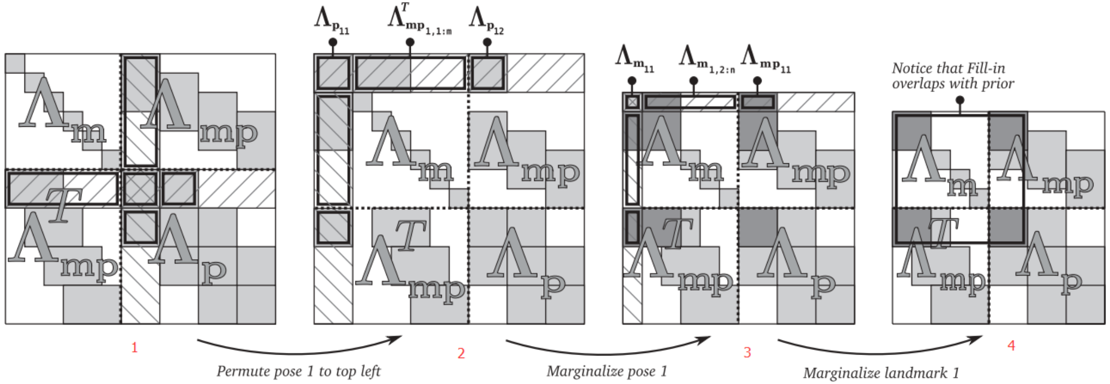


Table 3: Overall Running Time

Marginalization Algorithm	Average Running Time(s)
Optimization with Schur Complement Marginalization	0.0440574
Optimization with Given Rotations	0.00953969

Table 4: Marginalization Method Comparison

Marginalization Algorithm	Average Running Time(s)	Speed Rank
Full Schur Complement	0.0321067	5
Null Space with Given Rotations	0.0000867469	2
Null Space with HouseHolder QR	0.000116933	3
Null Space with Direct HouseHolder	0.000118446	4
Null Space with Projection Form	0.0000757341	1

Yang Y, Maley J, Huang G. Null-space-based marginalization: Analysis and algorithm[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 6749-6755.

边缘化的方法

边缘化和高斯推断

- 边缘化是把变量积分消掉，条件概率是把变量当作常量消去

$$p(\boldsymbol{\alpha},\boldsymbol{\beta})=\mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_\alpha\\ \boldsymbol{\mu}_\beta\end{bmatrix},\begin{bmatrix}\Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta}\\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta}\end{bmatrix}\right)=\mathcal{N}^{-1}\left(\begin{bmatrix}\boldsymbol{\eta}_\alpha\\ \boldsymbol{\eta}_\beta\end{bmatrix},\begin{bmatrix}\Lambda_{\alpha\alpha} & \Lambda_{\alpha\beta}\\ \Lambda_{\beta\alpha} & \Lambda_{\beta\beta}\end{bmatrix}\right)$$

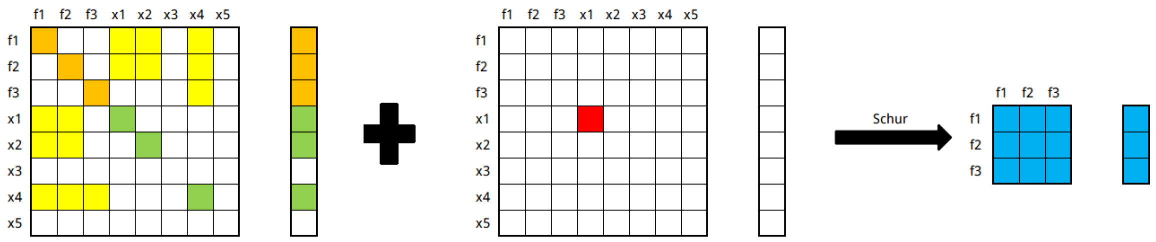
	MARGINALIZATION $p(\boldsymbol{\alpha})=\int p(\boldsymbol{\alpha},\boldsymbol{\beta})\,d\boldsymbol{\beta}$	CONDITIONING $p(\boldsymbol{\alpha}\mid\boldsymbol{\beta})=p(\boldsymbol{\alpha},\boldsymbol{\beta})/p(\boldsymbol{\beta})$
COVARIANCE FORM	$\boldsymbol{\mu}=\boldsymbol{\mu}_\alpha$ $\Sigma=\Sigma_{\alpha\alpha}$	$\boldsymbol{\mu}'=\boldsymbol{\mu}_\alpha+\Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}(\boldsymbol{\beta}-\boldsymbol{\mu}_\beta)$ $\Sigma'=\Sigma_{\alpha\alpha}-\Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}\Sigma_{\beta\alpha}$
INFORMATION FORM	$\boldsymbol{\eta}=\boldsymbol{\eta}_\alpha-\Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta$ $\Lambda=\Lambda_{\alpha\alpha}-\Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}$	$\boldsymbol{\eta}'=\boldsymbol{\eta}_\alpha-\Lambda_{\alpha\beta}\boldsymbol{\beta}$ $\Lambda'=\Lambda_{\alpha\alpha}$

- **点marg策略**

  - 边缘化使得点的残差变很少、最新一帧看不到、连续两帧外点，主帧被边缘化的点候选边缘化或删除
  - 如果是内点，并且逆深度协方差小于阈值则标记为边缘化
  - 其它的直接丢弃
  - 把挑出来的边缘化点构造H- HSC /b- bSC，加到HM，bM中

  - 注：点的边缘化会导致帧之间的Hessian变稠密，因此需要合理选择。
  - 注：DSO里将marg掉的点的所有观测都扔掉，不会移动host。

边缘化：



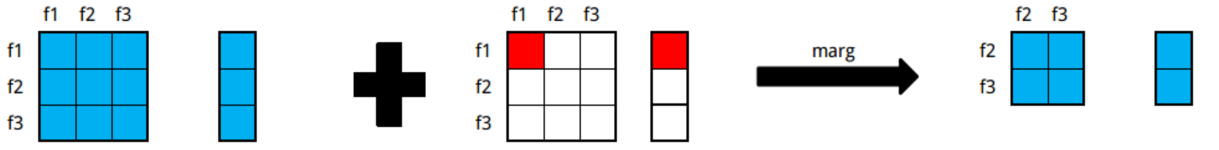
使用F10 F20 F30计算 使用R10 R20 R30计算

3、DSO边缘化策略

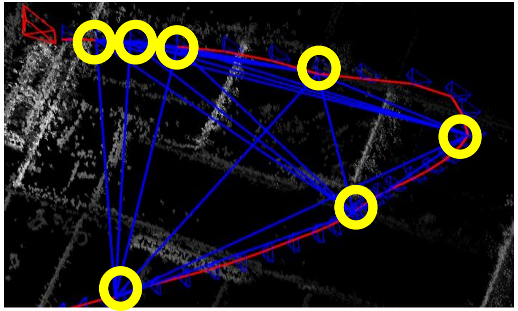
● 帧marg策略

- 留下的点(去除边缘化+删除)占当前可见所有点小于5%
- 和参考帧比曝光变化较大
- 保证滑窗内有5个关键帧
- 如果还大于7个关键帧，则边缘化掉到最新关键帧的距离占所有距离比最大的。保证良好的空间结构

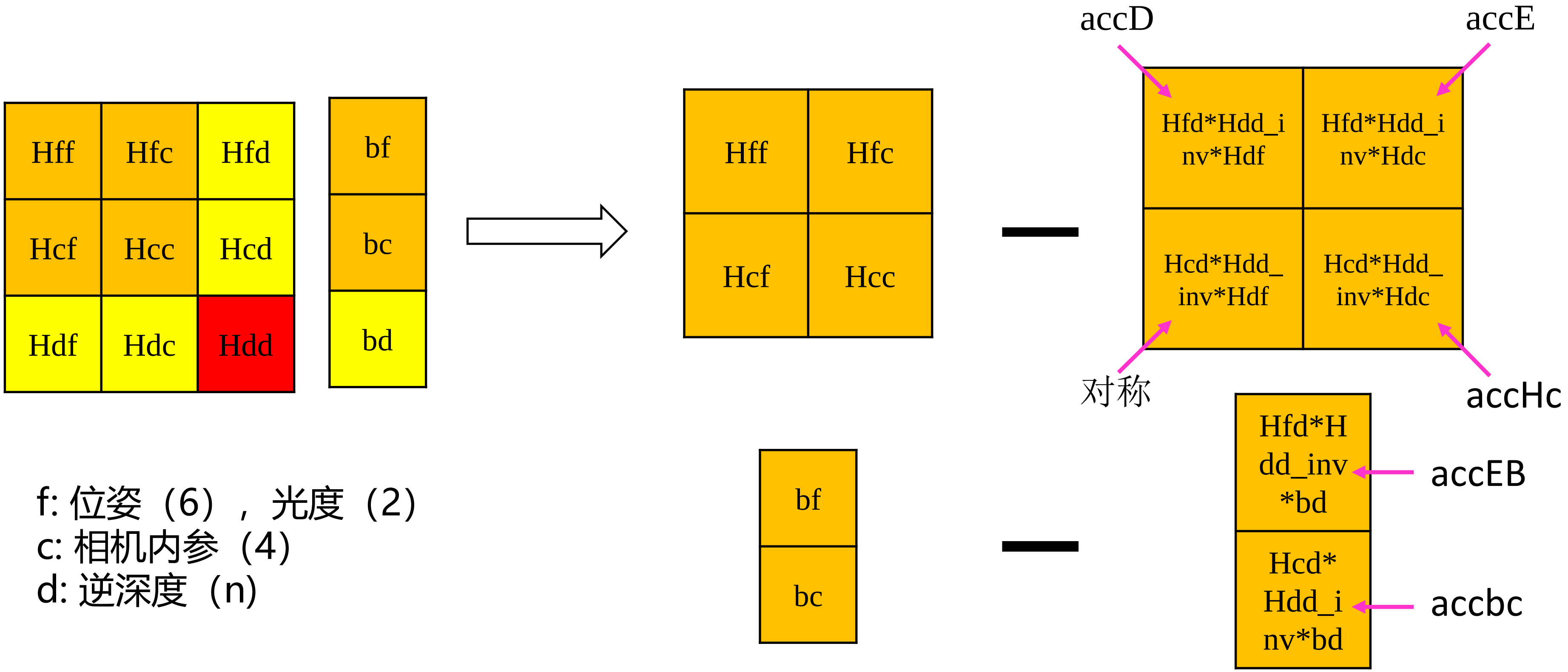
$$s\left(I_i\right)=\sqrt{d(i,1)}\sum_{j\in[3,n]\setminus\{i\}}(d(i,j)+\epsilon)^{-1}$$



边缘化的时候加上先验

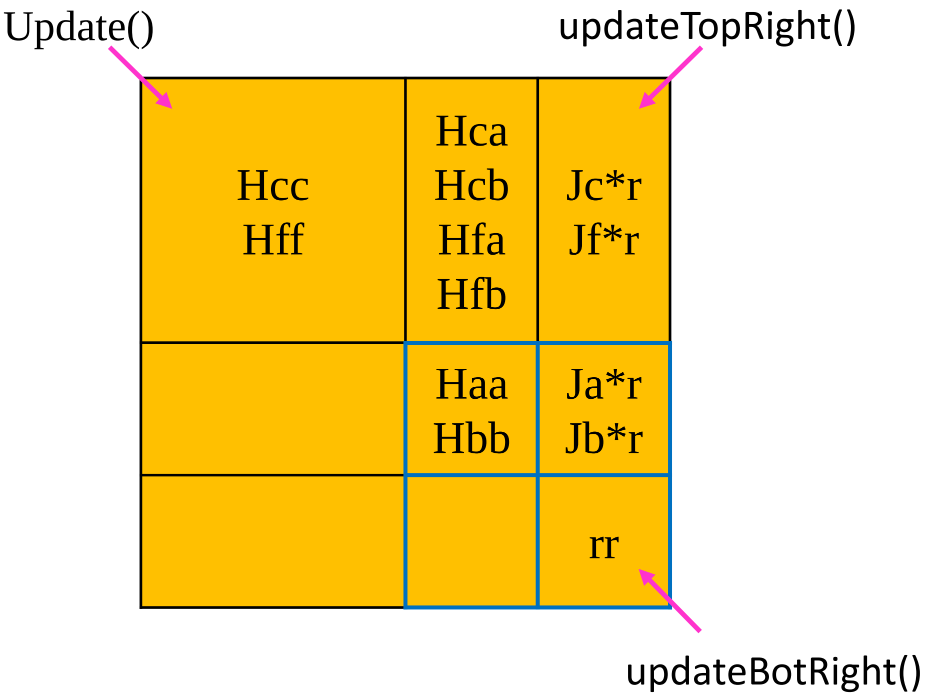


- **AccumulatedSCHessianSSE**

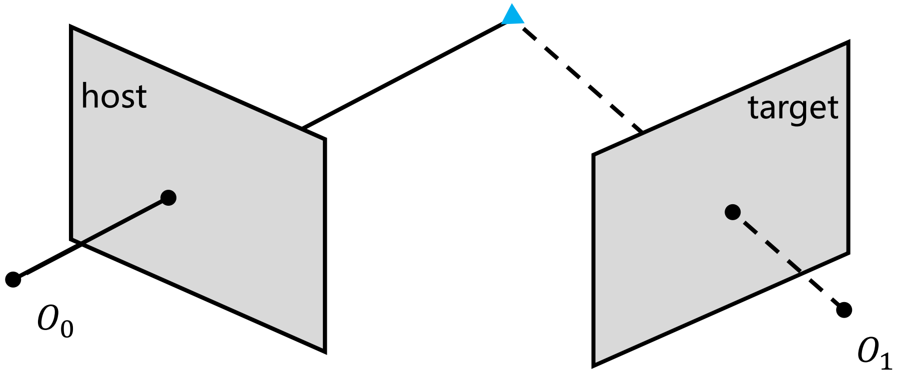


f: 位姿（6），光度（2）
c: 相机内参（4）
d: 逆深度（n）

- **AccumulatedTopHessianSSE**



- 为了保证滑窗内的每一帧都被优化，需要使得Hff对称
- 还会加上先验.



滑窗优化

1、滑窗优化

2、零空间的处理

3、DSO边缘化策略

4、FEJ策略

5、DSO中的solver

6、总结

FEJ介绍

- 普通的BA中在[0,k']内的信息矩阵

$$\mathbf{H}_{\mathrm{ba}}(k') = \sum_{(i,j)\in\mathcal{S}} \mathbf{J}_{ij}^{T}(k')\mathbf{R}_{ij}^{-1}\mathbf{J}_{ij}(k')$$

- 滑窗算法中的信息矩阵

$$\mathbf{H}_{\mathrm{mar}}(k') = \sum_{(i,j)\in\mathcal{S}_m} \mathbf{J}_{ij}^{T}(k)\mathbf{R}_{ij}^{-1}\mathbf{J}_{ij}(k) + \sum_{(i,j)\in\mathcal{S}_a(k')} \mathbf{J}_{ij}^{T}(k')\mathbf{R}_{ij}^{-1}\mathbf{J}_{ij}(k')$$

- 这两个不同状态使得某些可以消去的项变得不行。因此两个矩阵的秩变得不一样。

$$\operatorname{rank}(\mathbf{H}_{\mathrm{mar}}(k')) = \operatorname{rank}\left(\mathbf{H}_{\mathrm{ba}}(k')\right) + 3$$

- 这使得信息矩阵包含了更多的信息（沿着状态空间的更多方向），因此低估了状态的不确定性。使得该问题变得不一致且是次优的。

- 在当前状态和边缘化状态相连时，使用边缘化时刻的状态来计算后面的雅克比。这样保证了信息矩阵只有一个状态，这样导致了线性化误差，因此使用时还是要保证在较好的状态下。

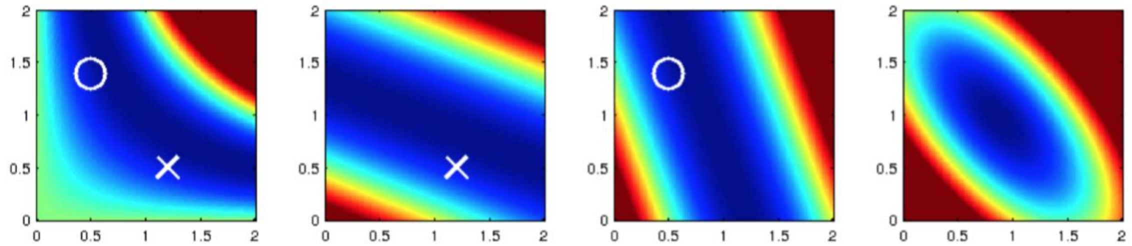
Dong-Si T C, Mourikis A I. Consistency analysis for sliding-window visual odometry[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 5202-5209.

4、FEJ策略

FEJ举例

Windowed, real-time optimization: Consistency.

(for now, let's assume we have initializations, and know which points to use and where they are visible.)



$$E = E_1 + E_2 = (xy-1)^2 + (xy-1)^2$$

$E'_1 = E_1$

Lin. around (0.5, 1.4)

$E'_2 = E_2$

Lin. around (1.2, 0.5)

$E' = E'_1 + E'_2$

nullspace disappears!

never combine linearizations around different linearization points,
especially in the presence of non-linear nullspaces!
It will render unobservable dimensions observable, and corrupt the system.

FEJ举例说明

- 类似SLAM中对能量进行线性化：

$$E = E_1 + E_2$$

$$E_1 = E_2 = (xy-1)^2$$

$$E_1' = E_2' = \left((x_0 y_0 - 1) + y_0(x - x_0) + x_0(y - y_0)\right)^2$$

- 对于能量在不同的点进行线性化展开：

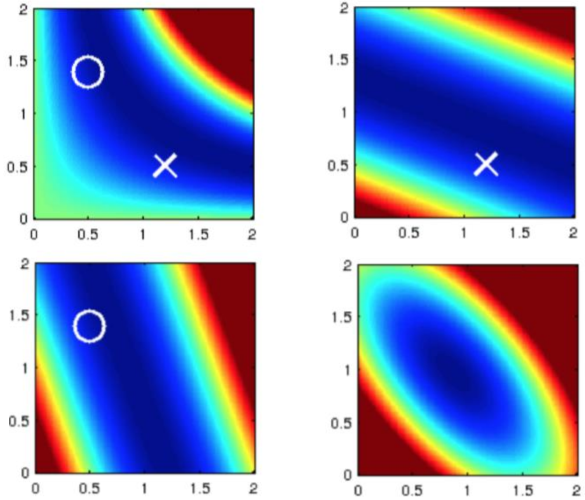
$$E_1' = (1.4x + 0.5y - 1.7)^2$$

$$E_2' = (0.5x + 1.2y - 1.6)^2$$

- 求得总能量：

$$E' = E_1' + E_2'$$

$$= (1.4x + 0.5y - 1.7)^2 + (0.5x + 1.2y - 1.6)^2$$



● DSO中FEJ的使用

- 因为DSO部分状态使用FEJ，所以需要保存不同状态的值。
- 固定的线性化点位置的相对位姿。

FrameFramePrecalc::PRE_RTll_0/PRE_tTll_0

- 优化更新状态后的相对位姿。

FrameFramePrecalc::PRE_RTll / PRE_tTll

- 固定线性化点处的光度参数b。

FrameFramePrecalc::PRE_b0_mode

- 其中位姿 / 光度参数使用固定线性化点，逆深度 / 内参 / 图像导数都没有固定线性化点。
- 逆深度每次都会重新设置线性化点，相当于没有固定，虽然代码写的很像。
- 残差更新，HM和bM是边缘化得到的，因为被边缘化后没办法project计算新的残差，因此使用一阶泰勒方式更新。

bM_top = (bM+ HM * getStitchedDeltaF()

滑窗优化

1、滑窗优化
2、零空间的处理
3、DSO边缘化策略
4、FEJ策略
5、DSO中的solver
6、总结

- **求解中的伴随性质**

  - 伴随性质

$$\mathrm{Exp}\left(\mathrm{Ad}_{\mathbf{T}} \cdot \xi\right) \doteq \mathbf{T}\mathrm{Exp}(\xi)\mathbf{T}^{-1}$$

  - 线性化点相对位姿和绝对位姿的关系:

$$\mathbf{T}_{th} = \mathbf{T}_{tw}\mathbf{T}_{hw}^{-1}$$

  - 相对位姿se3对host se3导数:

$$\mathrm{Exp}(\delta\xi_{th})\mathbf{T}_{th} = \mathbf{T}_{tw}(\mathrm{Exp}(\delta\xi_h)\mathbf{T}_{hw})^{-1}$$

$$\begin{aligned}
\mathrm{Exp}(\delta\xi_{th}) &= \mathbf{T}_{tw}\mathbf{T}_{hw}^{-1}\mathrm{Exp}(-\delta\xi_h)\mathbf{T}_{th}^{-1} \\
&= \mathbf{T}_{th}\mathrm{Exp}(-\delta\xi_h)\mathbf{T}_{th}^{-1} \\
&= \mathrm{Exp}\left(-\mathrm{Ad}_{\mathbf{T}_{\mathrm{th}}}(\delta\xi_h)\right)
\end{aligned}$$

  - 得到相对位姿和绝对位姿之间的关系

$$\delta\xi_{th} = -\mathrm{Ad}_{\mathbf{T}_{\mathrm{th}}}\delta\xi_h$$

$$\frac{\partial\xi_{th}}{\partial\xi_h} = -\mathrm{Ad}_{\mathbf{T}_{\mathrm{th}}}$$

- **求解中的伴随性质**
  - 相对位姿se3对target se3求导:

$$\mathrm{Exp}(\delta\xi_{th})\mathbf{T}_{th} = \mathrm{Exp}(\delta\xi_t)\mathbf{T}_{tw}\mathbf{T}_{hw}^{-1}$$

$$\mathrm{Exp}(\delta\xi_{th}) = \mathrm{Exp}(\delta\xi_t)\mathbf{T}_{tw}\mathbf{T}_{hw}^{-1}\mathbf{T}_{th}^{-1}$$

$$= \mathrm{Exp}(\delta\xi_t)$$

  - 得到相对位姿和绝对位姿之间的关系

$$\delta\xi_{th} = \delta\xi_t$$

$$\frac{\partial\xi_{th}}{\partial\xi_t} = \mathbf{I}$$

  - 相对光度参数对host和target求导
  - host和target之间的相对光度参数为

$$\delta a = -\frac{t_j e^{a_j}}{t_i e^{a_i}}, \quad \delta b = -b_j + \frac{t_j e^{a_j}}{t_i e^{a_i}} b_i$$

  - 相对量对绝对量求导

$$\frac{\partial\delta a}{\partial a_j} = -\frac{t_j e^{a_j}}{t_i e^{a_i}} = -e^{a_{ji}} \qquad \frac{\partial\delta a}{\partial a_i} = \frac{t_j e^{a_j}}{t_i e^{a_i}} = e^{a_{ji}}$$

$$\frac{\partial\delta b}{\partial b_j} = \frac{t_j e^{a_j}}{t_i e^{a_i}} = -1 \qquad \frac{\partial\delta b}{\partial b_i} = \frac{t_j e^{a_j}}{t_i e^{a_i}} = e^{a_{ji}}$$

- **先验说明**

- **EF->cPrior   EF->cDeltaF**
- 相机参数的先验信息：Hessian是很大的常数；delta估计值与设置的初值的差；

- **p->priorF p->deltaF**
- 点逆深度的先验信息：第一帧上的有(2500)，其它帧上的为0；deltaF值为0(因为线性化点被更新了)

- **EF->frames->prior   EF->frames->delta_prior**
- 位姿光度的先验信息：第一帧位姿Hessian常数，光度Hessian常数(1e14); 其它帧位姿Hessian为零，光度Hessian常数(1e12/1e8) ； delta估计值与固定线性化点的差;

- **fixLinearizationF()**
- 因为状态需要固定在线性化点位置，边缘化一个点前会重新线性化一次，这时得到的resF是使用最新状态线性化的，使用该函数减去J×delta得到在线性化点状态的residual。

5、DSO中的solver

- **各种solver**

$$\mathbf{Hx} = \mathbf{b}$$

- SOLVER_ORTHOGONALIZE_SYSTEM

进行Schur消元，使用正交投影$\mathbf{H}$和$\mathbf{b}$

- SOLVER_SVD
- SOLVER_SVD_CUT7

$$\mathbf{H}' = \mathbf{SHS} \quad \mathbf{b}' = \mathbf{Sb}$$

$$\mathbf{H}'\mathbf{x} = \mathbf{U\Sigma V^T x} = \mathbf{b}$$

$$\mathbf{\Sigma V^T x} = \mathbf{U^T b}$$

对奇异值较小的，置零处理。
若用CUT7，要保证后7列为零。

$$\mathbf{x} = \mathbf{V\Sigma'^{-1}U^T b}$$

- SOLVER_ORTHOGONALIZE_X_LATER
- SOLVER_ORTHOGONALIZE_X

进行Schur消元，使用正交投影$\mathbf{x}$

- SOLVER_REMOVE_POSEPRIOR

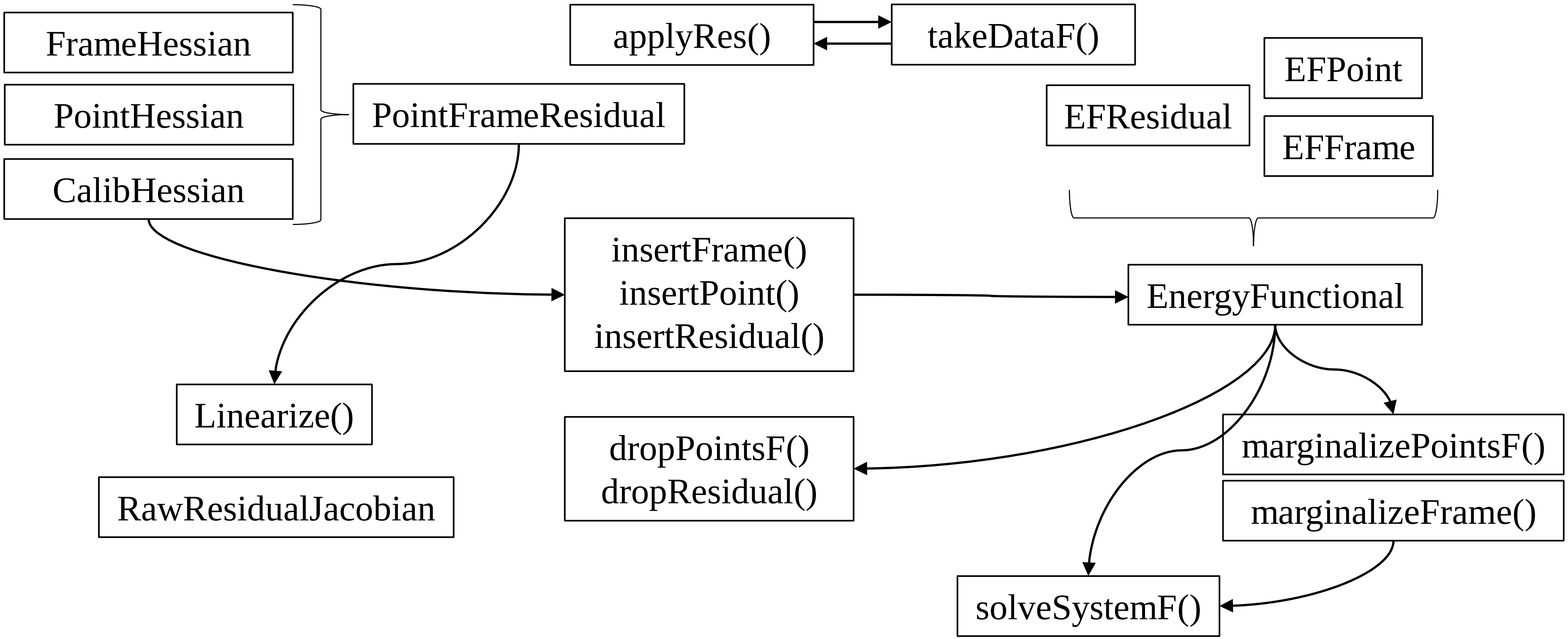
不使用第一帧上点和位姿的大先验

- SOLVER_ORTHOGONALIZE_POINTMARG
- SOLVER_ORTHOGONALIZE_FULL

5、DSO中的solver

optimize

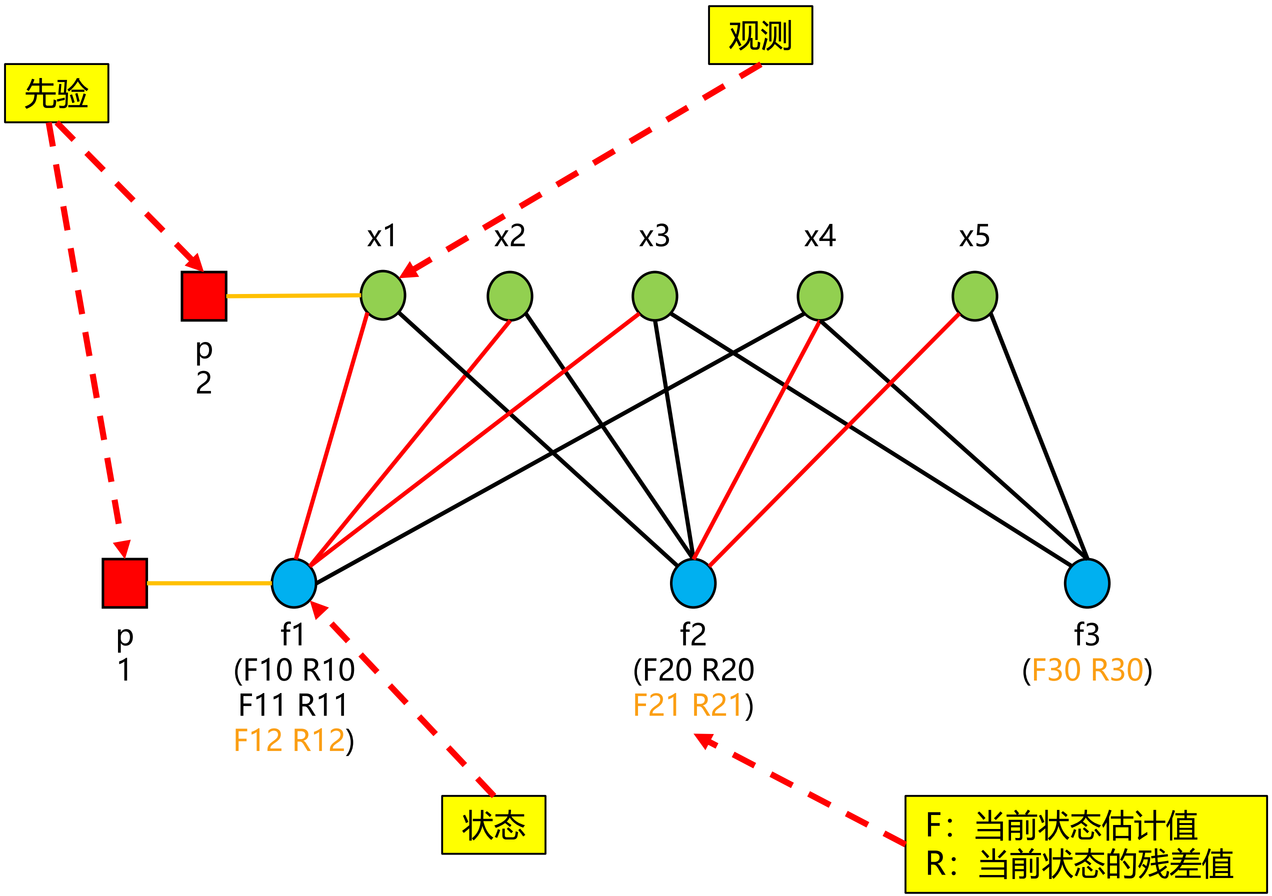
- **变量梳理**



5、DSO中的solver

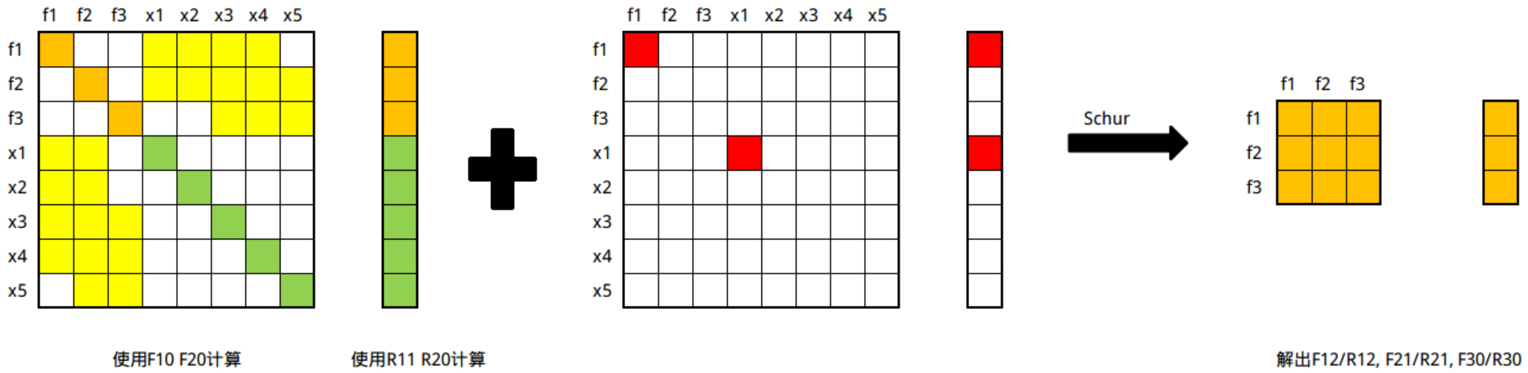
边缘化举例

f3为新加入的帧，完成优化后得到状态F12/R12, F21/R21, F30/R30，将F30/R30设置为**线性化点**



边缘化举例

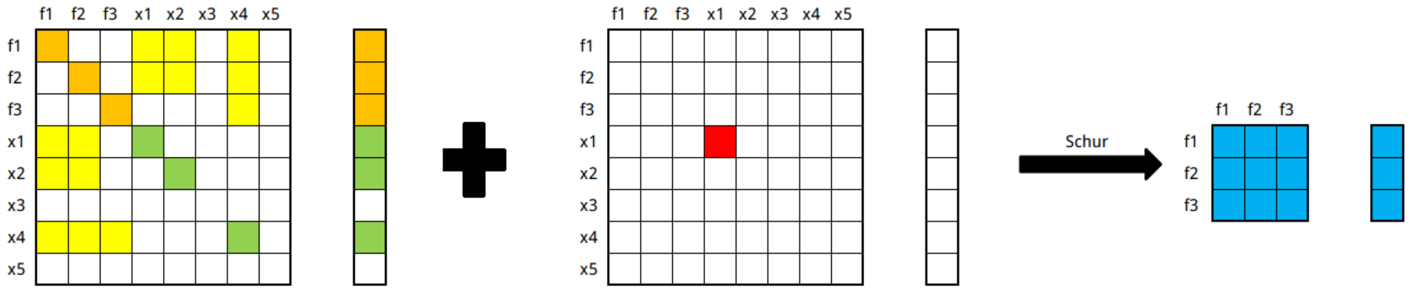
使用Hessian来表示上述过程



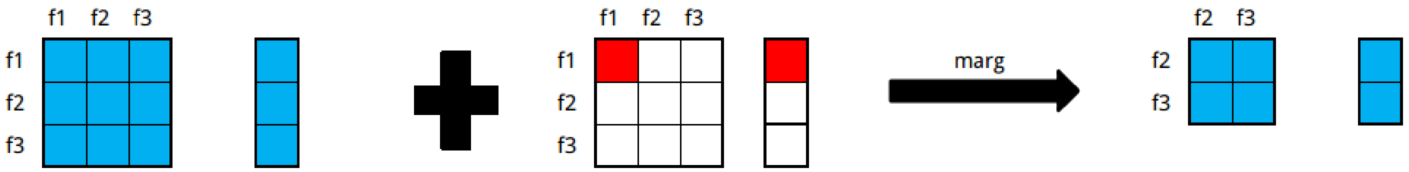
边缘化举例

x1、x2、x4边缘化掉，x3丢掉，构造边缘化先验HM，bM

边缘化：



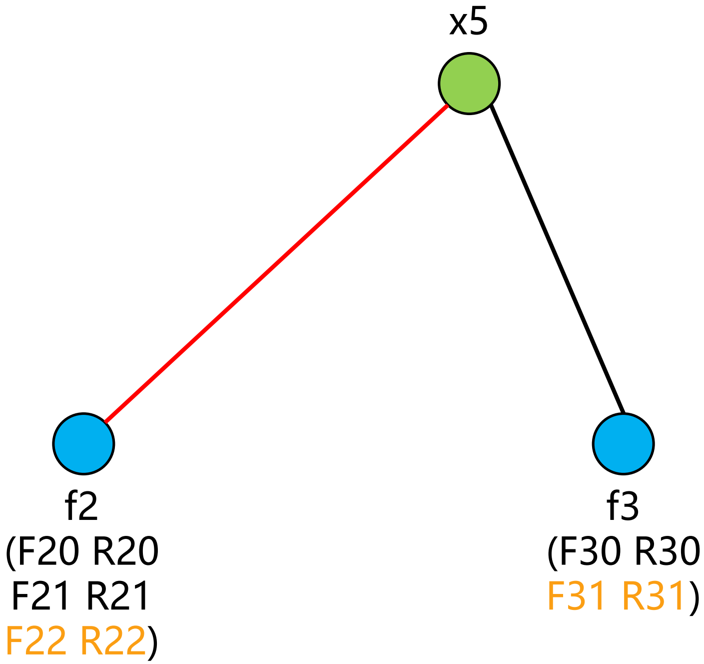
使用F10 F20 F30计算　　使用R10 R20 R30计算



边缘化的时候加上先验

边缘化举例

边缘化之后会得到边缘化的先验矩阵，然后求解得状态：F22/R22，F31/R31

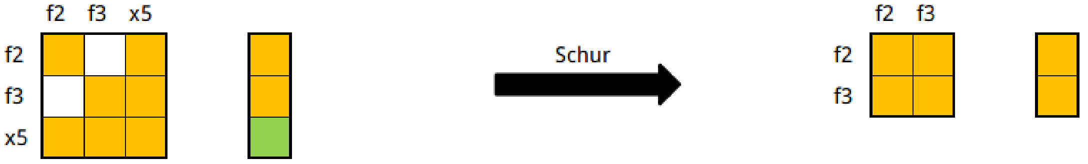


5、DSO中的solver

边缘化举例

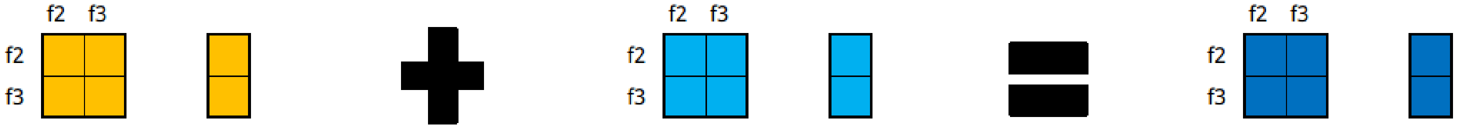
使用Hessian来表示上述过程

边缘化后：



使用F20 F30计算

使用R21 R30计算



使用b-Hdx计算

解出F22/R22, F31/R31

- **Code**

Residuals.cpp

OptimizationBackend/

HessianBlocks.cpp

FullSystem.cpp

FullSystemOptimize.cpp

FullSystemMarginalize.cpp

滑窗优化

1、滑窗优化
2、零空间的处理
3、DSO边缘化策略
4、FEJ策略
5、DSO中的solver
6、总结

6、总结

滑窗优化

- 留下的点(去除边缘化+删除)占所有点小于5%
- 和参考帧比曝光变化较大
- 保证滑窗内有5个关键帧
- 如果还大于7个关键帧，则边缘化掉到最新关键帧的距离占所有距离比最大的。保证良好的空间结构

3.3 标记边缘化帧

makeKeyFrame()

traceNewCoarse()

flagFramesForMarginalization()

- 加入关键帧，能量函数中insertFrame()，设置相对位姿和线性化点
- 构建新关键帧和之前关键帧之间的残差PointFrameResidual，并插入能量函数insertResidual()，DSO选择先全部构建，之后再删除

3.4 构建旧残差

makeKeyFrame()

添加帧和点后，重新梳理ID

- 在最新关键帧上生成距离地图(点附近第一层是1，第二层2,……)
- 满足搜索范围小于8，质量好，逆深度为正，删除外点和边缘化帧上的点
- 将点投影到距离地图上，大于阈值(阈值根据当前点数量调节)激活
- 使用LM优化选出来点的逆深度，将内点构建PointHessian和残差一起加入能量函数
- 删除未收敛的点

3.5 激活未成熟点

activatePointsMT()

ef->makeIDX()

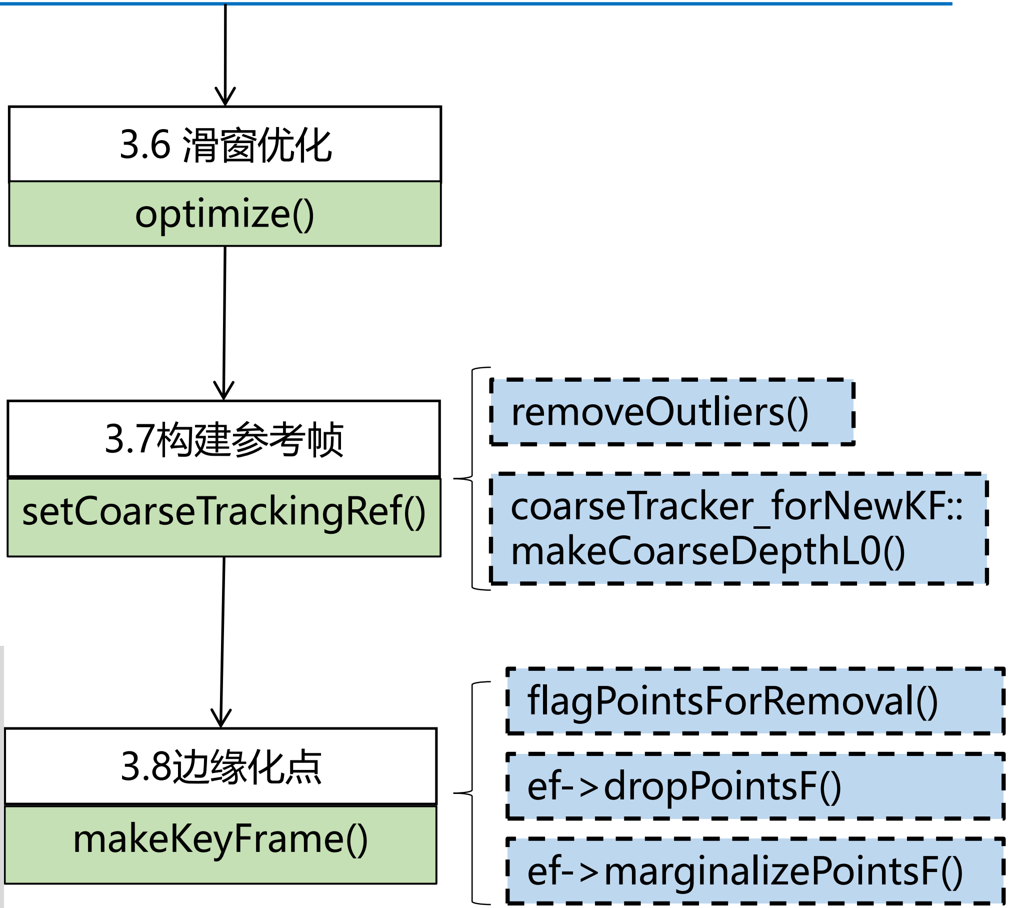
6、总结

滑窗优化

- 使用GN法对位姿、光度参数、逆深度、相机内参进行优化，由于边缘化需要维护两个H矩阵和b向量
- 其中位姿和光度参数使用FEJ，除了最新一帧，相关H矩阵固定在上一次优化，残差仍然使用更新后的状态求
- 被边缘化部分残差更新：b=b+H*delta
- 其中第一帧位姿和其上点逆深度，由于初始化具有先验，光度参数有先验
- 使用伴随性质将相对位姿变为世界系下的(local -> global)
- 减去求解出的增量零空间部分，防止在零空间乱飘

- 将最新帧设置为参考帧，并将所有的点向最新帧投影，并且在金字塔从下向上使用协方差加权生成逆深度，然后对于每一层使用周围点来尽可能生成像素逆深度，这样保证有足够多得点来跟踪，鲁棒。

- 边缘化使得点的残差变很少、最新一帧看不到、连续两帧外点，主帧被边缘化的点候选边缘化或删除
- 如果是内点，并且逆深度协方差小于阈值则标记为边缘化
- 其它的直接丢弃
- 把挑出来的边缘化点构造H- HSC /b- bSC，加到HM，bM中



6、总结

- 在最新帧第0层提取随机方向梯度最大的像素，并且构造成ImmaturePoint

3.9 提取新像素

makeNewTraces()

pixelSelector->makeMaps()

- 将被边缘化的帧的8个状态挪到右下角，然后计算Schur Complement，将其消掉
- 删除在被边缘化帧上的残差
- DSO里面操作的都是PoseGraph

3.10 边缘化帧

marginalizeFrame()

ef->marginalizeFrame()

Reference

- Nocedal J, Wright S. Numerical optimization[M]. Springer Science & Business Media, 2006.
- Yang Y, Maley J, Huang G. Null-space-based marginalization: Analysis and algorithm[C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017: 6749-6755.
- Roumeliotis S I, Kottas D G, Guo C, et al. Observability-constrained vision-aided inertial navigation: U.S. Patent 9,243,916[P]. 2016-1-26.
- Meyer C D. Matrix analysis and applied linear algebra[M]. Siam, 2000.
- Zhang Z, Gallego G, Scaramuzza D. On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation[J]. IEEE Robotics and Automation Letters, 2018, 3(3): 2710-2717.
- Dong-Si T C, Mourikis A I. Consistency analysis for sliding-window visual odometry[C]//2012 IEEE International Conference on Robotics and Automation. IEEE, 2012: 5202-5209.

结语

感谢各位聆听!

Thanks for Listening