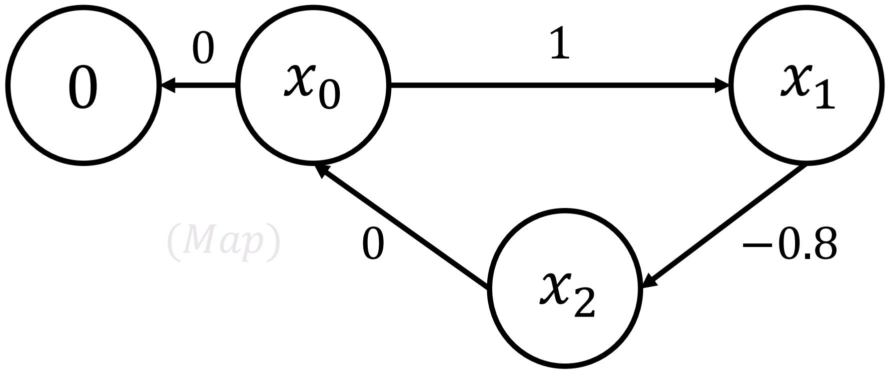
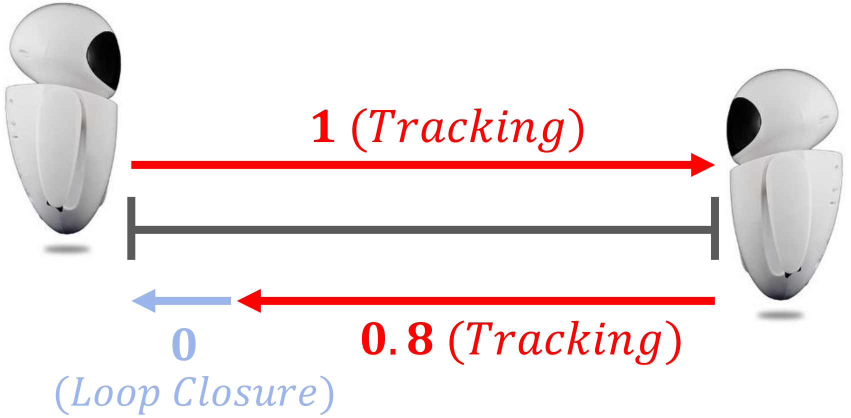
Robotic Navigation and Exploration

Week 6: SLAM Front-end

Min-Chun Hu anitahu@cs.nthu.edu.tw
CS, NTHU

Graph Optimization: 1D Example





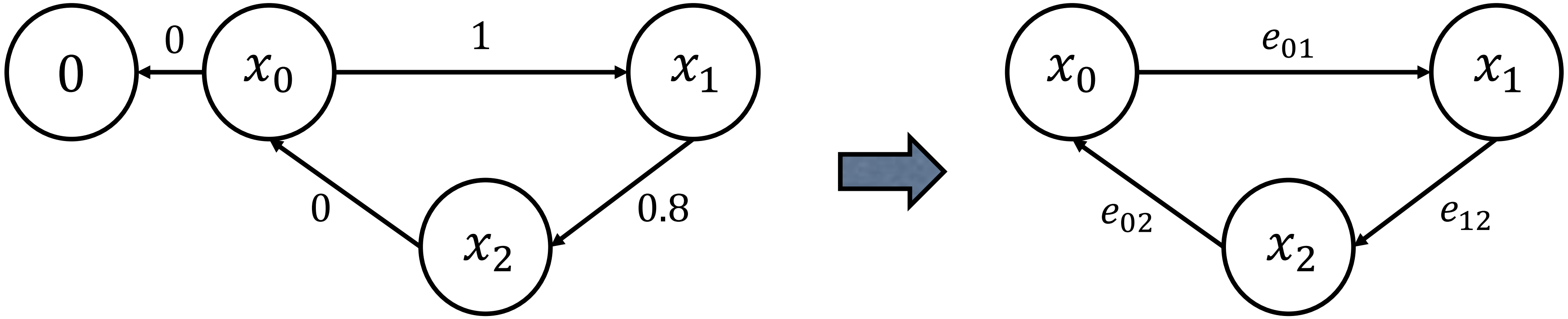
Error function

$$
\begin{aligned}
x_0 &= 0 \\
x_1 &= x_0 + 1 \\
x_2 &= x_1 - 0.8 \\
x_0 &= x_2 + 0
\end{aligned}
\qquad
\begin{aligned}
f_1 &= x_0 \\
f_2 &= x_1 - x_0 - 1 \\
f_3 &= x_2 - x_1 + 0.8 \\
f_4 &= x_0 - x_2
\end{aligned}
$$

$$\min_x \sum_i w_i f_i^2 = w_1 x_0^2 + w_2(x_1 - x_0 - 1)^2 + w_3(x_2 - x_1 + 0.8)^2 + w_4(x_0 - x_2)^2$$

(Optimization)

Graph Optimization: 1D Example



Error Function

$$e_{01} = x_1 - x_0 - 1$$
$$e_{12} = x_2 - x_1 - 0.8$$
$$e_{02} = x_0 - x_2$$

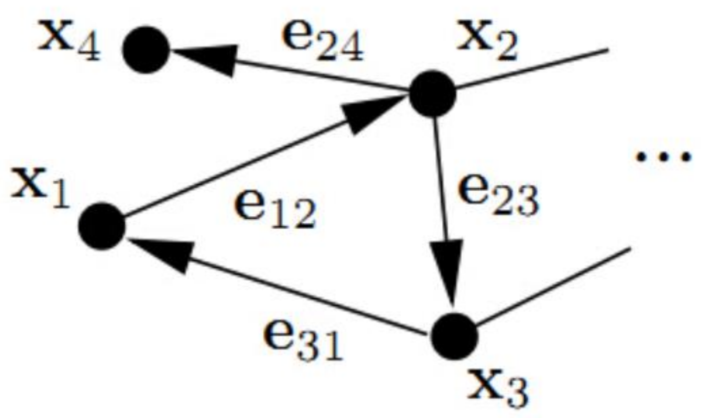
$$\min_x \sum_{i,j} w_{ij} e_{ij}^2 = w_{01}(x_1 - x_0 - 1)^2 + w_{12}(x_2 - x_1 + 0.8)^2 + w_{02}(x_0 - x_2)^2$$

Graph Optimization: General Form

$$\min_x \sum_{i,j} w_{ij} e_{ij}^2 = w_{01}(x_1 - x_0 - 1)^2 + w_{12}(x_2 - x_1 + 0.8)^2 + w_{02}(x_0 - x_2)^2$$

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in \mathcal{C}} \underbrace{\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})^\top \boldsymbol{\Omega}_{ij} \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})}_{\mathbf{F}_{ij}} \quad (1)$$

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbf{F}(\mathbf{x}). \quad (2)$$



$$\begin{aligned} \mathbf{F}(\mathbf{x}) = {} & \mathbf{e}_{12}^\top \boldsymbol{\Omega}_{12} \mathbf{e}_{12} \\ & + \mathbf{e}_{23}^\top \boldsymbol{\Omega}_{23} \mathbf{e}_{23} \\ & + \mathbf{e}_{31}^\top \boldsymbol{\Omega}_{31} \mathbf{e}_{31} \\ & + \mathbf{e}_{24}^\top \boldsymbol{\Omega}_{24} \mathbf{e}_{24} \\ & + \ldots \end{aligned}$$

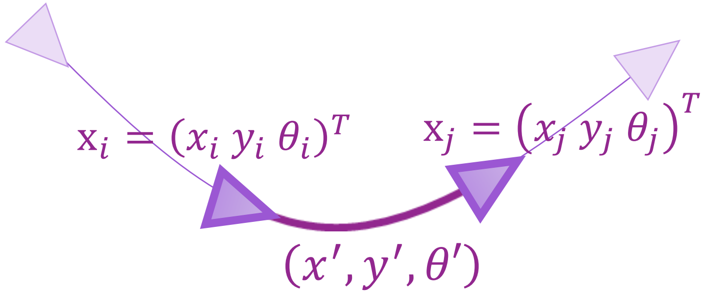
Graph Optimization for 2D Pose

- Consider the relation between two poses:

$$\begin{bmatrix} x_j \\ y_j \\ \theta_j \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} + \begin{bmatrix} R_i * \begin{bmatrix} x' \\ y' \end{bmatrix} \\ \theta' \end{bmatrix}, \text{ in which } R_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}$$

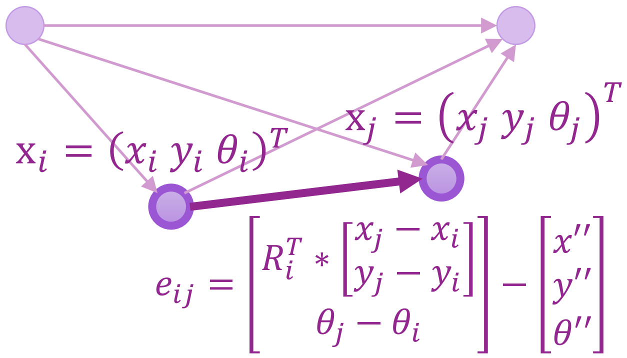
And get

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} R_i^T * \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ \theta_j - \theta_i \end{bmatrix}$$



- After measuring the transform (x'', y'', θ'') between two nodes, we can write down the error term:

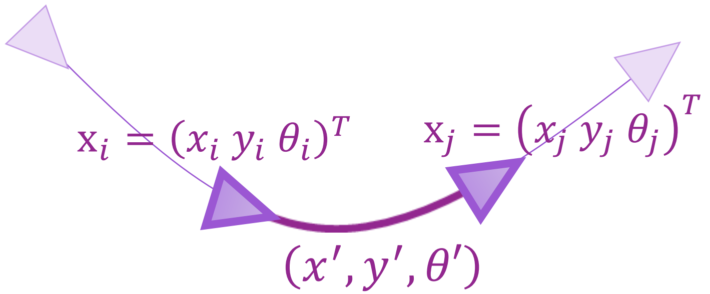
$$e_{ij} = \begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} - \begin{bmatrix} x'' \\ y'' \\ \theta'' \end{bmatrix} = \begin{bmatrix} R_i^T * \begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ \theta_j - \theta_i \end{bmatrix} - \begin{bmatrix} x'' \\ y'' \\ \theta'' \end{bmatrix}$$



Graph Optimization for 2D Pose

- The goal is to find the optimal poses

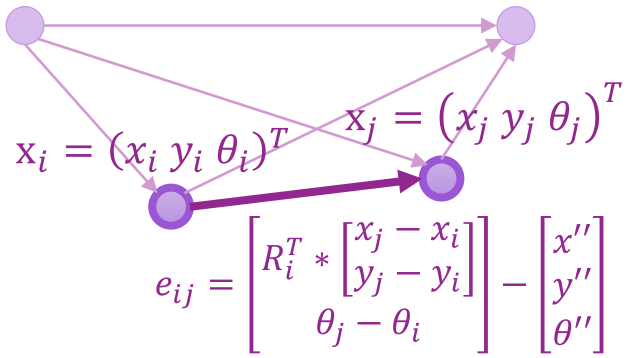
$$F = \sum_{i,j} e_{ij}^{\mathrm{T}} \Omega e_{ij} \qquad \begin{aligned} \mathrm{x} &= (x, y, \theta)^{\mathrm{T}} \\ \mathrm{x}^* &= \operatorname*{argmax}_{\mathrm{x}} F(\mathrm{x}) \end{aligned}$$



- Approximate the object function by 1st order Taylor:

$$F \approx \sum_{i,j} e_{ij}(\mathrm{x}_i + \Delta\mathrm{x}_i, \mathrm{x}_j + \Delta\mathrm{x}_j)^T \Omega e_{ij}(\mathrm{x}_i + \Delta\mathrm{x}_i, \mathrm{x}_j + \Delta\mathrm{x}_j)$$

$$= \sum_{i,j} \left(e_{ij}(x_i, x_j) + A_{ij}\Delta\mathrm{x}_i + B_{ij}\Delta\mathrm{x}_j\right)^T \Omega \left(e_{ij}(x_i, x_j) + A_{ij}\Delta\mathrm{x}_i + B_{ij}\Delta\mathrm{x}_j\right) = \bar{\mathrm{F}}$$



, in which

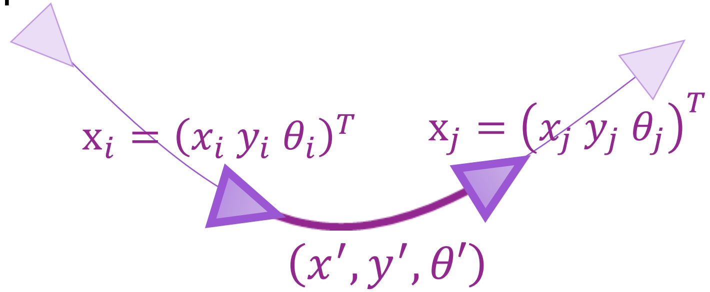
$$A_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}_i} = \begin{bmatrix} -R_i^T & \frac{\partial R_i^T}{\partial \theta_i}\begin{bmatrix} x_j - x_i \\ y_j - y_i \end{bmatrix} \\ 0 & -1 \end{bmatrix}_{3\times 3}, B_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}_j} = \begin{bmatrix} R_i^T & 0 \\ 0 & -1 \end{bmatrix}_{3\times 3}$$

Graph Optimization for 2D Pose

- Apply Gauss-Newton method, we solve the 1st order approximation of object function:

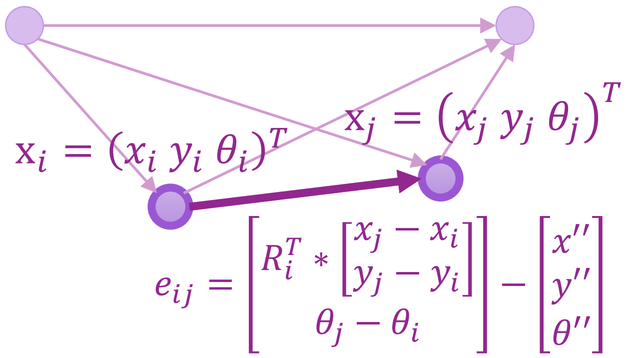
$$\frac{\partial \bar{F}}{\partial \Delta x_i} = A_{ij}^T \Omega A_{ij} \Delta x_i + A_{ij}^T \Omega B_{ij} \Delta x_j + A_{ij}^T \Omega e_{ij} = 0,$$

$$\frac{\partial \bar{F}}{\partial \Delta x_j} = B_{ij}^T \Omega A_{ij} \Delta x_i + B_{ij}^T \Omega B_{ij} \Delta x_j + B_{ij}^T \Omega e_{ij} = 0$$



- Transform the equation into matrix form:

$$\begin{bmatrix} A_{ij}^T \Omega A_{ij} & A_{ij}^T \Omega B_{ij} \\ B_{ij}^T \Omega A_{ij} & B_{ij}^T \Omega B_{ij} \end{bmatrix} * \begin{bmatrix} \Delta x_i \\ \Delta x_j \end{bmatrix} = \begin{bmatrix} -A_{ij}^T \Omega e_{ij} \\ -B_{ij}^T \Omega e_{ij} \end{bmatrix}$$



Solve the linear system by Cholesky Factorization

$$H\Delta x = -b \qquad\qquad (H + \lambda I)\Delta x = -b$$

$\mathbf{H} \approx \mathbf{J}^T\mathbf{J}$ (Gauss-Newton) (Levenberg-Marquardt)

Complete Algorithm

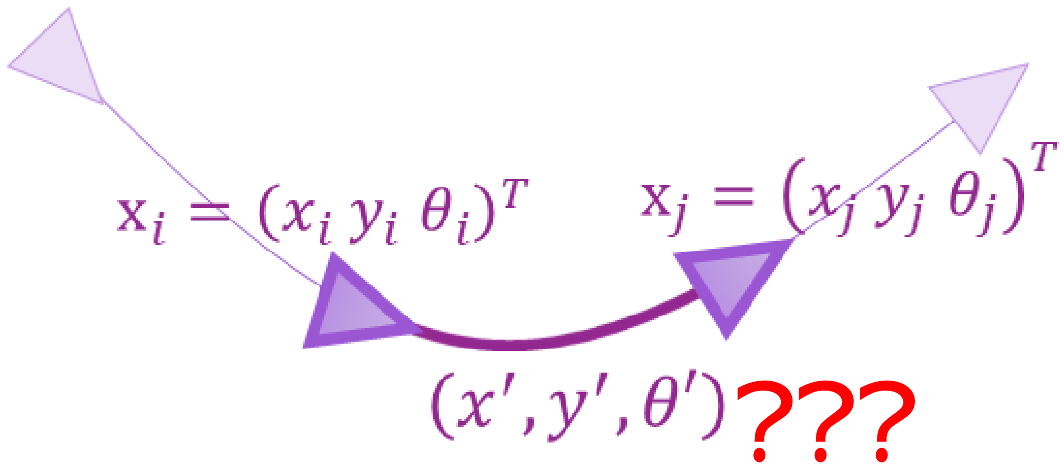
$$\mathbf{J}_{ij} = \left(\mathbf{0} \cdots \mathbf{0} \; \underbrace{\mathbf{A}_{ij}}_{\text{node } i} \; \mathbf{0} \cdots \mathbf{0} \; \underbrace{\mathbf{B}_{ij}}_{\text{node } j} \; \mathbf{0} \cdots \mathbf{0} \right).$$

$$\mathbf{H}_{ij} = \begin{pmatrix} \ddots & & & & \\ & \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} & \cdots & \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} & \cdots & \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \\ & & & & \ddots \end{pmatrix}$$

$$\mathbf{b}_{ij} = \begin{pmatrix} \vdots \\ \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \\ \vdots \\ \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \\ \vdots \end{pmatrix}$$

Require: $\breve{\mathbf{x}} = \breve{\mathbf{x}}_{1:T}$: initial guess. $\mathcal{C} = \{\langle \mathbf{e}_{ij}(\cdot), \boldsymbol{\Omega}_{ij} \rangle\}$: constraints
Ensure: $\mathbf{x}^*$: new solution, $\mathbf{H}^*$ new information matrix
// find the maximum likelihood solution
while $\neg$converged **do**
  $\mathbf{b} \leftarrow \mathbf{0}$ $\mathbf{H} \leftarrow \mathbf{0}$
  for all $\langle \mathbf{e}_{ij}, \boldsymbol{\Omega}_{ij} \rangle \in \mathcal{C}$ **do**
    // Compute the Jacobians $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ of the error function
    $\mathbf{A}_{ij} \leftarrow \left. \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i} \right|_{\mathbf{x}=\breve{\mathbf{x}}}$ $\mathbf{B}_{ij} \leftarrow \left. \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j} \right|_{\mathbf{x}=\breve{\mathbf{x}}}$
    // compute the contribution of this constraint to the linear system
    $\mathbf{H}_{[ii]} \mathrel{+}= \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij}$ $\mathbf{H}_{[ij]} \mathrel{+}= \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij}$
    $\mathbf{H}_{[ji]} \mathrel{+}= \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij}$ $\mathbf{H}_{[jj]} \mathrel{+}= \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij}$
    // compute the coefficient vector
    $\mathbf{b}_{[i]} \mathrel{+}= \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$ $\mathbf{b}_{[j]} \mathrel{+}= \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$
  end for
  // keep the first node fixed
  $\mathbf{H}_{[11]} \mathrel{+}= \mathbf{I}$
  // solve the linear system using sparse Cholesky factorization
  $\boldsymbol{\Delta}\mathbf{x} \leftarrow \text{solve}(\mathbf{H} \, \boldsymbol{\Delta}\mathbf{x} = -\mathbf{b})$
  // update the parameters
  $\breve{\mathbf{x}} \mathrel{+}= \boldsymbol{\Delta}\mathbf{x}$
end while
$\mathbf{x}^* \leftarrow \breve{\mathbf{x}}$
$\mathbf{H}^* \leftarrow \mathbf{H}$
// release the first node
$\mathbf{H}^*_{[11]} \mathrel{-}= \mathbf{I}$
return $\langle \mathbf{x}^*, \mathbf{H}^* \rangle$

How to get the transformation ?



Scan-to-Scan Registration

- Given two matching points sets p_i and q_i, we aims to minimize the least square of registration error:

$$J = \frac{1}{2}\sum_{i=1}^{n}\|q_i - Rp_i - t\|^2$$

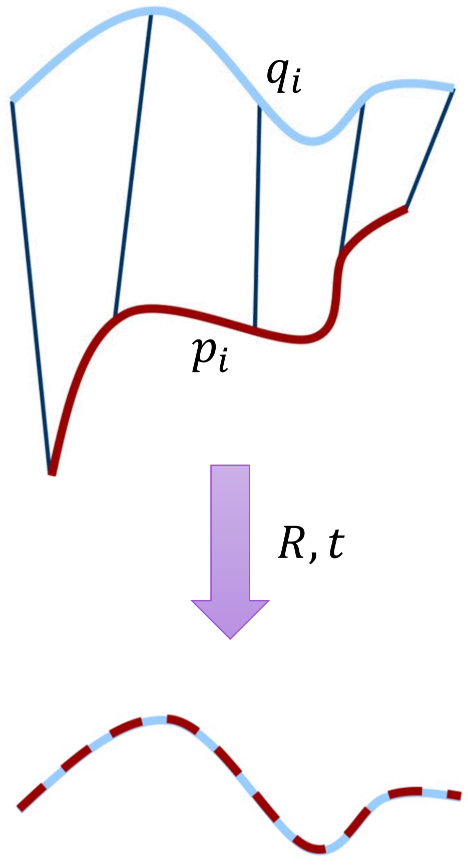
- Define the mean of points sets μ_p and μ_q, we can get

$$\frac{1}{2}\sum_{i=1}^{n}\|q_i - Rp_i - t\|^2 = \frac{1}{2}\sum_{i=1}^{n}\left\|q_i - Rp_i - t - (\mu_q - R\mu_p) + (\mu_q - R\mu_p)\right\|^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i - \mu_q - R(p_i - \mu_p)\right) + (\mu_q - R\mu_p - t)\right\|^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i - \mu_q - R(p_i - \mu_p)\right)\right\|^2 + \|\mu_q - R\mu_p - t\|^2 \cancel{+ 2\left(q_i - \mu_q - R(p_i - \mu_p)\right)^T(\mu_q - R\mu_p - t)}$$

$$\boxed{\begin{aligned}&\sum_{i=1}^{n}\left(q_i - \mu_q - R(p_i - \mu_p)\right)^T(\mu_q - R\mu_p - t) = (\mu_q - R\mu_p - t)^T\sum_{i=1}^{n}\left(q_i - \mu_q - R(p_i - \mu_p)\right)\\ &= (\mu_q - R\mu_p - t)^T\left(n\mu_q - n\mu_q - R(n\mu_p - n\mu_p)\right) = 0\end{aligned}}$$

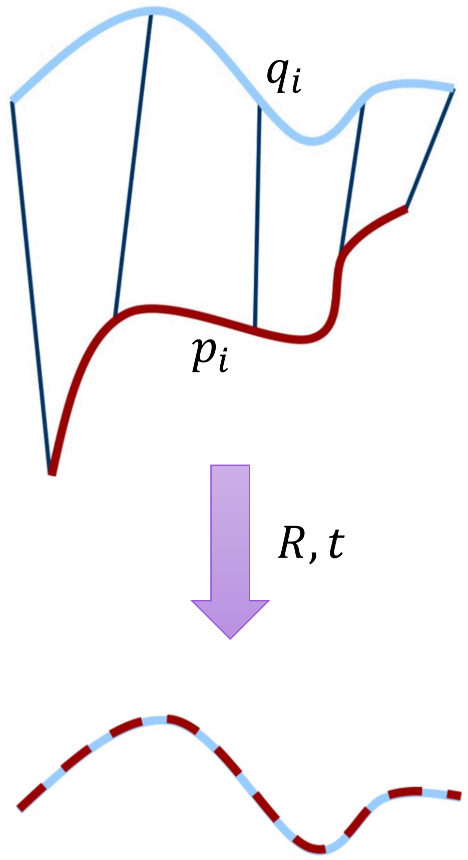


Scan-to-Scan Registration

- Define the relative location p_i' and q_i', the objective function becomes:

$$\frac{1}{2}\sum_{i=1}^{n}\left\|\left(q_i-\mu_q-R\left(p_i-\mu_p\right)\right)\right\|^2+\left\|\mu_q-R\mu_p-t\right\|^2$$
$$=\frac{1}{2}\sum_{i=1}^{n}\|(q_i'-Rp_i')\|^2+\left\|\mu_q-R\mu_p-t\right\|^2$$

$$\boxed{\begin{aligned} p_i' &= p_i-\mu_p, \\ q_i' &= q_i-\mu_q \end{aligned}}$$

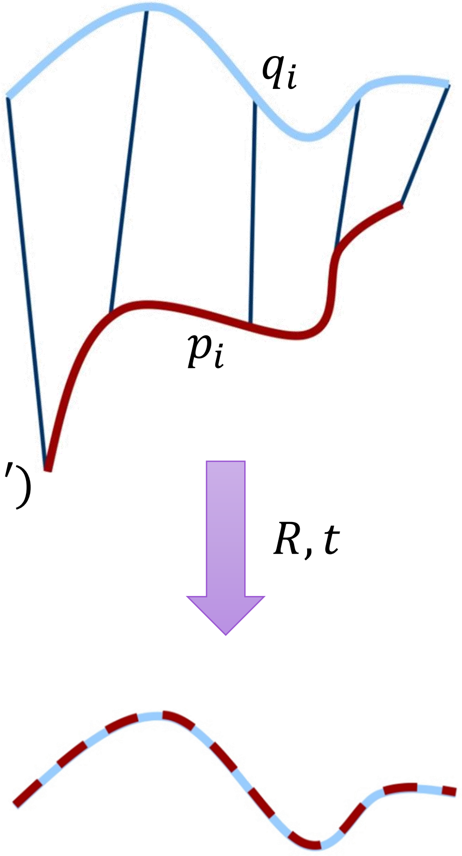


- Divide the optimization process into two steps:

1. Rotation $R^* = \underset{\mathrm{R}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\|(q_i'-Rp_i')\|^2$

2. Translation $t^* = \mu_q - R^*\mu_p$

Scan-to-Scan Registration



- Solve the rotation term:

$$R^* = \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \|(q_i' - Rp_i')\|^2 = \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (q_i'^T q_i' + p_i'^T R^T R p_i' - 2 q_i'^T R p_i')$$

$$= \underset{R}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} (q_i'^T q_i' + p_i'^T p_i' - 2 q_i'^T R p_i') = \underset{R}{\operatorname{argmin}} \sum_{i=1}^{n} -q_i'^T R p_i'$$

- Minimizing the function is equivalent to maximizing

$$F = \sum_{i=1}^{n} q_i'^T R p_i' = Trace\left(\sum_{i=1}^{n} R q_i'^T p_i'\right) = Trace(RH)$$

, where $H = \sum_{i=1}^{n} q_i'^T p_i'$

Scan-to-Scan Registration

- we can solve the rotation by the SVD decomposition of H :

$$\underset{R}{\operatorname{argmax}}\, Trace(RH) \Rightarrow H = U\Lambda V^T \Rightarrow R^* = VU^T$$

- Proof:

Lemma:
For any positive definite matrix AA^T
, and any orthonormal matrix B,

$$Trace(AA^T) \geq Trace(BAA^T)$$

Proof of Lemma:
Let a_i be the $\boldsymbol{ith}$ column of A. Then

$$Trace(BAA^T) = Trace(A^T BA) = \sum_i a_i^T (Ba_i)$$

The Cauchy-Schwarz Inequality:

$$a_i^T (Ba_i) \leq \sqrt{(a_i^T a_i)(a_i^T B^T B a_i)} = a_i^T a_i$$

Hence, $Trace(BAA^T) \leq \sum_i a_i^T a_i = Trace(AA^T)$

SVD decomposition of H :

$$H = U\Lambda V^T$$

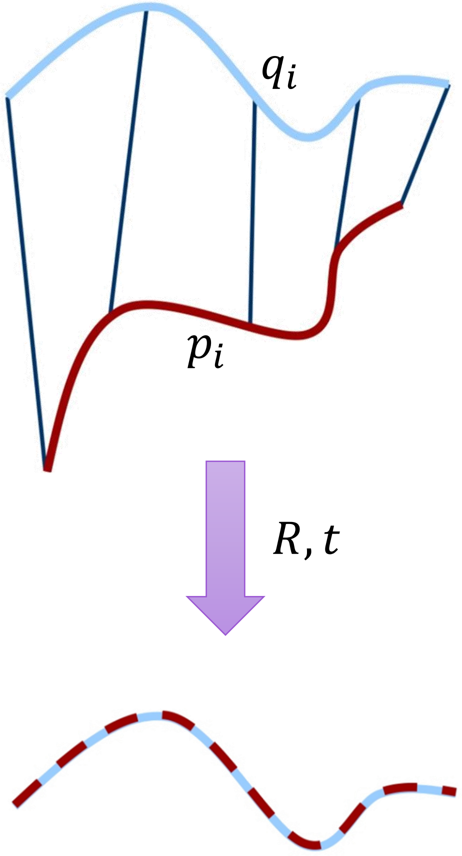
Set $X = VU^T$, and we have

$$XH = VU^T U\Lambda V^T = V\Lambda V^T$$ (positive definite)

From the Lemma, for ant orthonormal matrix B

$$Trace(XH) \geq Trace(\boldsymbol{B}XH)$$

Any other rotation



Theorem C.1 **(Cauchy–Schwarz)** *Let V be a linear space with inner product $\langle ., . \rangle$, then for each $\mathbf{a}, \mathbf{b} \in V$ we have:*

$$|\langle \mathbf{a}, \mathbf{b} \rangle|^2 \leq ||\mathbf{a}|| \cdot ||\mathbf{b}||.$$

Proof If $\langle \mathbf{a}, \mathbf{b} \rangle = 0$ then the result is self evident. We therefore assume that $\langle \mathbf{a}, \mathbf{b} \rangle = \alpha \neq 0$, α may of course be complex. We start with the inequality

$$||\mathbf{a} - \lambda\alpha\mathbf{b}||^2 \geq 0$$

where λ is a real number. Now,

$$||\mathbf{a} - \lambda\alpha\mathbf{b}||^2 = \langle \mathbf{a} - \lambda\alpha\mathbf{b}, \mathbf{a} - \lambda\alpha\mathbf{b} \rangle.$$

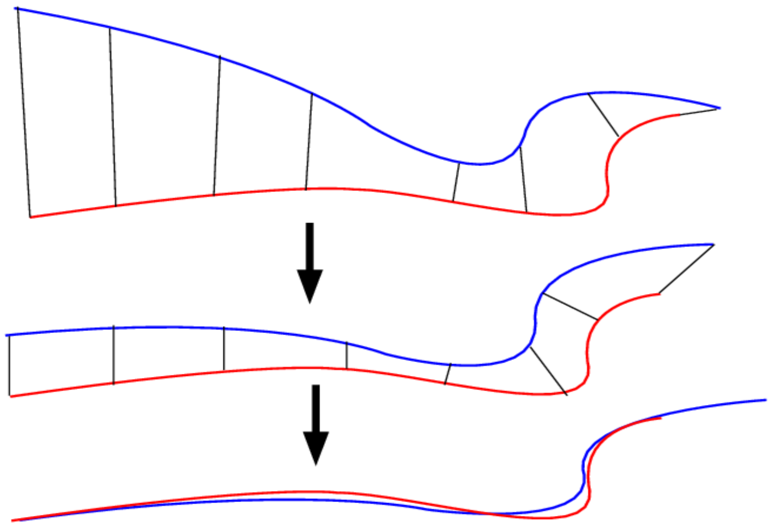
We use the properties of the inner product to expand the right hand side as follows:-

$$\begin{aligned}
\langle \mathbf{a} - \lambda\alpha\mathbf{b}, \mathbf{a} - \lambda\alpha\mathbf{b} \rangle = \langle \mathbf{a}, \mathbf{a} \rangle - \lambda\langle \alpha\mathbf{b}, \mathbf{a} \rangle - \lambda\langle \mathbf{a}, \alpha\mathbf{b} \rangle + \lambda^2|\alpha|^2\langle \mathbf{b}, \mathbf{b} \rangle \geq 0 \\
\text{so } ||\mathbf{a}||^2 - \lambda\alpha\langle \mathbf{b}, \mathbf{a} \rangle - \lambda\bar{\alpha}\langle \mathbf{a}, \mathbf{b} \rangle + \lambda^2|\alpha|^2||\mathbf{b}||^2 \geq 0 \\
\text{i.e. } ||\mathbf{a}||^2 - \lambda\alpha\bar{\alpha} - \lambda\bar{\alpha}\alpha + \lambda^2|\alpha|^2||\mathbf{b}||^2 \geq 0 \\
\text{so } ||\mathbf{a}||^2 - 2\lambda|\alpha|^2 + \lambda^2|\alpha|^2||\mathbf{b}||^2 \geq 0.
\end{aligned}$$

Scan-to-Scan Registration

- Iterative Closest Points (ICP) Algorithm

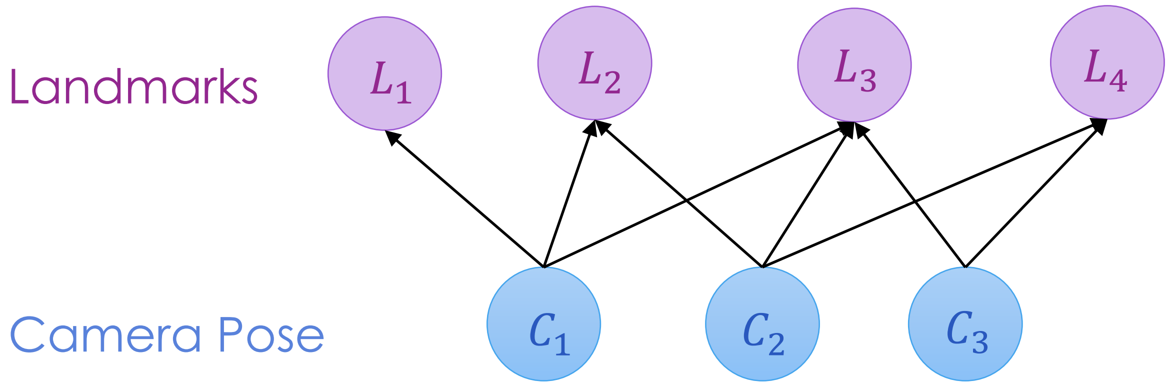
Given two points sets P and Q



Initialize $R_0 = I, t_0 = 0$
Build the kd-tree of Q
Repeat
  Transform the points set $\widehat{p_i} = R_k p_i + t_k$
  Search the nearest points pairs $[q_i, \widehat{p_i}]$
  Compute mean of points sets and the relative location $\widehat{p_i}' = \widehat{p_i}' - \mu_{\widehat{p}}$ =and $q_i' = q_i - \mu_q$
  SVD Decomposition: $H = U\Lambda V^T$, where $H = \sum_{i=1}^{n} q_i'^T \widehat{p_i}'$
  Get the optimize transformation $R^* = VU^T$ and $t^* = \mu_q - R^*\mu_p$
  Update the transformation $R_k = R^* R_{k-1}$ and $t_k = R^* t_{k-1} + t^*$
Until Convergence

Graph Optimization for Map and Pose

- Bundle Adjustment
- The bipartite optimization graph



- Given observation model $z_{ij} = h(C_i, L_j)$, the objective is to minimize the observation error:

$$F = \sum_{ij} \left\| z_{ij}^{obs} - h(C_i, L_j) \right\|^2$$

Sparse Hessian and Marginalization

- The Jacobian matrix of observation error and the approximated Hessian:

$$J_{ij} = \frac{\partial e_{ij}}{\partial \mathrm{x}} = [0, \dots, 0, \underbrace{\frac{\partial e_{ij}}{\partial C_i}, 0, \dots,}_{\text{Camera Pose}} \underbrace{0, 0, \dots, 0, \frac{\partial e_{ij}}{\partial L_j}, 0, \dots, 0}_{\text{Landmarks}}]$$

$$H \cong J^T J = \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ji} & H_{jj} \end{bmatrix}$$ (Arrow-Like Matrix)

- Schur Elimination and Marginalization

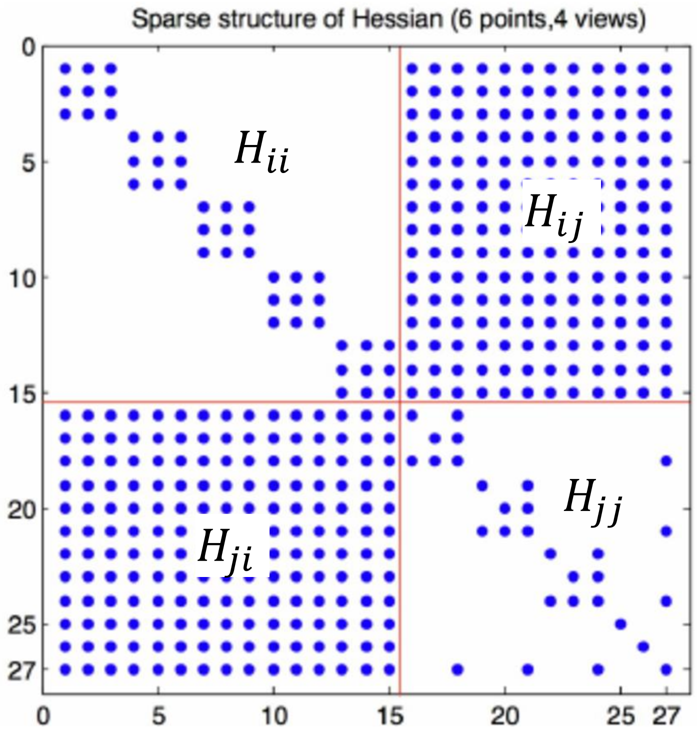
$$H\Delta\mathrm{x} = -b \rightarrow \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}$$

$$\begin{bmatrix} I & -H_{ij}H_{jj}^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} H_{ii} & H_{ij} \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} I & -H_{ij}H_{jj}^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

$$\begin{bmatrix} H_{ii} - H_{ij}H_{jj}^{-1}H_{ij}^T & 0 \\ H_{ij}^T & H_{jj} \end{bmatrix} \begin{bmatrix} \Delta\mathrm{x}_C \\ \Delta\mathrm{x}_L \end{bmatrix} = \begin{bmatrix} v - H_{ij}H_{jj}^{-1}w \\ w \end{bmatrix}$$

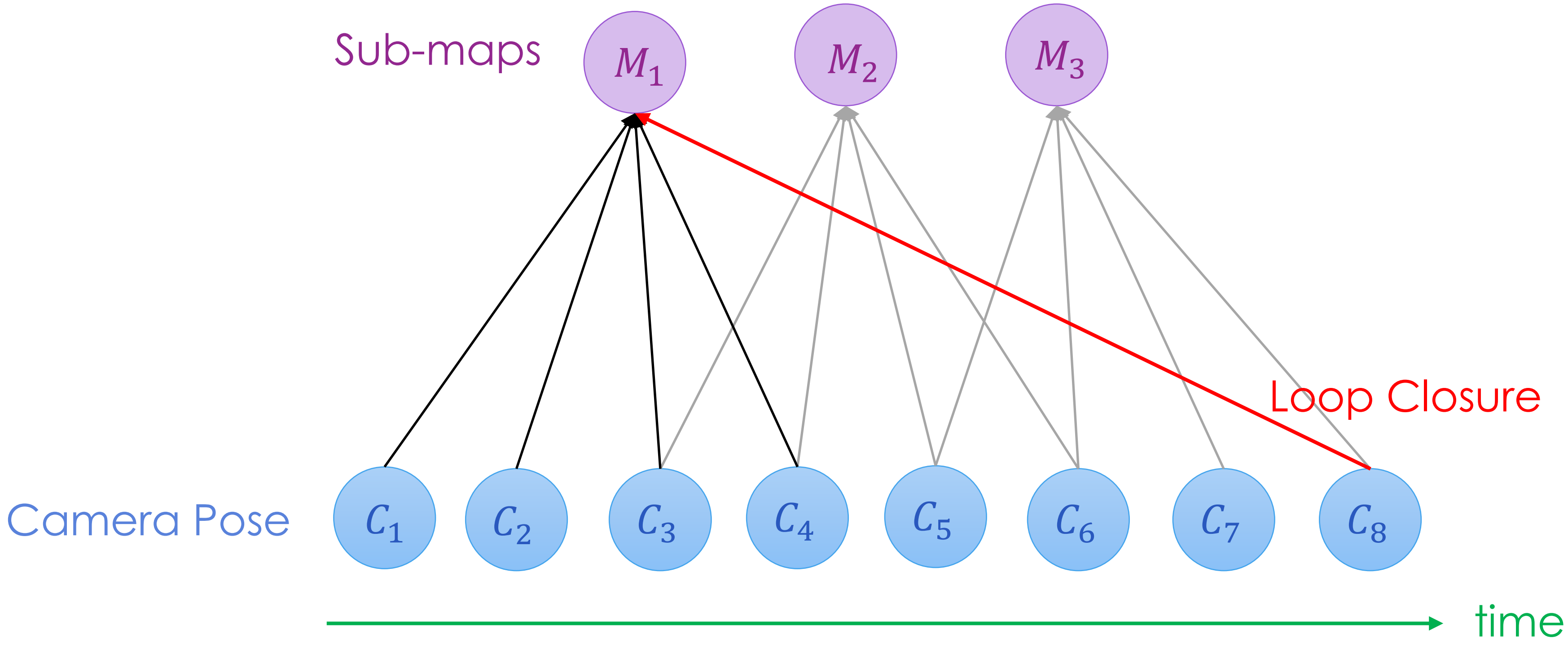
$$[H_{ii} - H_{ij}H_{jj}^{-1}H_{ij}^T]\Delta\mathrm{x}_C = v - H_{ij}H_{jj}^{-1}w$$

Easy to compute !!



Graph Optimization for Grid-based SLAM

- Karto-SLAM (Open-Source) / Cartographer (Google)



Scan-to-Map Matching

- Define the Robot Pose State $\xi = (p_x, p_y, \psi)^T$ and the Optimization Objective:

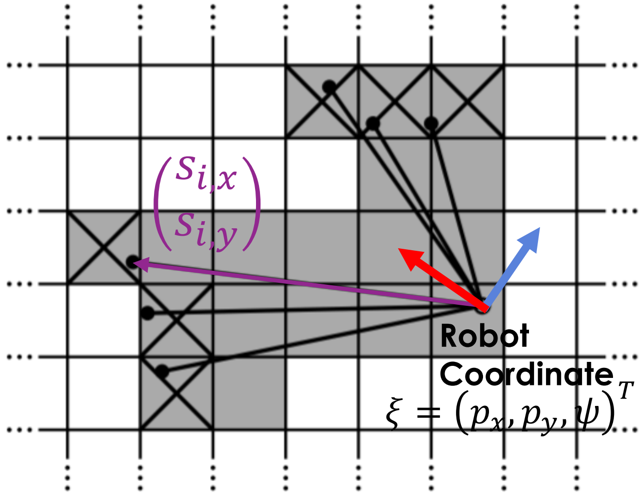
$$\xi^* = \operatorname{argmin}_\xi \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \text{ , where } \quad S_i(\xi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

- Apply the 1st order Taylor approximation

$$\sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \approx \sum_{i=1}^{n} \left[1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right]^2$$

- Partial Derivative to $\Delta\xi$

$$2 \sum_{i=1}^{n} \left[\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M(S_i(\xi)) - \nabla M(s_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right] = 0$$



Scan-to-Map Matching

- Solving the problem by GN methods:

$$2\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[1 - M\big(S_i(\xi)\big) - \nabla M(s_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\Delta\xi\right] = 0$$

$$\underbrace{\left[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[\nabla M\big(S_i(\xi)\big)\frac{\partial S_i(\xi)}{\partial \xi}\right]}_{H}\underbrace{\Delta\xi}_{\Delta x} = \underbrace{\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T[1 - M(S_i(\xi))]}_{-b}$$

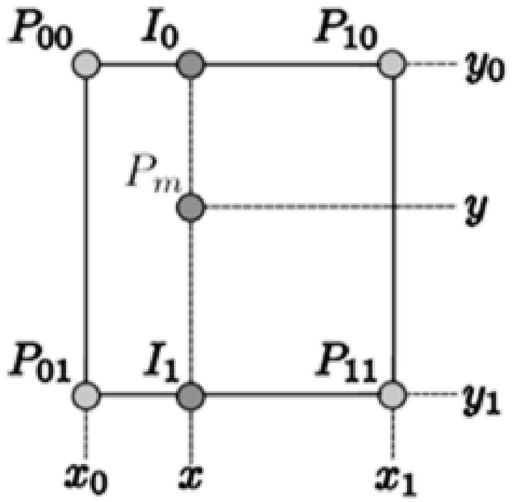
$$\Delta\xi = H^{-1}\sum_{i=1}^{n}\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T[1 - M(S_i(\xi))]$$

$$\boxed{\frac{\partial S_i(\xi)}{\partial \xi} = \begin{pmatrix} 1 & 0 & -\sin(\psi)\, s_{i,x} - cos(\psi)\, s_{i,y} \\ 0 & 1 & \cos(\psi)\, s_{i,x} - \sin(\psi)\, s_{i,y} \end{pmatrix}}$$

, where $H = \left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]^T\left[\nabla M(S_i(\xi))\frac{\partial S_i(\xi)}{\partial \xi}\right]$

Scan-to-Map Matching

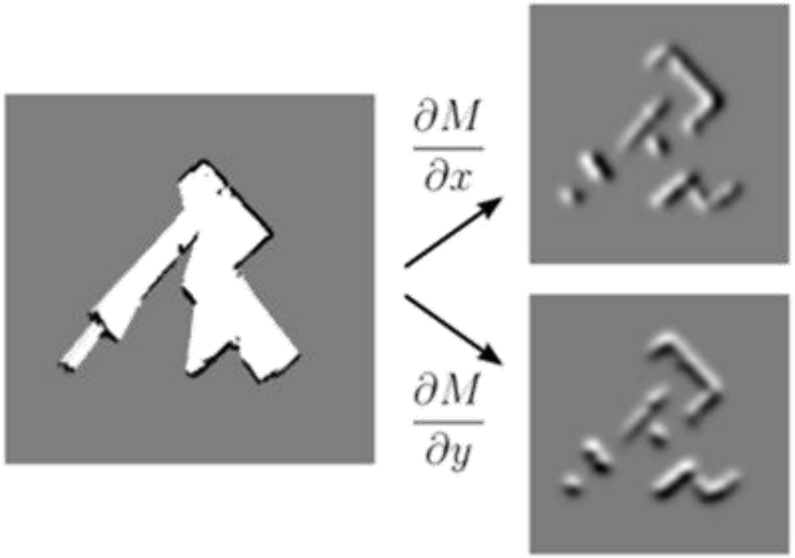
- The derivative of map with respect to location.



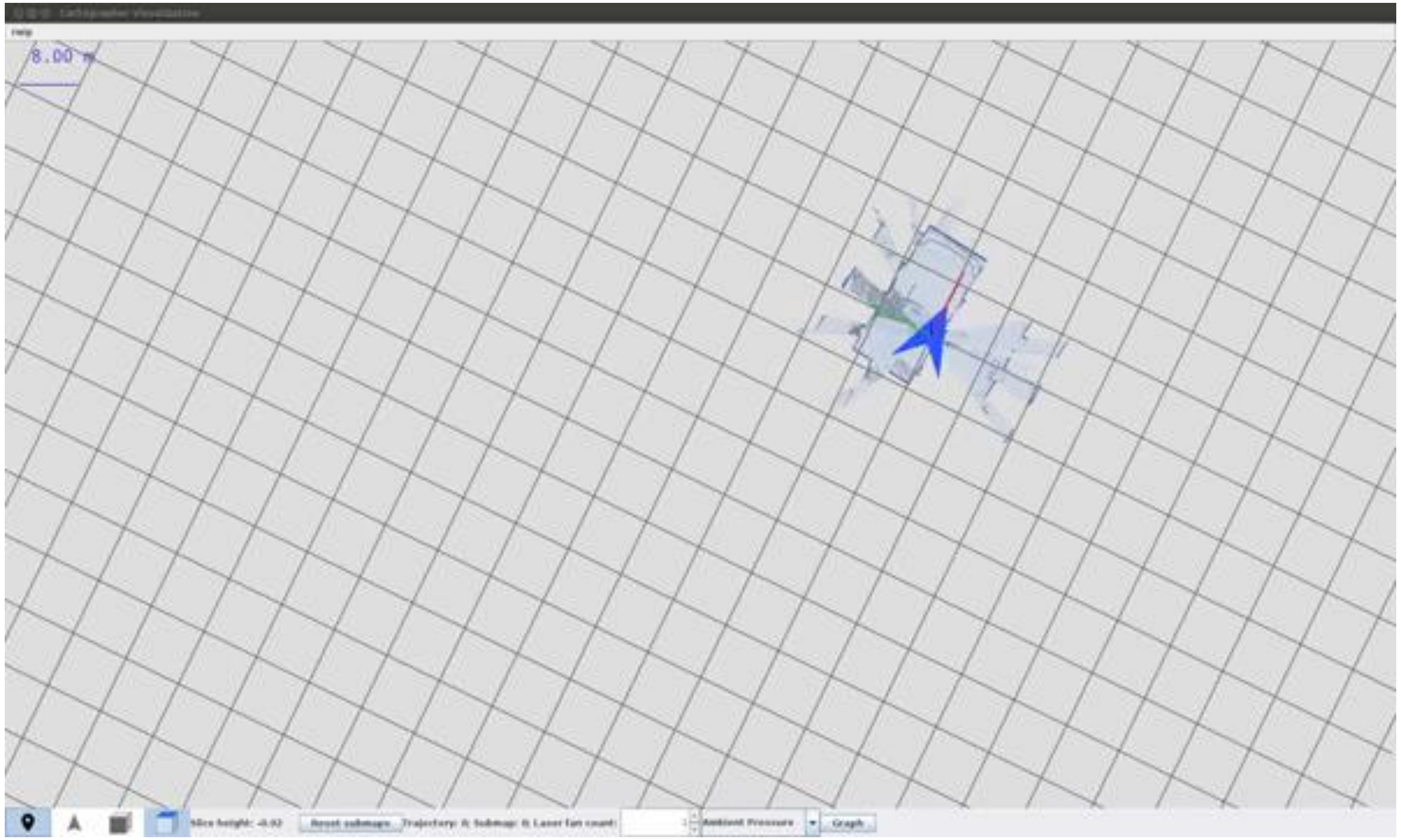
$$M(P_m) \approx \frac{y - y_0}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{11}) + \frac{x_1 - x}{x_1 - x_0}M(P_{01})\right) + \frac{y_1 - y}{y_1 - y_0}\left(\frac{x - x_0}{x_1 - x_0}M(P_{10}) + \frac{x_1 - x}{x_1 - x_0}M(P_{00})\right)$$

$$\frac{\partial M}{\partial x}(P_m) \approx \frac{y - y_0}{y_1 - y_0}(M(P_{11}) - M(P_{01})) + \frac{y_1 - y}{y_1 - y_0}(M(P_{10}) - M(P_{00}))$$

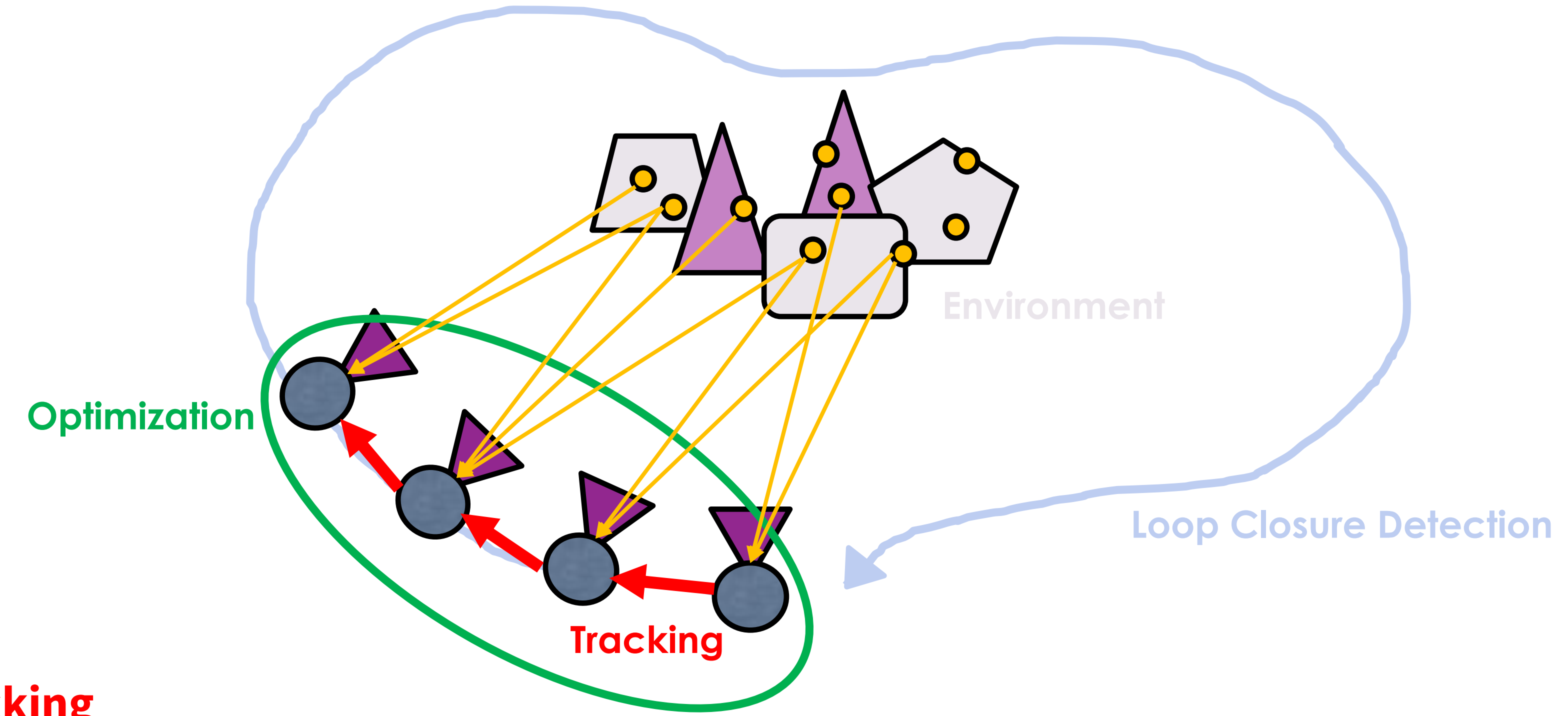
$$\frac{\partial M}{\partial y}(P_m) \approx \frac{x - x_0}{x_1 - x_0}(M(P_{11}) - M(P_{10})) + \frac{x_1 - x}{x_1 - x_0}(M(P_{01}) - M(P_{00}))$$



Cartographer Demo



SLAM Overview



Pose Tracking

Using continuous measurement to estimate the movement

Local Optimization

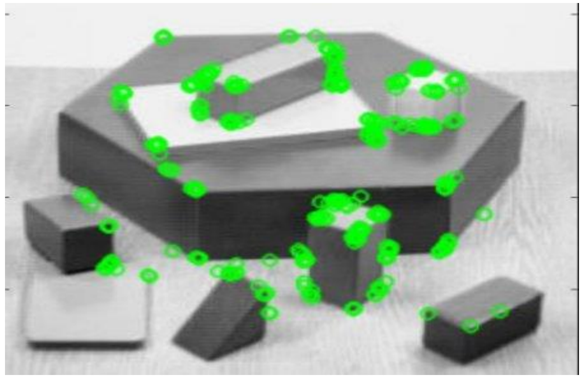
Using several measurement to optimize the error of the map

Loop Closure Detection

Detecting the loop to stabilize the global structure

Information from Image Data

Sparse

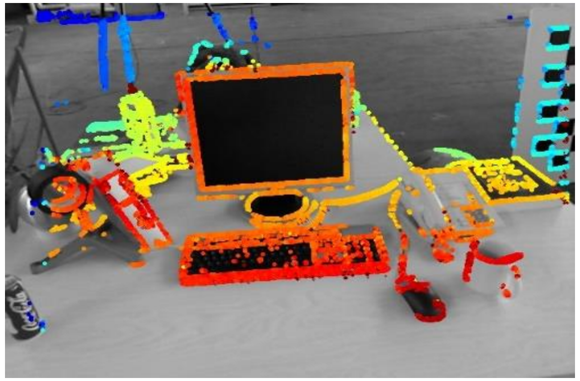


Sparse Feature Points

Dense

All Points

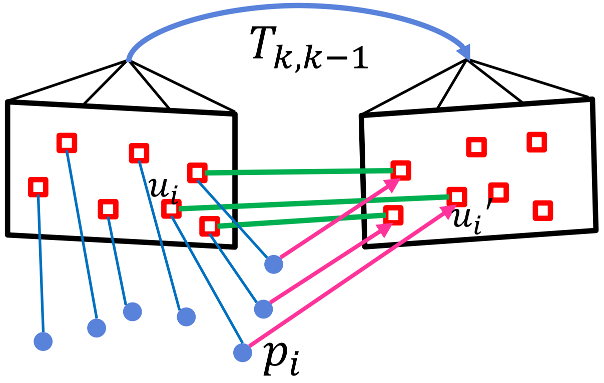
Semi-Dense



Important Points

Objective Function

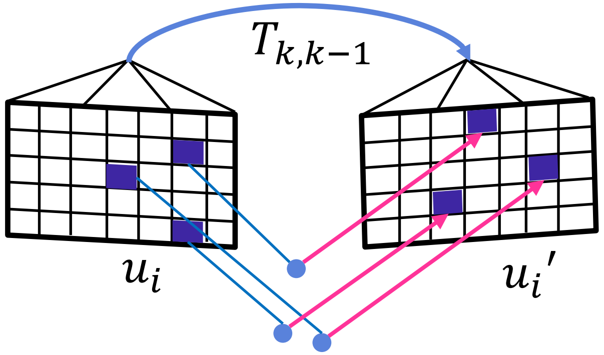
Indirect Method



$$T_{k,k-1} = \underset{T}{argmin} \sum_{i}^{N} ||u_i' - \pi p_i||^2$$

Minimize Geometric Error (Reprojection)

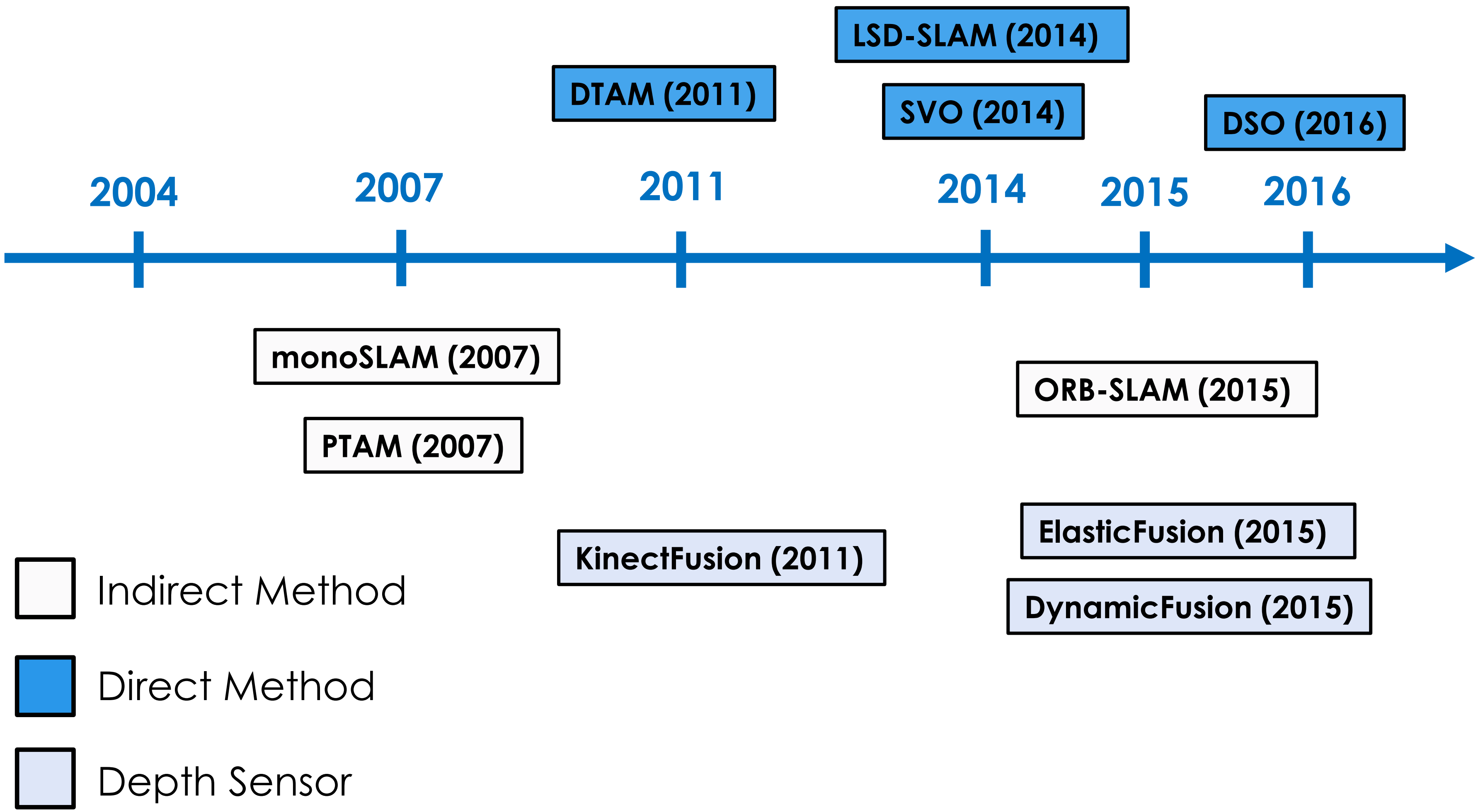
Direct Method



$$T_{k,k-1} = \underset{T}{argmin} \sum_{i}^{N} ||I_k(u_i') - I_{k-1}(u_i)||^2$$

Minimize Photometric Error (Pixel Grayscale)

History of Visual SLAM



History of Visual SLAM

2004 — 2007 — 2011 — 2014 — 2015 — 2016

PTAM (2007)

First real-time monocular SLAM algorithm.
Separate the system into two thread: tracking and mapping. The pipeline is the basis of modern SLAM system.

DTAM (2011)

First dense monocular SLAM algorithm.
Using GPU to accelerate the computation and build dense point cloud.

KinectFusion (2011)

First depth SLAM algorithm.
Using the volumetric fusion map to construct complete and beautiful dense 3D point cloud.

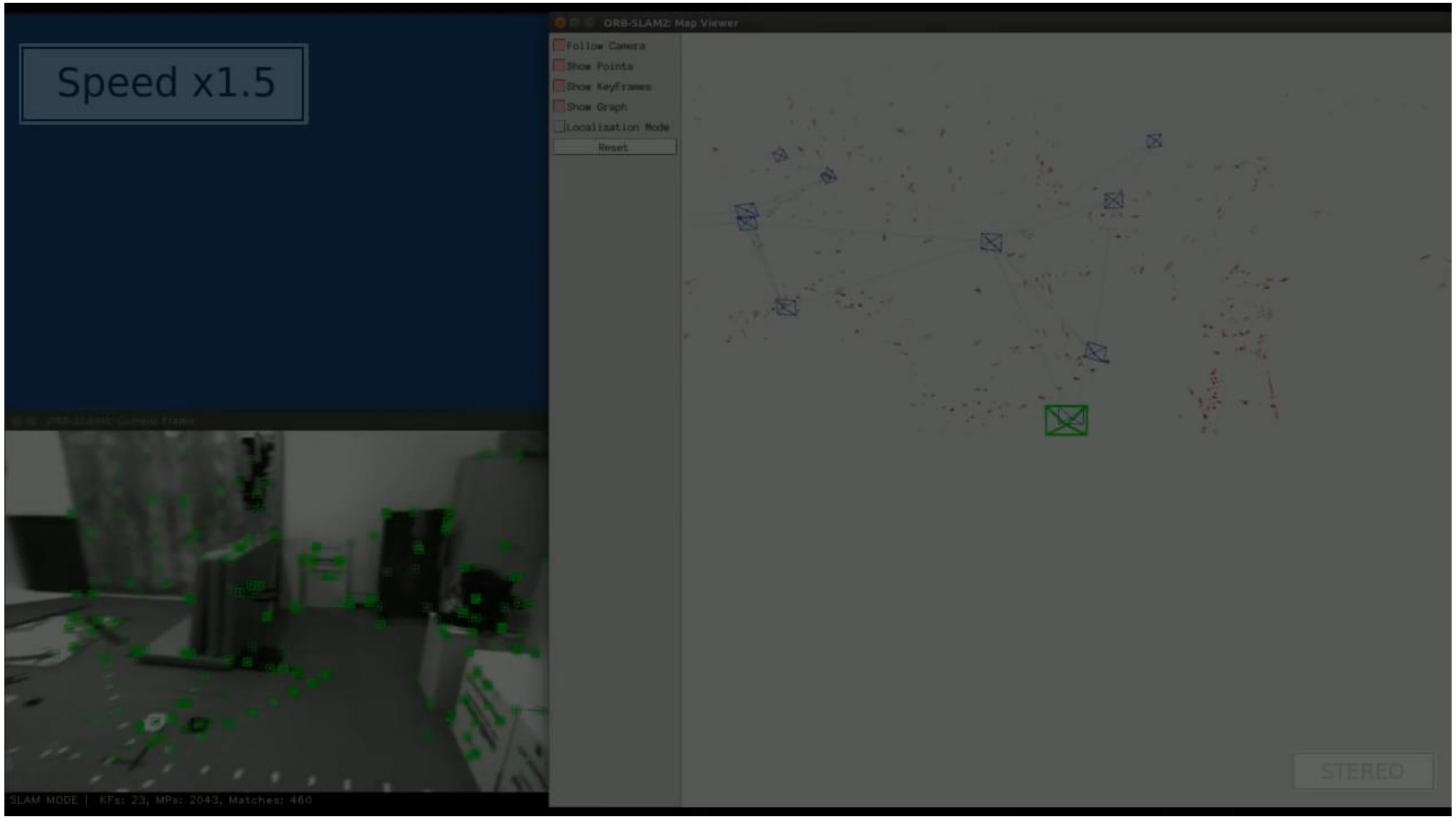
LSD-SLAM (2014)

Improve the speed of DTAM by only building the **semi-dense map** of whole image.

ORB-SLAM (2015)

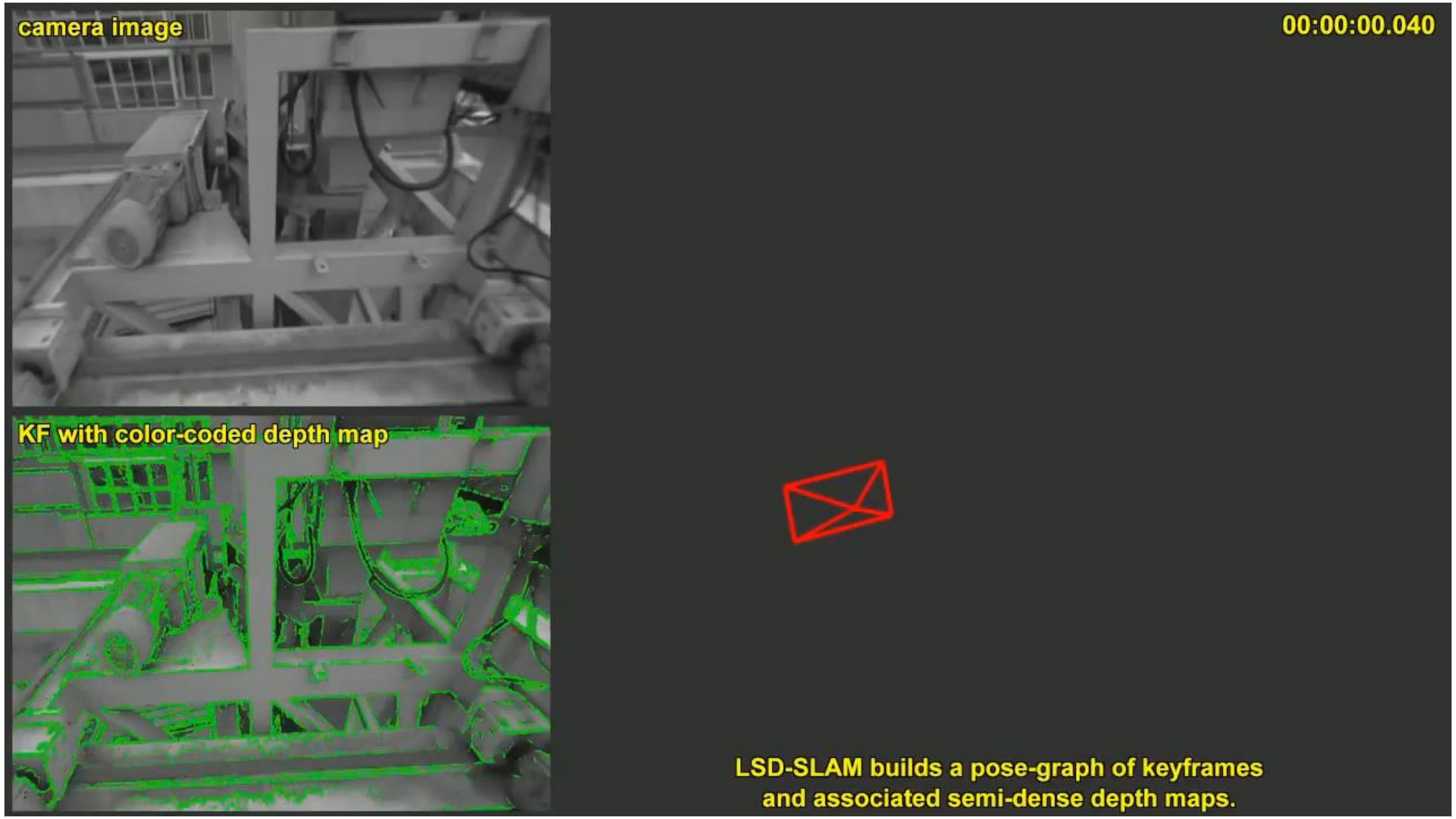
Assembles recent researches of **feature-based SLAM**. Use similar pipeline as PTAM. A stable and reliable monocular SLAM system.

ORB-SLAM



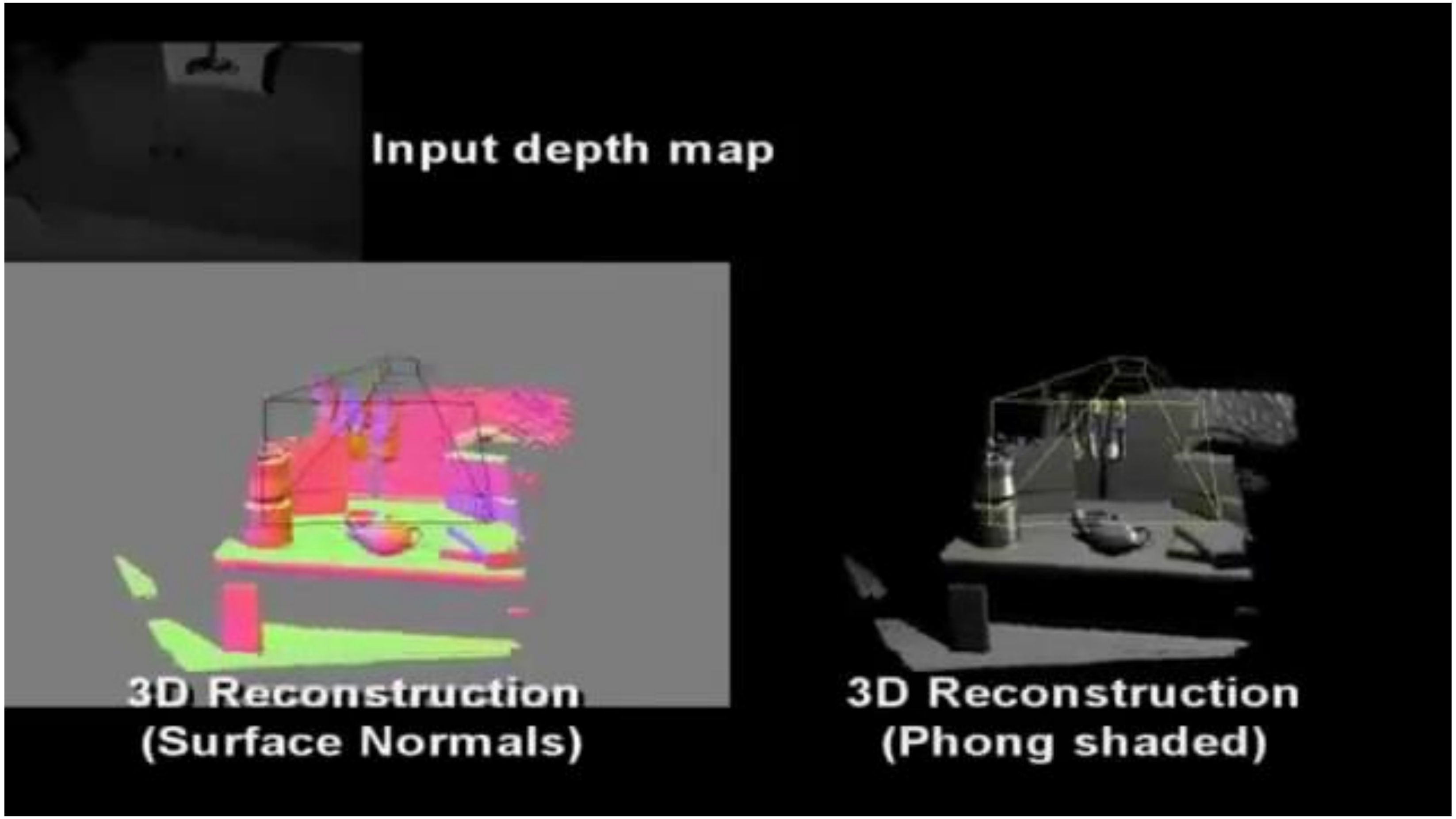
https://www.youtube.com/watch?v=IuBGKxgaxS0

LSD-SLAM



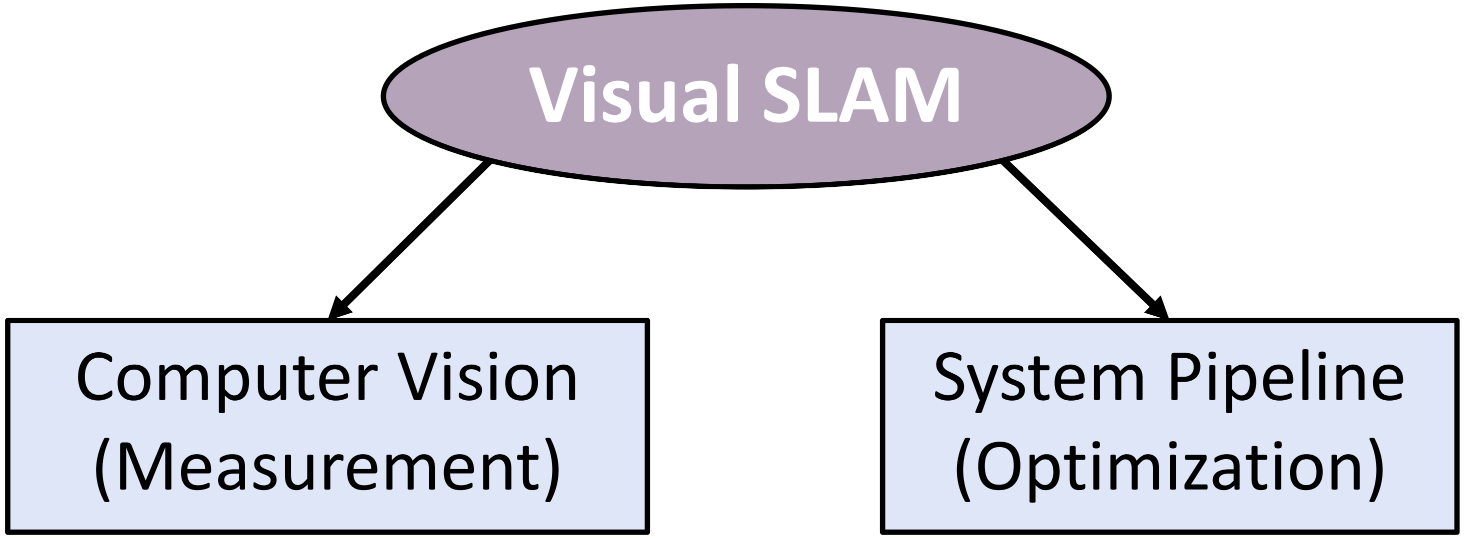
https://www.youtube.com/watch?v=GnuQzP3gty4

Kinect Fusion



https://www.youtube.com/watch?v=KOUSSIKUJ-A

Feature-based Visual SLAM



Computer Vision (Measurement)

- **Feature Points Matching**
- **Perspective-n-Points**
- **Bundle Adjustment**
- **Epipolar Geometry**

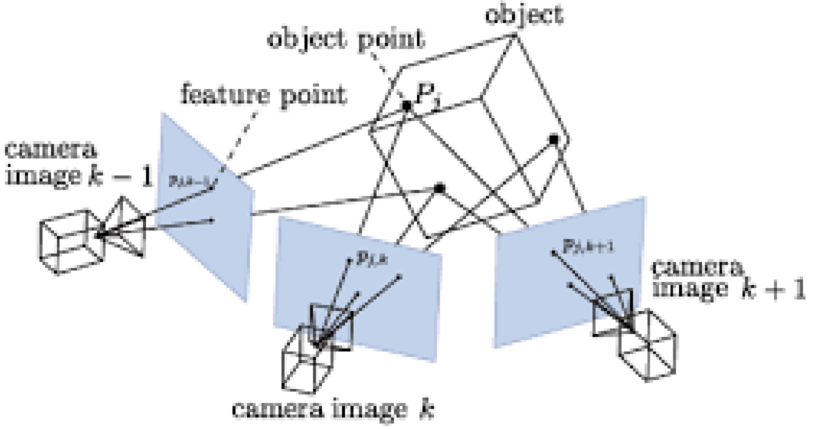
System Pipeline (Optimization)

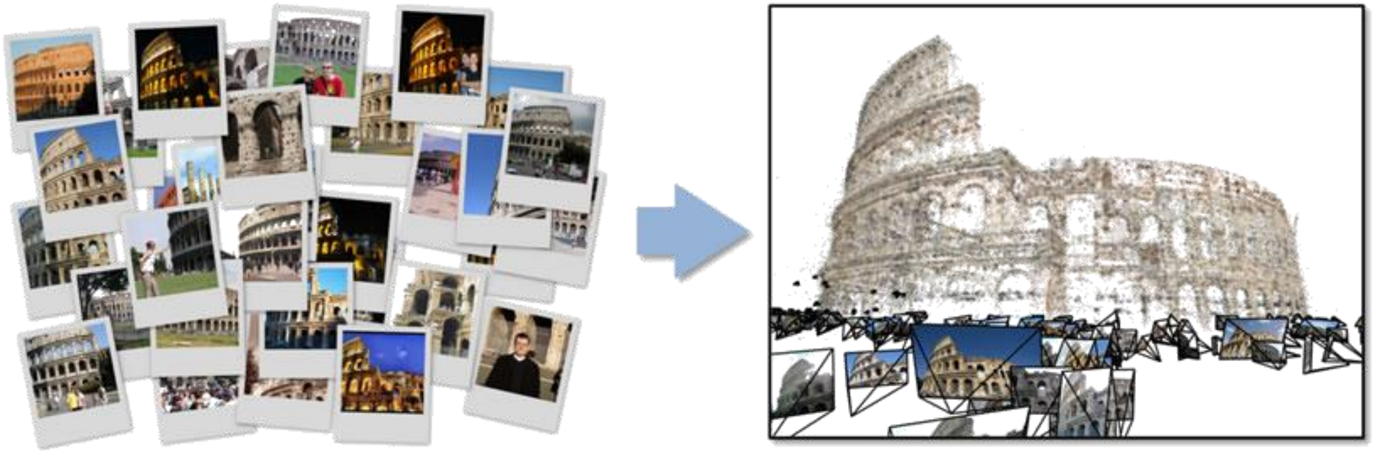
- **Map storage**
- **Graph Optimization**
- **Loop Closure Detection**

Computer Vision / Multi-View Geometry

Structure from Motion (SfM)

- Structure from motion: automatic recovery of **camera motion** and **scene structure** from two or more images.

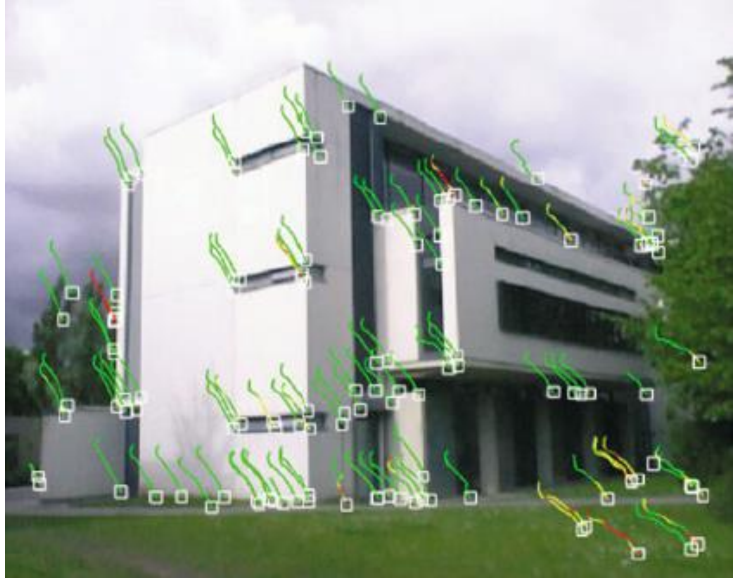




SfM Pipeline

2D feature tracking → 3D estimation → optimization (bundle adjust) → geometry fitting

- Step 1: Track Features
  - Detect good features (SIFT)
  - Find correspondences between frames
    - Lucas & Kanade-style motion estimation
    - window-based correlation
    - SIFT matching

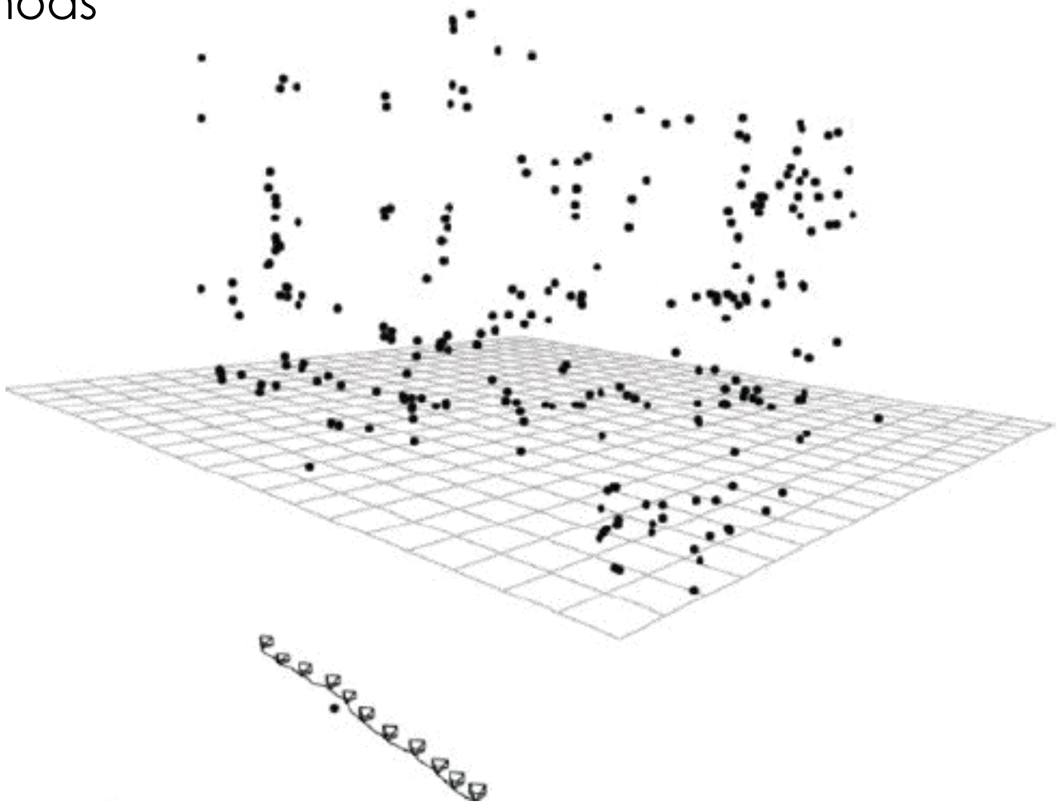


SfM Pipeline

- Step 2: Estimate Motion and Structure
  - Simplified projection model
  - 2 or 3 views at a time

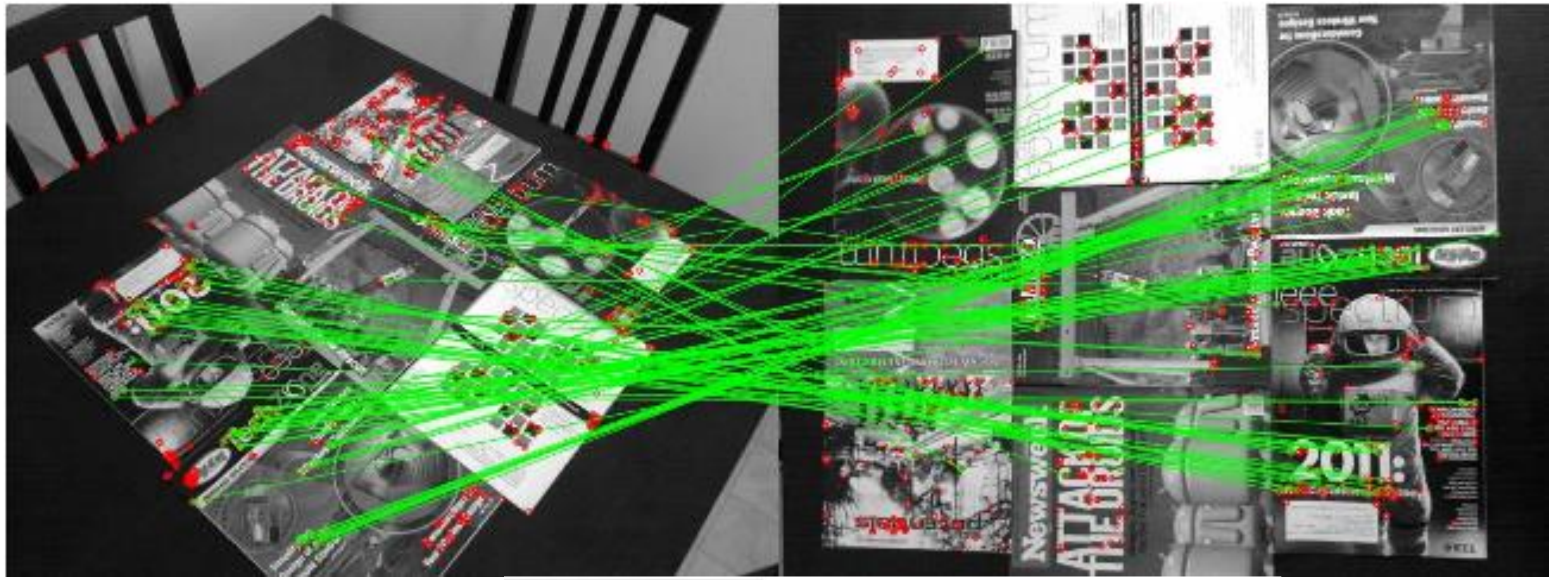
SfM Pipeline

- Step 3: Optimization to refine estimation
  - "Bundle adjustment" in photogrammetry
  - Other iterative methods

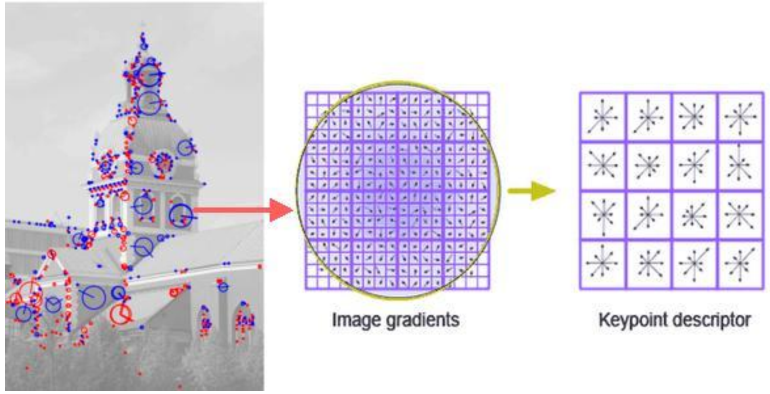


Feature Points Matching

Feature Points Detection/Description



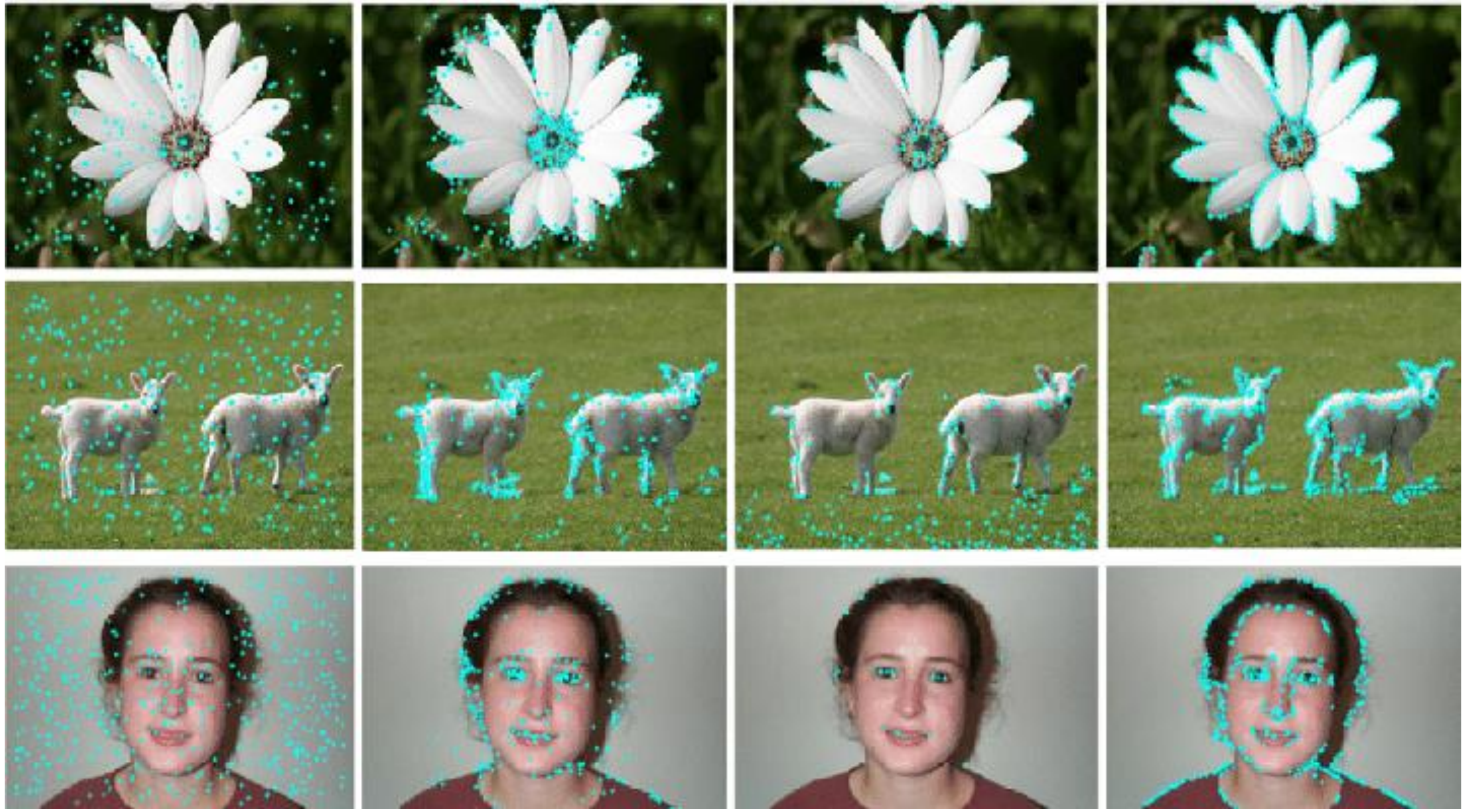
SIFT, SURF, ORB



Feature Point Extraction

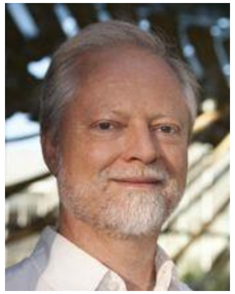
Popular Feature Extractors

SURF	SIFT	Harris	CEFF

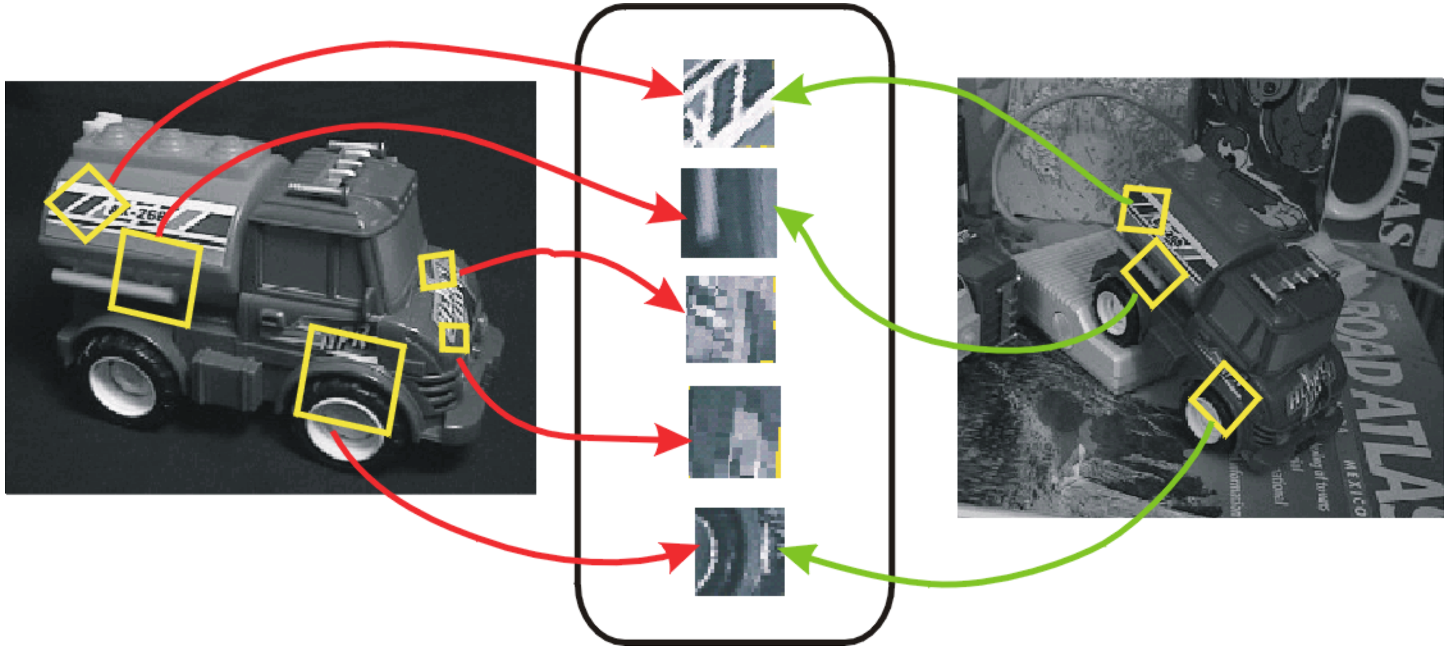


[Ref] Nawaz, Mehmood, et al. Clustering based one-to-one hypergraph matching with a large number of feature points. *Signal Processing: Image Communication*, 2019, 74: 289-298.

Idea of SIFT



- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters

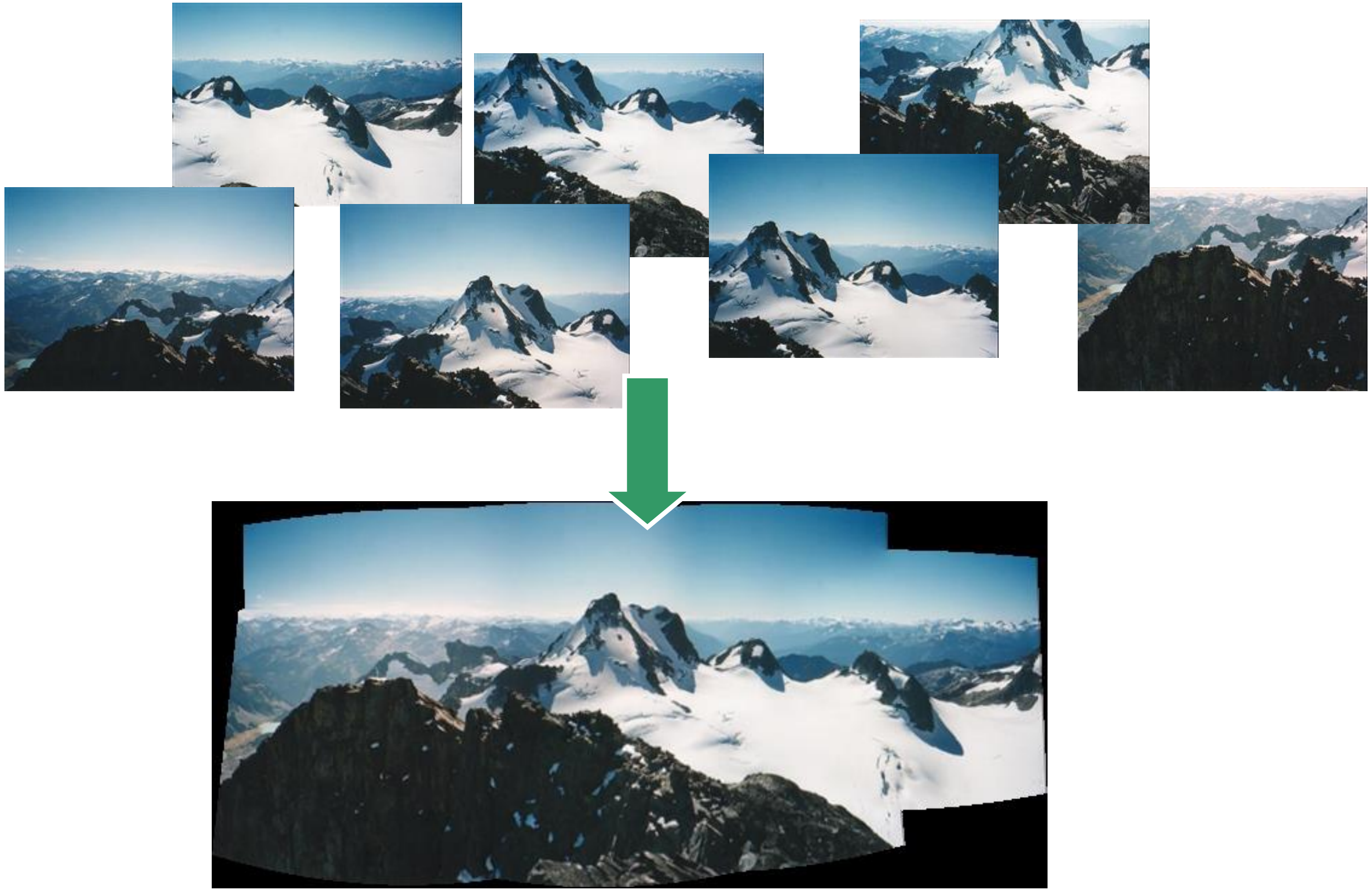


SIFT Features

[Ref] Lowe, David G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 2004, 60.2: 91-110.

Application: Object Recognition (Matching)

Application: Image Stitching



Application: Photosynth

Claimed Advantages of SIFT

- **Locality**
  - features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness**
  - individual features can be matched to a large database of objects
- **Quantity**
  - many features can be generated for even small objects
- **Efficiency**
  - close to real-time performance
- **Extensibility**
  - can easily be extended to wide range of other feature types, with each adding robustness

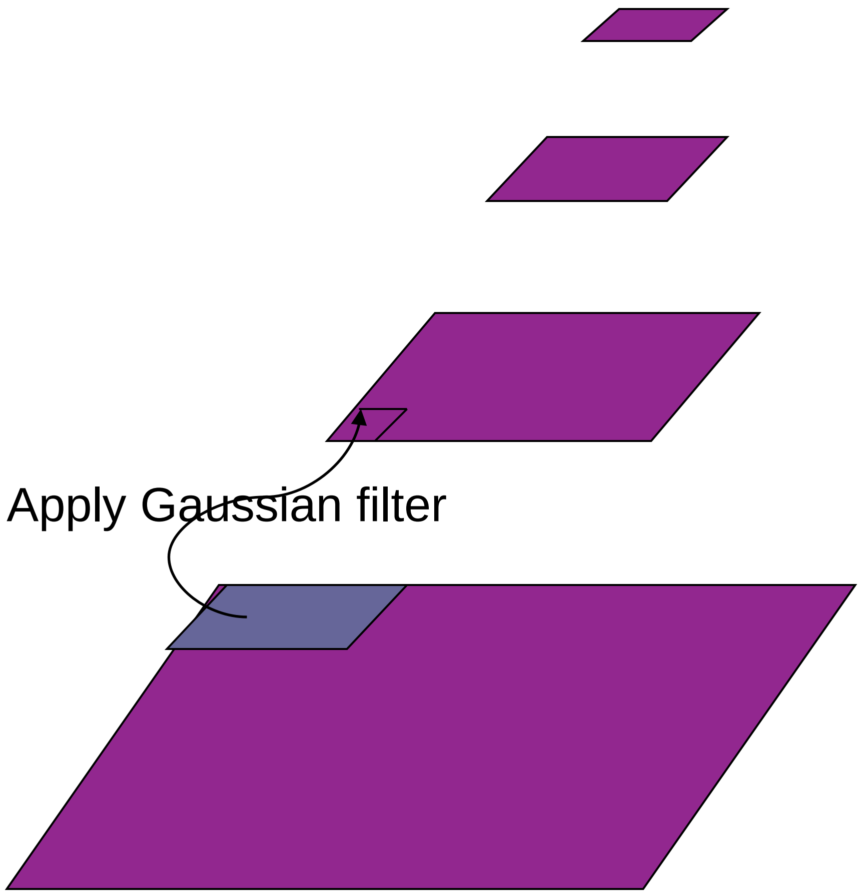
4 Steps of SIFT

- **Scale-space extrema detection**
  – Search over multiple scales and image locations
- **Keypoint localization**
  – Fit a model to determine location and scale
  – Select keypoints based on a measure of stability
- **Orientation assignment**
  – Compute best orientation(s) for each keypoint region
- **Keypoint descriptor**
  – Use local image gradients at selected scale and rotation to describe each keypoint region

1. Scale-space Extrema Detection

- Goal:
  - Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- Method:
  - Search for stable features across multiple scales using a continuous function of scale.
- Prior work has shown that under a variety of assumptions, the best function is a Gaussian function.
- The scale space of an image is a function $L(x,y,\sigma)$ that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

Gaussian Pyramid

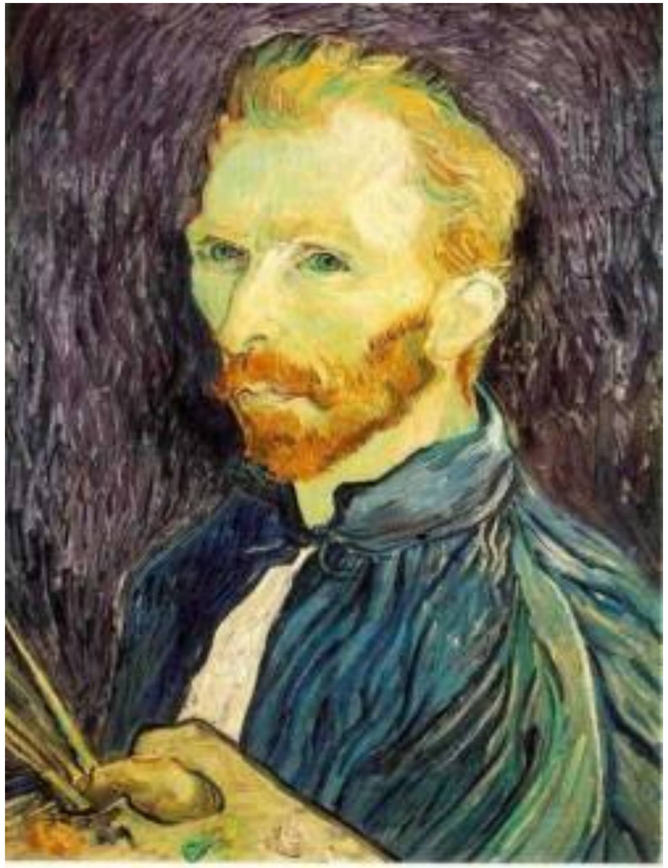


And so on.

At 2nd level, each pixel is the result of applying a Gaussian mask to the first level and then subsampling to reduce the size.

Bottom level is the original image.

Example

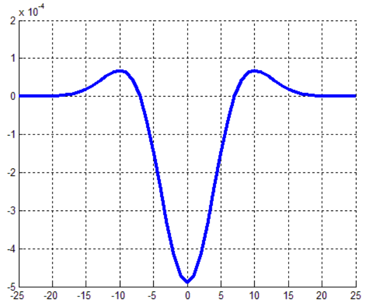


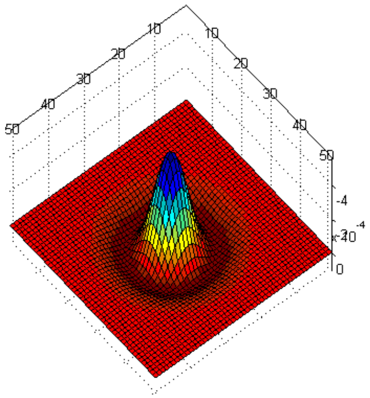
Gaussian 1/2

G 1/4

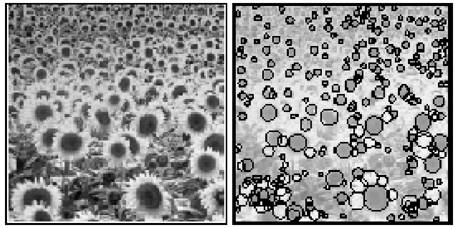
G 1/8

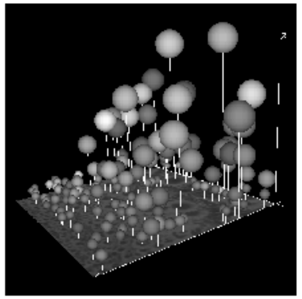
Lowe's Scale-space Interest Points





- **Laplacian of Gaussian** kernel
  - Scale normalized
  - Proposed by Lindeberg
- Scale-space detection
  - Find local maxima across scale/space
  - A good "blob" detector



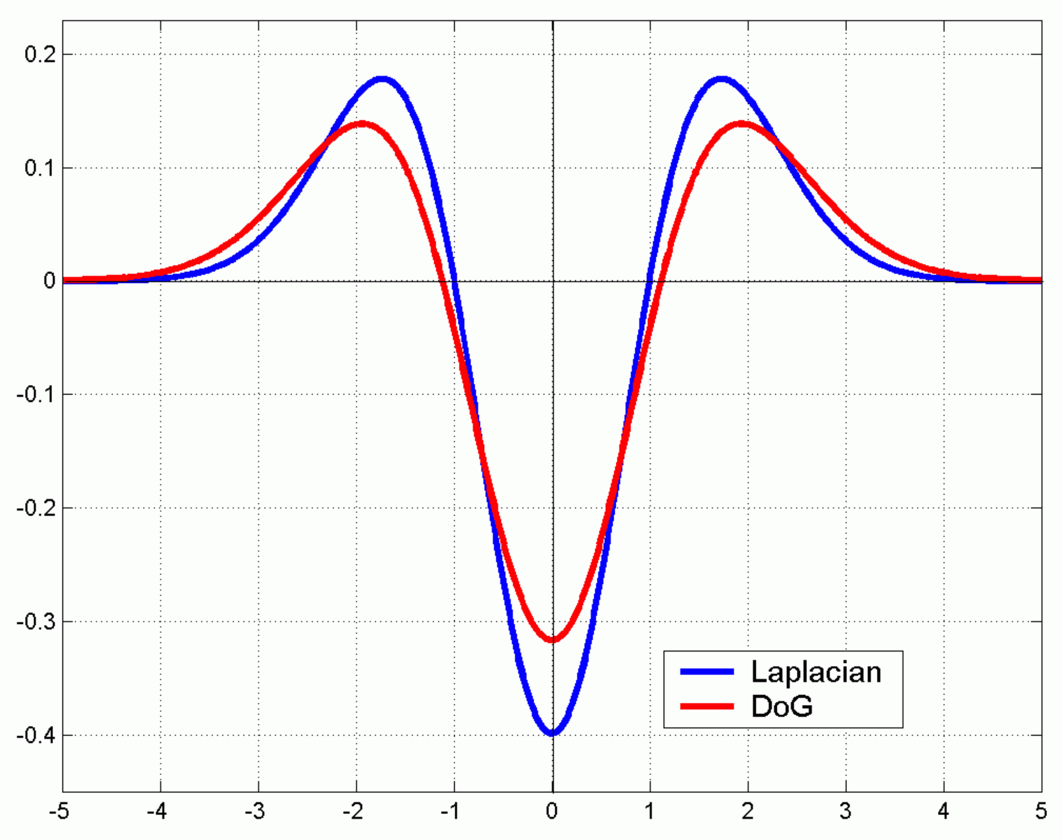


$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

$$\Delta[G_\sigma(x, y) * f(x, y)] = [\Delta G_\sigma(x, y)] * f(x, y)$$

$$LoG = \Delta G_\sigma(x, y) = \frac{\partial^2 G_\sigma(x, y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x, y)}{\partial y^2} = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} e^{-(x^2+y^2)/2\sigma^2}$$

Lowe's Scale-space Interest Points: Difference of Gaussians



$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2 + y^2}{2\sigma^2})$$

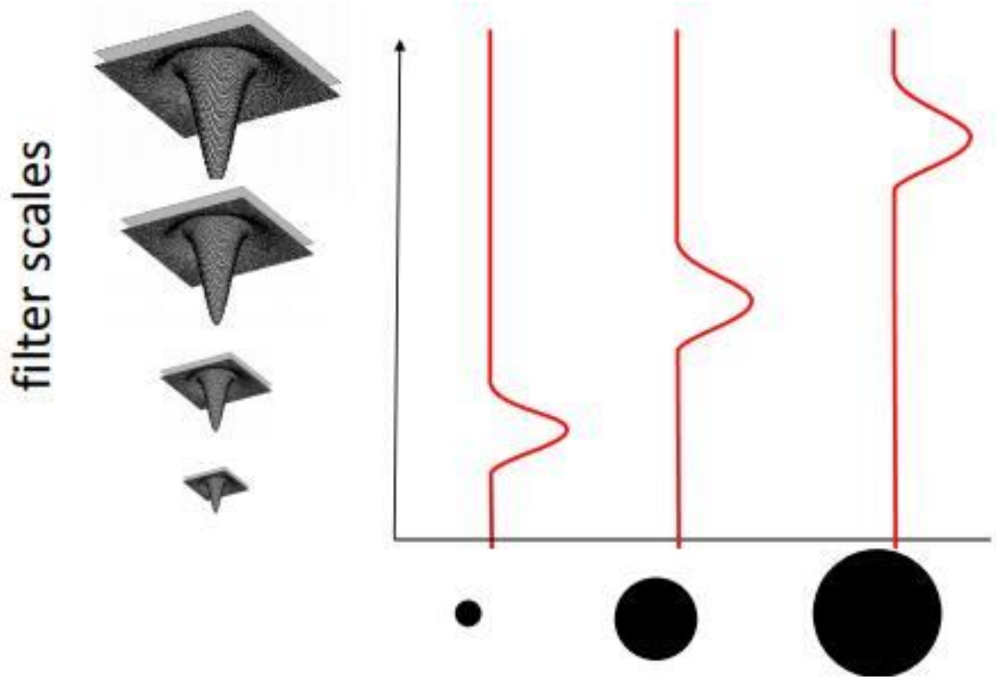
$$DoG = G_{\sigma 1} - G_{\sigma 2} = \frac{1}{\sqrt{2\pi}}[\frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2}]$$

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

$$\frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

Lowe's Scale-space Interest Points: Difference of Gaussians



Lowe's Pyramid Scheme

$\sigma_{s+1}=2^{(s+1)/s}\sigma_0$

$\vdots$

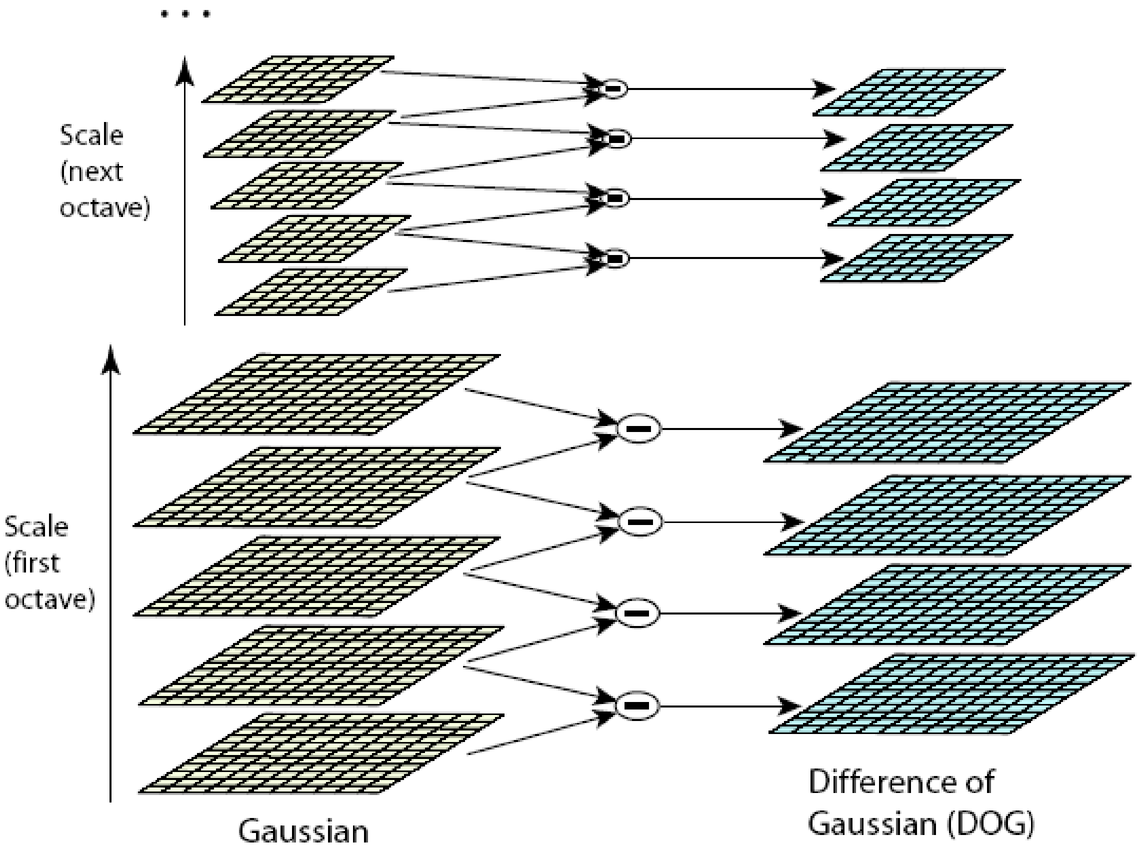
$\sigma_i=2^{i/s}\sigma_0$

$\vdots$

$\sigma_2=2^{2/s}\sigma_0$

$\sigma_1=2^{1/s}\sigma_0$

σ_0

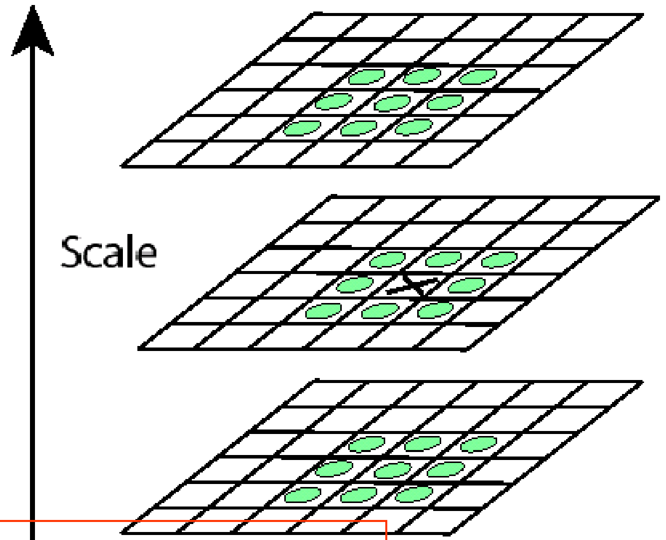


The parameter **s** determines the number of images per octave

2. Key point localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below

s+2 difference images.
Ignore top and bottom.
Search s planes.



For each max or min found, output is the **location** and the **scale**.

2. Keypoint Localization

- There are still a lot of points, some of them are not good enough
  - The locations of keypoints may be not accurate

Taylor series expansion

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^{\mathbf{T}} \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x}$$

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}$$

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}.$$

Eliminating the Edge Response

- Reject flats by a gradient threshold:
  - $|D(\hat{\mathbf{x}})| < 0.03$
- Reject edges by a ratio threshold: $\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Let α be the eigenvalue with larger magnitude and β the smaller.

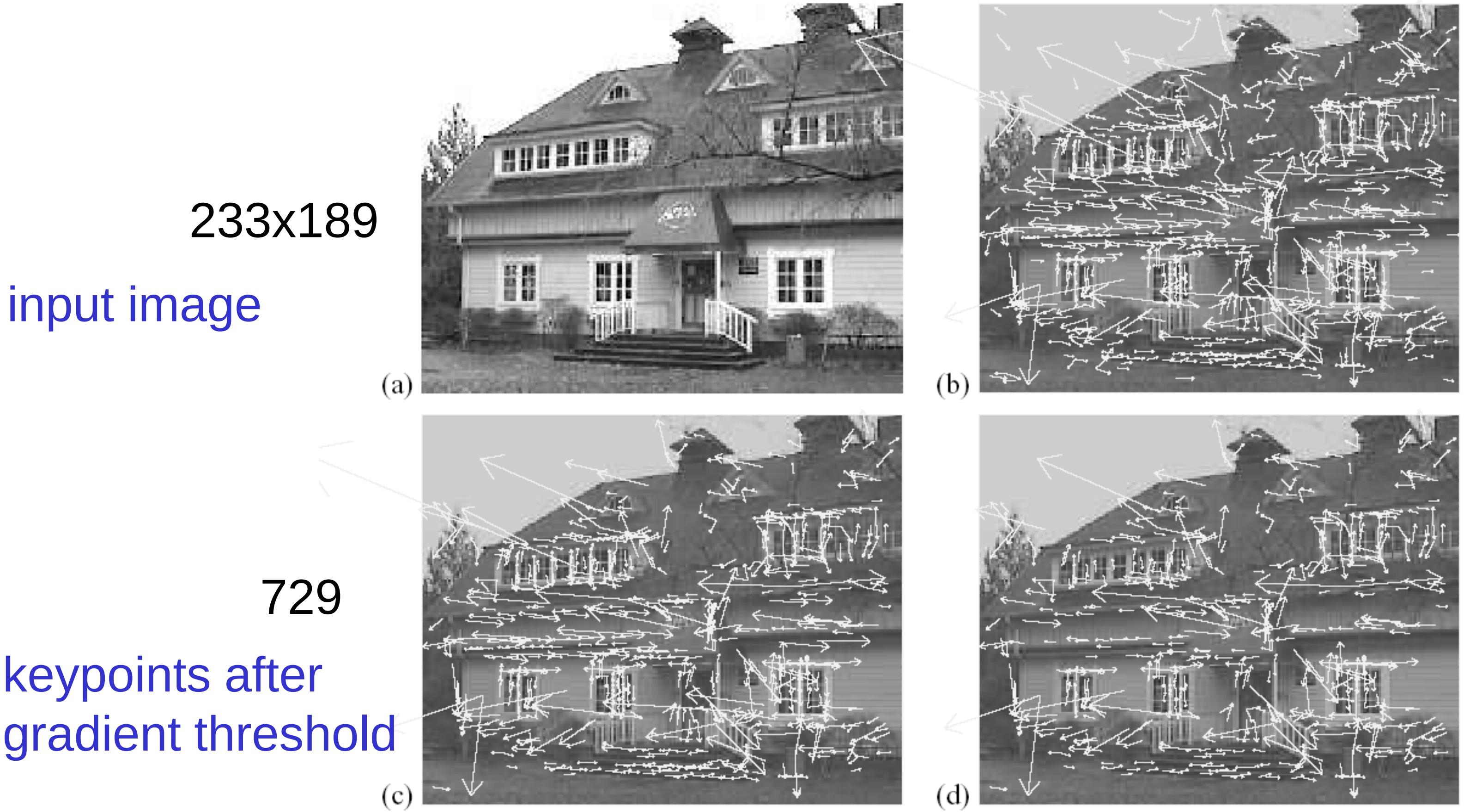
$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

Let $r = \alpha/\beta$.
So $\alpha = r\beta$

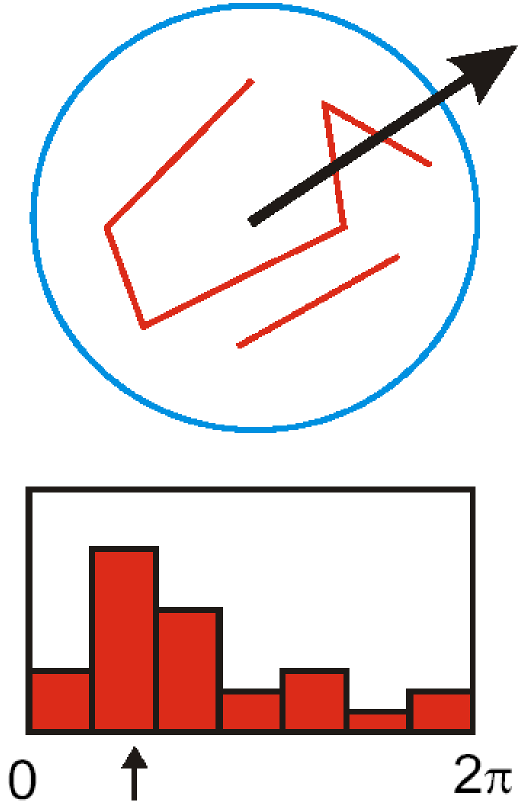
$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

  - $r < 10$

Eliminating the Edge Response



3. Orientation assignment



- Create histogram of local gradient directions at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates

  (x, y, scale, orientation)

If 2 major orientations, use both.

Orientation Assignment

- Assign an orientation to each keypoint, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation
- Compute magnitude and orientation on the Gaussian smoothed images

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

4. Keypoint Descriptors

- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

Lowe's Keypoint Descriptor
(shown with 2 X 2 descriptors over 8 X 8)

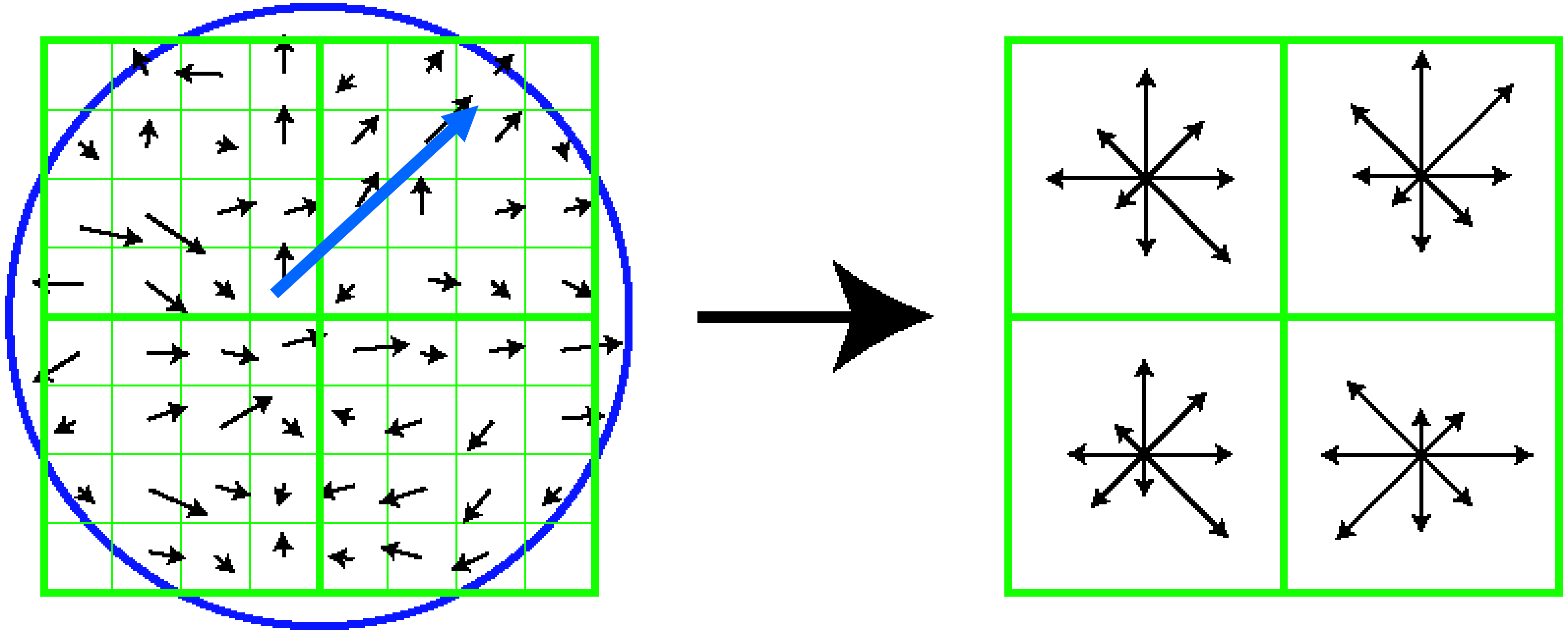


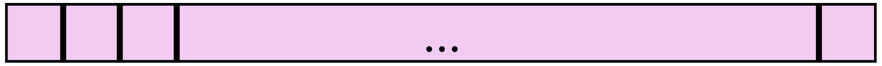
Image gradients

Keypoint descriptor

In experiments, 4x4 arrays of 8 bin histogram is used,
a total of 128 features for one keypoint

Lowe's Keypoint Descriptor

- Use the normalized region about the keypoint
- Compute gradient magnitude and orientation at each point in the region
- Weight them by a Gaussian window overlaid on the circle
- Create an orientation histogram over the 4 X 4 subregions of the window
- 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives a vector of 128 values.



Application on Object Recognition

- The SIFT features of training images are extracted and stored
- For a query image
  1. Extract SIFT feature
  2. Efficient nearest neighbor indexing
  3. 3 keypoints, Geometry verification (RANSAC)

Extensions

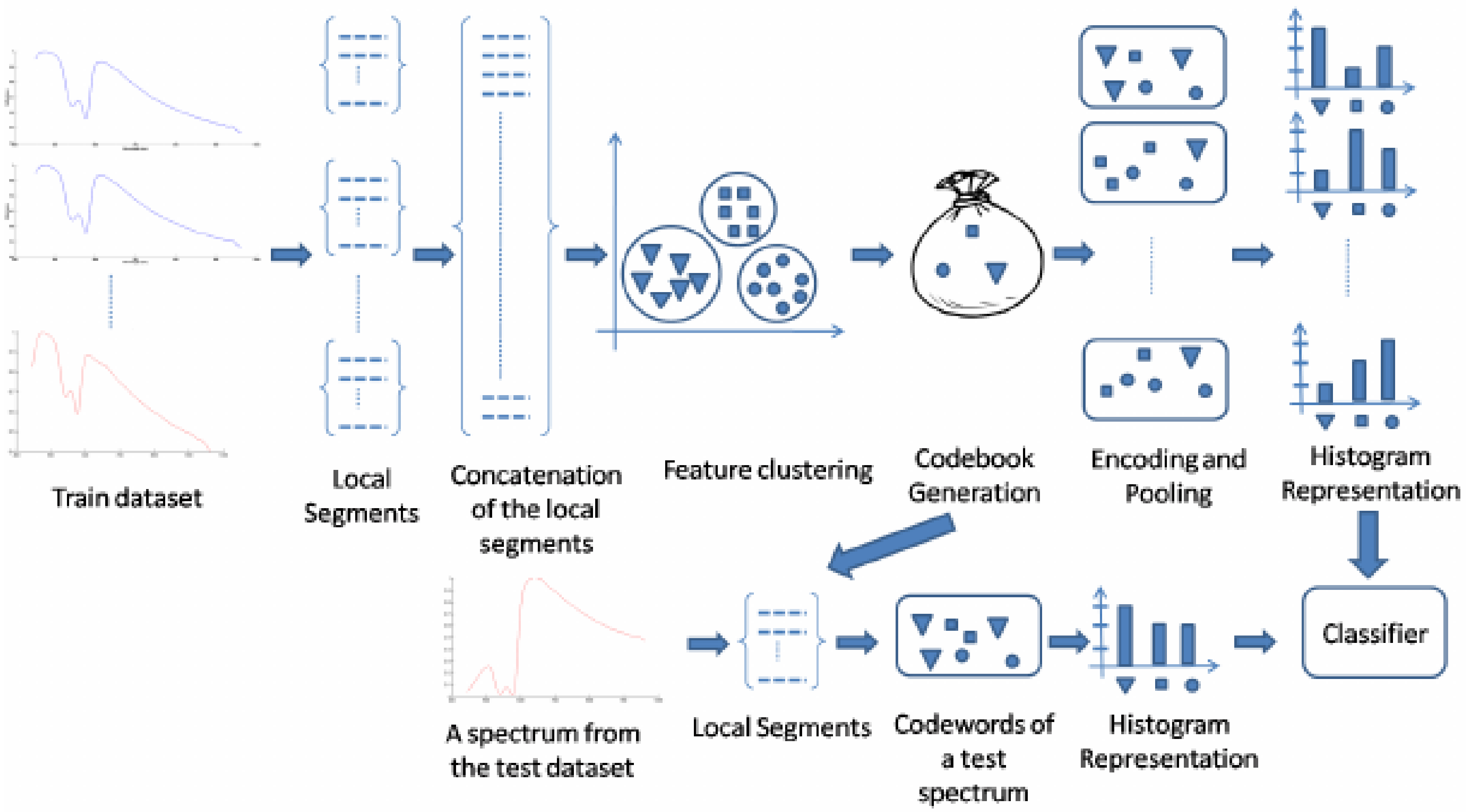
- PCA-SIFT

1. Work on patches wit size 41*41 pixels
2. Compute vertical and horizontal gradient for all pizels (2*39*39 dimensions)
3. Use PCA to project it to 20 dimensions

SURF

- Approximate SIFT
- Works almost equally well
- Very fast

Bag of Words



Image/ Document

RANSAC

RANdom SAmple Consensus

repeat

select minimal sample (8 matches)

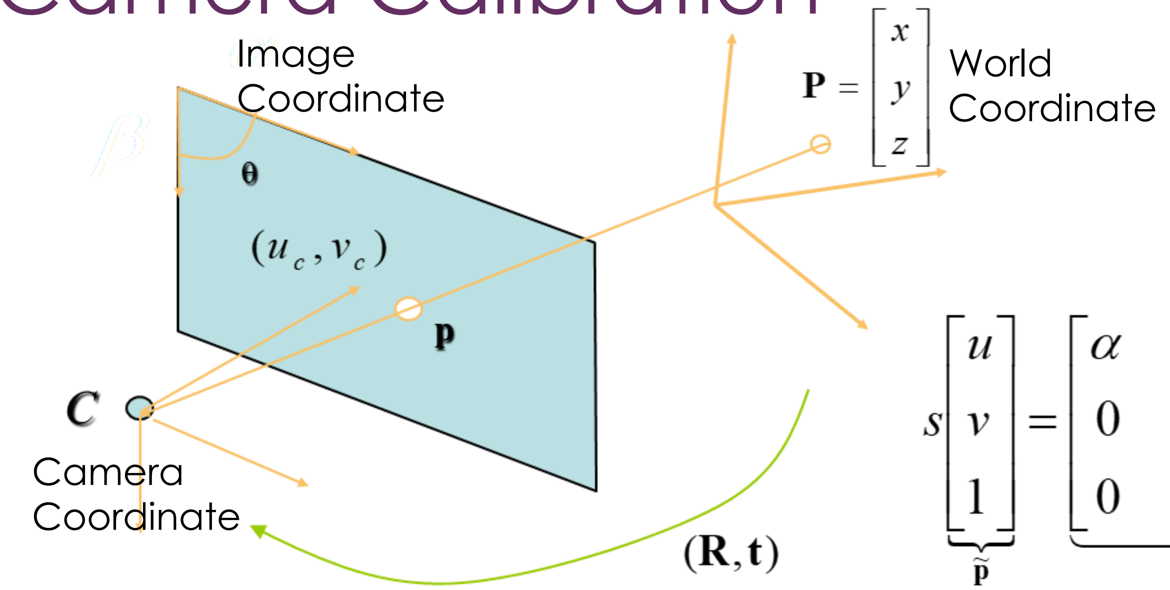
compute solution(s) for F

determine inliers

until Γ(*#inliers*,*#samples*)>95% or too many times

compute F based on all inliers

Camera Calibration



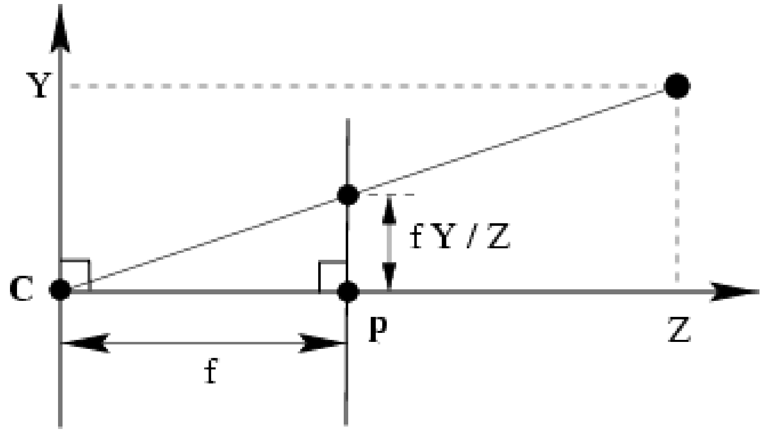
$$s\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\tilde{\mathbf{p}}} = \underbrace{\begin{bmatrix} \alpha & \gamma & u_c \\ 0 & \beta & v_c \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}}_{[\mathbf{R}\ \ \mathbf{t}]} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\tilde{\mathbf{P}}}$$

- Intrinsics:
  - scale factor
  - focal length
  - aspect ratio
  - principle point
  - radial distortion

- Extrinsics
  - optical center
  - camera orientation

A camera is calibrated when intrinsics/extrinsics are known.

Pinhole Camera Projection Model



$$x = \frac{fX}{Z} \qquad y = \frac{fY}{Z}$$

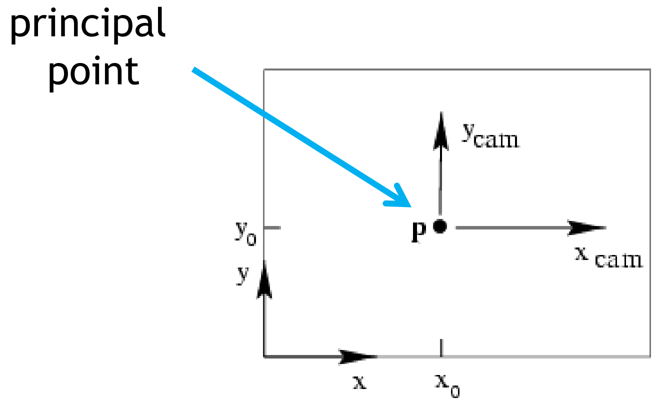
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsic matrix}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{extrinsic matrix}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \mathbf{x} \sim \mathbf{K}[\mathbf{I}|0]\mathbf{X}$$

intrinsic matrix

extrinsic matrix (Camera Coordinate =World Coordinate)

Principal Point Offset

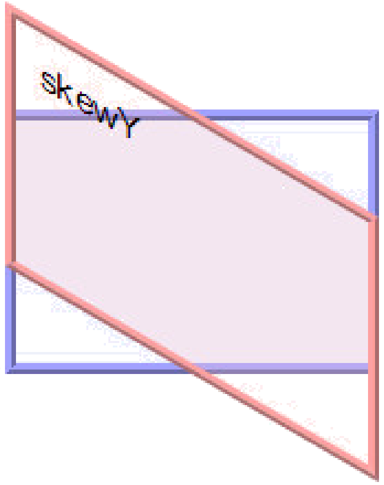


$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} \sim \mathbf{K}[\mathbf{I}|0]\mathbf{X}$$

Intrinsic Matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \mathbf{K} = \begin{bmatrix} fa & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

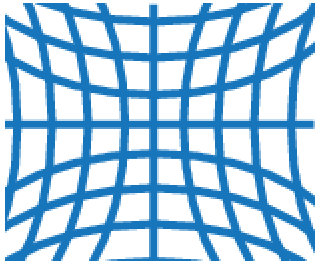


Good enough for modeling the camera projection?

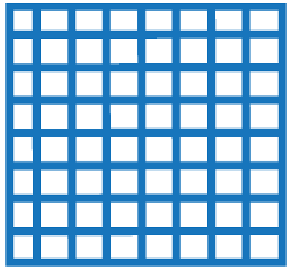
a : aspect ratio (for non-square pixels)
s : skew (for non-rectangular pixels)

$$x_{distorted} = x(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

$$y_{distorted} = y(1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

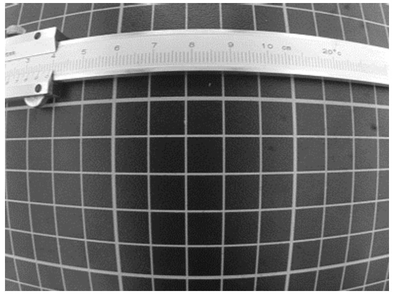


Negative radial distortion "pincushion"

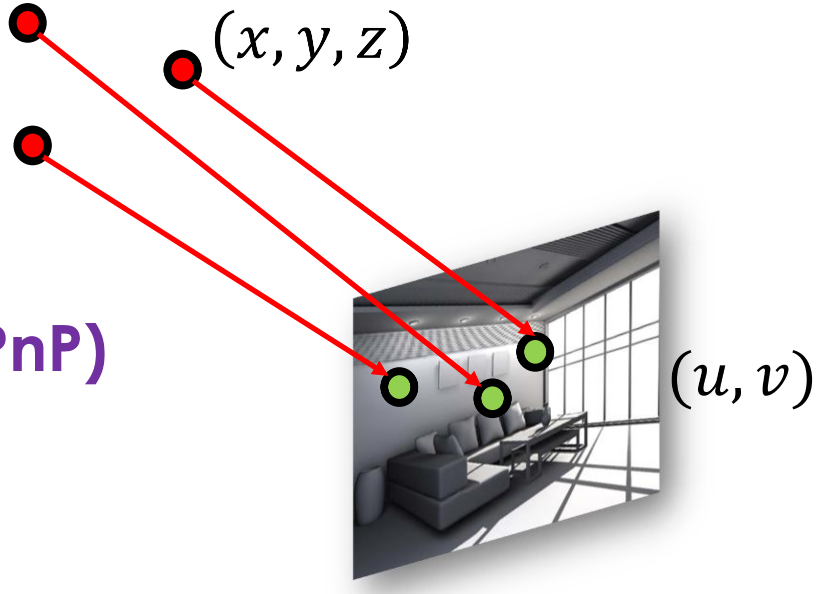


No distortion

Positive radial distortion "barrel"



Transformation Matrix Estimation by Reprojection

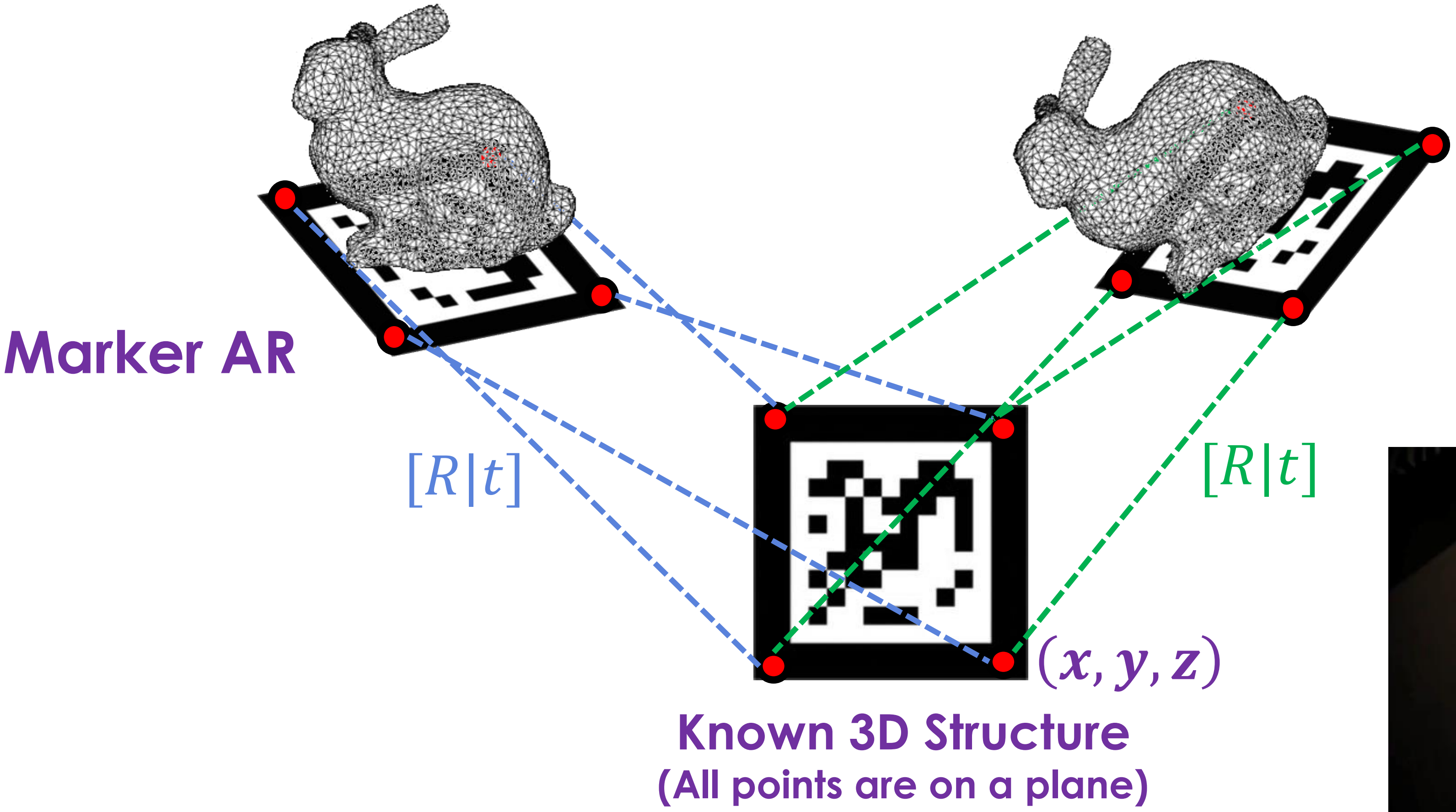


Perspective-n-Point (PnP)

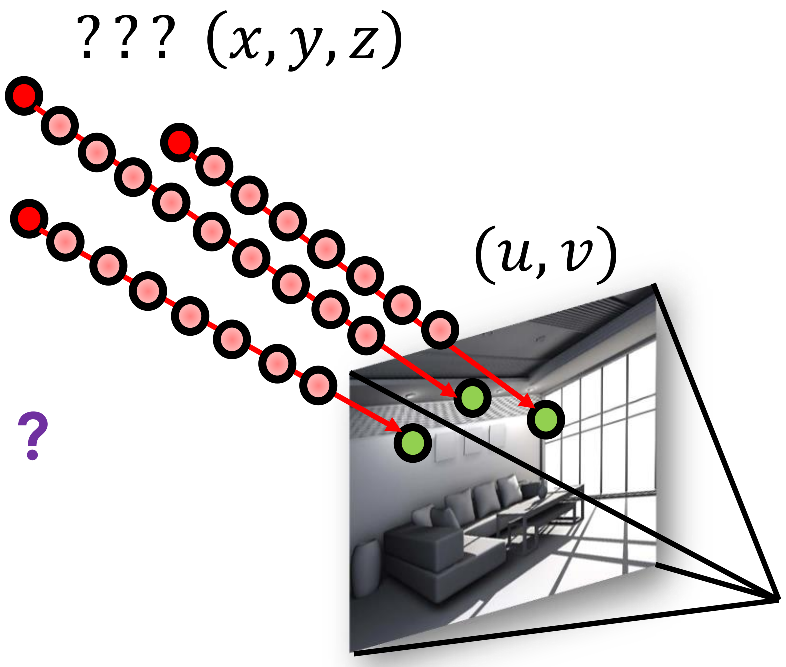
$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Matrix}} \underbrace{\boxed{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}}_{\text{Extrinsic Matrix}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Unknown Need 3 Points to Solve (P3P)

Transformation Matrix Estimation by Reprojection



Transformation Estimation by Reprojection



Unknown Structure ?

Transformation Matrix Estimation by Reprojection

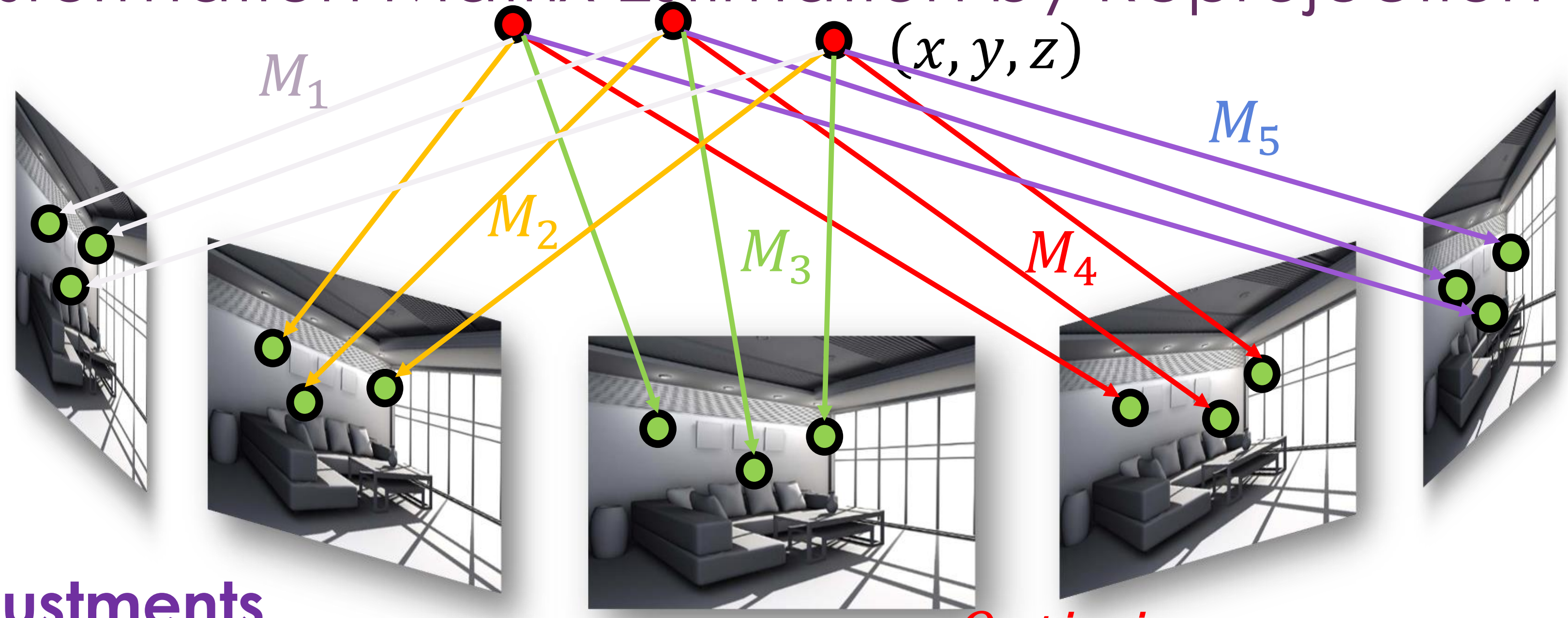
$$sx_1 = KM_1P$$
$$sx_2 = KM_2P$$
$$\vdots$$

Solve M

Direct Optimize → Bundle Adjustment

Reduce P → Epipolar Geometry

Transformation Matrix Estimation by Reprojection



Bundle Adjustments

Optimize

Camera Matrix

$$\boxed{s}\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \boxed{\underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{M} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}$$

Unknown Structure Initialization

$$\overrightarrow{\mathbf{C}_0\mathbf{p}_0} \cdot \left(\overrightarrow{\mathbf{C}_0\mathbf{C}_1} \times \overrightarrow{\mathbf{C}_1\mathbf{p}_1}\right) = 0$$

$$\mathbf{p}_0 \cdot (\mathbf{t} \times \mathbf{R}\mathbf{p}_1) = 0$$

$$\mathbf{p}_0^T \boxed{[\mathbf{t}]_\times} \mathbf{R}\mathbf{p}_1 = 0$$

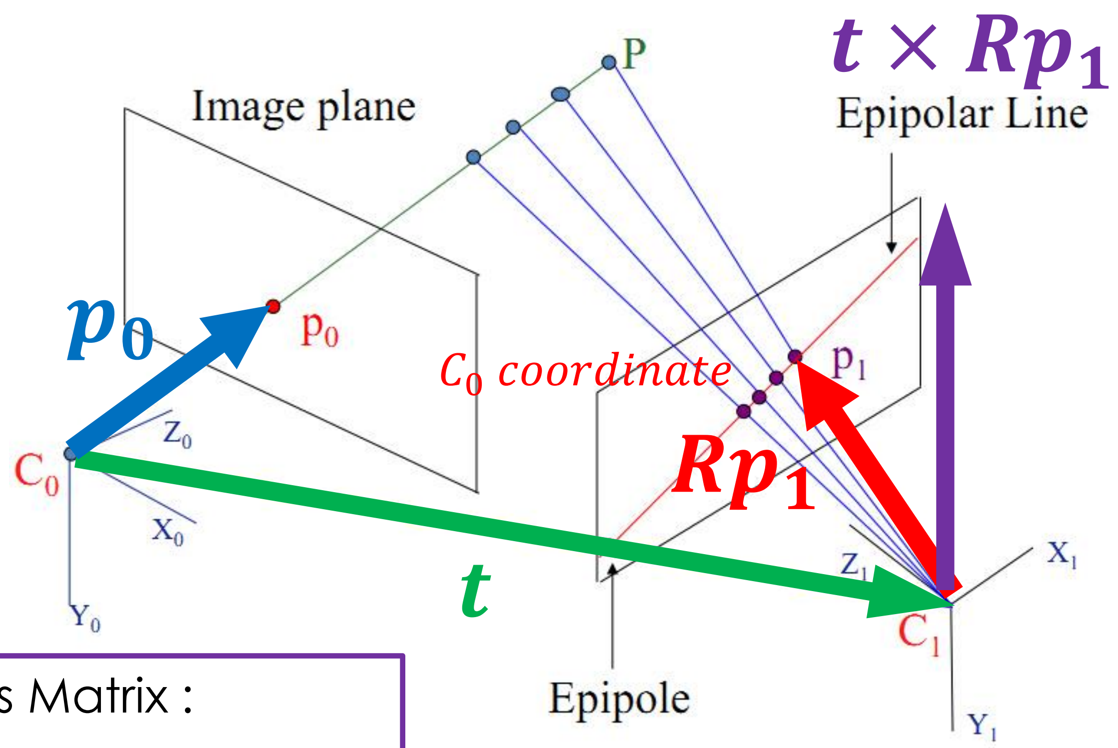
$$\mathbf{p}_0^T \boxed{\mathbf{E}} \mathbf{p}_1 = 0$$

Essential Matrix

Cross Matrix :

$$[t]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

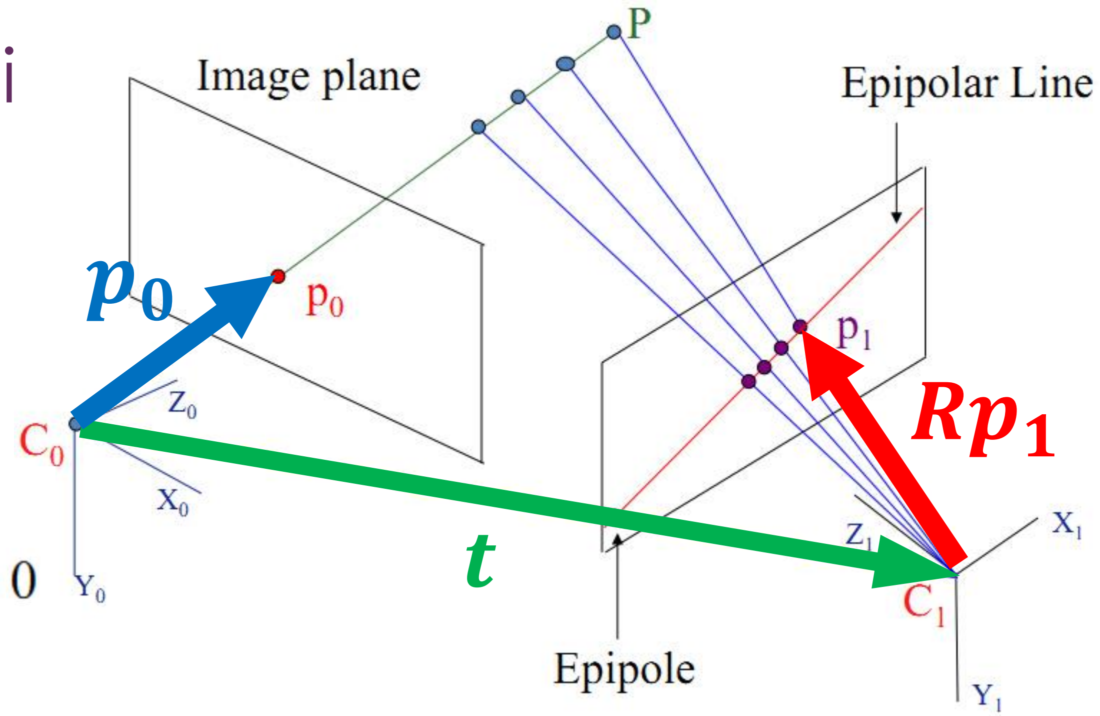
Epipolar Geometry



Unknown Structure Initi

Epipolar Geometry

$$\mathbf{p}_0^T \, \mathbf{E} \, \mathbf{p}_1 = 0$$



$$\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0$$

Write as **A x = 0**, where **x** = (E11, E12, E13, … , E33)

$$\begin{pmatrix} x_0x_1 & x_0y_1 & x_0 & y_0x_1 & y_0y_1 & y_0 & x_1 & y_1 & 1 \end{pmatrix} \begin{pmatrix} E_{11} \\ E_{12} \\ E_{13} \\ \vdots \\ E_{33} \end{pmatrix} = 0$$

Unknown Structure Initialization

Essential Matrix Decomposition

Scale

$$E = [t]_\times R = U\Sigma V^T = U \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \boxed{0} \end{bmatrix} V^T$$

Loss of Rank

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, ZW = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

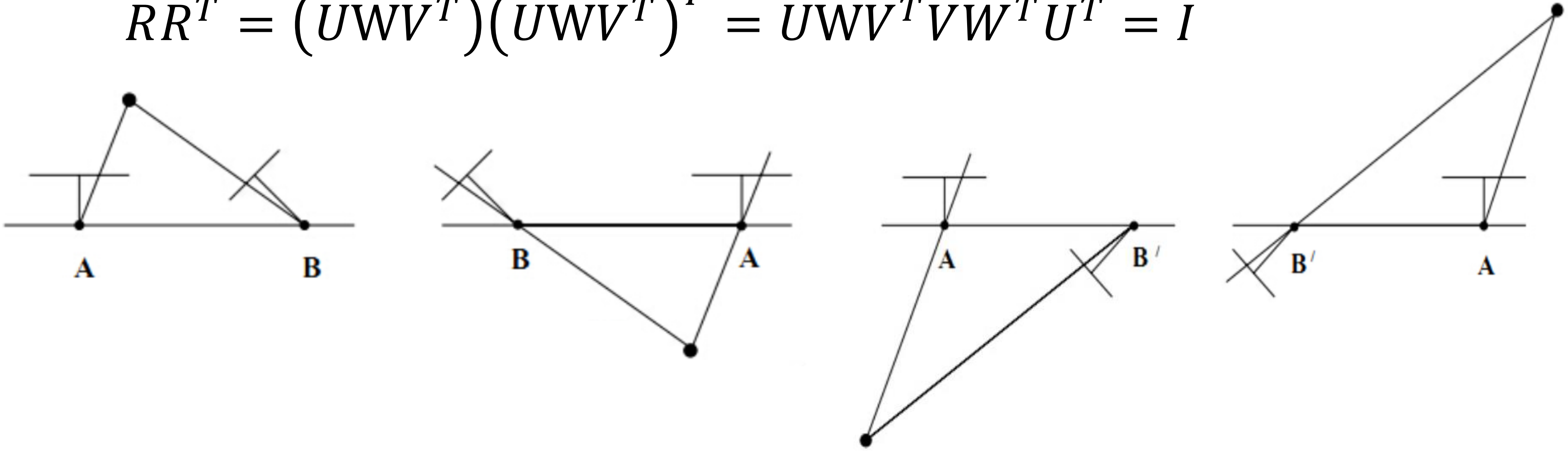
$$E = UZWV^T = \underbrace{(UZU^T)}_{[\boldsymbol{t}]_\times}\underbrace{(UWV^T)}_{\boldsymbol{R}} = [\boldsymbol{t}]_\times R$$

Unknown Structure Initialization

Essential Matrix Decomposition

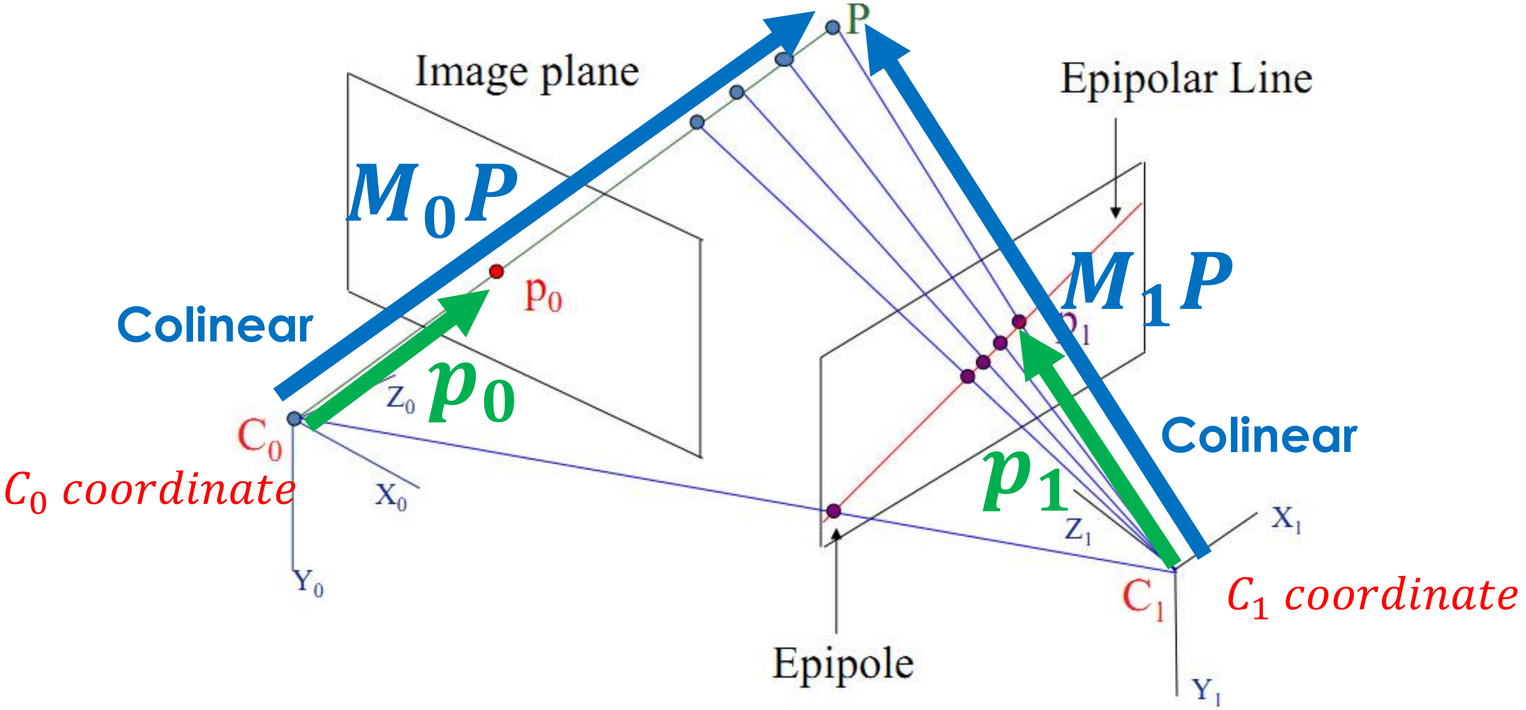
$$\begin{bmatrix} 0 & u_{33} & -u_{23} \\ -u_{33} & 0 & u_{13} \\ u_{23} & -u_{13} & 0 \end{bmatrix} \longrightarrow [\boldsymbol{t}]_\times$$

$$RR^T = \left(UWV^T\right)\left(UWV^T\right)^T = UWV^TVW^TU^T = I$$



Unknown Structure Initialization

3D Structure Recovering (Triangulation)



$$M_0 = [R_0|t_0]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}, M_1 = [R_1|t_1]$$

$$\begin{aligned} p_0 \times M_0P = 0 \\ p_1 \times M_1P = 0 \end{aligned} \Longrightarrow \begin{pmatrix}[(p_0)_\times]M_0 \\ [(p_1)_\times]M_1\end{pmatrix} P = 0$$

Unknown Structure Initialization

Scaling Problem

- If the 3D coordinates of P are unknown and we measure only is projection in 2 images:
  - Compute the essential matrix using 5 or more points
  - Problem is of dimension 5: i.e. up to a global scale factor
- This is the same as the (old) Hollywood effect