

Bundle Adjustment

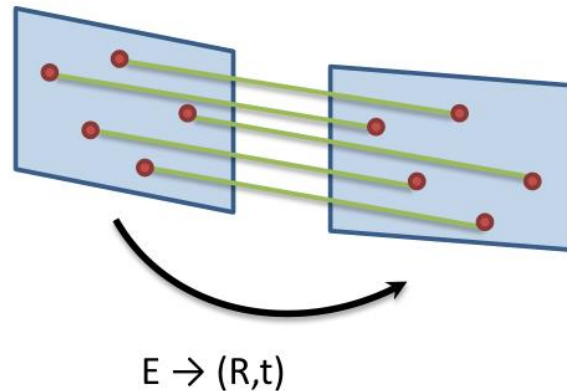
(10.2)

최종호

2019.03.17

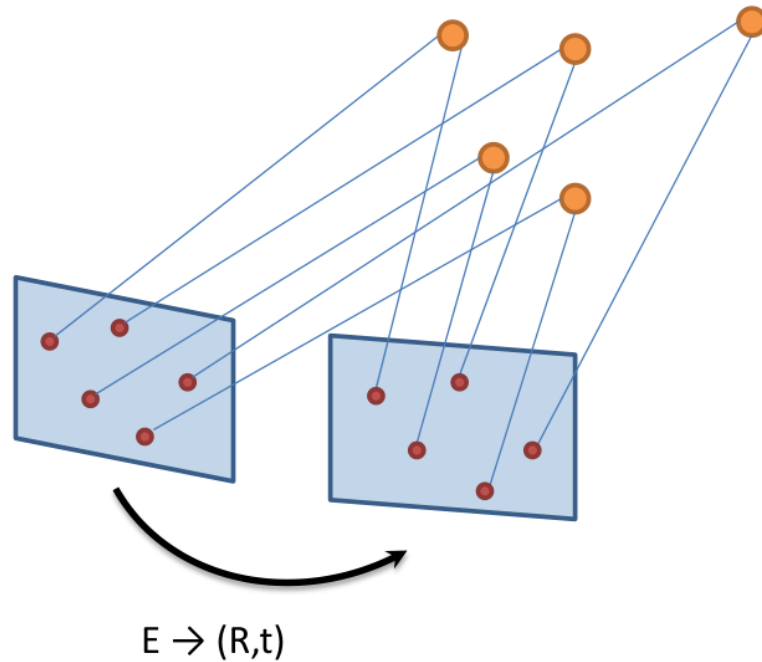
Feature Based Visual Odometry (Recap - 7장)

- Relative camera pose from 2 views:
 - Fundamental matrix
 - Essential matrix



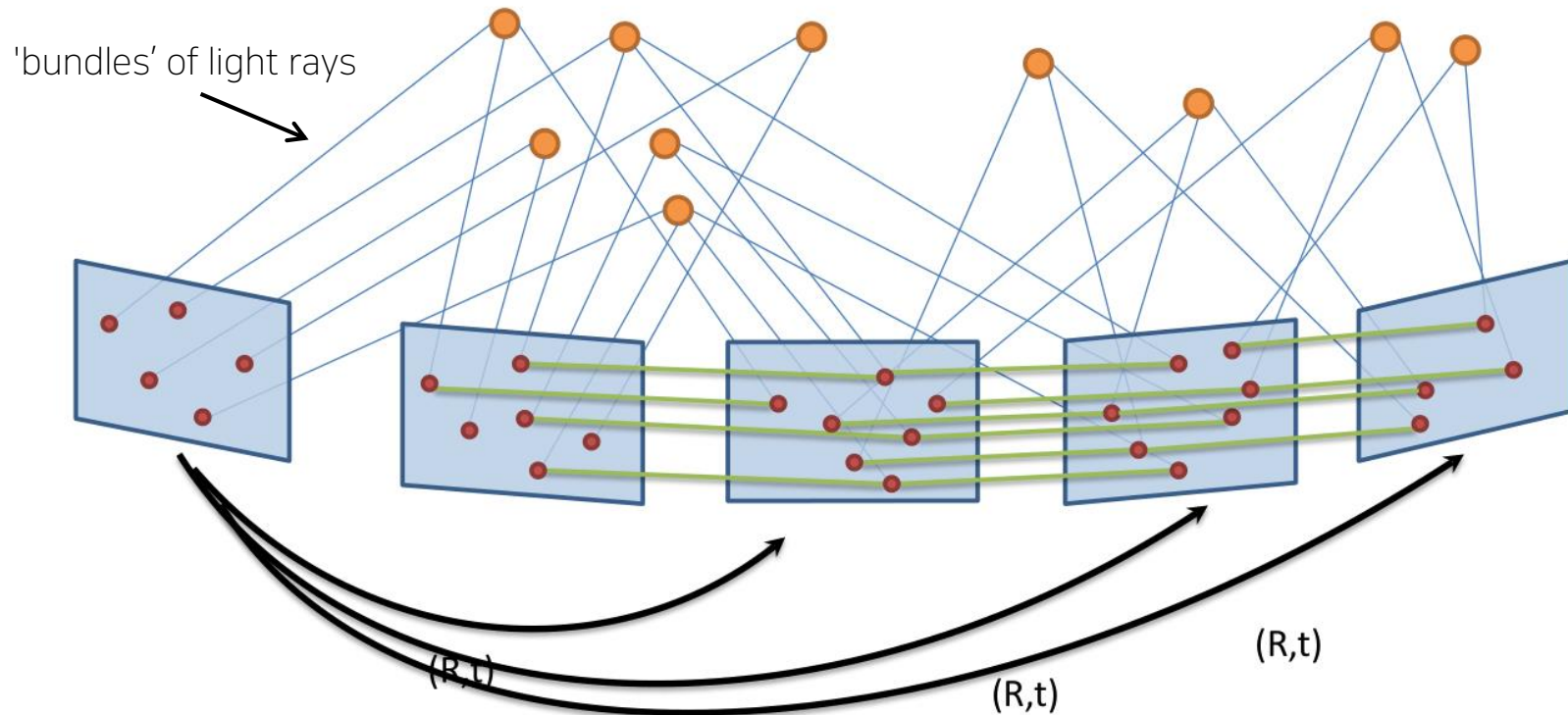
Feature Based Visual Odometry (Recap - 7장)

- 3D Points:
 - Triangulation



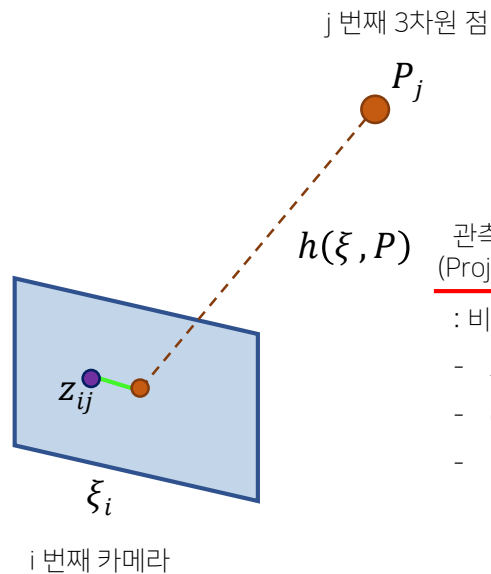
Feature Based Visual Odometry (Recap - 7장)

- Subsequent views:
 - Bundle adjustment - 모든 카메라 포즈와 3차원 점을 조정(최적화)한다.



최적화하고자 하는 BA Cost Function은?

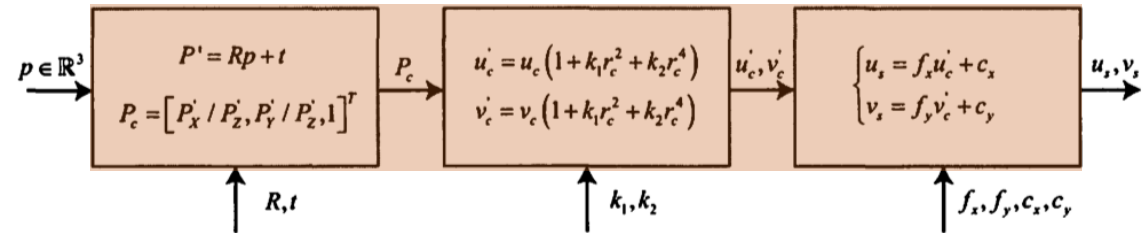
- Total reprojection error를 줄이는 방향으로 최적화하라.



관측 모델
(Projection)

: 비선형 모델 (non-linear optimization 활용)

- 초기값에서 시작하여,
- cost function을 줄이는 방향으로,
- 점진적 방정식을 통해 증가분($\Delta\xi, \Delta P$) 계산



$$z = h(x, y). \quad (10.40)$$

$$e = z - h(\xi, p). \quad (10.41)$$

$$\text{minimize} \longrightarrow \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|e_{ij}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \underbrace{\|z_{ij} - h(\xi_i, p_j)\|^2}_{\text{reprojection error}}. \quad (10.42)$$

증가분(Δ) 계산을 위한 점진적 방정식은?

- Cost function에서 최적화할 변수:

$$\mathbf{x} = [\xi_1, \dots, \xi_m, p_1, \dots, p_n]^T. \quad (10.43)$$

$$\mathbf{x}_c = [\xi_1, \xi_2, \dots, \xi_m]^T \in \mathbb{R}^{6m},$$

$$\mathbf{x}_p = [p_1, p_2, \dots, p_n]^T \in \mathbb{R}^{3n},$$

$$e(\mathbf{x} + \Delta \mathbf{x}) \approx e(\mathbf{x}) + \mathbf{J} \Delta \mathbf{x}. \quad (7.36)$$

$$\mathbf{J} = [\mathbf{F} \ \mathbf{E}]$$

\mathbf{F} : 현재 cost function에 카메라 포즈(ξ)로 편미분
 \mathbf{E} : 현재 cost function에 3차원 점(P)로 편미분

$$\frac{1}{2} \|f(\mathbf{x} + \Delta \mathbf{x})\|^2 \approx \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|\mathbf{e}_{ij} + \mathbf{F}_{ij} \Delta \xi_i + \mathbf{E}_{ij} \Delta p_j\|^2. \quad (10.44)$$

이 값을 줄일 수 있는 증가분을 계산

근데 어떻게 계산?

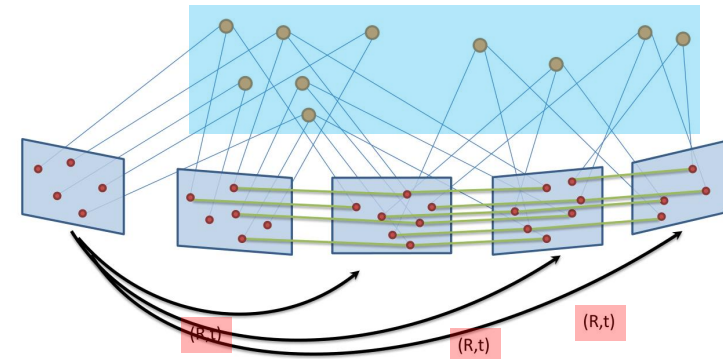
Gauss-Newton 또는 Levenberg-Marquardt 방법을 활용하면 점진적 방정식(10.48) 계산 가능

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}$$

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}$$

점진적 방정식 $\longrightarrow \mathbf{H} \Delta \mathbf{x} = \mathbf{g}.$

$$(10.48)$$



\mathbf{F} 와 \mathbf{E} 에 대한 유도는 [7.7.3 절](#) 참고

식 (10.42)에 증가분($\Delta \xi, \Delta P$)을 추가

식 (10.48)에 대한 유도는 [6.2 절](#) 참고

H(Hessian Matrix)는 어떤 모양?

- 예) Gauss-Newton

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} = \begin{bmatrix} \mathbf{F}^T \mathbf{F} & \mathbf{F}^T \mathbf{E} \\ \mathbf{E}^T \mathbf{F} & \mathbf{E}^T \mathbf{E} \end{bmatrix}. \quad (10.50)$$

$$\mathbf{J} = \begin{bmatrix} \overset{\text{F}}{\underbrace{\frac{\partial \mathbf{h}(\xi(\mathbf{R}, \mathbf{t}), \mathbf{P})}{\partial \mathbf{R}}}_{3}} & \underbrace{\frac{\partial \mathbf{h}(\xi(\mathbf{R}, \mathbf{t}), \mathbf{P})}{\partial \mathbf{t}}}_{3} & \overset{\text{E}}{\underbrace{\frac{\partial \mathbf{h}(\xi(\mathbf{R}, \mathbf{t}), \mathbf{P})}{\partial \mathbf{P}}}_{3}} \end{bmatrix} \begin{matrix} \updownarrow \\ 2 \text{ (u,v)} \end{matrix} \quad \text{포즈와 포인트가 한 개씩 일 때}$$

희소성(sparsity) $\longrightarrow \mathbf{J}_{ij}(\mathbf{x}) = \left(\mathbf{0}_{2 \times 6}, \dots, \mathbf{0}_{2 \times 6}, \underbrace{\frac{\partial e_{ij}}{\partial \xi_i}}_{3}, \mathbf{0}_{2 \times 6}, \dots, \mathbf{0}_{2 \times 3}, \dots, \mathbf{0}_{2 \times 3}, \underbrace{\frac{\partial e_{ij}}{\partial p_j}}_{3}, \mathbf{0}_{2 \times 3}, \dots, \mathbf{0}_{2 \times 3} \right). \quad (10.51)$

: 행렬의 값이 대부분 0인 경우

포즈와 포인트가 여러 개씩 일 때

: 3차원 점 P_j 가 로봇 위치 ξ_i 에서 보이는데
i 번째 카메라 포즈와 j 번째 랜드마크에서만 보이고
나머지 변수의 미분은 모두 0이라는 사실을 보여준다.

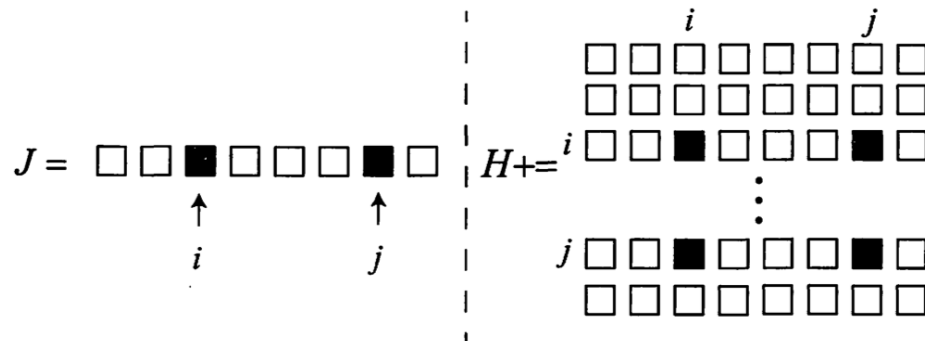
$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix}. \quad (10.53)$$

H(Hessian Matrix)는 어떤 모양?

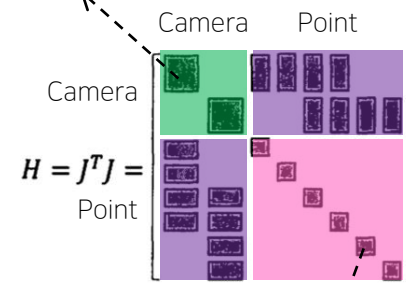
- 예) Gauss-Newton

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}.$$

(10.53)



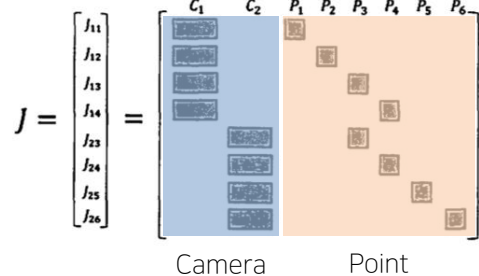
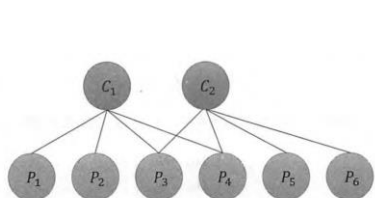
6x6 블록



3x3 블록

H의 희소성은 J로부터 야기

J_{ij} 가 i, j 에서 0이 아닌 블록일 때,
 H 는 (i,i) , (i,j) , (j,i) , (j,j) 에 위치한 블록만 0 이 아닌 값을 갖는다.



H_{11} 은 camera-camera로 구성된 대각 블록 행렬
 H_{22} 은 point-point로 구성된 대각 블록 행렬
 H_{12}/H_{21} 은 observations에 따라 sparse 또는 dense 행렬

H(Hessian Matrix)는 어떤 모양?

- 예) Gauss-Newton

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}. \quad (10.53)$$

A **B**

B^T

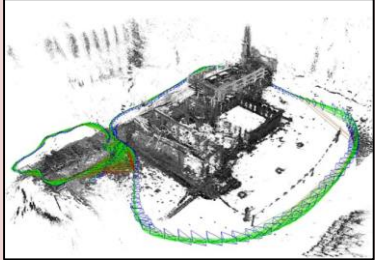
J^TJ =

C

Goal: $\Delta x = \underline{H^{-1}}g$

☹ Main computational bottle neck

Large-scale 최적화 문제를 헤쳐 나갈 전략이 필요하다!



In real world,
일반적으로 랜드마크의 수가 카메라 포즈보다 훨씬 크다!

연산량 줄이기 위해 슈어 보수(Schur Complement)를 활용?

block triangular matrix factorization

- 점진적 방정식에 대응하는 선형시스템:

$$H\Delta x = g. \longrightarrow \begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}. \quad (10.56)$$

- 연산량 줄이기 위한 과정:

- 1) 먼저 Δx_c 계산 (전진 대입; forward substitution)
- 2) 이후 Δx_p 계산 (후진 대입; backward substitution)

연산량 줄이기 위해 슈어 보수(Schur Complement)를 활용?

block triangular matrix factorization

- Δx_c (camera poses 증가분) 계산:

선형시스템에서 E 를 소거하기 위해
양변에 **행렬** 곱 (forward substitution)

$$\begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} I & -EC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}. \quad (10.57)$$

marginalization

$$\begin{bmatrix} B - EC^{-1}E^T & 0 \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} v - EC^{-1}w \\ w \end{bmatrix}. \quad (10.58)$$

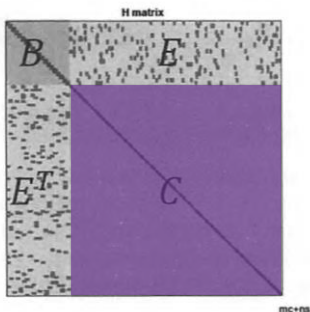
첫 번째 방정식을 풀면
 Δx_p 가 소거된 Δx_c 에 대한 방정식을 유도

$$[B - EC^{-1}E^T] \Delta x_c = v - EC^{-1}w. \quad (10.59)$$

Schur complement of C

(Sparse and symmetric positive definite matrix)

YES! C 행렬은 대각 블록 행렬이며 각 블록은 3x3 행렬
대각 블록 행렬의 역행렬을 계산할 때는 각 블록별로 독립적인 inverse 연산 가능
일반 블록 행렬의 역행렬을 계산하는 것보다 상대적으로 쉽다.



그런데, C^{-1} 을 계산하는 건 쉬운가?

연산량 줄이기 위해 슈어 보수(Schur Complement)를 활용?

block triangular matrix factorization

- Δx_p (3D points 증가분) 계산:

두 번째 방정식을 풀면
 Δx_p 에 대한 방정식을 유도

$$\begin{bmatrix} \mathbf{B} - \mathbf{E}\mathbf{C}^{-1}\mathbf{E}^T & \mathbf{0} \\ \mathbf{E}^T & \mathbf{C} \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} \mathbf{v} - \mathbf{E}\mathbf{C}^{-1}\mathbf{w} \\ \mathbf{w} \end{bmatrix}. \quad (10.58)$$

Already computed

이미 계산한 Δx_c 를 활용해
 Δx_p 를 계산 (backward substitution)

$$\Delta x_p = \mathbf{C}^{-1}(\mathbf{w} - \mathbf{E}^T \Delta x_c)$$

슈어 보수로 얻은 선형 방정식을 어떻게 계산?

- 전통적인 numerical matrix analysis 활용:

$$AX = b \quad [B - EC^{-1}E^T] \Delta x_c = v - EC^{-1}w. \quad (10.59)$$

- LU factorization $\rightarrow A = LU$

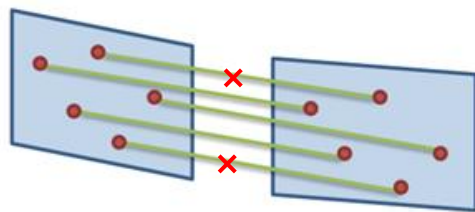
$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{pmatrix} = \begin{pmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & \vdots \\ L_{m1} & L_{m2} & \cdots & L_{mr} \end{pmatrix} \begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_r \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & \cdots & \cdots & U_{1n} \\ & U_{22} & \cdots & \cdots & U_{2n} \\ & & \ddots & & \vdots \\ & & & \ddots & \vdots \\ & & & & U_{rn} \end{pmatrix}$$

- $Cholesky$ factorization $\rightarrow A = LL^T$ (if A is symmetric positive definite, we can further set $D = 1$)
 - Standard decomposition method for almost all unconstrained optimizations
(including bundle adjustment)

효율적 최적화를 위한 Robust Cost Function 설계?

- Cost function:

- 문제) 주어진 데이터가 잘못된 매칭 정보(outlier)일 경우, e_{ij} 값이 지나치게 커질 수 있다!

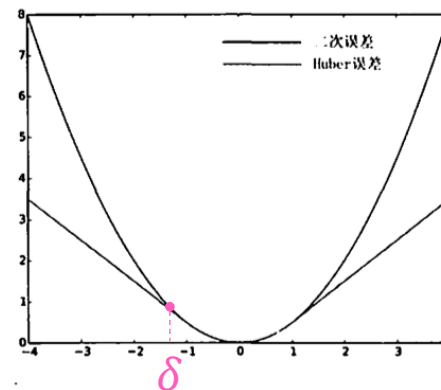


$$\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|e_{ij}\|^2 = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n \|z_{ij} - h(\xi_i, p_j)\|^2.$$

- Robust cost function:

- 해결) Robust kernel(Huber, Cauchy, ...)을 활용하여 error가 큰 데이터의 영향력을 줄일 수 있다!

Huber Kernel $\longrightarrow H(e) = \begin{cases} \frac{1}{2}e^2 & \text{当 } |e| \leq \delta, \\ \delta(|e| - \frac{1}{2}\delta) & \text{其他} \end{cases} \quad (10.61)$



Q & A

Thank you