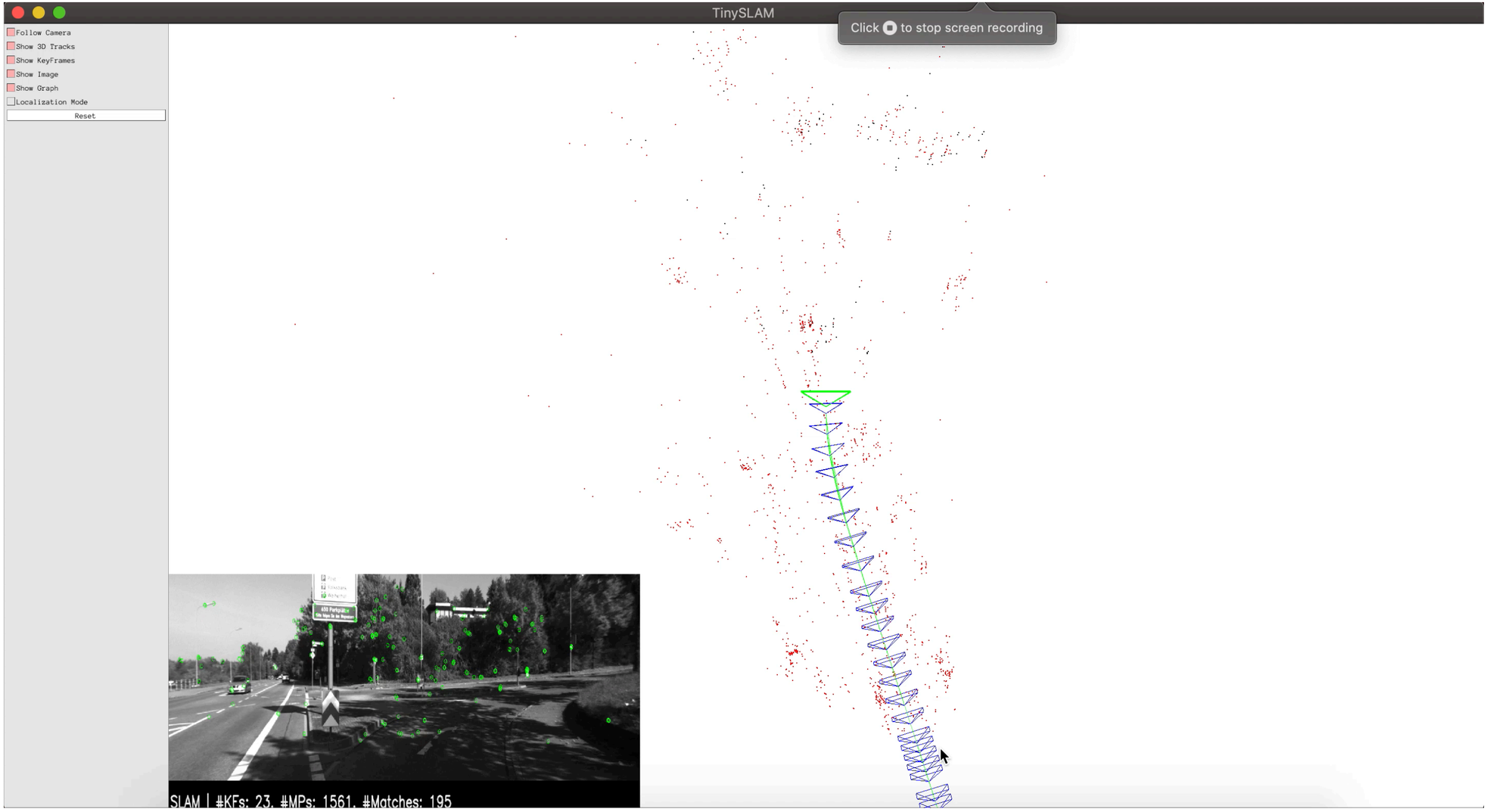
# A Survey on Deep Learning for Localization and Mapping

黄卓斐

# What is SLAM


TinySLAM
Click to stop screen recording
Follow Camera
Show 3D Tracks
Show KeyFrames
Show Image
Show Graph
Localization Mode
Reset
SLAM | #KFs: 23, #MPs: 1561, #Matches: 195

# The SLAM Problem

**SLAM** is the process by which a robot **builds a map** of the environment and, at the same time, uses this map to **compute its location**

- **Localization:** inferring location given a map
- **Mapping:** inferring a map given a location
- **SLAM:** learning a map and locating the robot simultaneously

# The SLAM Problem

**Input:**

- A sequence of images or video

$$\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\}$$

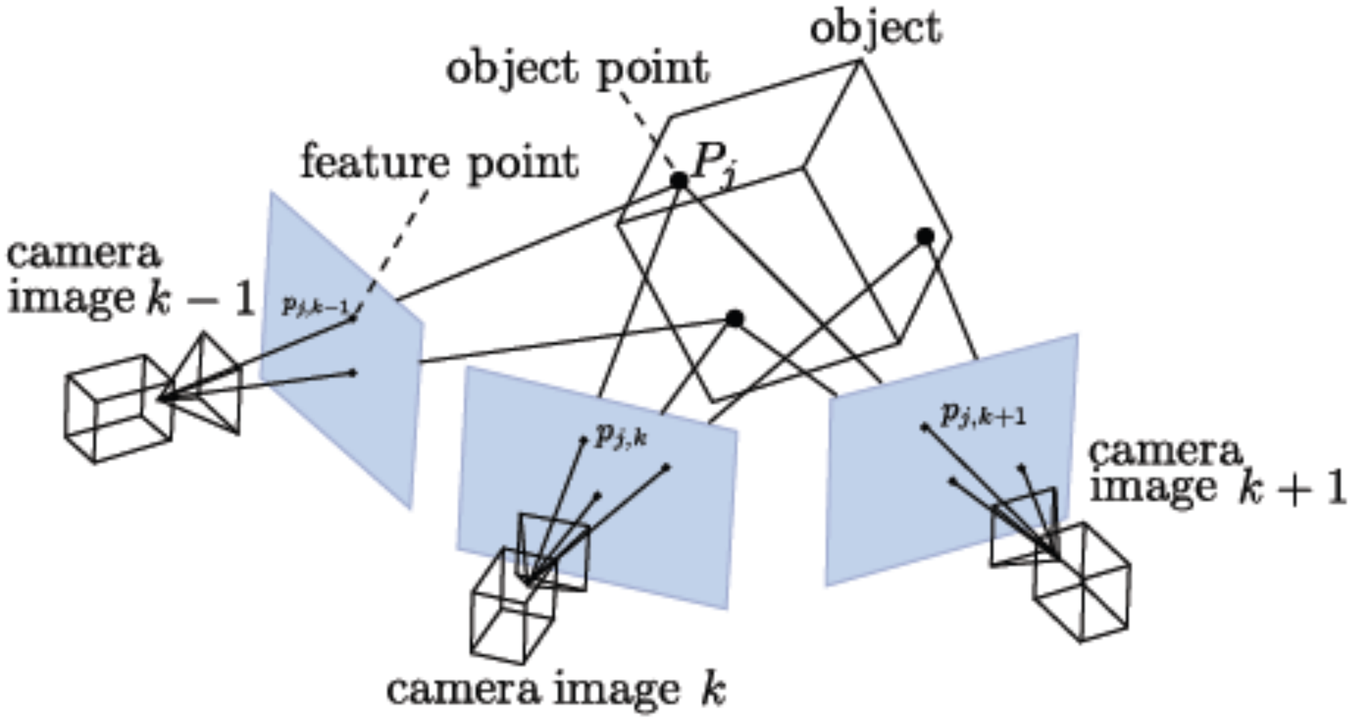
**Output:**

- Map of features

$$\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$$

- Path of the robot(Poses of cameras)

$$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\}$$



# Outline

I. Visual Odometry: Monocular SLAM

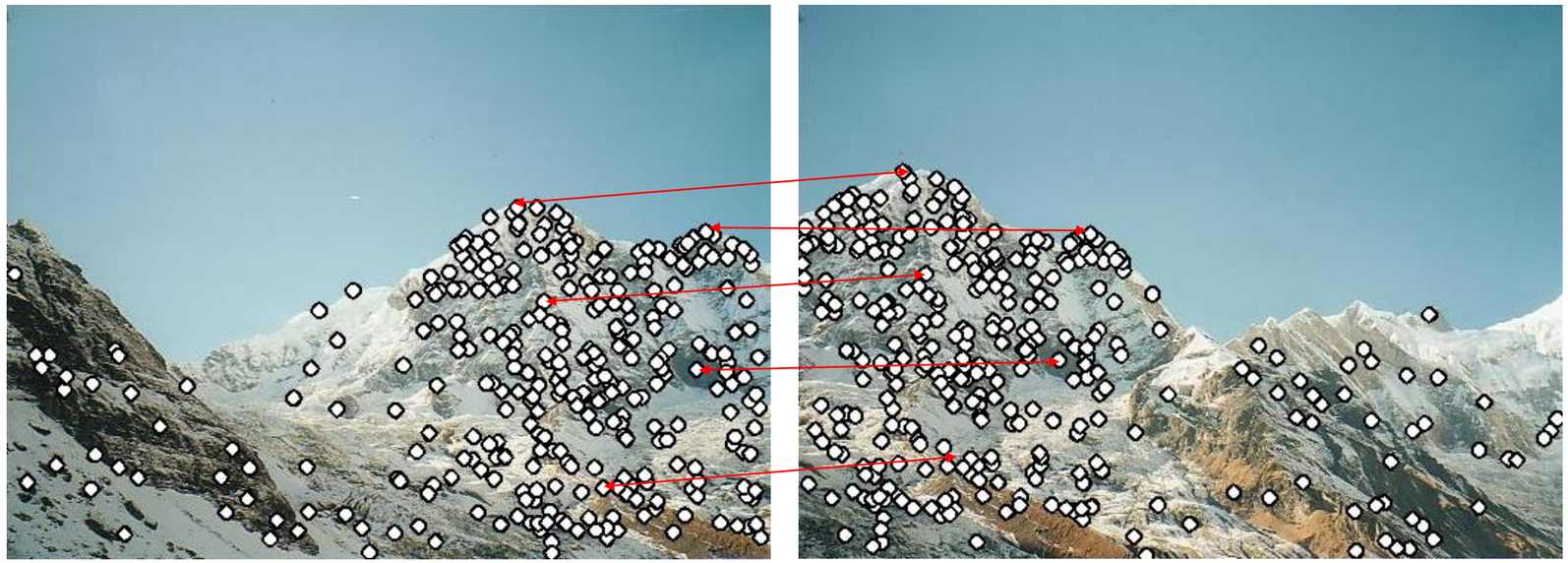
II. Inertial Measurement Unit and Pre-Integration

III. Visual-Inertial Alignment and Bundle Adjustment

IV. Learning Based Feature Extractor

# Matching with Features

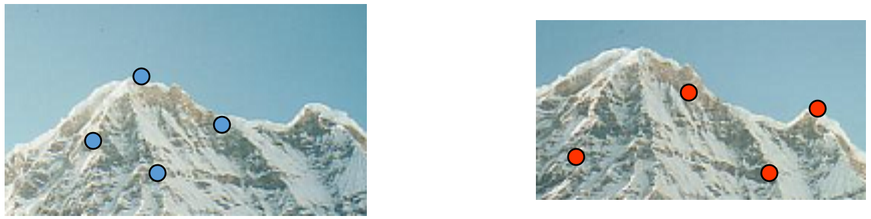
- Detect feature points in both images
- Find corresponding pairs



# Matching with Features

- Task 1:
  - Detect the same point *independently* in both images

counter-example:

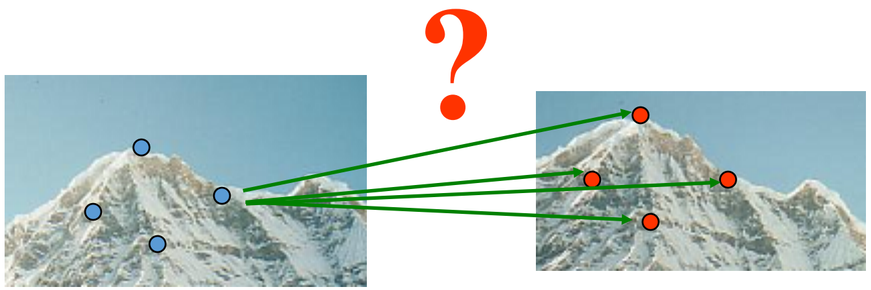


no chance to match!

We need a repeatable **detector**

# Matching with Features

- Task 2:
  - For each point correctly recognize the corresponding one

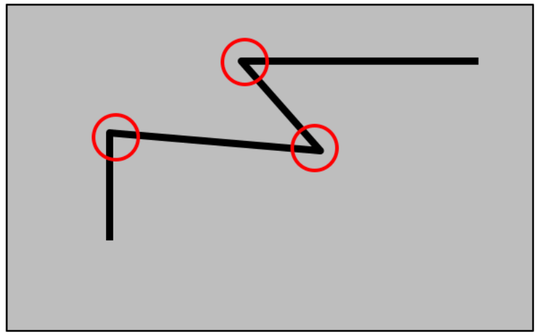



We need a reliable and distinctive **descriptor**

# Harris Detector

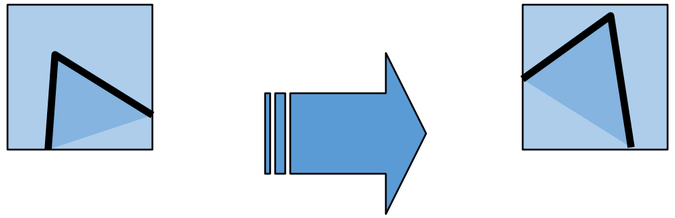
- Feature point detection
  - Harris corner detector

- Why not use in SLAM ?
  - Rotation invariance?
  - Scale invariance?



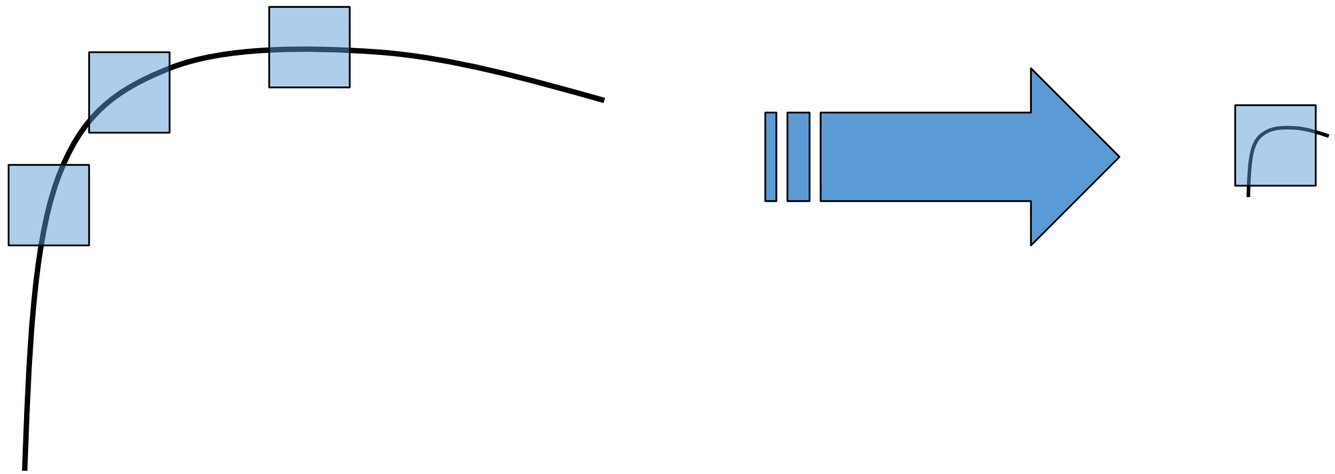
# Harris Detector: Some Properties

- Rotation invariance?



# Harris Detector: Some Properties

- Not invariant to *image scale*!



All points will be classified as edges

Corner !

# Real-Time SLAM

- What's a "good feature"?
  - Satisfies brightness constancy—looks the same in both images
  - Has sufficient texture variation
  - Does not have too much texture variation
  - Does not deform too much over time – Efficiency
- A suitable choice: ORB

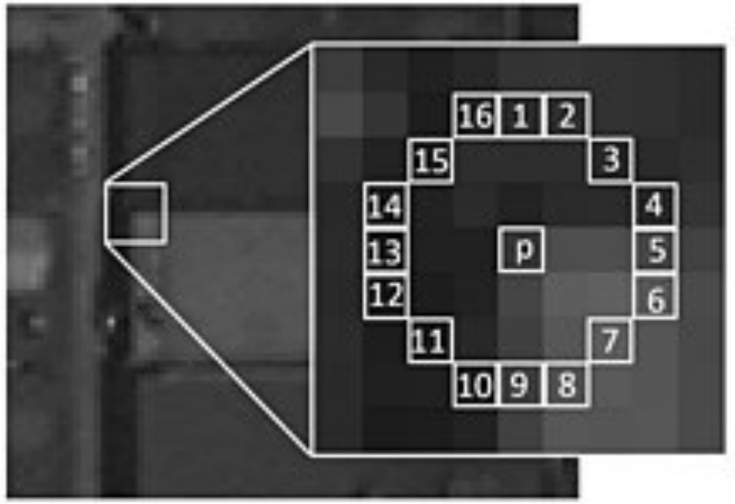
# ORB

- What's "ORB extractor"?
  - Oriented FAST and Rotated BRIEF
  - Representative Real-Time feature extractor
  - FAST: Scale and Rotation Invariant
  - BRIEF: Binary Descriptor

- Extract 1000 feature points in a single image
  - SIFT: 5228.7 ms
  - SURF: 217.3 ms
  - ORB: 15.3 ms

# FAST Detector

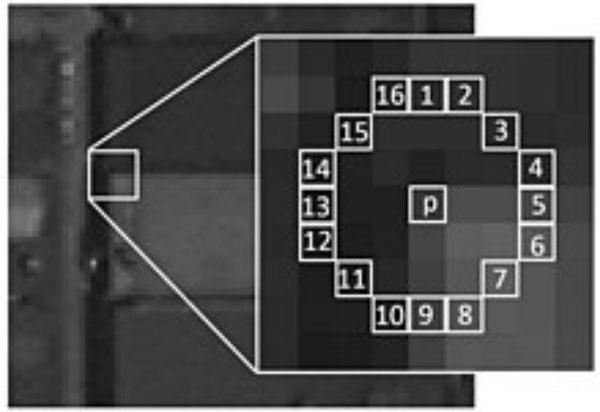
- Given a pixel $p$ in an array:
  - Compare the brightness of $p$: $I_p$ to surrounding 16 pixels that are in a small circle around $p$.
  - If more than N pixels are brighter than 1.2 $I_p$, or darker than 0.8 $I_p$, pixel $p$ will be selected as a FAST keypoint.
  - Apply non-maximal suppression after one-round detection.




N=12: FAST-12
N=11: FAST-11
N=9: FAST-9

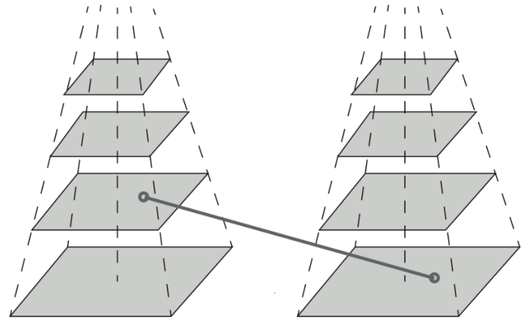
# Why NOT FAST

- No scale invariance
  - The radius of circle = 3 is fixed.




- No rotation invariance
  - FAST corner has NO direction.

- We need a better description of scale and rotation on FAST corners.
  - Scale: Generate Pyramid, detect corners on each level
  - Rotation: Intensity Centroid



# Intensity Centroid

- The moments of a patch B are defined as:

$$m_{pq} = \sum_{(x,y) \in B} x^p y^q I(x,y)$$

- The center of mass of the patch as centroid:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

- Construct a vector from the corner's center O to the centroid. The orientation of the patch B is then given by:

$$\theta = atan2(m_{01}, m_{10})$$

x

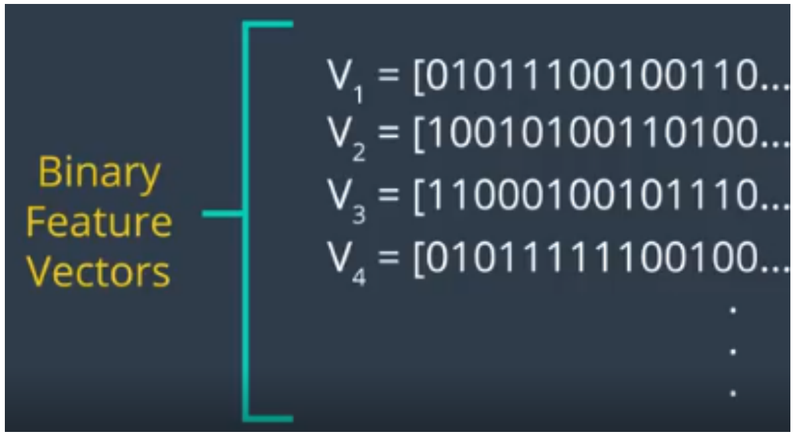
y

| | | | | |
|---|---|---|---|---|
| I(-2, -2) | I(-1, -2) | I(0, -2) | I(1, -2) | I(2, -2) |
| I(-2, -1) | I(-1, -1) | I(0, -1) | I(1, -1) | I(2, -1) |
| I(-2, 0) | I(-1, 0) | I(0, 0) | I(1, 0) | I(2, 0) |
| I(-2, 1) | I(-1, 1) | I(0, 1) | I(1, 1) | I(2, 1) |
| I(-2, 2) | I(-1, 2) | I(0, 2) | I(1, 2) | I(2, 2) |

Patch B

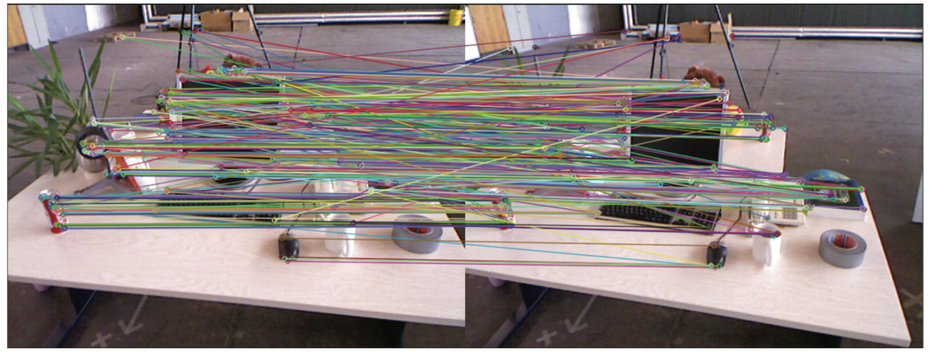
# BRIEF descriptor

- Binary vector only contains 0 and 1
- 128~512 bit long
- 1st pixel: Gaussian distribution centered around the keypoint with a strandard deviation or spread of $\sigma$.
- 2nd pixel: Gaussian distribution centered around the first pixel with a standard deviation or spread of $\sigma/2$.

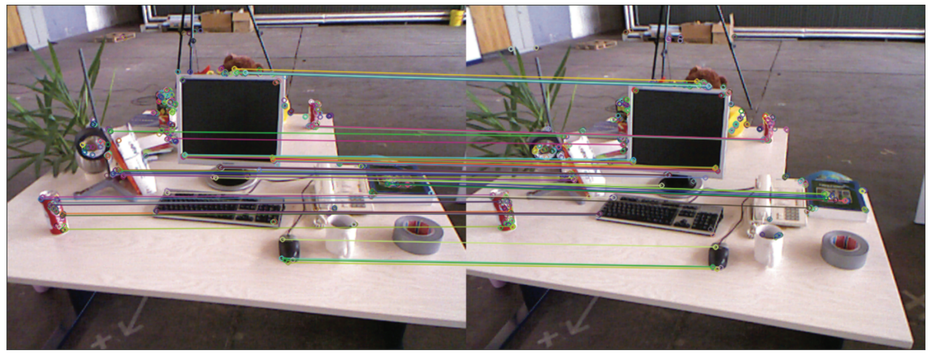



# ORB Matching

- Hamming distance of BRIEF descriptors
- Brute Force, FLANN, …
- Filter mismatches
  - If $dist < 2\, min_{dist}$ , keep
  - Otherwise, discard
- Obtain a set of matches of pixels: $\{p_l, p_r\}$



Before filter

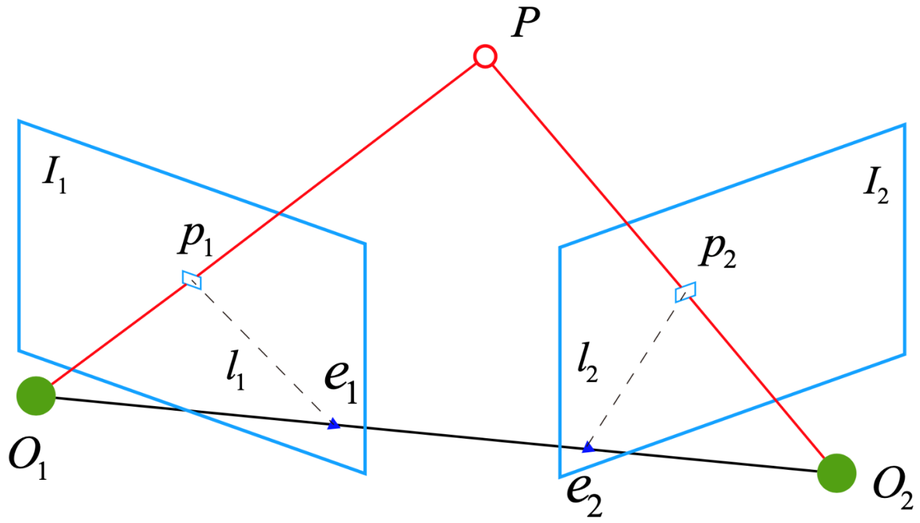


After filter

# Epipolar Geometry

- Known intrinsic camera matrix K
- $x_1 = K^{-1}p_1$ and $x_2 = K^{-1}p_2$
- Epipolar constraint:

$$\mathbf{E} = \mathbf{t}^{\wedge}\mathbf{R}, \quad \mathbf{F} = \mathbf{K}^{-\mathrm{T}}\mathbf{E}\mathbf{K}^{-1}, \quad \mathbf{x}_2^{\mathrm{T}}\mathbf{E}\mathbf{x}_1 = \mathbf{p}_2^{\mathrm{T}}\mathbf{F}\mathbf{p}_1 = 0$$

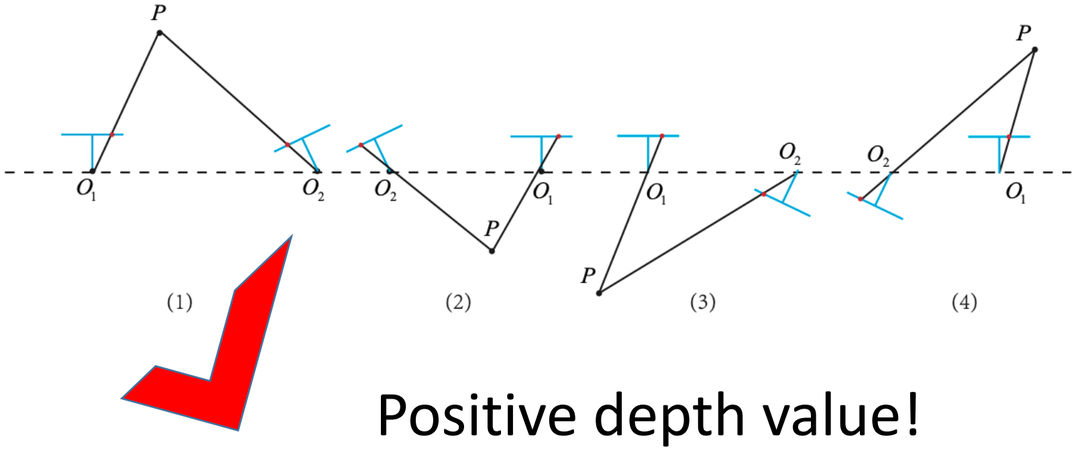



- Eight-Point Algorithm: solves E or F
- SVD: $\mathbf{E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$ , 2 solutions

$$\mathbf{t}_1^{\wedge} = \mathbf{U}\mathbf{R}_Z(\tfrac{\pi}{2})\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}, \quad \mathbf{R}_1 = \mathbf{U}\mathbf{R}_Z^{\mathrm{T}}(\tfrac{\pi}{2})\mathbf{V}^{\mathrm{T}}$$

$$\mathbf{t}_2^{\wedge} = \mathbf{U}\mathbf{R}_Z(-\tfrac{\pi}{2})\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}}, \quad \mathbf{R}_2 = \mathbf{U}\mathbf{R}_Z^{\mathrm{T}}(-\tfrac{\pi}{2})\mathbf{V}^{\mathrm{T}}$$

- The same for –E, also 2 solutions




Positive depth value!

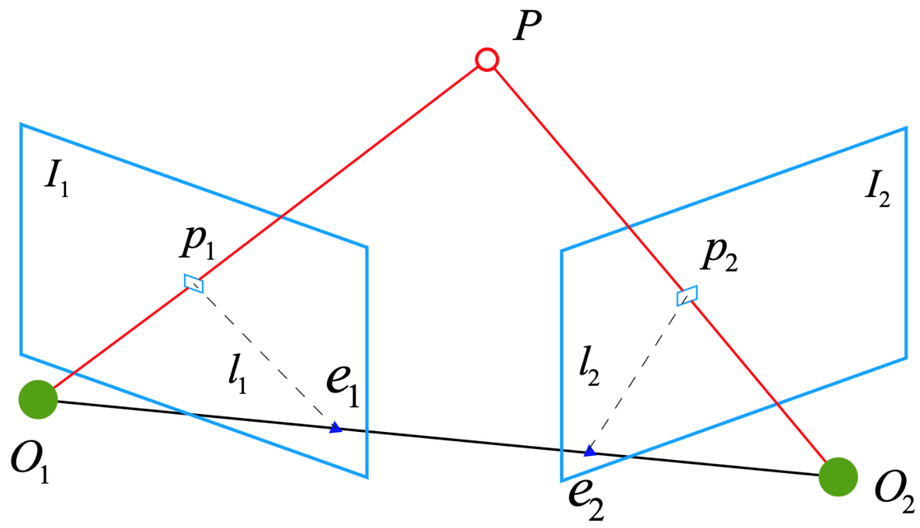
# Epipolar Geometry

- Epipolar constraint: $E = t^{\wedge}R, \quad x_2^T E x_1 = 0$

$$t^{\wedge} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

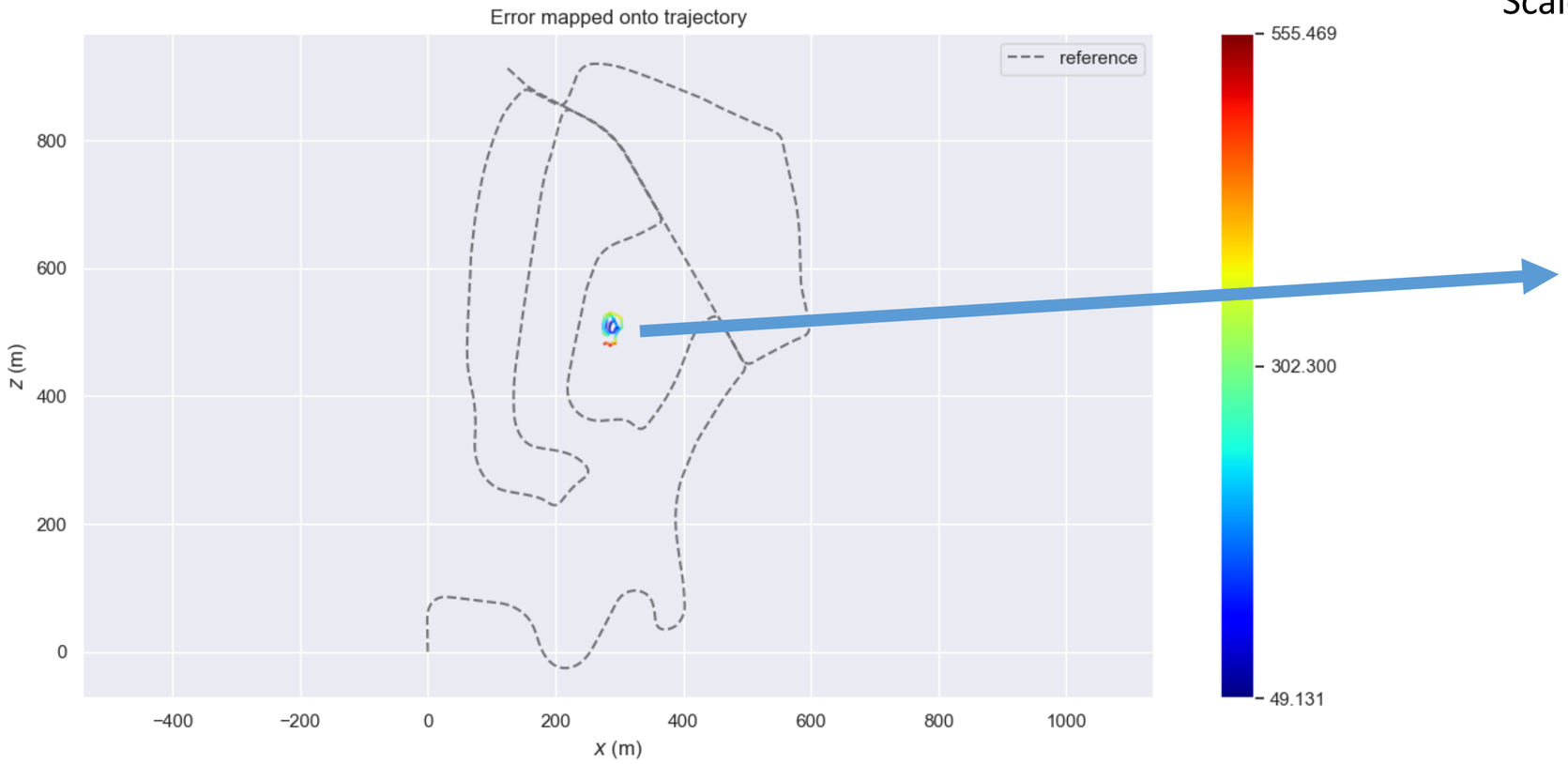
- Rotation matrix $R$ keeps self constraint
- Scale $t$ by any proportional constant, it still satisfies epipolar constraint.
- It causes scale ambiguity in monocular.

- Eg.: $t = 1.0$, 1.0 cm or 1.0 m?



# Scale Ambiguity

- Compare trajectory with groundtruth on KITTI with Monocular SLAM




Scale factor: 17.95

# Homography Matrix

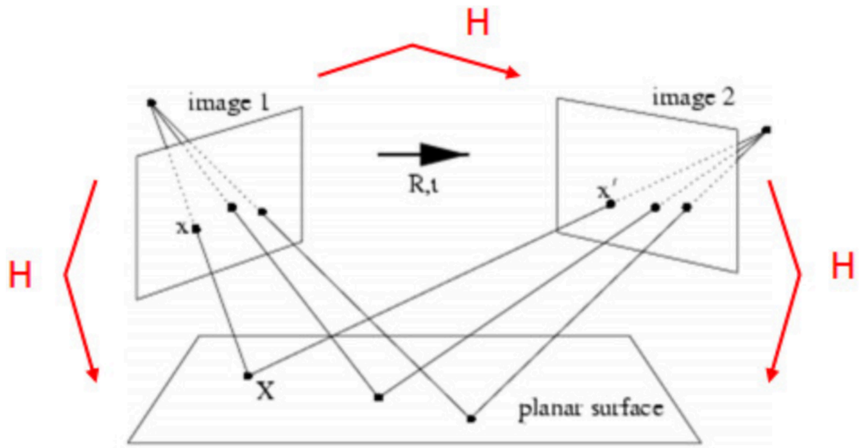
- Epipolar constraint: $E = t^{\wedge}R, \quad x_2^T E x_1 = 0$

$$t^{\wedge} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

- What if $t = 0$? Self-rotation
- $E = 0$, cannot solve $R$.

- Another estimation: Homography Matrix

- Assumption: all of the feature points in the scene falls on the same plane (such as walls, ground, etc.)

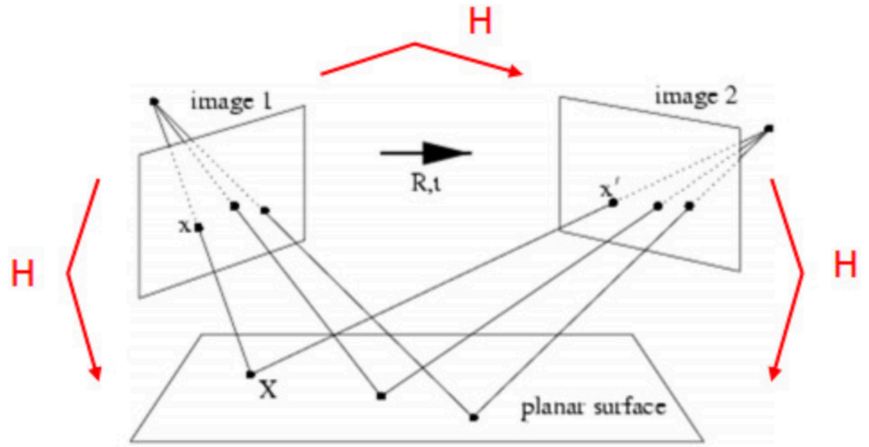


# Homography Matrix

- Assumption: all of the feature points $P$ in the scene falls on the same plane: $n^T P + d = 0$.

$$-\frac{\mathbf{n}^{\mathrm{T}}\mathbf{P}}{d} = 1$$

$$\begin{aligned}
\mathbf{p}_2 &= \mathbf{K}(\mathbf{R}\mathbf{P} + \mathbf{t}) \\
&= \mathbf{K}\left(\mathbf{R}\mathbf{P} + \mathbf{t}\cdot(-\frac{\mathbf{n}^{\mathrm{T}}\mathbf{P}}{d})\right) \\
&= \mathbf{K}\left(\mathbf{R} - \frac{\mathbf{t}\mathbf{n}^{\mathrm{T}}}{d}\right)\mathbf{P} \\
&= \mathbf{K}\left(\mathbf{R} - \frac{\mathbf{t}\mathbf{n}^{\mathrm{T}}}{d}\right)\mathbf{K}^{-1}\mathbf{p}_1.
\end{aligned}$$



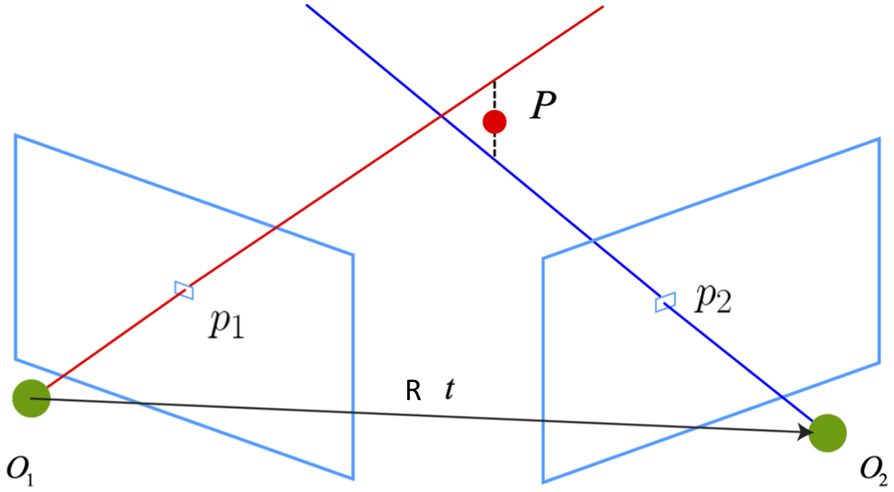

$$\mathbf{p}_2 = \mathbf{H}\mathbf{p}_1$$

# Triangulation

- Observe the same landmark at different locations
- Infer the distance of the landmark from the observed location

- **Input:**
  - Relative pose [R, t] between 2 cameras
  - Two corresponding pixels $p_1$ and $p_2$

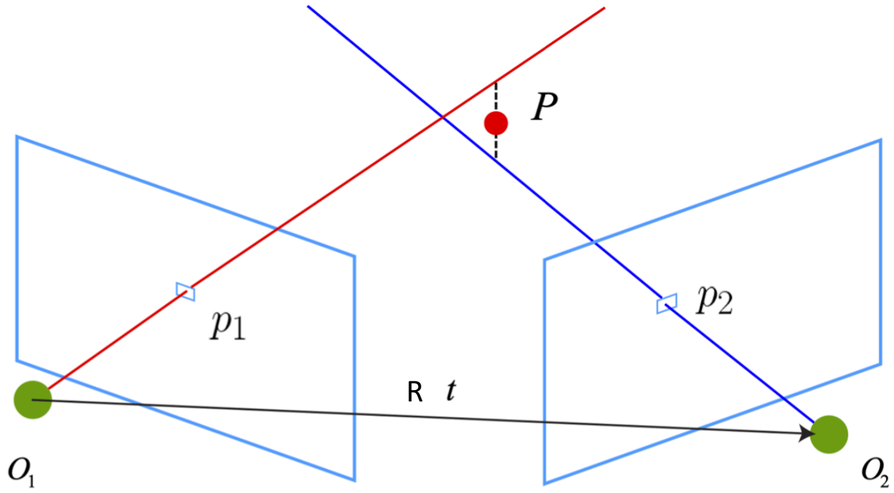
- **Output:**
  - Depth values of P in both cameras $s_1$ and $s_2$



# Triangulation

- Known intrinsic camera matrix K
- Normalized: $x_1 = K^{-1}p_1$ and $x_2 = K^{-1}p_2$
- $s_2 x_2 = s_1 R x_1 + t$
- $0 = s_2 \hat{x_2} x_2 = s_1 \hat{x_2} R x_1 + \hat{x_2} t$

Solve $s_1$




$$\mathbf{a}^{\wedge} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

$a^{\wedge}\, a$ means outer-product

Still scale ambiguity!

# Monocular Visual SLAM

- Limitations of pure visual SLAM:
  - Hard for localization when failing to detect sufficient feature points
  - Coordinate with respect to the first camera, not world coordinate
- Fusion with IMU:
  - Give a better pose estimation when tracking lost occurs in visual SLAM
  - Solve scale ambiguity
  - Align with world coordinate

# Outline

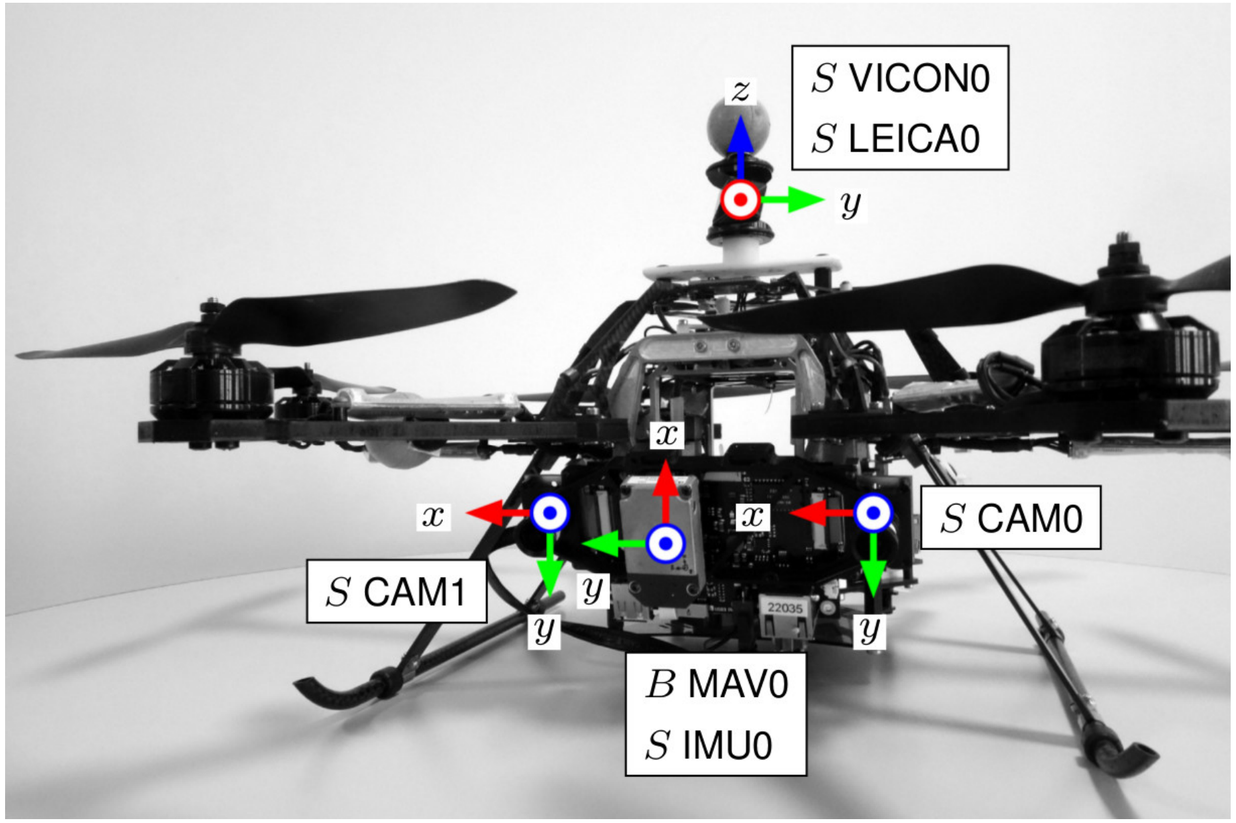
I. Visual Odometry: Monocular SLAM

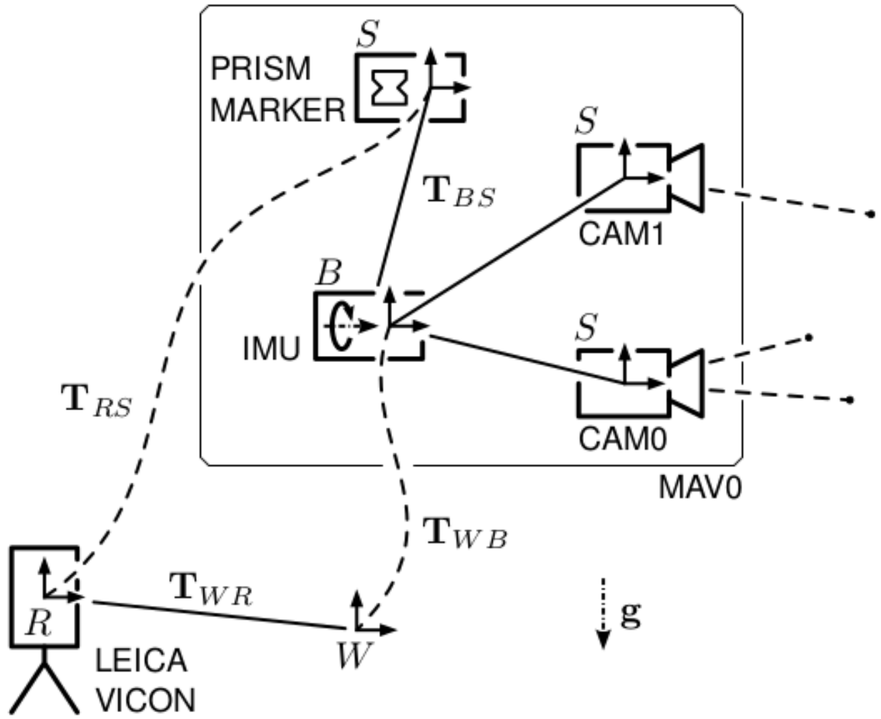
II. Inertial Measurement Unit and Pre-Integration

III. Visual-Inertial Alignment and Bundle Adjustment

IV. Learning Based Feature Extractor

# EuRoC: Camera-IMU Body



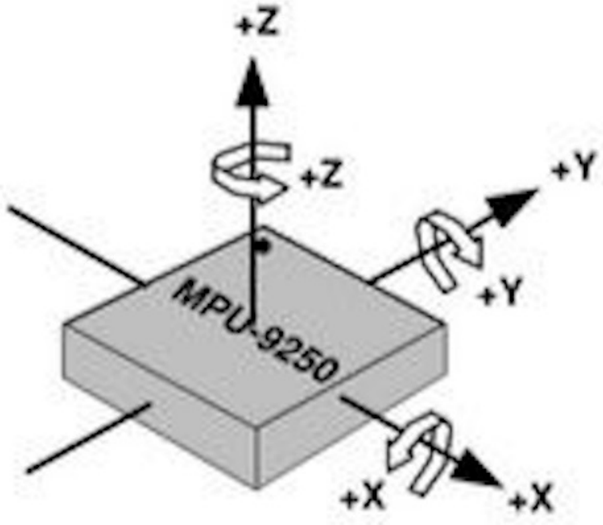



# IMU -- Inertial Navigation

- 6 D.o.F sensor model
  - Gyroscope: $\omega = (\omega_x, \omega_y, \omega_z)$
  - Accelerometer: $a = (a_x, a_y, a_z)$
- Types of sensor fusion
  - Loose fusion
  - Tight fusion

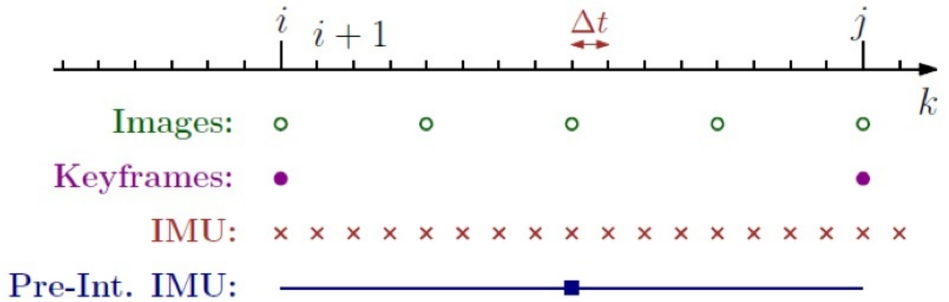
$$\hat{a}_t = a_t + b_{a_t} + R_w^t g^w + n_a$$

$$\hat{w}_t = w_t + b_{w_t} + n_w$$



# Loose Fusion

- Simple idea: Directly pre-integrate a physical trajectory(Position P, Velocity V, Rotation Q) based on angular velocity $\omega$ and acceleration $a$ .
- $[P, V, Q] = PreIntegration(\omega, a)$




- Often not work well in many scenarios (Why? )
  - Sensor Bias can never be ignored

# Loose Fusion

$$\hat{a}_t = a_t + b_{a_t} + R_w^t g^w + n_a$$

$$\hat{w}_t = w_t + b_{w_t} + n_w$$

- Gyroscope Error Model:
- $\omega = \tilde{\omega} - b_\omega - \eta_\omega$

$\omega$: true value

$\tilde{\omega}$: observed value

$b_\omega$: bias (random walk)

$\eta_\omega$: Gaussian Noise $\sim N(0, \sigma^2)$

Optimize $b_\omega$ :
-> Tight fusion

# Pre-integration

- In world coordinate: Position, Velocity, Rotation

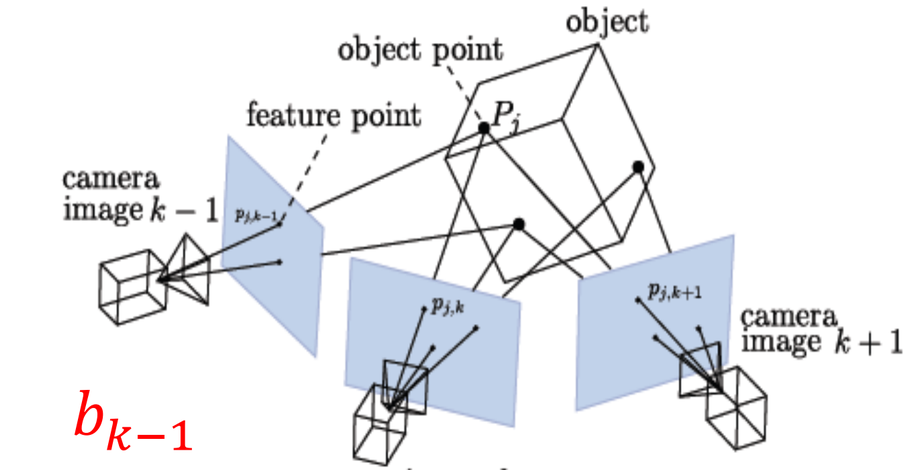
$$p^w_{b_{k+1}} = p^w_{b_k} + v^w_{b_k}\Delta t_k + \iint_{t\in[t_k,t_{k+1}]}(R^w_t(\hat{a}_t - b_{a_t} - n_a) - g^w)dt^2$$

$$v^w_{b_{k+1}} = v^w_{b_k} + \int_{t\in[t_k,t_{k+1}]}(R^w_t(\hat{a}_t - b_{a_t} - n_a) - g^w)dt$$

$$q^w_{b_{k+1}} = q^w_{b_k} \otimes \int_{t\in[t_k,t_{k+1}]} \frac{1}{2} q^{b_k}_t \otimes \begin{bmatrix} \hat{w}_t - b_{w_t} - n_w \\ 0 \end{bmatrix} dt$$

$$= q^w_{b_k} \otimes \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\Omega(\hat{w}_t - b_{w_t} - n_w) q^{b_k}_t dt$$

$$\Omega(\omega) = \begin{bmatrix} -[\omega]_\times & \omega \\ -\omega^T & 0 \end{bmatrix}, [\omega]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$



# Pre-integration

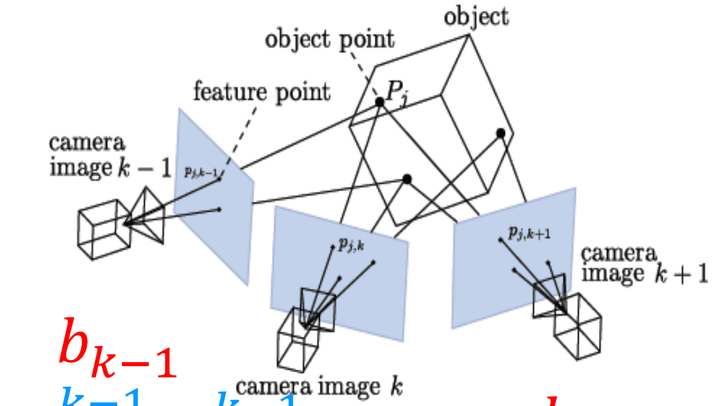
- Transform coordinate from world to $b_k$ : $R_w^{b_k}$

$$R_w^{b_k} p_{b_{k+1}}^w = R_w^{b_k}(p_{b_k}^w + v_{b_k}^w \Delta t_k - \frac{1}{2} g^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k}$$
$$R_w^{b_k} v_{b_{k+1}}^w = R_w^{b_k}(v_{b_k}^w - g^w \Delta t_k) + \beta_{b_{k+1}}^{b_k}$$
$$q_w^{b_k} q_{b_{k+1}}^w = \gamma_{b_{k+1}}^{b_k}$$

where

$$\alpha_{b_{k+1}}^{b_k} = \iint_{t \in [t_k, t_{k+1}]} R_t^{b_k}(\hat{a}_t - b_{a_t} - n_a) dt^2$$
$$\beta_{b_{k+1}}^{b_k} = \iint_{t \in [t_k, t_{k+1}]} R_t^{b_k}(\hat{a}_t - b_{a_t} - n_a) dt$$
$$\gamma_{b_{k+1}}^{b_k} = \iint_{t \in [t_k, t_{k+1}]} \frac{1}{2} \Omega(\hat{w}_t - b_{w_t} - n_w) \gamma_{b_k}^t dt$$

avoids duplicated propagation of IMU states

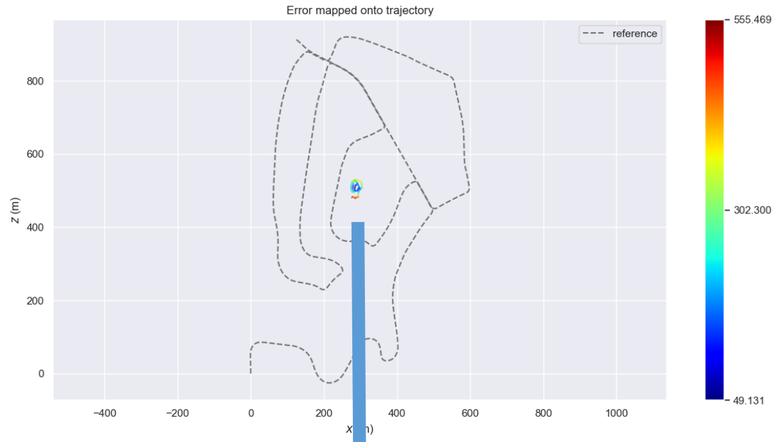



First order approximation

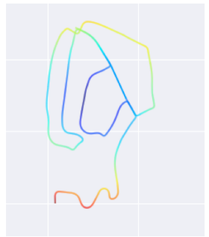
$$\alpha_{b_{k+1}}^{b_k} = \hat{\alpha}_{b_{k+1}}^{b_k} + J_{b_a}^{\alpha} \delta b_a + J_{b_w}^{\alpha} \delta b_w$$
$$\beta_{b_{k+1}}^{b_k} = \hat{\beta}_{b_{k+1}}^{b_k} + J_{b_a}^{\beta} \delta b_a + J_{b_w}^{\beta} \delta b_w$$
$$\gamma_{b_{k+1}}^{b_k} = \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix}$$

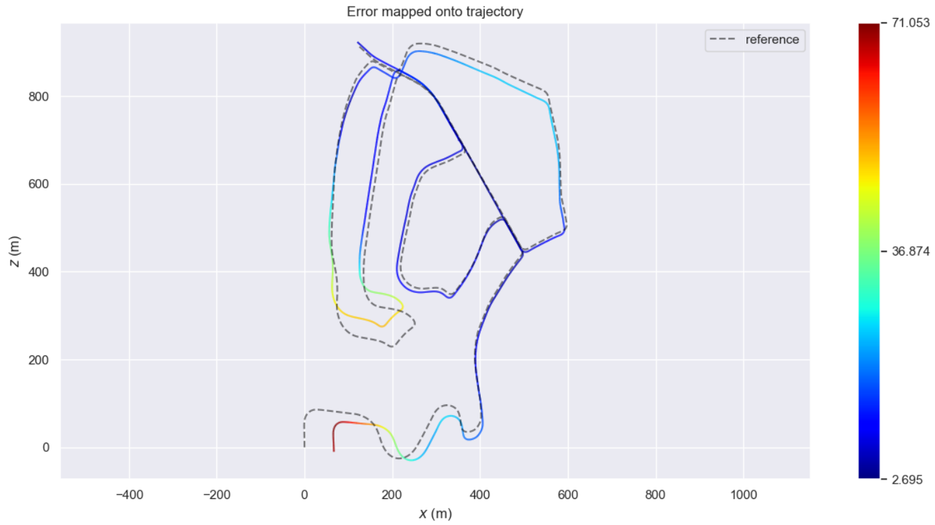
# Tight Fusion

- Recall Scale Ambiguity problem in Monocular Visual SLAM
- How we solve scale ambiguity if we add IMU pre-integration?




Scale factor: 17.95






Scale consistency

# Outline

I. Visual Odometry: Monocular SLAM

II. Inertial Measurement Unit and Pre-Integration

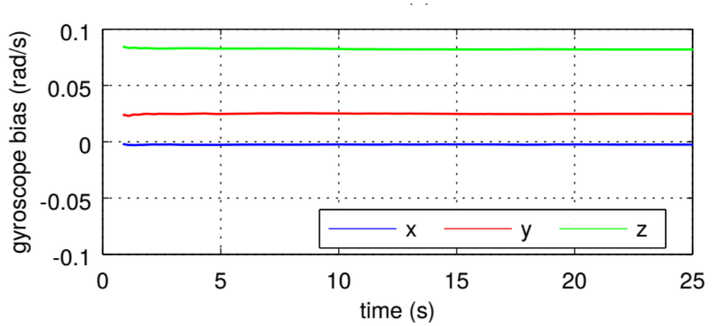
III. Visual-Inertial Alignment and Bundle Adjustment

IV. Learning Based Feature Extractor

V. Conclusions and future directions

# Tight Fusion

- Observation: During a short period, sensor bias $b_\omega$ almost keeps unchanged.

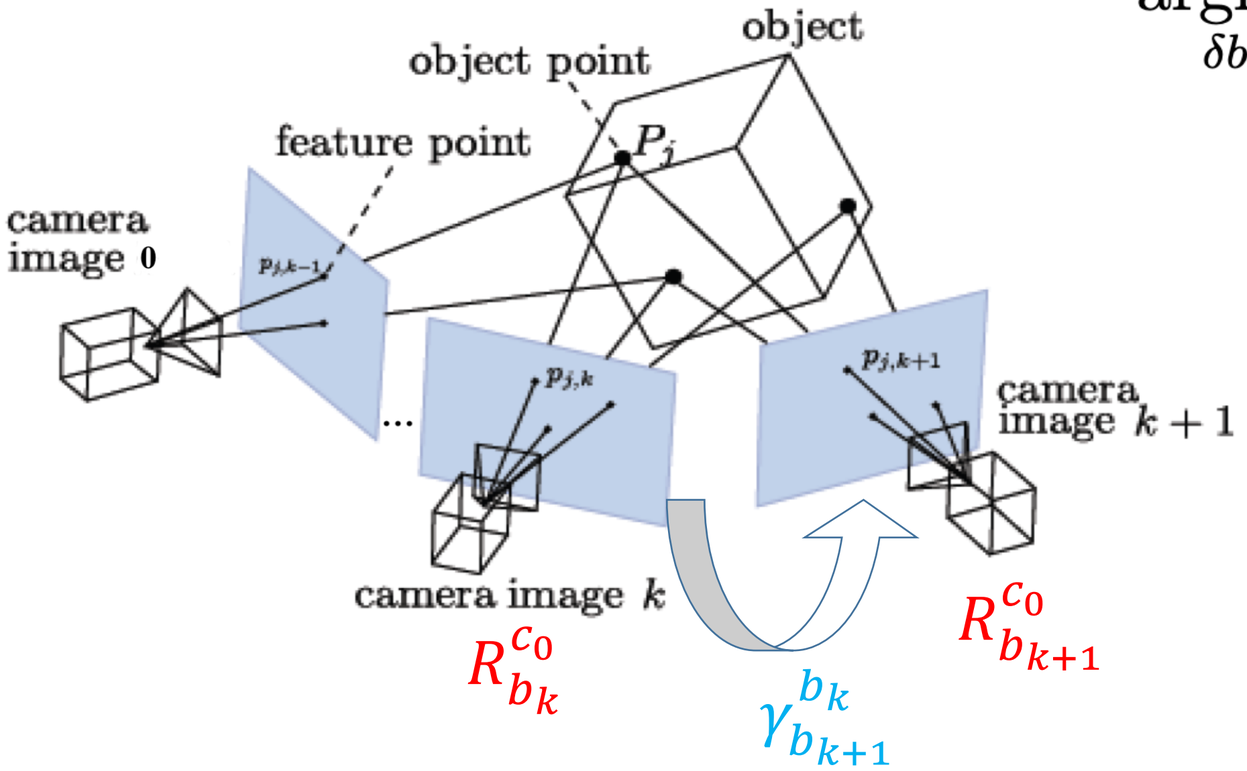



- Visual-Inertial Initialization terms:
  - gyroscope bias: $b_\omega$
  - velocity: $v_{b_k}^{b_k}$
  - scale: $s$
  - gravity vector in first camera: $g^{c_0}$

Ignore bias $b_a$

# Gyroscope bias

- Input: $R_{b_k}^{c_0}, R_{b_{k+1}}^{c_0}$ from Visual Monocular SLAM, $\gamma_{b_{k+1}}^{b_k}$ from IMU pre-integration between $b_k$ and $b_{k+1}$
- Output: gyroscope bias $\delta b_w$



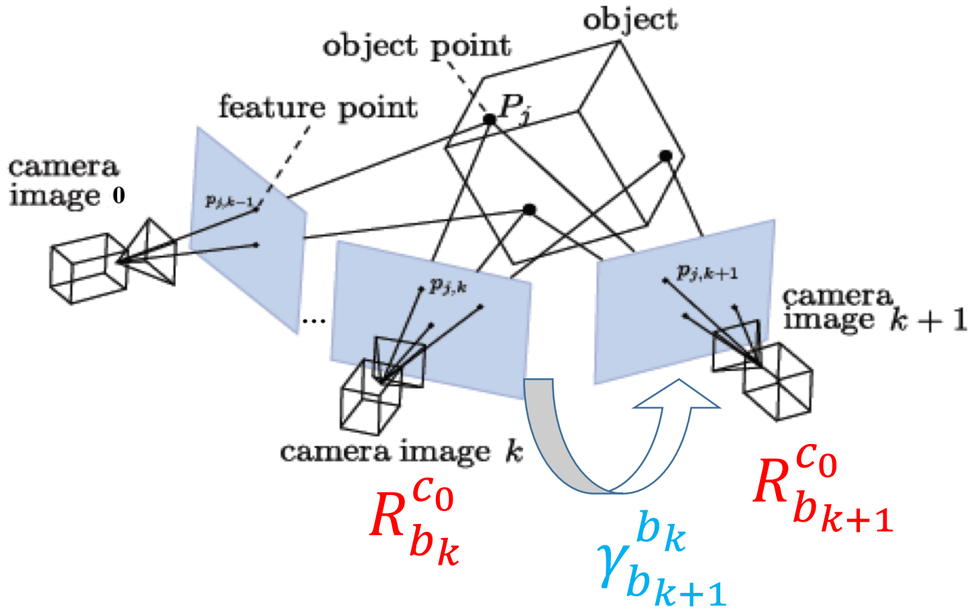

$$\underset{\delta b_w}{\operatorname{argmin}} \sum_{k=1}^{N-1} \| \, {q_{b_{k+1}}^{c_0}}^{-1} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} \, \|^2$$

$$\gamma_{b_{k+1}}^{b_k} = \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix}$$

# Gyroscope bias

$$\underset{\delta b_w}{\operatorname{argmin}} \sum_{k=1}^{N-1} \| \, {q_{b_{k+1}}^{c_0}}^{-1} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} \, \|^2$$

w.r.t $$\gamma_{b_{k+1}}^{b_k} = \hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix}$$




$${q_{b_{k+1}}^{c_0}}^{-1} \otimes q_{b_k}^{c_0} \otimes \gamma_{b_{k+1}}^{b_k} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \longrightarrow I_{3\times3}$$

$$\gamma_{b_{k+1}}^{b_k} = {q_{b_k}^{c_0}}^{-1} \otimes q_{b_{k+1}}^{c_0} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

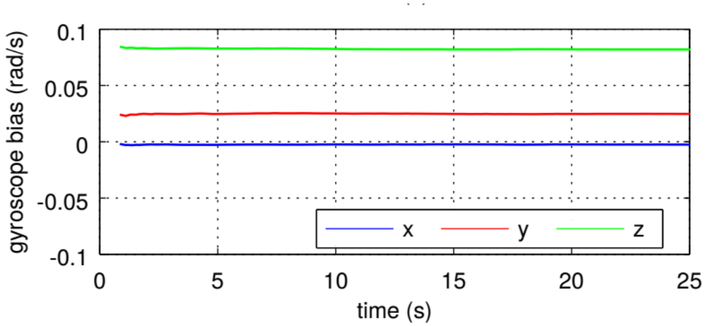
$$\hat{\gamma}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_w \end{bmatrix} = {q_{b_k}^{c_0}}^{-1} \otimes q_{b_{k+1}}^{c_0} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$J_{b_w}^{\gamma} \delta b_w = 2({\hat{\gamma}_{b_{k+1}}^{b_k}}^{-1} \otimes {q_{b_k}^{c_0}}^{-1} \otimes q_{b_{k+1}}^{c_0})_{vec}$$

$${J_{b_w}^{\gamma}}^{T} J_{b_w}^{\gamma} \delta b_w = 2 {J_{b_w}^{\gamma}}^{T} ({\hat{r}_{b_{k+1}}^{b_k}}^{-1} \otimes {q_{b_k}^{c_0}}^{-1} \otimes q_{b_{k+1}}^{c_0})_{vec}$$

# Tight Fusion

- Observation: During a short period, sensor bias $b_\omega$ almost keeps unchanged.




- Visual-Inertial Initialization terms:
  - gyroscope bias: $b_\omega$
  - velocity: $v_{b_k}^{b_k}$
  - scale: $s$
  - gravity vector in first camera: $g^{c_0}$

# Velocity, Scale, Gravity Vector

$$\mathcal{X}_I^{3(n+1)+3+1} = [v_{b_0}^{b_0}, v_{b_1}^{b_1}, ..., v_{b_n}^{b_n}, g^{c_0}, s]$$

$$R_w^{b_k} p_{b_{k+1}}^w = R_w^{b_k}(p_{b_k}^w + v_{b_k}^w \Delta t_k - \frac{1}{2} g^w \Delta t_k^2) + \alpha_{b_{k+1}}^{b_k}$$

$$R_w^{b_k} v_{b_{k+1}}^w = R_w^{b_k}(v_{b_k}^w - g^w \Delta t_k) + \beta_{b_{k+1}}^{b_k}$$

Replace w with $c_0$

$$\alpha_{b_{k+1}}^{b_k} = R_{c_0}^{b_k}(s p_{b_{k+1}}^{c_0} - s p_{b_k}^{c_0} + \frac{1}{2} g^{c_0} \Delta t_k^2 - R_{b_k}^{c_0} v_{b_k}^{b_k} \Delta t_k)$$

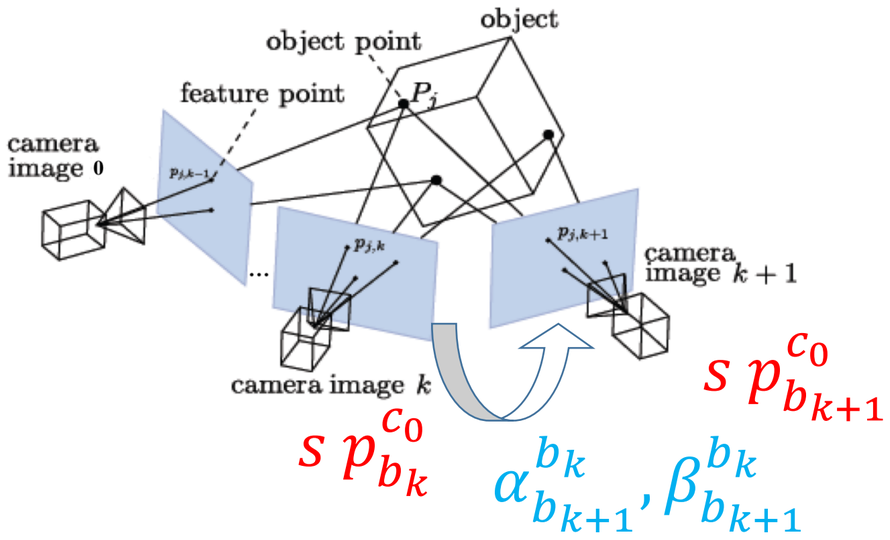
$$\beta_{b_{k+1}}^{b_k} = R_{c_0}^{b_k}(R_{b_{k+1}}^{c_0} v_{b_{k+1}}^{b_{k+1}} + g^{c_0} \Delta t_k - R_{b_k}^{c_0} v_{b_k}^{b_k})$$

$$s p_{b_k}^{c_0} = s p_{c_k}^{c_0} - R_{b_k}^{c_0} p_c^b$$

$$R_{c_0}^{b_k}(p_{c_{k+1}}^{c_0} - p_{c_k}^{c_0}) s - v_{b_k}^{b_k} \Delta t_k + \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 g^{c_0} = \alpha_{b_{k+1}}^{b_k} + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b - p_c^b$$

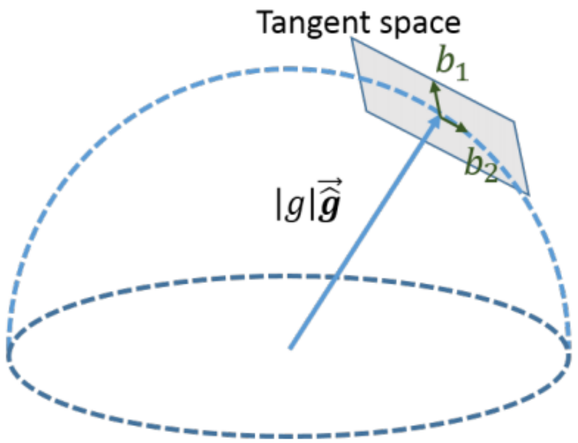
$$R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} v_{b_{k+1}}^{b_{k+1}} + R_{c_0}^{b_k} \Delta t_k g^{c_0} - R_{c_0}^{b_k} R_{b_k}^{c_0} v_{b_k}^{b_k} = \beta_{b_{k+1}}^{b_k}$$

$$\begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2} R_{c_0}^{b_k} \Delta t_k^2 & R_{c_0}^{b_k}(p_{c_{k+1}}^{c_0} - p_{c_k}^{c_0}) \\ -I & R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k} \Delta t_k & 0 \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}} \\ g^{c_0} \\ s \end{bmatrix} = \begin{bmatrix} \alpha_{b_{k+1}}^{b_k} + R_{c_0}^{b_k} R_{b_{k+1}}^{c_0} p_c^b - p_c^b \\ \beta_{b_{k+1}}^{b_k} \end{bmatrix}$$



# Gravity Refinement

- We have estimated gravity vector in the first frame: $g_{c_0}$
- $\|g\|$ = 9.8 is an inner constraint, thus $g_{c_0}$ needs refinement.

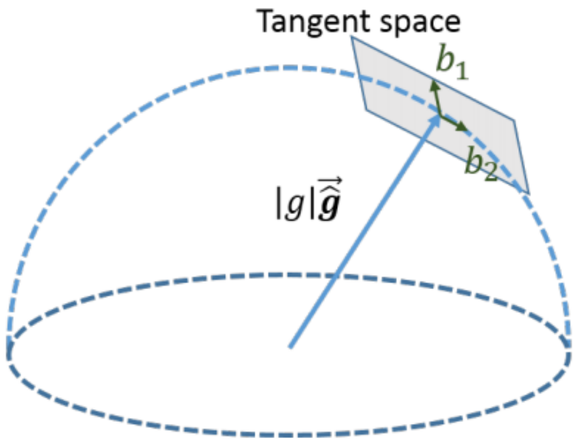



- find two orthogonal basis on spanning tangent space

$$\hat{g} = ||g||\bar{\hat{g}} + w_1 b_1 + w_2 b_2 = ||g||\bar{\hat{g}} + \vec{b}^{3\times 2} w^{2\times 1}$$

# Gravity Refinement

$$\hat{g} = ||g||\bar{\hat{g}} + w_1 b_1 + w_2 b_2 = ||g||\bar{\hat{g}} + \vec{b}^{3\times 2} w^{2\times 1}$$




Substitute $g_{c_0}$

$$\begin{bmatrix} -I\Delta t_k & 0 & \frac{1}{2}R_{c_0}^{b_k}\Delta t_k^2 & R_{c_0}^{b_k}(p_{c_{k+1}}^{c_0} - p_{c_k}^{c_0}) \\ -I & R_{c_0}^{b_k}R_{b_{k+1}}^{c_0} & R_{c_0}^{b_k}\Delta t_k & 0 \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}} \\ g^{c_0} \\ s \end{bmatrix} = \begin{bmatrix} \alpha_{b_{k+1}}^{b_k} + R_{c_0}^{b_k}R_{b_{k+1}}^{c_0}p_c^b - p_c^b \\ \beta_{b_{k+1}}^{b_k} \end{bmatrix}$$

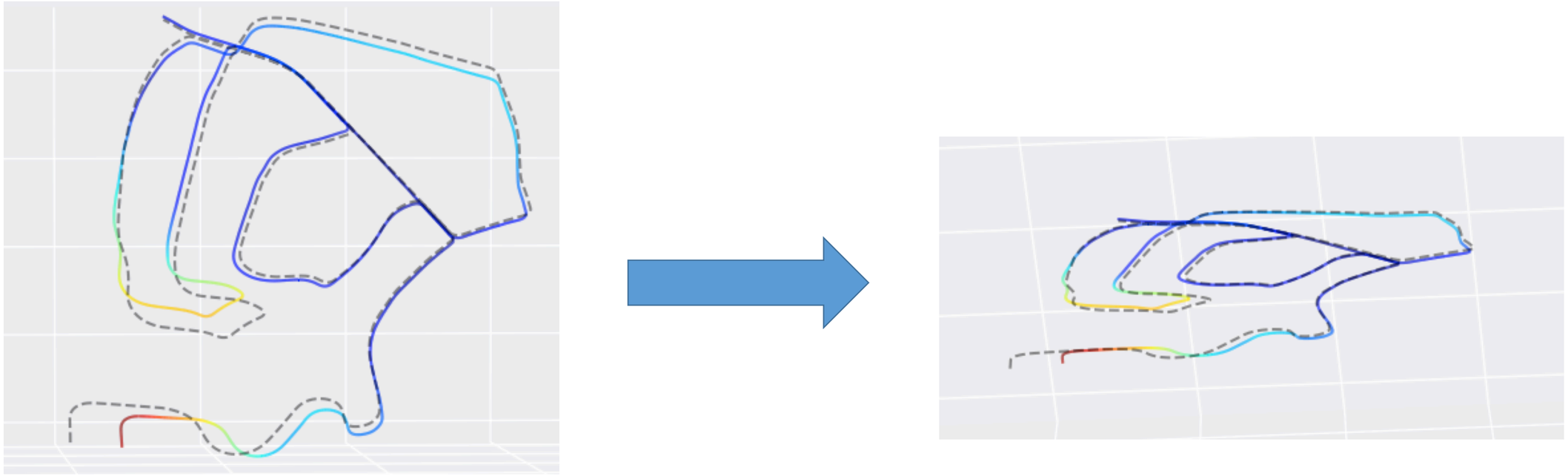
refine $g$ until converge

$$\begin{bmatrix} -\mathbf{I}\Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}_{c_0}^{b_k}\Delta t_k^2\mathbf{b} & \mathbf{R}_{c_0}^{b_k}(\overline{\mathbf{p}}_{b_{k+1}}^{c_0} - \overline{\mathbf{p}}_{b_k}^{c_0}) \\ -\mathbf{I} & \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0} & \mathbf{R}_{c_0}^{b_k}\Delta t_k\mathbf{b} & \mathbf{0} \end{bmatrix} \begin{bmatrix} v_{b_k}^{b_k} \\ v_{b_{k+1}}^{b_{k+1}} \\ \omega \\ s \end{bmatrix} = \begin{bmatrix} \hat{\alpha}_{b_{k+1}}^{b_k} - \mathbf{p}_c^b + \mathbf{R}_{c_0}^{b_k}\mathbf{R}_{b_{k+1}}^{c_0}\mathbf{p}_c^b - \frac{1}{2}\mathbf{R}_{c_0}^{b_k}\Delta t_k^2 \cdot g \cdot \bar{\hat{\mathbf{g}}} \\ \hat{\beta}_{b_{k+1}}^{b_k} - \mathbf{R}_{c_0}^{b_k}\Delta t_k \cdot g \cdot \bar{\hat{\mathbf{g}}} \end{bmatrix}$$

# Gravity Refinement

Refined g: $\hat{g} = ||g||\bar{\hat{g}} + w_1 b_1 + w_2 b_2 = ||g||\bar{\hat{g}} + \vec{b}^{3\times 2} w^{2\times 1}$

Align g with [0, 0, -1] in world coordinate to obtain the rotation between $c_0$ and $w$.



# Bundle Adjustment

Now we have got all camera poses + 3D landmark mapping

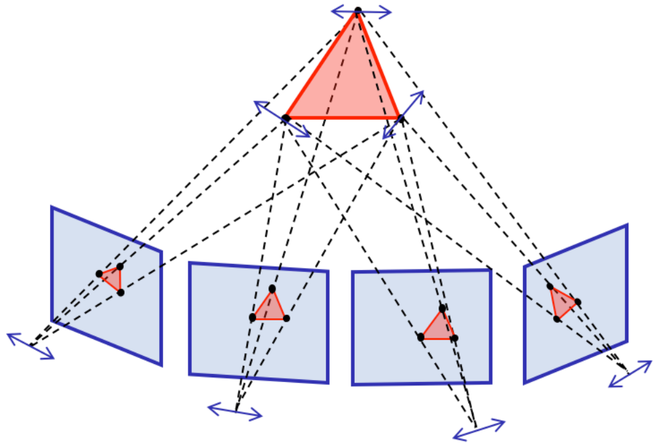
To avoid accumulation of error drift, we need optimization based on some constraints.

Input: 3D landmarks, $\{p_j \mid j = 1, \dots, N\}$

camera poses, $\{c_i \mid i = 1, \dots, M\}$

# Bundle Adjustment

Non-linear optimization problem

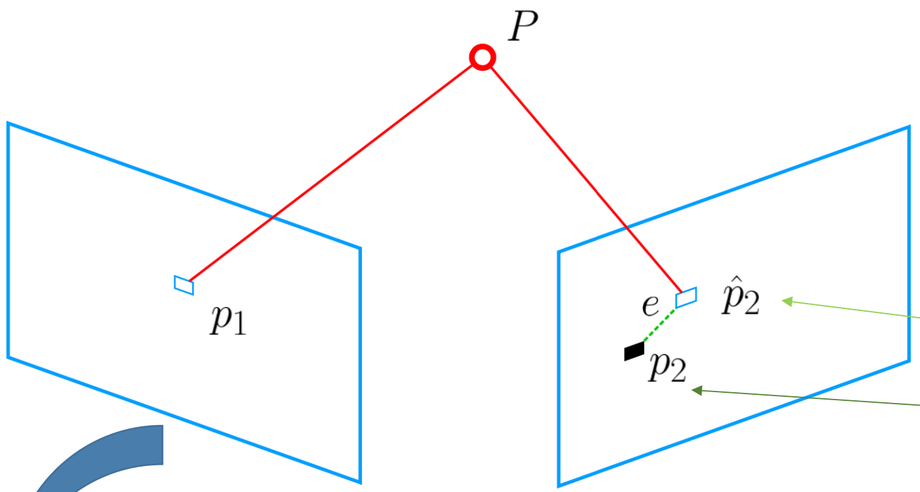


Input: 3D landmarks, $\{p_j \mid j = 1, \ldots, N\}$

camera poses, $\{c_i \mid i = 1, \ldots, M\}$

Constraint: Reprojection Error

# Reprojection Error

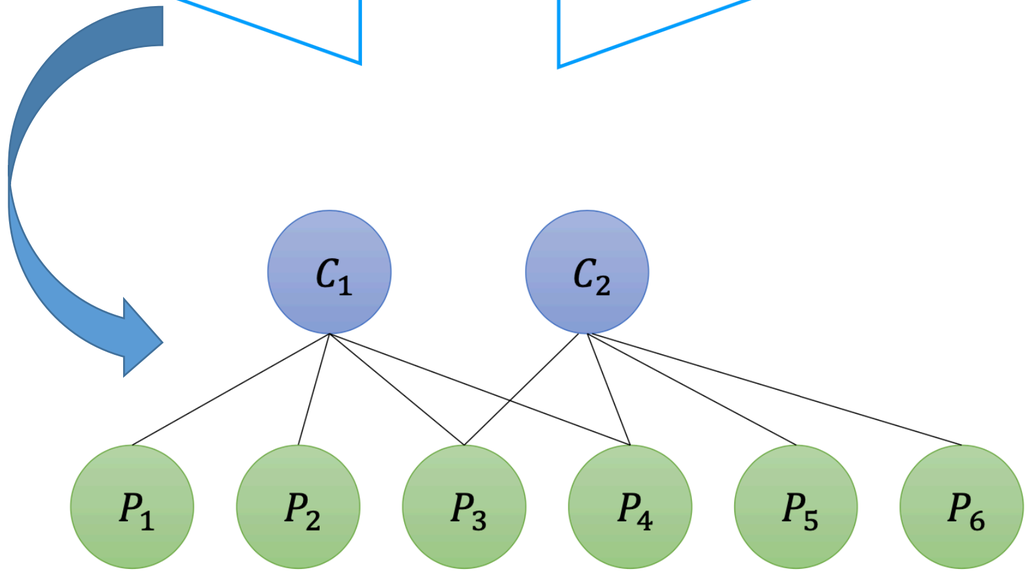



Input: 3D landmarks, $\{p_j \mid j = 1, \ldots, N\}$

camera poses, $\{\xi_i \in SE(3) \mid i = 1, \ldots, M\}$

Constraint: Reprojection Error

$$e_{ij} = z_{ij} - \pi(\xi_i, \mathbf{p}_j)$$



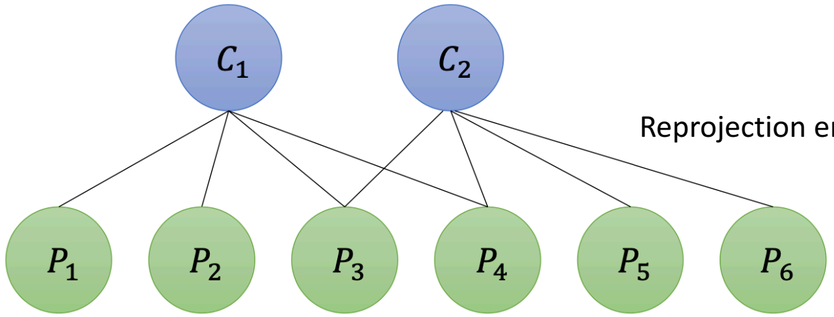

Combined error:

$$\mathbf{E} = \frac{1}{2}\sum_i \sum_j \| e_{ij} \|^2 = \frac{1}{2}\sum_i \sum_j \| z_{ij} - \pi(\xi_i, \mathbf{p}_j) \|^2$$

# General Graph Optimization

Camera poses vertex:

3D landmarks vertex:



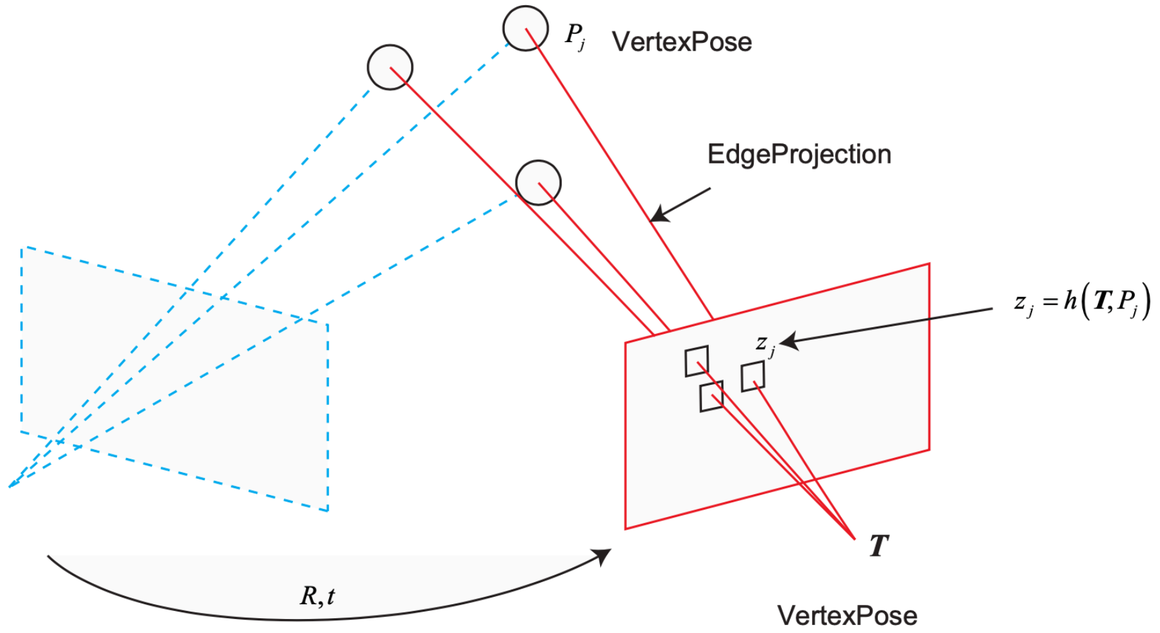

Reprojection error edge

$$\theta = (\xi_1, \ldots, \xi_m, \mathbf{p}_1, \ldots, \mathbf{p}_n)^T$$

$$\xi_i \in SE(3), 1 \leq i \leq m$$

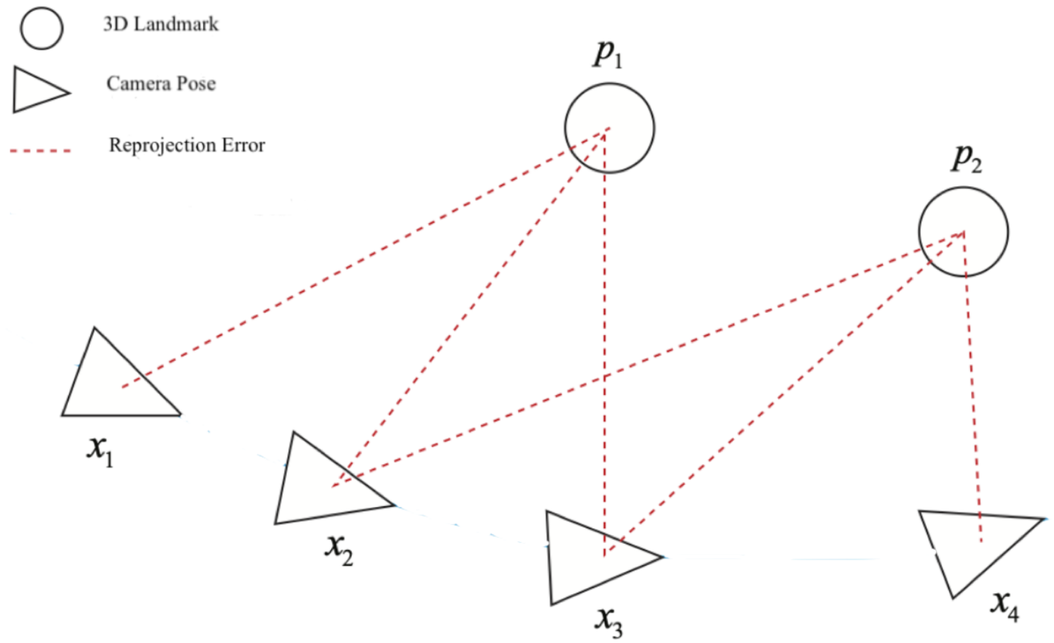
$$\mathbf{p}_j \in \mathbb{R}^3, 1 \leq j \leq n$$

Minimize the energy of graph:
Sum of all edges

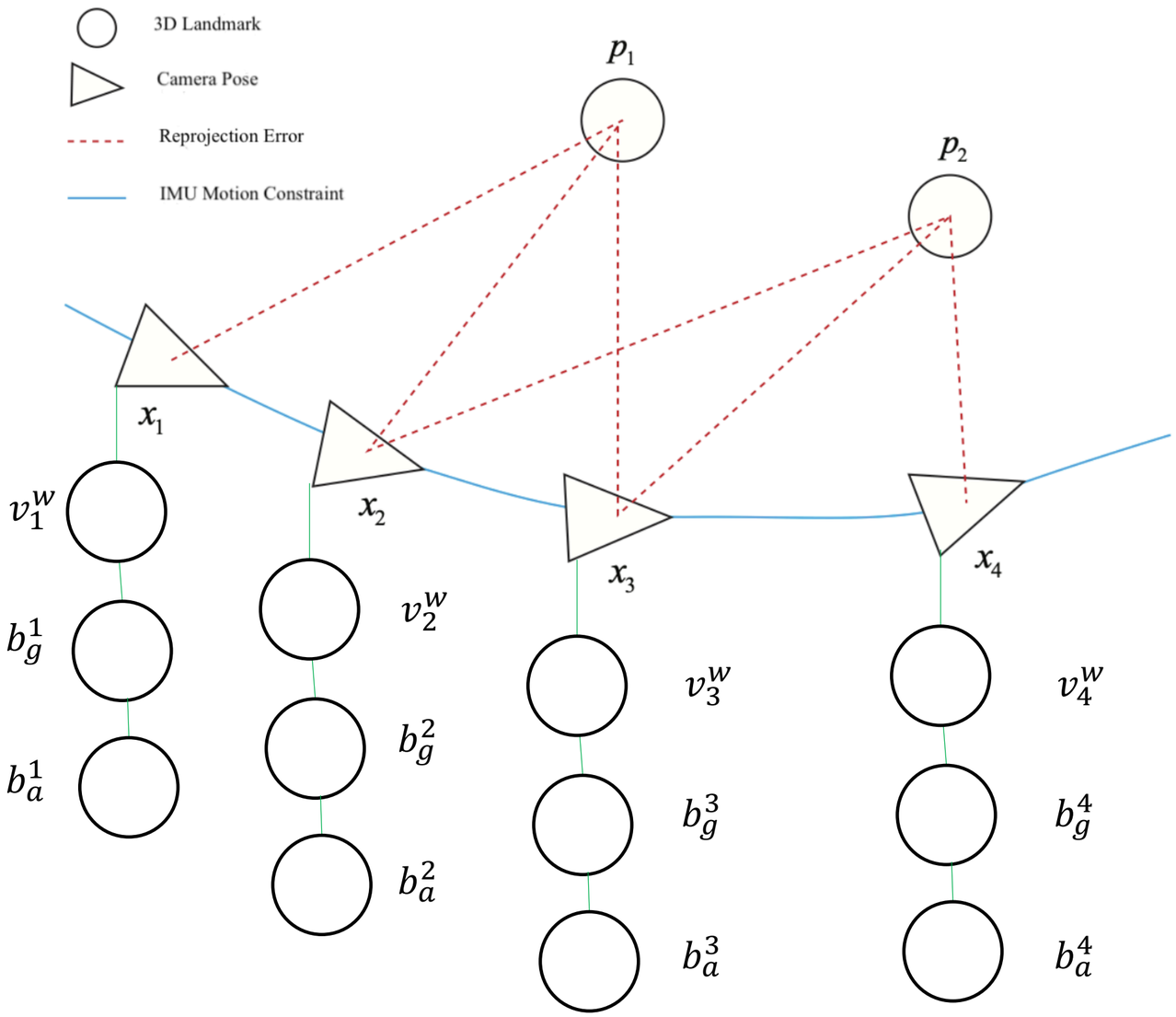


# Solve VIO BA

How to do BA when fuse IMU?



# Solve VIO BA




## Optimization Terms:

$$\theta = [\theta_c, \theta_p]$$

$$\theta_c = \{\mathbf{R}_{b_i}^w, \mathbf{p}_{b_i}^w, \mathbf{v}_{b_i}^w, \mathbf{b}_g^i, \mathbf{b}_a^i\}$$

$$\theta_p = \{\mathbf{p}_j\}_{1 \le j \le n}$$

$$\mathbf{E}_{IMU}(i, i+1) = \rho_h([\mathbf{e}_R^T \mathbf{e}_v^T \mathbf{e}_p^T]\Omega_I[\mathbf{e}_R^T \mathbf{e}_v^T \mathbf{e}_p^T]^T) + \rho_h(\mathbf{e}_b^T \Omega_R \mathbf{e}_b)$$

# Outline

I. Visual Odometry: Monocular SLAM

II. Inertial Measurement Unit and Pre-Integration

III. Visual-Inertial Alignment and Bundle Adjustment

IV. Learning Based Feature Extractor

# Extractor Framework

Structure of network used for feature extraction: including detector and descriptor

H x W

Conv layer 1
Conv layer 2
…
Conv layer N

H x W Keypoint map

256 x H x W descriptor map

# Extractor Framework

- Drawbacks [SuperPoint, 2018]:
  - Hard for real-time SLAM – calculation load for float descriptor
  - Work well in visual tracking, but not pose estimation in SLAM

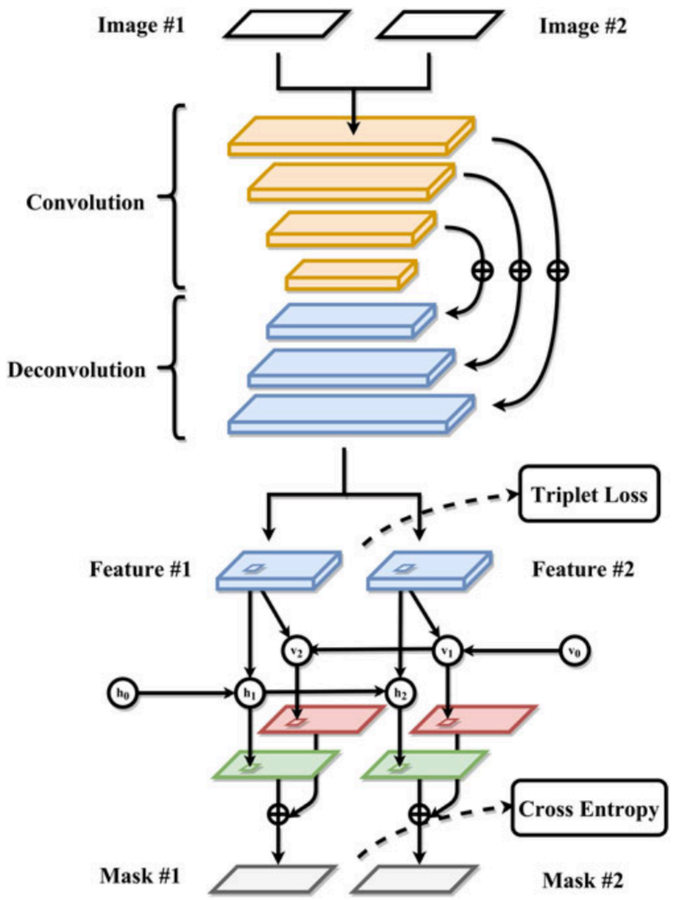
H x W

Conv layer 1
Conv layer 2
…
Conv layer N

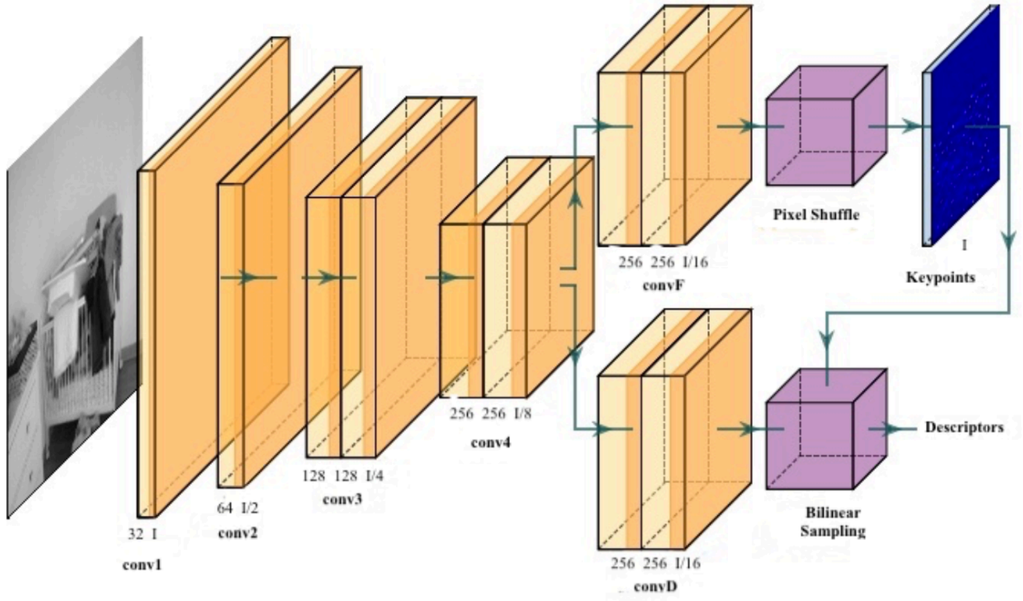
H x W Keypoint map

256 x H x W descriptor map

# Pairwise Extractor Framework

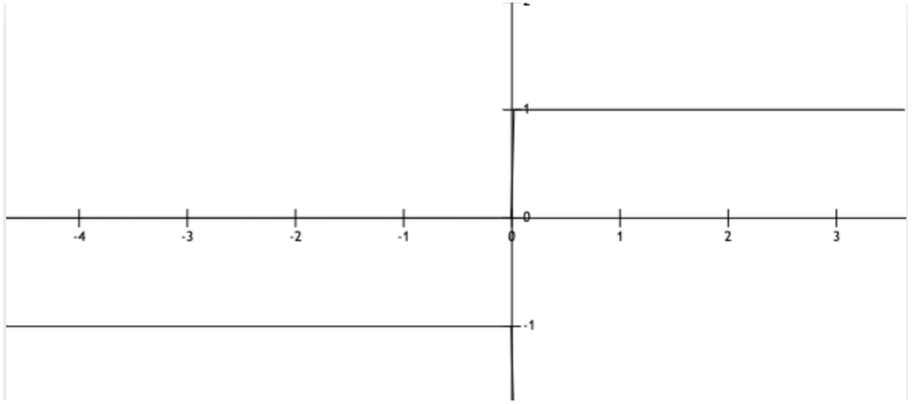



- Geometric Correspondence Network (GCN)
  - Use FCN with ResNet-50 as backbone: dense feature extraction
  - Input image pair rather than a single image while training
  - Use pose groundtruth between image pairs
  - Binary activation layer on descriptor extraction

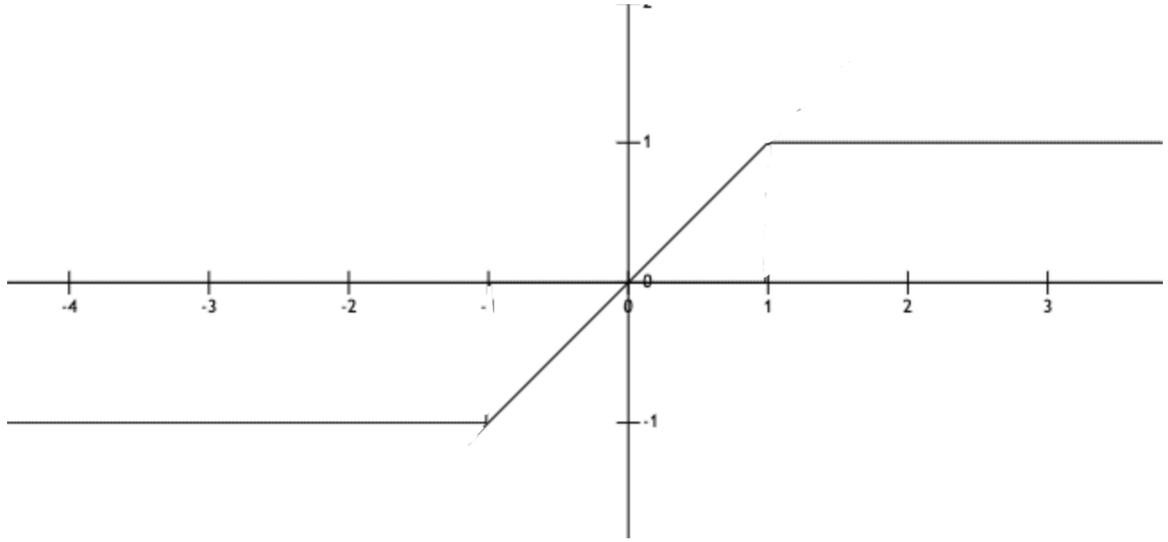


# Binary Activation

- Similar to BRIEF descriptor
  - 256 bits, only contains 0 and 1
  - Hamming distance
- Forward: $$\mathbf{b}(\mathbf{x}) = sign(\mathbf{f}(\mathbf{x})) = \begin{cases} +1 & \mathbf{f}(\mathbf{x}) \geq 0 \\ -1 & otherwise \end{cases}$$




- Backward: $$\frac{\partial \mathbf{b}}{\partial \mathbf{f}} = \frac{\partial L_{metric}}{\partial \mathbf{b}} \mathbf{1}_{|\mathbf{f} \leq 1|}$$



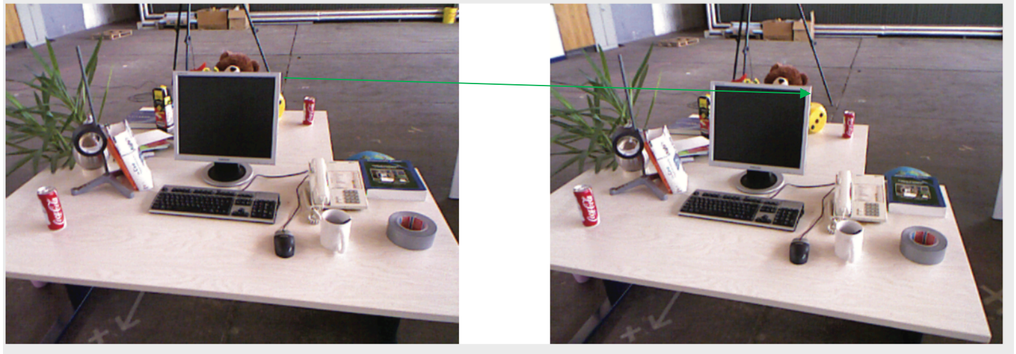
# Pairwise Extractor Framework

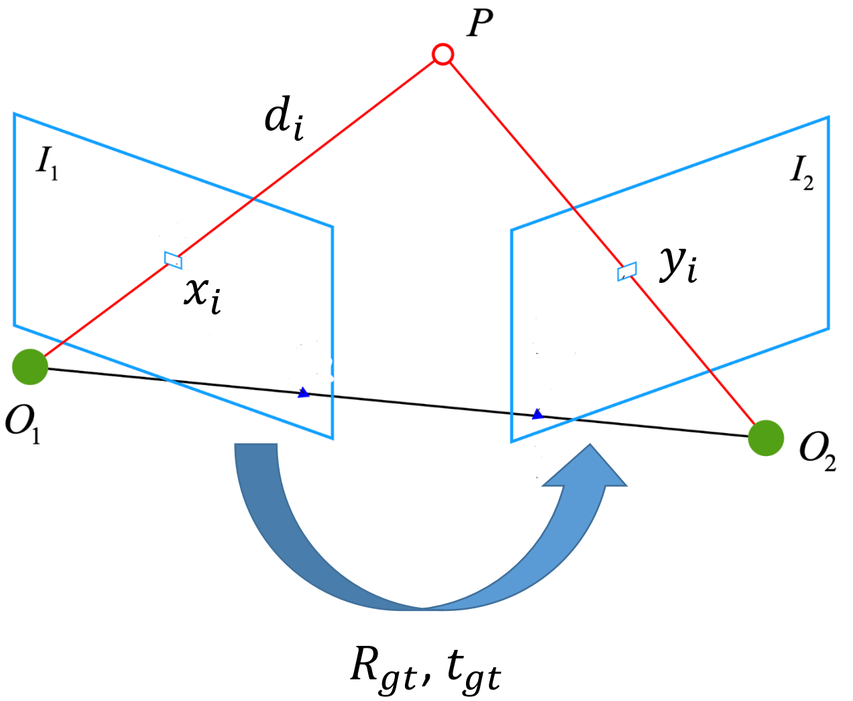
- In left image: $x_i = [u_i, v_i]$, depth $d_i$
- In right image: $y_i = [u_i', v_i']$
- 2D-3D transformation:

$$\pi(\mathbf{x_i}, d_i) = \left[\frac{d_i(u_i - c_x)}{f_x}, \frac{d_i(v_i - c_y)}{f_y}, d_i\right]^T$$

- Warping function:

$$y_i = \pi^{-1}(R_{gt} \cdot \pi(x_i, d_i) + t_{gt})$$

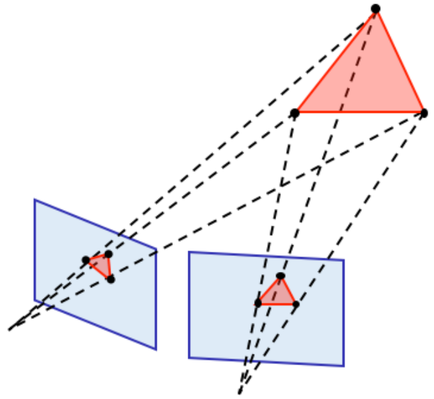


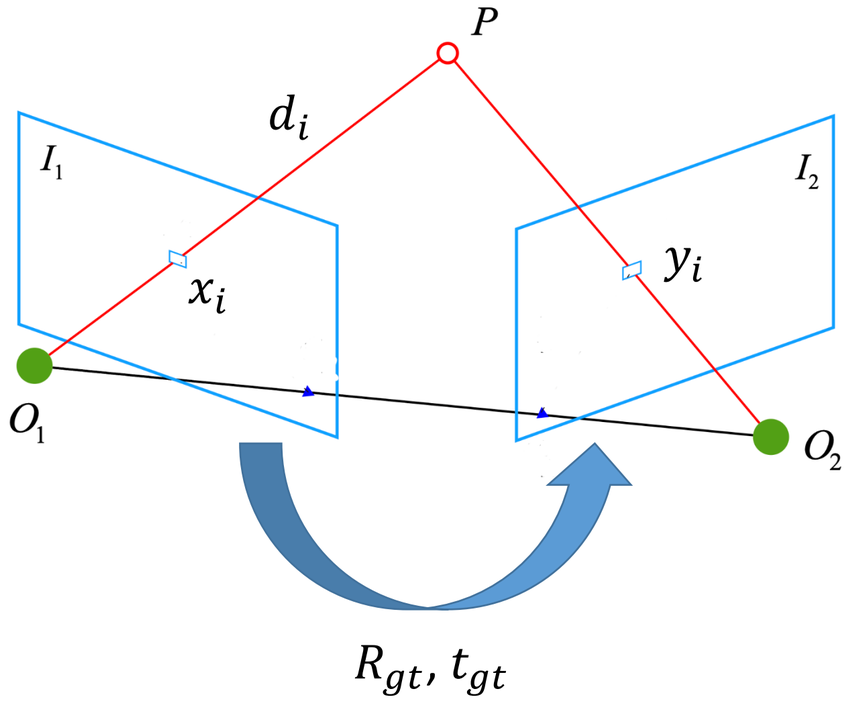


# Refine Pose Groundtruth

- self-annotated data with only vision information
- pose groundtruth: $R_{gt}, t_{gt}$
  - misalignment at frame level
  - needs refinement and calibration

- extract matched SIFT features on $I_1$, $I_2$
- set $R_{gt}, t_{gt}$ as initial values, bundle adjustment to optimize relative pose $R, t$



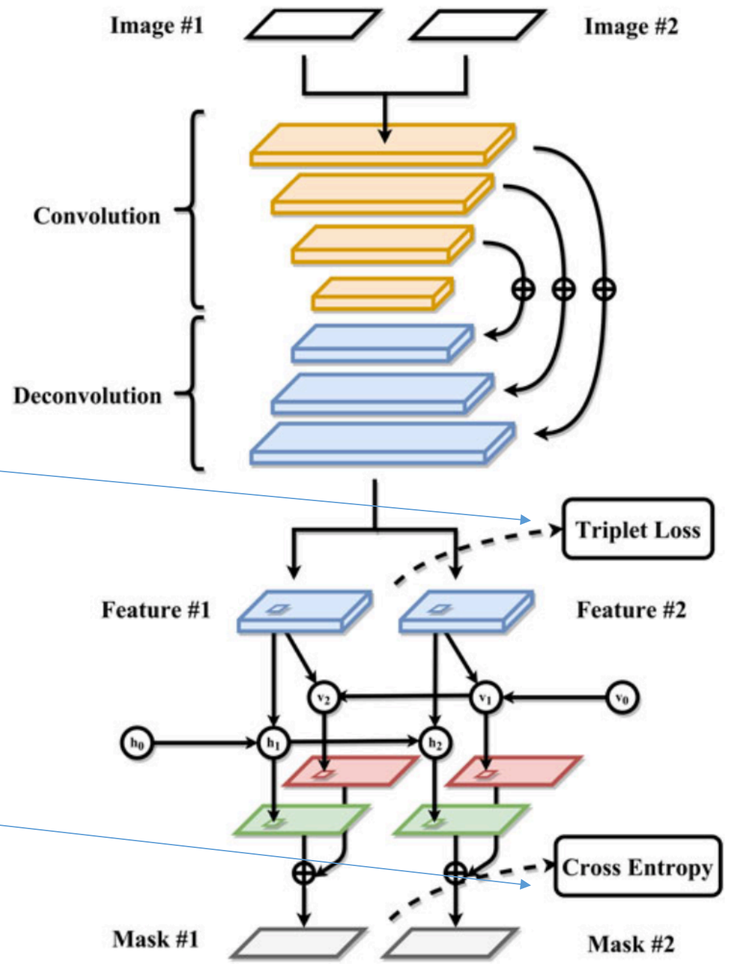


# Design Loss Function

- Generate labels as sample for each image
  - SHI-Thomas corners on each 16x16 grids
  - Obtain dense samples

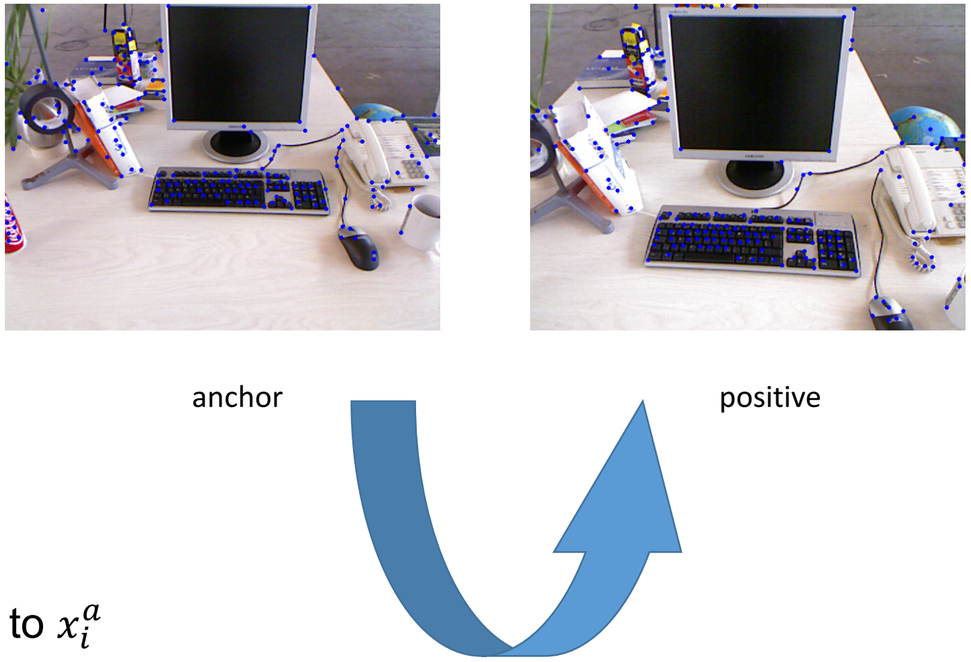
- Descriptor loss $L_{metric}$
  - deep metric learning
  - triplet loss

- Detector loss $L_{mask}$
  - weighted cross entropy



# Descriptor Loss

- Deep Metric Learning
- Triple set: <anchor, positive, negative>
  - Anchor: $x_i^a$
    - Detected feature points in the left image
  - Positive: $x_i^p$
    - Warped points from left $x_i^a$ to right image by using $(R_{gt}, t_{gt})$
    - $x_i^p = \pi^{-1}(R_{gt} \cdot \pi(x_i^a, d_i) + t_{gt})$
  - Negative: $x_i^n$
    - Detected feature points in the right image: $y_j$
    - For each $x_i^a$, find the sample $x_i^n$ from $y_j$ with minimum distance to $x_i^a$




- Triplet Loss:

$$L_{metric} = \sum_i \max\left(0, d_{x_i^a, x_i^p} - d_{x_i^a, x_i^n} + m\right)$$

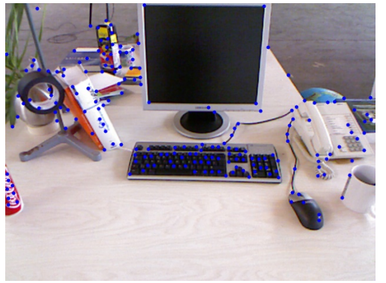
# Detector Loss

- For each pixel $x_i$ with label $c_{x_i} \in \{0, 1\}$ in the left image
- Warp all feature points $x_i$ from left to right image
  - $x_i^p = \pi^{-1}(R_{gt} \cdot \pi(x_i, d_i) + t_{gt})$
- Weighted Cross Entropy Loss:

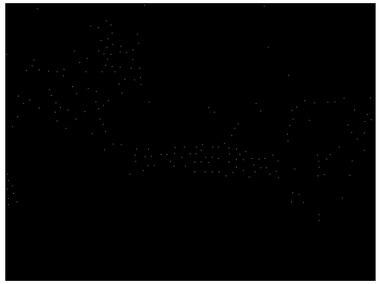
$$L_{mask} = L_{ce}(\mathbf{o_1}, \mathbf{x}) + L_{ce}(\mathbf{o_2}, \mathbf{x^p})$$

$$L_{ce}(\mathbf{o}, \mathbf{x}) = -\sum_i [\alpha_1 c_{x_i} \log(\mathbf{o}(\mathbf{x_i})) + \alpha_2 (1 - c_{x_i} \log(1 - \mathbf{o}(\mathbf{x_i})))]$$

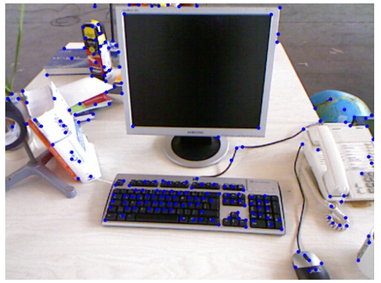
- $o_1, o_2$ are predicted keypoints by network
- $\alpha_1, \alpha_2$ are balanced factors:
  - the number of key-points (thousands)
  - the total number of pixels (hundreds of thousands)



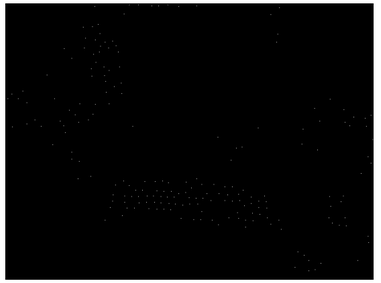
anchor



left label



positive



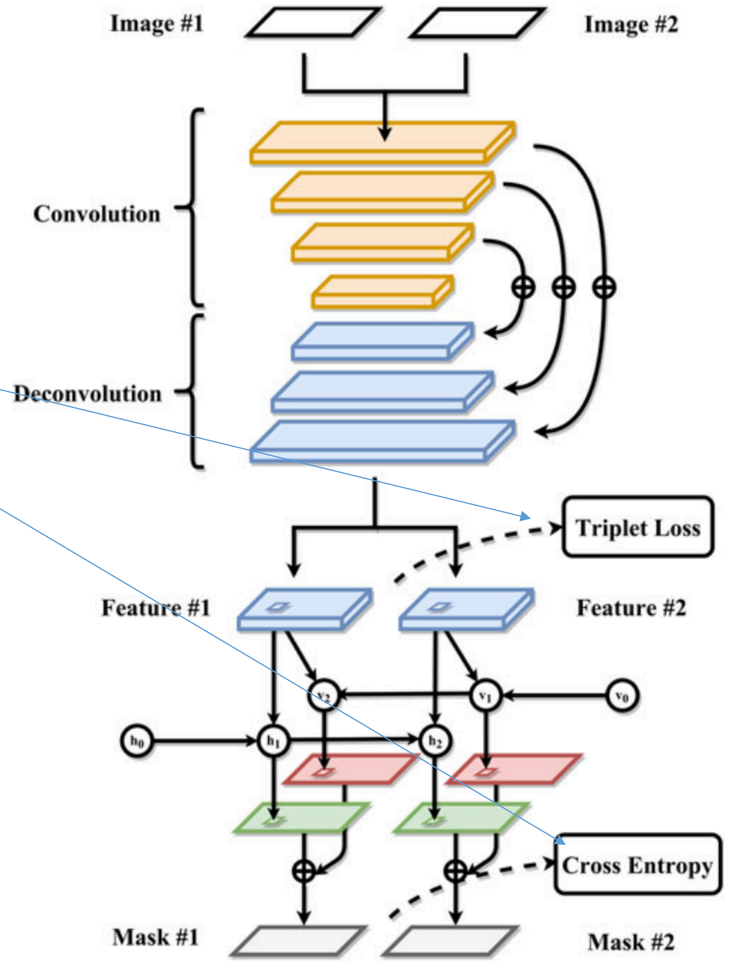
right label

# Design Loss Function

- Combined Loss = $\omega_1 L_{metric} + \omega_2 L_{mask}$

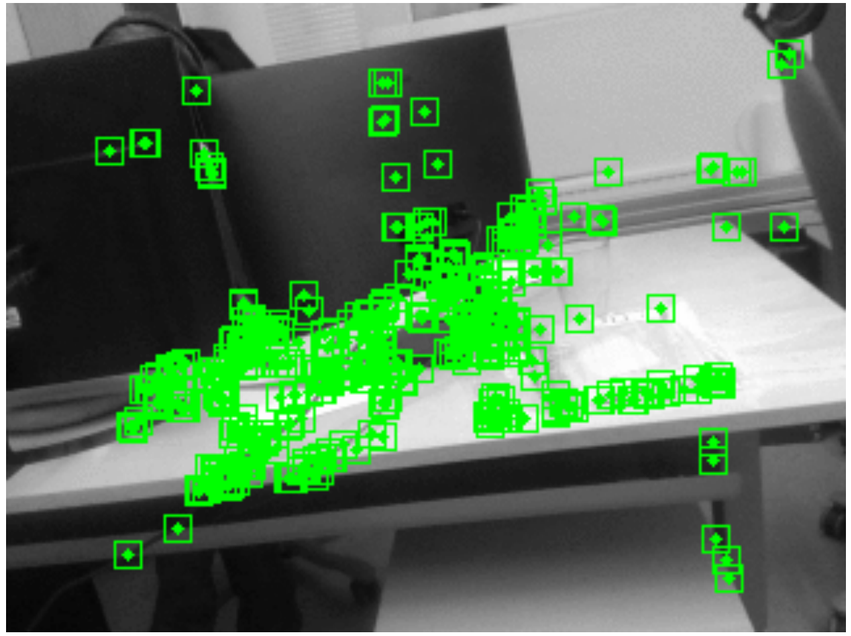
| Dataset | GCN | SuperPoint | ORB |
|---|---|---|---|
| fr1_floor | **0.015m** | - | 0.080m |
| fr1_desk | **0.037m** | 0.166m | 0.151m |
| fr1_360 | **0.059m** | - | 0.278m |
| fr3_long_office | **0.046m** | 0.105m | 0.090m |
| fr3_large_cabinet | **0.064m** | 0.195m | 0.097m |
| fr3_nst | **0.018m** | 0.055m | 0.061m |
| fr3_nnf | **0.221m** | - | - |

Absolute trajectory error in TUM



# Compare Performance

ORB



Learning-Based

- Thanks!