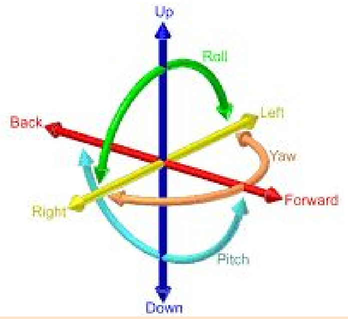
法律声明

LiDAR Odometry
&
Loop Closure Detection

LiDAR Odometry

Challenges

- Ego-motion Estimation Requires



6DOF, precision, low-draft

Robust to Lighting Changes

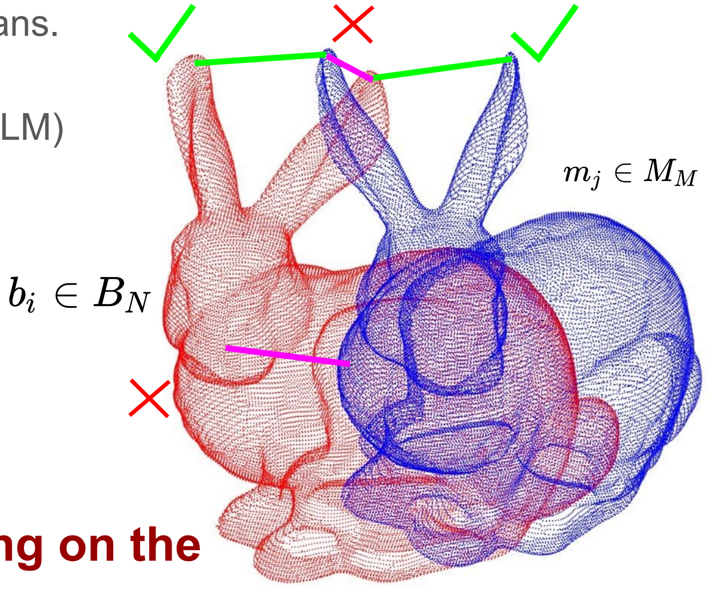
Real-time, high frequency, small but factor

Robust to aggressive motion

Scan-to-Scan Match

- Point-to-point ICP
  - Compute corresponding points between two scans.
  - Searching closest for $O(N \log M)$
  - Use Newton-gradient or Levenberg-Marquardt (LM) Calculate the transformation iteratively.

$$T \leftarrow \underset{T}{\operatorname{argmin}} \{ \sum_i w_i || T \cdot b_i - m_i ||^2 \};$$

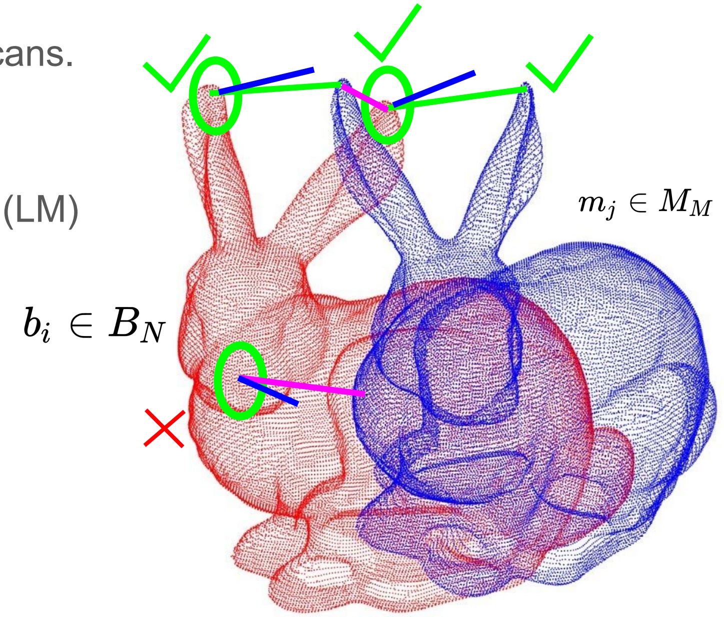


The accuracy of ICP method is highly relying on the initial transformation.

Scan-to-Scan Match

- Point-to-plane ICP
  - Compute corresponding points between two scans.
  - Searching closest for $O(N \log M)$
  - **Calculate the surfnorm for each point in *B.***
  - Use Newton-gradient or Levenberg-Marquardt (LM) Calculate the transformation iteratively.

$$T \leftarrow \underset{T}{\operatorname{argmin}} \{ \sum_i w_i || \eta_i \cdot (T \cdot b_i - m_i) ||^2 \}$$



Scan-to-Scan Match

Surface Normal Definition

Given a point-cloud P, for point p_i we have,

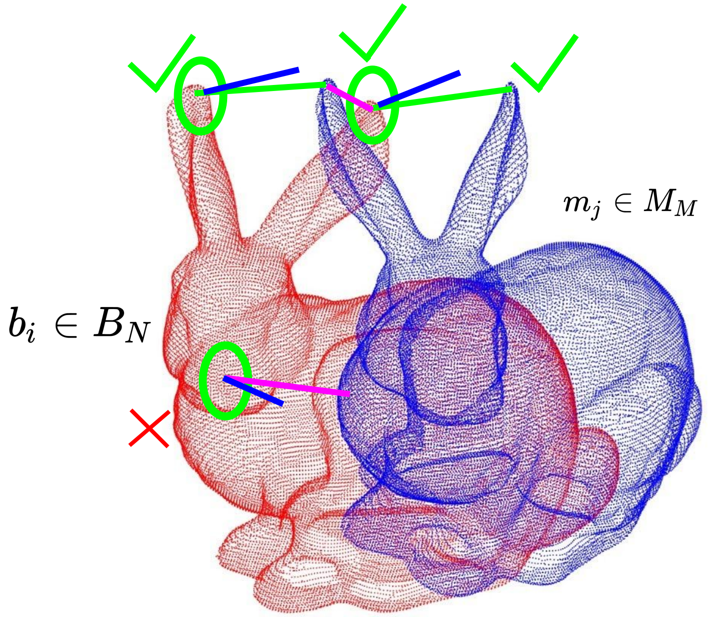
$$\overline{x}_i = \frac{1}{k}\sum_{i=1}^{k} p_i, \tag{1}$$

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{x}_i)\cdot(p_i - \overline{x}_i)^T, \tag{2}$$

$$C \cdot v_j = \lambda_j \cdot v_j, \lambda_1 \leq \lambda_2 \leq \lambda_3, j \in \{1, 2, 3\} \tag{3}$$

and the surface normal of point p_i, could be estimated by,

$$c_i = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} \tag{4}$$



Scan-to-Scan Match

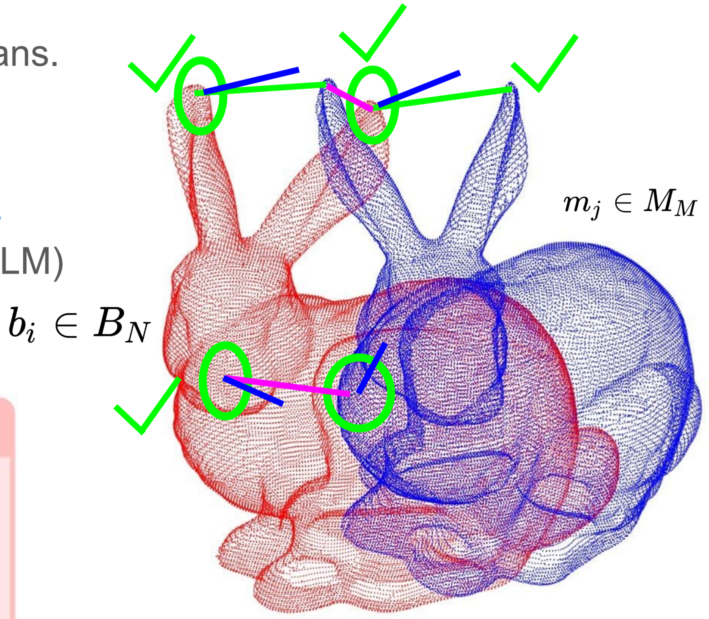
- Plane-to-plane ICP
  - Compute corresponding points between two scans.
  - Searching closest for $O(N \log M)$
  - **Calculate the surfnorm for each point in *B.***
  - **Calculate the surfnorm for the relatives in *M.***
  - Use Newton-gradient or Levenberg-Marquardt (LM)
    Calculate the transformation iteratively.

Points Sets definition

$$\hat{M} = \{\hat{m}_i\}, \hat{B} = \{\hat{b}_i\}, \hat{m}_i = \mathbf{T}^* \hat{b}_i \qquad (5)$$

$$m_i = N(\hat{m}_i, C_i^M), b_i = N(\hat{b}_i, C_i^B) \qquad (6)$$

$$d_i^{\mathbf{T}} = m_i - \mathbf{T} b_i \qquad (7)$$



Scan-to-Scan Match

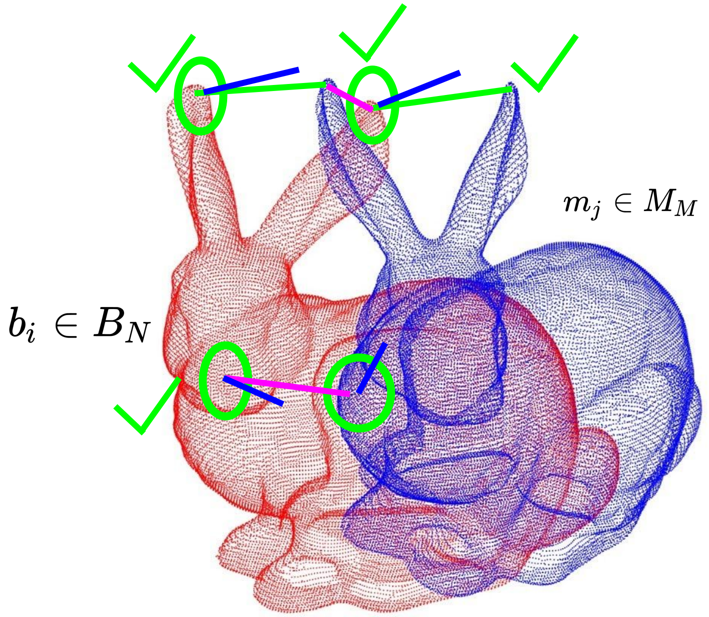
Optimal Transformation Case

$$\begin{aligned} d_i^{\mathbf{T}^*} &= m_i - \mathbf{T}^* b_i \\ &\sim N(\hat{m} - \mathbf{T}^*\hat{b}_i, C_i^M + (\mathbf{T}^*)C_i^B(\mathbf{T}^*)') \qquad (8) \\ &= N(0, C_i^M + (\mathbf{T}^*)C_i^B(\mathbf{T}^*)') \end{aligned}$$

Maximum Likelihood Estimation

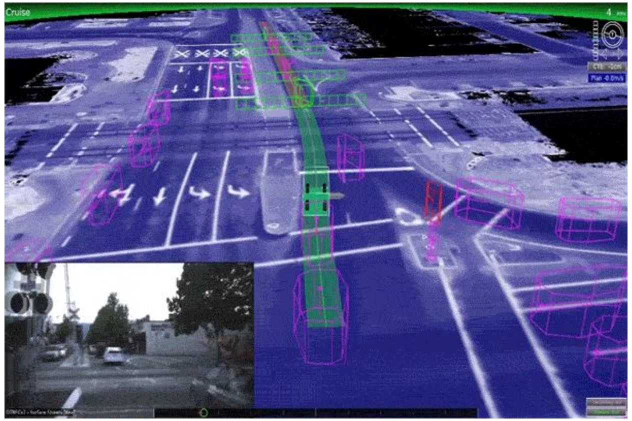
$$\mathbf{T} = \arg\max_{\mathbf{T}} \prod_i^N p(d_i^{\mathbf{T}}) = \arg\max_{\mathbf{T}} \sum_i^N \log(p(d_i^{\mathbf{T}})) \qquad (9)$$

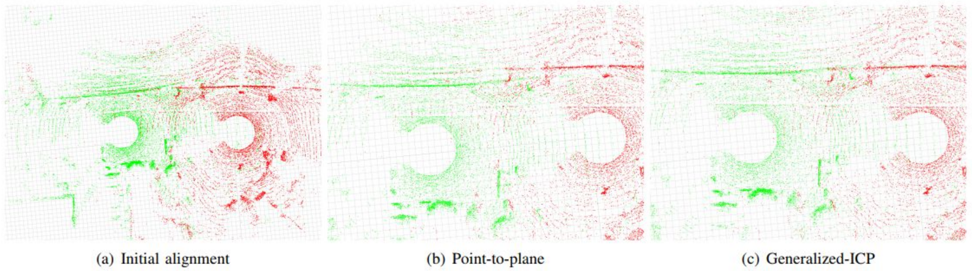
$$\arg\max_{\mathbf{T}} \sum_i^N (d_i^{\mathbf{T}})'(C_i^M + \mathbf{T}C_i^B\mathbf{T})^{-1} d_i^{\mathbf{T}} \qquad (10)$$



Why not ICP method for LiDAR odometry

- Point-to-point ICP: fast but not robust.
- Point-to-Plane ICP: Moderate Speed with robust for plane search.
- Plane-to-Plane ICP: Slowest but most robust for accurate searching.

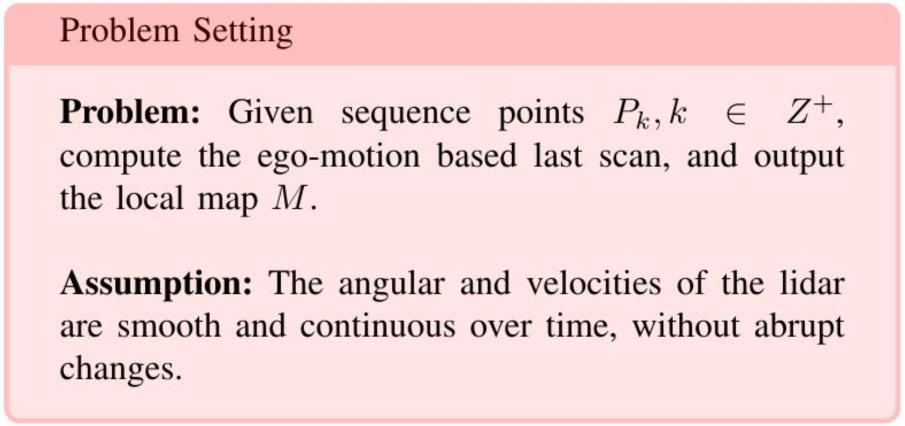




Segal, Aleksandr, Dirk Haehnel, and Sebastian Thrun. "Generalized-ICP." *Robotics: science and systems*. Vol. 2. No. 4. 2009.

LiDAR Odometry

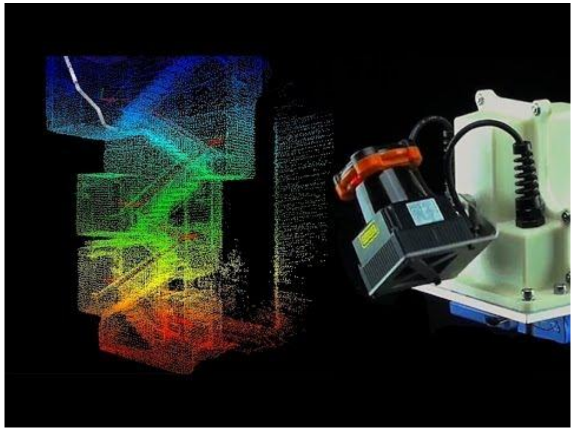
- LiDAR odometry property
  - 6-DOF, precision, low-drift
  - Real-time, high frequency
  - Robust to lighting changes



Problem Setting

Problem: Given sequence points $P_k, k \in Z^+$, compute the ego-motion based last scan, and output the local map M.

Assumption: The angular and velocities of the lidar are smooth and continuous over time, without abrupt changes.

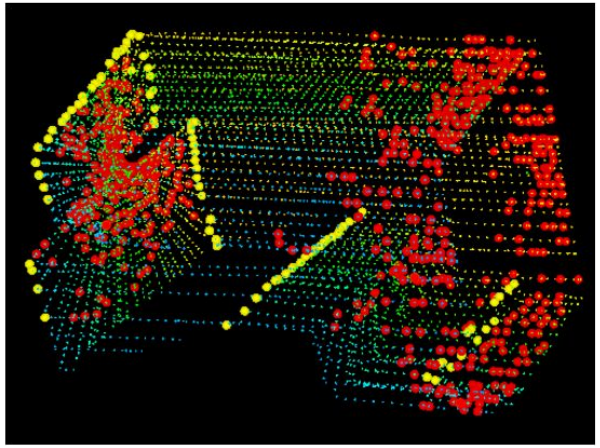


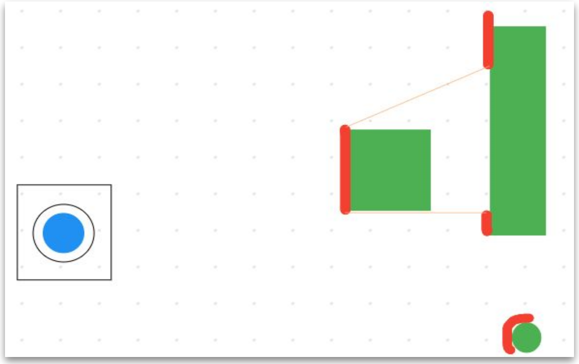
How to select efficient points

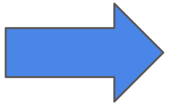
Curvature Definition

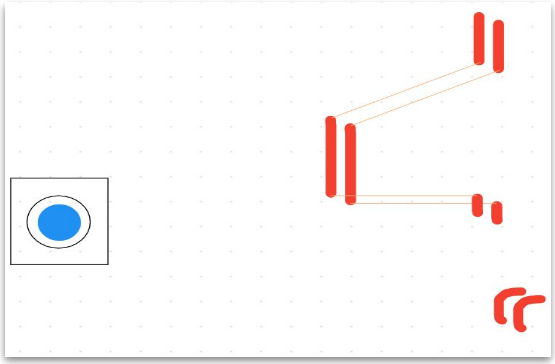
Assume a LiDAR scan contain X^L with N sweeps (i.e. VLP-16 has 16 sweeps), the curvature of the point i-th point in n-th sweep $X^L_{(n,i)}$ could by obtained by,

$$c_{(n,i)} = \frac{1}{|S| \cdot \|X^L_{(n,i)}\|} \| \sum_{j \in S, j \neq i} (X^L_{(n,i)} - X^L_{(n,j)}) \|. \quad (11)$$









Surf points

Corner points

How to select efficient points

Corner Points

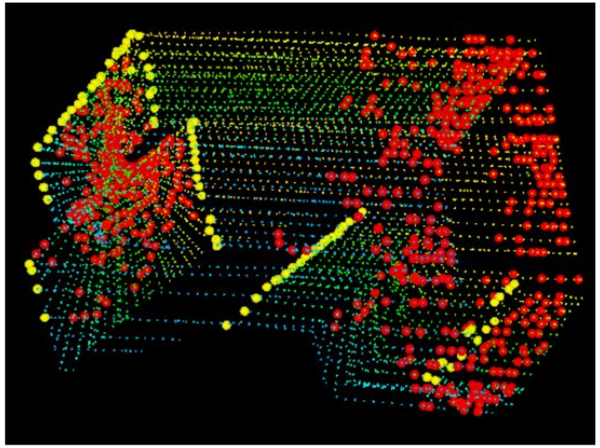
X_k^L is the current scan, and X_{k-1}^L is the last scan, the distance for corner point $X_{(k,i)}^L$ could be estimated by,

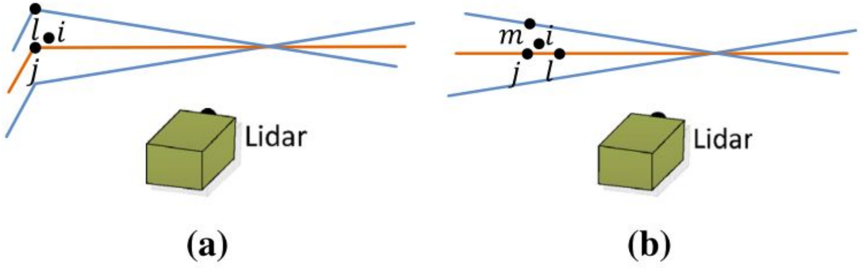
$$d_\varepsilon = \frac{|(\tilde{X}_{(k,i)}^L - \hat{X}_{(k-1,j)}^L) \times (\tilde{X}_{(k,i)}^L - \hat{X}_{(k-1,l)}^L)|}{(\hat{X}_{(k-1,j)}^L - \hat{X}_{(k-1,l)}^L)} \tag{12}$$

Surf Points

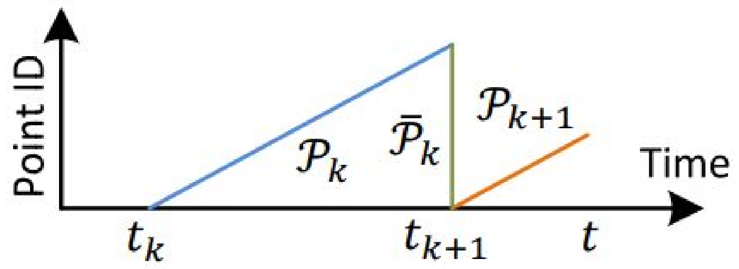
X_k^L is the current scan, and X_{k-1}^L is the last scan, the distance for surf point $X_{(k,i)}^L$ could be estimated by,

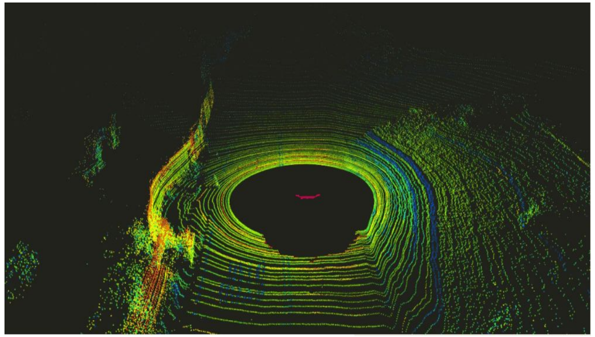
$$d_\zeta = \frac{\begin{bmatrix} ((\tilde{X}_{(k-1,i)}^L - \hat{X}_{(k-1,l)}^L) \times (\tilde{X}_{(k-1,j)}^L - \hat{X}_{(k-1,m)}^L)) \\ (\hat{X}_{(k,i)}^L - \hat{X}_{(k-1,j)}^L) \end{bmatrix}}{|(\tilde{X}_{(k-1,i)}^L - \hat{X}_{(k-1,l)}^L) \times (\tilde{X}_{(k-1,j)}^L - \hat{X}_{(k-1,m)}^L)|} \tag{13}$$

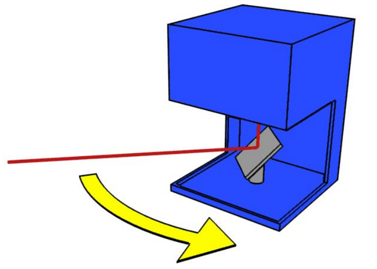


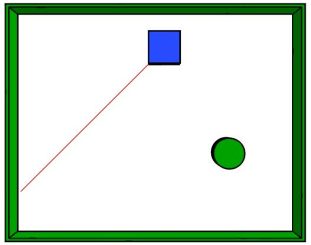


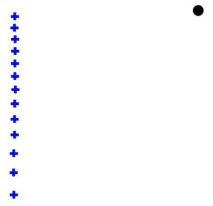
Points interpolation











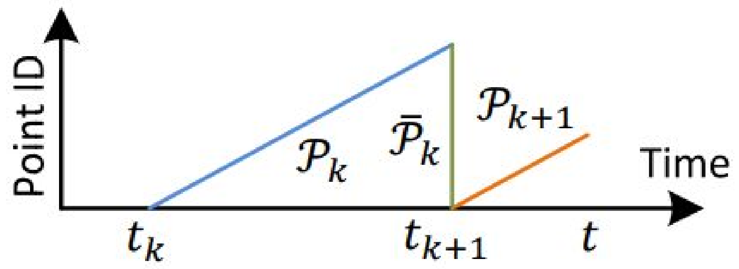
1m

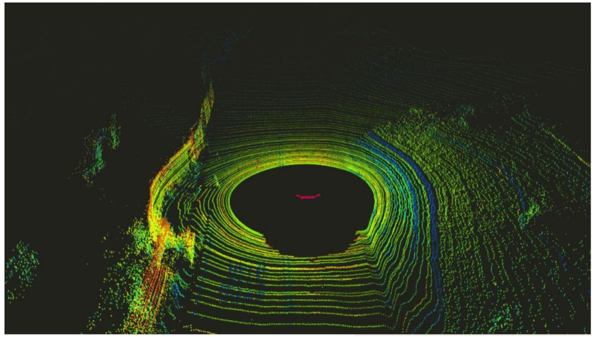
Point Missing Calculation

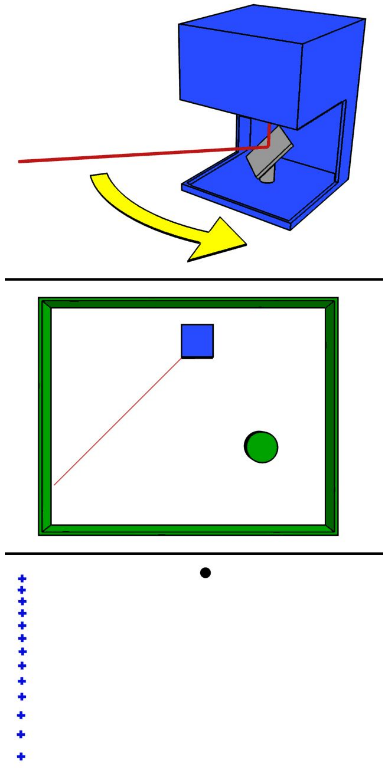
Assume we have a Velodyne-64E LiDAR, with norm motor speed at $600rmp = 62.8rad/sec$, which is mounted on a car moving at $10m/s$. If we know the LiDAR data publishing frequency at 10Hz,

Question: what's the missing distance for the first point and the last point within a same scan?

Points interpolation







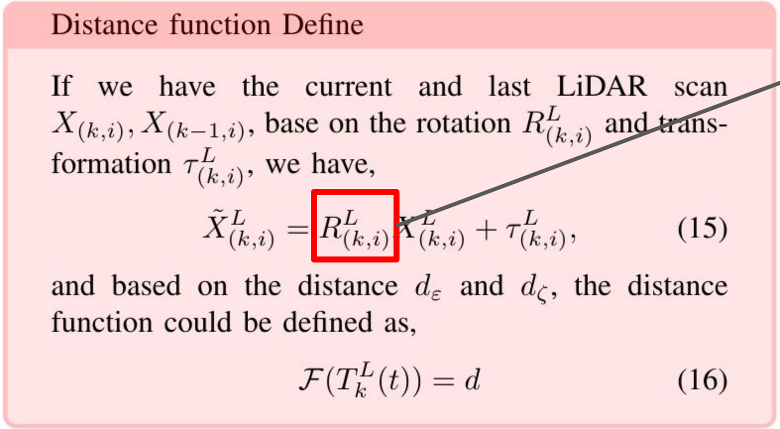
Points Interpolation

Given a point $i, i \in P_k$, let $t_{(k,i)}$ be its time stamp, and let $\mathbf{T}^L_{(k,i)}$ be the pose transformation between $[t_k, t_{(k,i)}]$. $\mathbf{T}^L_{(k,i)}$ could be obtained by linear interpolation of $\mathbf{T}^L_k(t)$,

$$\mathbf{T}^L_{(k,i)} = \frac{t_{(k,i)} - t_k}{t - t_k} \mathbf{T}^L_k(t). \qquad (14)$$

Motion Estimation

- What we get now
  - Calculate the curvature of the given points.
  - Extract the **Corner points** and the **Surf points**.
  - Calculate the distance of relative corner points and surf points.
  - Points interpolation based on estimated transformation.
- Estimate the transformation.



Distance function Define

If we have the current and last LiDAR scan $X_{(k,i)}, X_{(k-1,i)}$, base on the rotation $R^L_{(k,i)}$ and transformation $\tau^L_{(k,i)}$, we have,

$$\tilde{X}^L_{(k,i)} = \boxed{R^L_{(k,i)}} X^L_{(k,i)} + \tau^L_{(k,i)}, \qquad (15)$$

and based on the distance d_ε and d_ζ, the distance function could be defined as,

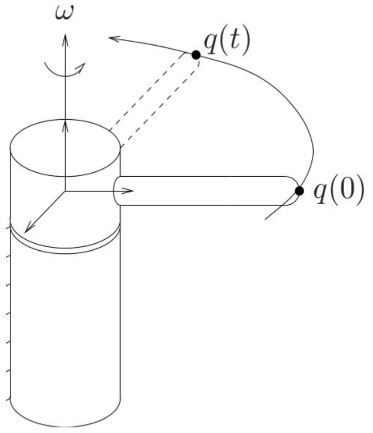
$$\mathcal{F}(T^L_k(t)) = d \qquad (16)$$

Instead of estimating angle, estimate the angular velocity

Re-define the rotation matrix

LM optimization method

Rodrigues' Formula



Exponential coordinates for rotation

Let $\omega \in R^3$ be a unit vector which specifies the direction of rotation and let $\theta \in R$ be the angle of rotation in radians. If the body moves in a constant unit velocity about the axis ω, the velocity of the point $\dot{q}$ could be written as,

$$\dot{q}(t) = \omega \times q(t) = \hat{\omega} q(t), \tag{17}$$

where $\hat{\omega}$ is the **skew symmetric** matrix, and $\hat{\omega}^T = -\hat{\omega}$. Since Eq(17) is a time-invariant linear differential equation, so we can integrated into,

$$q(t) = \exp^{\hat{\omega} t} q(0), \tag{18}$$

$$\exp^{\hat{\omega} t} = I + \hat{\omega} t + \frac{(\hat{\omega} t)^2}{2!} + \frac{(\hat{\omega} t)^3}{3!} + \ldots \tag{19}$$

This means, if we rotate the object about the axis ω with at a unit velocity for θ units of time, the net rotation will be,

$$R(\omega, \theta) = \exp^{\hat{\omega} \theta}. \tag{20}$$

Lemma 1

Lemma 1: Given $\hat{a} \in SO(3)$, we have the following relations:

$$\hat{a}^2 = aa^T - \|a\|^2 I \tag{21}$$

$$\hat{a}^3 = -\|a\|^2 \hat{a} \tag{22}$$

Rodrigues' Formula

Utilizing lemma 1, set $a = \omega\theta, \|\omega\| = 1$, we will become:

$$\hat{\omega}^{2k+1} = \hat{\omega} \cdot (-1)^k \tag{23}$$

$$\hat{\omega}^{2k+2} = \hat{\omega}^2 \cdot (-1)^k \tag{24}$$

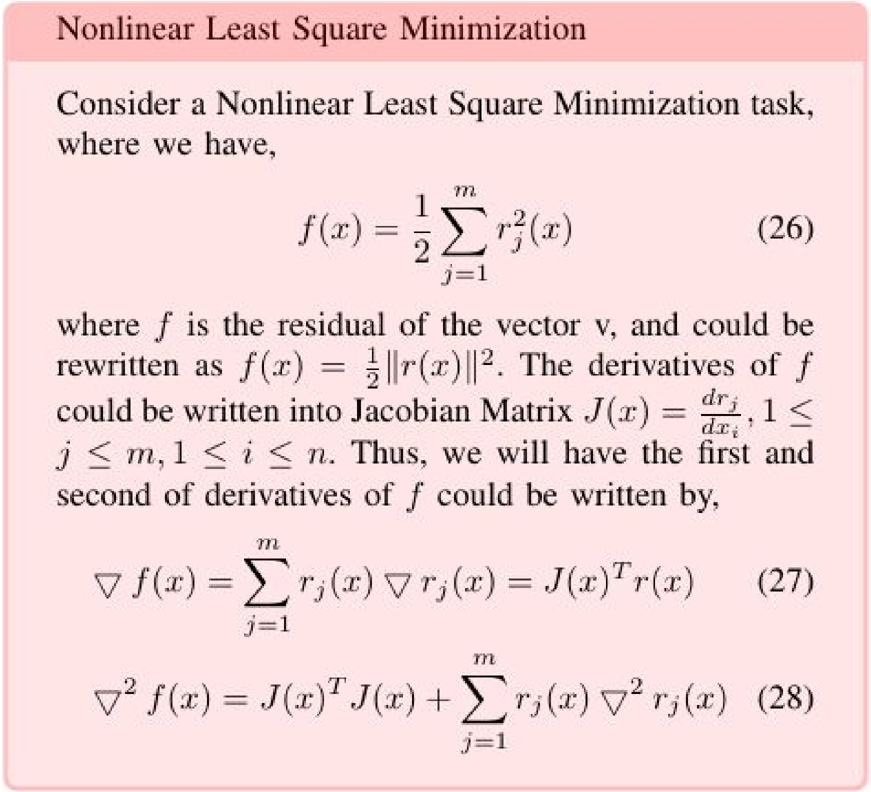
Thus, for original Eq(20) will be transformed into,

$$\begin{aligned} \exp^{\hat{\omega}\theta} &= I + (\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} \ldots)\hat{\omega} + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \frac{\theta^6}{6!} \ldots)\hat{\omega}^2 \\ &= I + \hat{\omega} \sin\theta + \hat{\omega}^2 (1 - \cos\theta) \\ &= I + \frac{\hat{\omega}}{\|\hat{\omega}\|} \sin(\|\omega\|\theta) + \frac{\hat{\omega}}{\|\hat{\omega}\|}(1 - \cos(\|\omega\|\theta)) \end{aligned} \tag{25}$$

$$\mathrm{d}R(\omega, \theta) = R(\omega, \theta) \cdot \mathrm{d}\hat{\omega},$$

$$\hat{\omega} = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix}$$

Levenberg-Marquardt Algorithm



Nonlinear Least Square Minimization

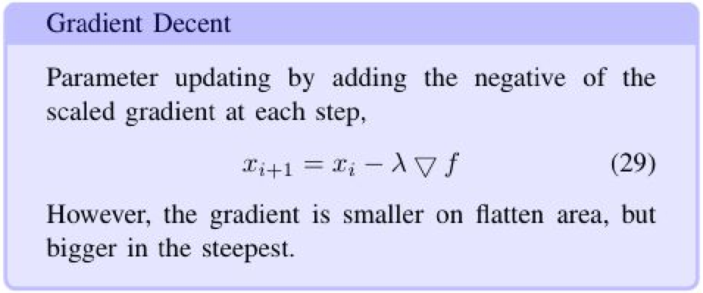
Consider a Nonlinear Least Square Minimization task, where we have,

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{26}$$

where f is the residual of the vector v, and could be rewritten as $f(x) = \frac{1}{2}\|r(x)\|^2$. The derivatives of f could be written into Jacobian Matrix $J(x) = \frac{dr_j}{dx_i}, 1 \leq j \leq m, 1 \leq i \leq n$. Thus, we will have the first and second of derivatives of f could be written by,

$$\bigtriangledown f(x) = \sum_{j=1}^{m} r_j(x) \bigtriangledown r_j(x) = J(x)^T r(x) \tag{27}$$

$$\bigtriangledown^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \bigtriangledown^2 r_j(x) \tag{28}$$

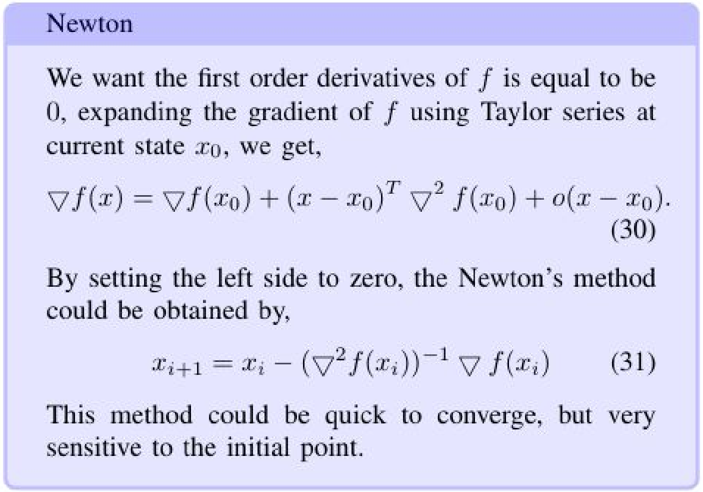


Gradient Decent

Parameter updating by adding the negative of the scaled gradient at each step,

$$x_{i+1} = x_i - \lambda \bigtriangledown f \tag{29}$$

However, the gradient is smaller on flatten area, but bigger in the steepest.



Newton

We want the first order derivatives of f is equal to be 0, expanding the gradient of f using Taylor series at current state x_0, we get,

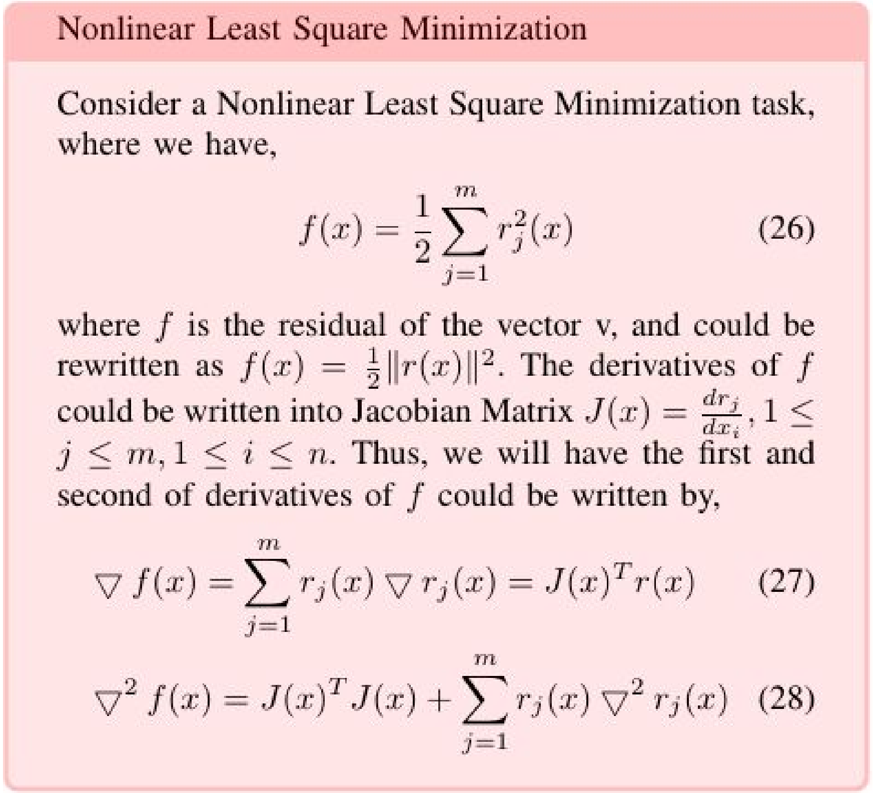
$$\bigtriangledown f(x) = \bigtriangledown f(x_0) + (x - x_0)^T \bigtriangledown^2 f(x_0) + o(x - x_0). \tag{30}$$

By setting the left side to zero, the Newton's method could be obtained by,

$$x_{i+1} = x_i - (\bigtriangledown^2 f(x_i))^{-1} \bigtriangledown f(x_i) \tag{31}$$

This method could be quick to converge, but very sensitive to the initial point.

Levenberg-Marquardt Algorithm



Nonlinear Least Square Minimization

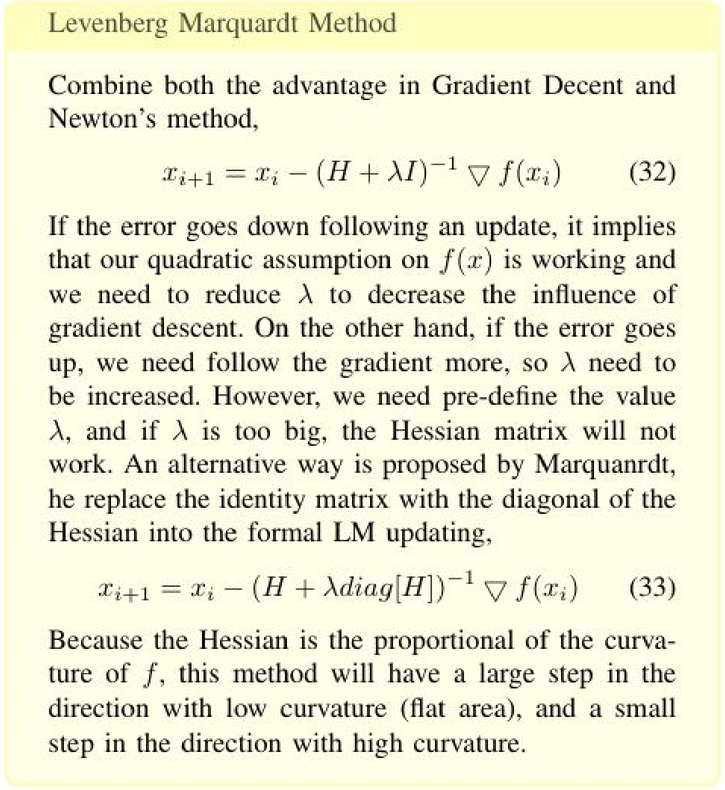
Consider a Nonlinear Least Square Minimization task, where we have,

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \qquad (26)$$

where f is the residual of the vector v, and could be rewritten as $f(x) = \frac{1}{2}\|r(x)\|^2$. The derivatives of f could be written into Jacobian Matrix $J(x) = \frac{dr_j}{dx_i}, 1 \leq j \leq m, 1 \leq i \leq n$. Thus, we will have the first and second of derivatives of f could be written by,

$$\bigtriangledown f(x) = \sum_{j=1}^{m} r_j(x) \bigtriangledown r_j(x) = J(x)^T r(x) \qquad (27)$$

$$\bigtriangledown^2 f(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x) \bigtriangledown^2 r_j(x) \qquad (28)$$



Levenberg Marquardt Method

Combine both the advantage in Gradient Decent and Newton's method,

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \bigtriangledown f(x_i) \qquad (32)$$

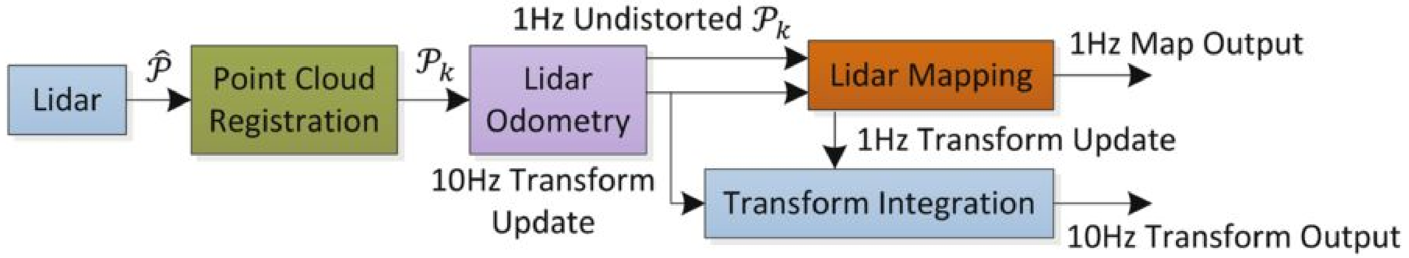
If the error goes down following an update, it implies that our quadratic assumption on $f(x)$ is working and we need to reduce λ to decrease the influence of gradient descent. On the other hand, if the error goes up, we need follow the gradient more, so λ need to be increased. However, we need pre-define the value λ, and if λ is too big, the Hessian matrix will not work. An alternative way is proposed by Marquanrdt, he replace the identity matrix with the diagonal of the Hessian into the formal LM updating,

$$x_{i+1} = x_i - (H + \lambda diag[H])^{-1} \bigtriangledown f(x_i) \qquad (33)$$

Because the Hessian is the proportional of the curvature of f, this method will have a large step in the direction with low curvature (flat area), and a small step in the direction with high curvature.

LiDAR Odometry Pipeline

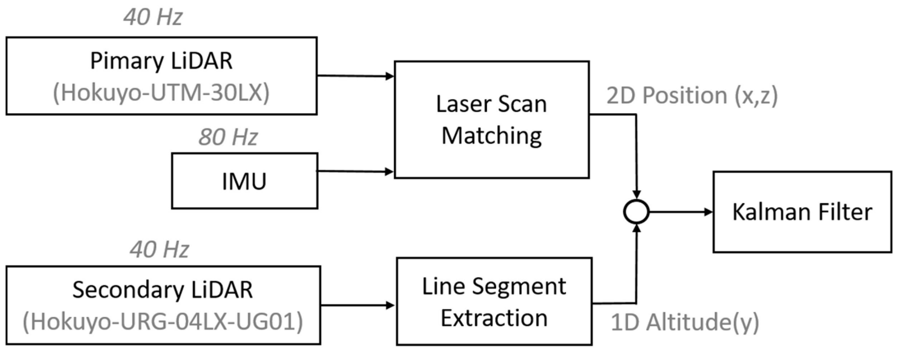
- Pipeline
    - Calculate the curvature of new scan.
    - Point interpolation
    - Distance calculation for Edge and Surf points
    - Use LM to estimate transformation matrix iteratively.

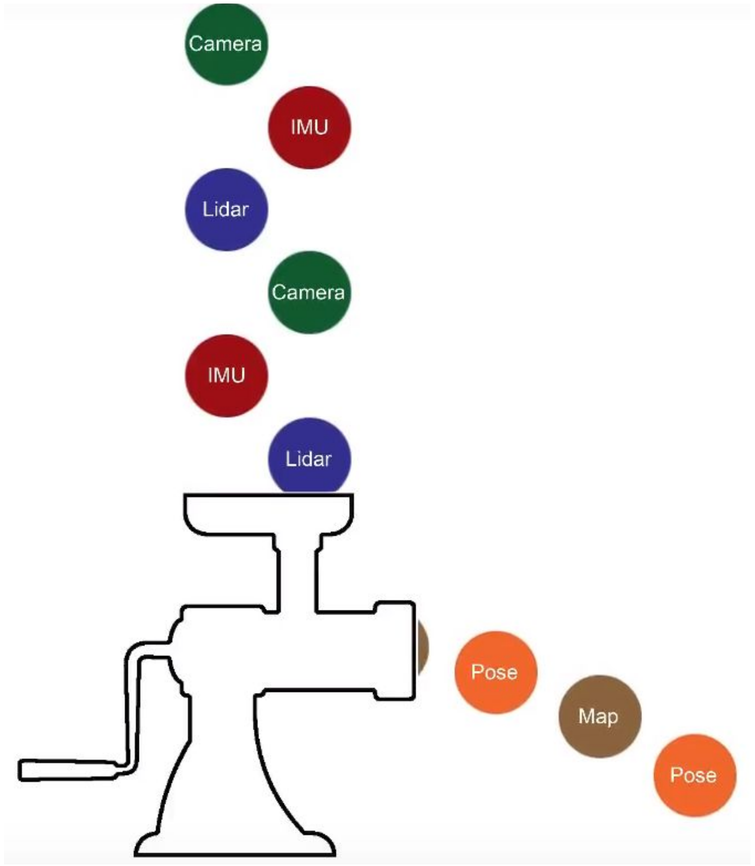


Sensor Fusion Pipeline

Filter Pipeline

- Sequence Processing in Filter Method
  - Sequence processing.
  - **Synchronized** problem

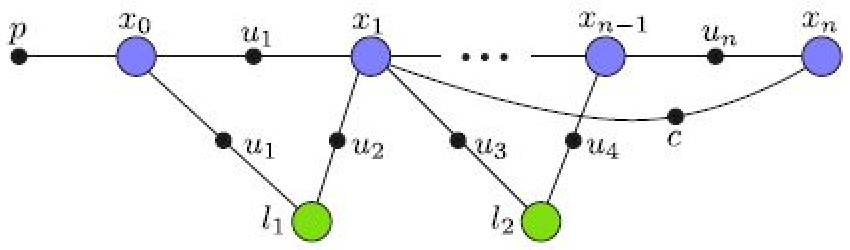


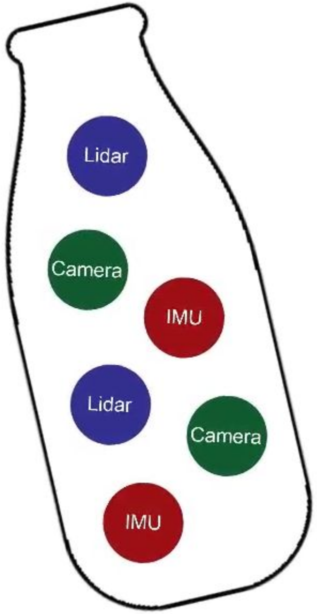


Sun, Shu-Li, and Zi-Li Deng. "Multi-sensor optimal information fusion Kalman filter." *Automatica* 40.6 (2004): 1017-1023.

Factor Graph Pipeline

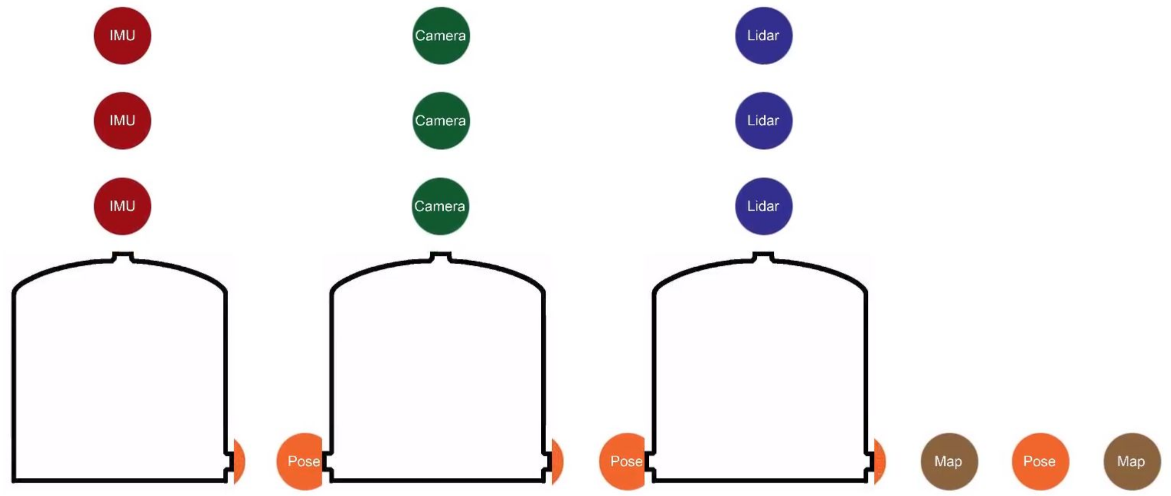
- Joint Estimation in Factor Graph
  - Consider all the constraints in a joint framework.
  - Try to minimize the joint error in FG at each step.
  - Limitation in large Scale





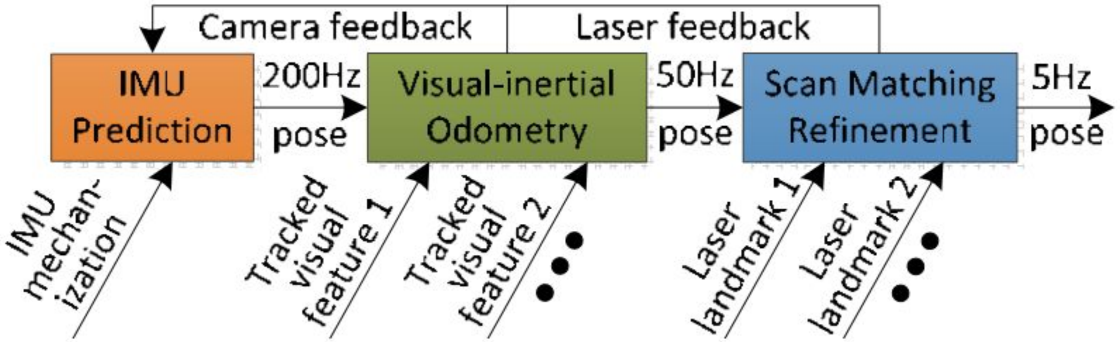
Hierarchical Pipeline

- Hierarchical pose estimation
    - Solved for motion from high to low frequency.
    - Each module reinforces from the previous modules.
    - Bypass modules to handle sensor degradation.



Hierarchical Pipeline

- Hierarchical pose estimation
  - Starting with IMU mechanization for prediction.
  - Visual-inertial coupled method estimate ego-motion.
  - Scan matching for further refines the estimation and generate map.



Homework

Understanding the framework of LOAM

Zhang, Ji, and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." *Robotics: Science and Systems*. Vol. 2. 2014.

Zhang, Ji, and Sanjiv Singh. "Visual-lidar odometry and mapping: Low-drift, robust, and fast." *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015.

Zhang, Ji, and Sanjiv Singh. "Low-drift and real-time lidar odometry and mapping." *Autonomous Robots* 41.2 (2017): 401-416.

https://github.com/laboshinl/loam_velodyne.git