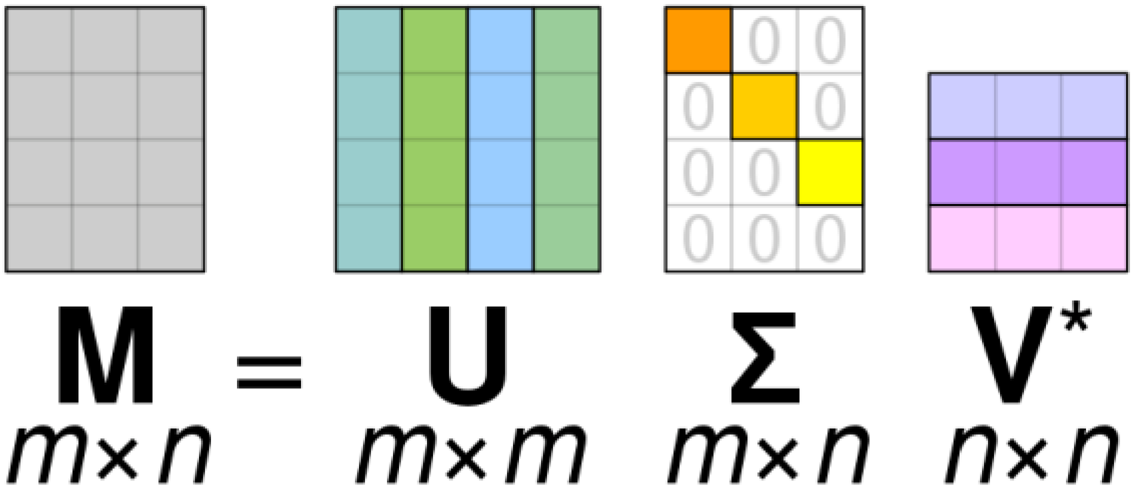
Catatan Skripsi

Singular Value Decomposition (SVD)

Semua matriks dapat diuraikan menjadi 3 matriks baru,

$$SVD(A) = U\Sigma U^T$$
$$A = \text{input matrix}$$
$$U = \text{orthogonal matrix (m x m), left singular values}$$
$$\Sigma = \text{diagonal singular values matrix (m x n)}$$
$$V = \text{orthogonal matrix (n x n), right singular values}$$



Orthogonal Matrix

Matrix yang kolom dan barisnya merupakan vektor orthonormals

$$Q^TQ = QQ^T = I, \text{where } Q^T = Q^{-1}$$

Suatu vektor dapat dikatakan orthonormal apabila memiliki magnitude = 1 dan dot product (·) = 0.

Contoh pada matriks rotasi :

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Matriks rotasi merupakan orthogonal karena $RR^T = I$, dan $R^T = R^{-1}$

Singular Values

Singular values pada dasarnya merupakan akar dari eigenvalues $(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_n)$ dari A^TA, maka singular valuesnya adalah $\sigma_1 = \sqrt{\lambda}_1, \sigma_2 = \sqrt{\lambda}_2, \sigma_3 = \sqrt{\lambda}_3, \ldots, \sigma_n = \sqrt{\lambda}_n$. Matriks A^TA merupakan matriks symmetric $n \times n$.

Matriks diagonal Σ, memiliki urutan σ_n dari paling besar ke paling kecil dan ≥ 0.

Rank Matrix

Rank matriks merupakan nilai kolom atau baris yang independen secara linear.

Contoh :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, R_2 = R_2 - 4R_1$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 7 & 8 & 9 \end{bmatrix}, R_3 = R_3 - 7R_1$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{bmatrix}, R_2 = \frac{R_2}{R_3}$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & -6 & -12 \end{bmatrix}, R_3 = R_3 + 6R3$$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

Maka, jumlah rank dari matriks A adalah 2 (non-zeros baris atau kolom).

How To

1. Hitung A^TA adalah matriks simetrik, maka eigenvectornya orthogonal. Untuk mendapatkan matriks V^T
2. Hitung eigenvalues (λ) dengan $\det(A^TA - \lambda I) = 0$, maka diperoleh eigenvalues $\lambda_1, \lambda_2, .., \lambda_n$
3. Hitung eigenvector (v) dengan menggunakan $\lambda_{1:n}$. Agar magnitude eigenvector = 1, tiap element pada eigenvector dibagi dengan $\sqrt{v_i^2 + v_j^2 + \ldots + v_n^2}$
4. Hitung nilai singular value $\sigma_i = \sqrt{\lambda_i}$

5. Hitung langkah 1 sampai 3, dengan AA^T untuk mendapatkan matriks U

Singular Value Decomposition - Iterative Closest Point (SVD-ICP)

$$X = \{x_1, x_2, x_3, \ldots, x_n\}$$
$$Y = \{y_1, y_2, y_3, \ldots, y_n\}$$
$$E = \underset{R,t}{\operatorname{argmin}} \sum_{i=1}^{n} ||y_i - (Rx_i + t)||^2$$

E adalah fungsi loss, X adalah titik korespondensi pada pointcould source dan P adalah titik korespondensi pada pointcloud reference.

$$\mu_x = \frac{1}{N} \sum_{n=1}^{N} x_i$$
$$\mu_y = \frac{1}{N} \sum_{n=1}^{N} y_i$$

μ_x dan μ_p merupakan center of mass dari pointcloud X dan P. Kemudian, pointcloud X dan P pada koordinat lokal dengan mengurakan tiap titik X dan P terhadap center of mass-nya.

$$X' = x_i - \mu_x = \{x'_i\}$$
$$Y' = y_i - \mu_y = \{y'_i\}$$
$$W = \sum_{i=1}^{N} x'_i y'^T_i$$
$$W = \begin{bmatrix} cov(x_x, y_x) & cov(x_x, y_y) \\ cov(x_y, y_x) & cov(x_y, y_y) \end{bmatrix}$$

W merupakan cross-covariance, cross-covariance memberikan informasi bagaimana perubahan pada koordinat di titik p jika koordinat titik x berubah. Cross-covariance yang ideal adalah matriks identitas, dikarenakan pada kondisi ideal tidak ada korelasi antara axis-x dan axis-y.

$$R = VU^T$$
$$t = \mu_y - R\mu_x$$

Vektor Translasi $(t = \mu_p - R\mu_x)$

Turunkan E terhadap t,

$$\frac{\partial E}{\partial t} = 2\sum_{i=1}^{n}(y_i - Rx_i - t) = 0$$
$$t = y_i - Rx_i$$

Matriks Rotasi $\left(R = VU^T\right)$

Rotasi R mencapai optimal jika translasi $t = 0$,

$$R = \underset{R \in SO(n)}{argmin} \sum_{i=1}^{n} \|Rx_i - y_i\|^2$$

$$\begin{aligned}\|Rx_i - y_i\|^2 &= (Rx_i - y_i)^T(Rx_i - y_i)\\ &= x_i^TR^TRx_i - y_i^TRx_i - x_i^TR^Ty_i - y_i^Ty_i\\ &= x_i^Tx_i - y_i^TRx_i - x_i^TR^Ty_i - y_i^Ty_i\end{aligned}$$

$x_i^TR^Ty_i$ adalah operasi skalar, sehingga $a = a^T$

$$x_i^TR^Ty_i = \left(x_i^TR^Ty_i\right)^T = y_i^TRx_i$$

$$\underset{R \in SO(d)}{\operatorname{argmin}} \sum_{i=1}^{n} \|Rx_i - y_i\|^2 = \underset{R \in SO(d)}{\operatorname{argmin}} \sum_{i=n}^{n} x_i^Tx_i - 2y_i^TRx_i - y_i^Ty_i$$

untuk mendapatkan matriks rotasi R, persamaan $x_i^Tx_i$ dan $y_i^Ty_i$ dapat diabaikan. Sehingga, untuk meminimalkan nilai χ^2, sama dengan memasimalkan nilai :

$$\underset{R \in SO(d)}{\operatorname{argmax}} \sum_{i=1}^{n} y_i^TRx_i$$

$$\sum_{i=1}^{n} y_i^TRx_i = \operatorname{tr}(Y^TRX) = \operatorname{tr}(RXY^T) = \operatorname{tr}(RW)$$

dengan sifat trace pada matrix $\operatorname{tr}(AB) = \operatorname{tr}(BA)$ dan $XY^T = W$, sehingga

$$SVD(W) = U\Sigma V^T, \text{subtitute to } tr(RXY^T)$$
$$\operatorname{tr}(RU\Sigma V^T) = \operatorname{tr}(\Sigma V^TRU)$$

tetapkan matriks $M = V^TRU$, M merupakan matriks orthogonal. Sehingga,

$$\operatorname{tr}(\Sigma V^TRU) = \operatorname{tr}(\Sigma M)$$

dengan menerapkan Cauchy-Schwarz inequality pada $\operatorname{tr}(\Sigma M)$,

$$\mathrm{tr}(\Sigma M) = \sum_{i=1}^{r} \sigma_i m_{ii}$$

$$\left(\sum_{i=1}^{r} \sigma_i m_{ii}\right)^2 \leq \sum_{i=1}^{r} \sigma_i^2 \sum_{i=1}^{r} m_{ii}^2$$

$$\sum_{i=1}^{r} \sigma_i m_{ii} \leq \sqrt{\sum_{i=1}^{r} \sigma_i^2 \sum_{i=1}^{r} m_{ii}^2}$$

$$\sum_{i=1}^{r} \sigma_i m_{ii} \leq \sum_{i=1}^{r} \sigma_i^2$$

dikarenakan M merupakan matriks orthogonal, maka $m_{ii}^2 = |m_{ii}| \leq 1$. Sehingga, untuk memperoleh nilai maksimum m_{ii} bernilai 1. Seperti yang dijelaskan diawal bahwa M merupakan matriks orthogonal, sehinggga m_{ii} bernilai 1 jika $M = I$

$$M = I = V^T RU$$
$$R^T = V^T U$$
$$R = UV^T$$

References :

- Sorkine-Hornung, O., & Rabinovich, M. (2017). Least-squares rigid motion using svd. *Computing*, *1*(1), 1-5. https://igl.ethz.ch/projects/ARAP/svd_rot.pdf
- Arun, K. S., Huang, T. S., & Blostein, S. D. (1987). Least-Squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI*-9(5), 698–700. https://doi.org/10.1109/tpami.1987.4767965

Levenberg Marquardt - Point to Line Iterative Closest Point (LM-PLICP)

Gradient-Descent

$$x_{i+1} = x_i + \lambda \nabla_{f(x)}$$

Newton-Rapshon

$$x_{i+1} = x_i + (\nabla^2_{f(x_i)})^{-1} \nabla_{f(x)}$$

Gauss-Newton

$$x_{i+1} = x_i + (J_{f(x)}^T J_{f(x)})^{-1} \nabla_{f(x)}$$

Levenberg-Marquardt

$$x_{i+1} = x_i + (\nabla_{f(x_i)}^2 + \lambda \text{ diag}(\nabla_{f(x_i)}^2))^{-1} \nabla_{f(x)}$$

Point to Line Iterative Closest Point

$$E = \underset{R,t}{\text{argmin}} \sum_{i=1}^{n} ||(y_i - Rx_i + t)n_{y,i}||^2$$

$n_{y,i}$ vektor normal pada point cloud reference dihitung dengan $\frac{-dy}{dx}$

Approximation Hessian $H = J^T J$

Fungsi loss E merupakan fungsi non-linear *differentiable* dan smooth dikarenakan terdapat matriks rotasi R, sehingga Taylor-Expansion valid untuk menyelesaikan fungsi non-linear E

$$\text{Taylor-Expansion :}$$
$$F(x + \Delta x) = F(x) + \frac{1}{1!}F'(x)\Delta x + \frac{1}{2!}F''(x)\Delta x^2 + \ldots$$

Sehingga,

$$y(x) = f(x; s)$$
$$r_i = y_i - f(x_i; s)$$

$f(x; s)$ merupakan fungsi non-linear, r_i merupakan residual, y_i merupakan target, dan s merupakan state yaitu (t_x, t_y, θ) yang akan dioptimisasi, dengan **First Order Taylor-Expansion** untuk menghitung perubahan pada state.

$$f(x_i; s') = \underbrace{f(x_i; s)}_{y_i - r_i} + \frac{\partial f}{\partial s}(s' - s) = y_i$$

Agar $f(x_1; s') = y_i$ maka $\frac{\partial f}{\partial s}(s' - s) = r_i$, ubah s menjadi (t_x, t_y, θ),

$$
\begin{aligned}
\frac{\partial f}{\partial t_x}\Delta t_x + \frac{\partial f}{\partial t_y}\Delta t_y + \frac{\partial f}{\partial \theta}\Delta\theta &= r_1 \\
\frac{\partial f}{\partial t_x}\Delta t_x + \frac{\partial f}{\partial t_y}\Delta t_y + \frac{\partial f}{\partial \theta}\Delta\theta &= r_2 \\
\vdots \quad + \quad \vdots \quad + \quad \vdots \quad &= \vdots \\
\frac{\partial f}{\partial t_x}\Delta t_x + \frac{\partial f}{\partial t_y}\Delta t_y + \frac{\partial f}{\partial \theta}\Delta\theta &= r_n
\end{aligned}
$$

Sehingga dapat disederhanakan menjadi,

$$
\begin{matrix} x = x_1 \\ \vdots \\ x = x_n \end{matrix}
\begin{bmatrix}
\frac{\partial f}{\partial t_x} & \frac{\partial f}{\partial t_y} & \frac{\partial f}{\partial \theta} \\
\vdots & \vdots & \vdots \\
\frac{\partial f}{\partial t_x} & \frac{\partial f}{\partial t_y} & \frac{\partial f}{\partial \theta}
\end{bmatrix}
\begin{bmatrix} \Delta t_x \\ \Delta t_y \\ \Delta\theta \end{bmatrix}
=
\begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}
$$

Sehingga,

$$J\Delta s = r$$

Dikarenakan matriks J bukan matriks persegi, sehingga matriks J ***non-invertible*** matriks. Agar dapat menyelesaikan persamaan tersebut, kedua sisi dikalikan dengan J^T, menjadi

$$
\begin{aligned}
(J^T J)\Delta s &= J^T r \\
\Delta s &= (J^T J)^{-1} J^T r
\end{aligned}
$$

References :

- METHODS FOR NON-LINEAR LEAST SQUARES PROBLEMS
- Nonlinear Least Squares

Pose Graph SLAM

$$e_{ij}(x) = z_{ij} - \hat{z}(x_i, x_j)$$

$e_{ij}(x)$ adalah fungsi error, z berdasarkan measurement model, $\hat{z}$ berdasarkan model matematika, dan x merupakan state. Diasumsikan error pada data sensor memiliki distribusi Gaussian,

$$p(x) = \frac{1}{\sqrt{2\pi^n |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Omega (x-\mu)\right) \sim \mathcal{N}(\mu, \Sigma)$$

$\Omega = \Sigma^{-1}$ merupakan information matrix, sehingga error merupakan multivariate normal distribution dengan $\mu = \hat{z}$ dan varians Σ. Dengan menerapkan log-likelihood pada $p(x)$, state optimal x^* dapat diperoleh dengan Maximum Likelihood Estimization, x optimal apabila $\ln(p(x))$ mencapai nilai maksimum.

$$x^* = \operatorname{argmax} p(x) = \underset{x}{\operatorname{argmin}}\ e_{ij}^T \Omega_{ij} e_{ij}$$

Dikarenakan adanya kumulatif error pada trajectory, yang merupakan error bukan hanya pada 1 pose melainkan terdapat kemungkinan error terjadi lebih dari 1 pose. Sehingga fungsi error merupakan jumlah dari setiap error pada setiap pose,

$$F(x) = e_{ij}^T \Omega_i e_{ij}$$
$$x^* = \underset{x}{\operatorname{argmin}} \sum_{i,j} F(x_{ij})$$

Persamaan tersebut tidak memiliki solusi pasti dikarenakan terdapat matriks rotasi yang mengindikasikan persamaan non-linear. Permasalahan dapat diselesaikan dengan menggunakan optimisasi dan dilinearkan menggunakan Taylor-Expansion.

$$F_{ij}(x + \Delta x) = e_{ij}(x + \Delta x)^T \Omega_{ij} e_{ij}(x + \Delta x)^T$$

Dengan menggunakan **First-Order Taylor Expansion**,

$$e_{ij}(x + \Delta x) = e_{ij} + J_{ij} \Delta x$$

Subtitusikan ke $F_{ij}(x + \Delta x)$,

$$\begin{aligned}
F_{ij}(x + \Delta x) &= (e_{ij} + J_{ij}\Delta x)^T \Omega_{ij} (e_{ij} + J_{ij}\Delta x) \\
&= (e_{ij}^T + (J_{ij}\Delta x)^T) + (\Omega_{ij} e_{ij} + \Omega_{ij} J_{ij} \Delta x)(\\
&= (e_{ij}^T + \Delta x^T J_{ij}^T)(\Omega_{ij} e_{ij} + \Omega_{ij} J_{ij} \Delta x) \\
&= e_{ij}^T \Omega_{ij} e_{ij} + e_{ij}^T \Omega_{ij} J_{ij} \Delta x + \Delta x^T J_{ij}^T \Omega_{ij} e_{ij} + \Delta x^T J_{ij}^T \Omega_{ij} J_{ij} \Delta x \\
&= e_{ij}^T \Omega_{ij} e_{ij} + e_{ij}^T \Omega_{ij} J_{ij} \Delta x + \left(\Delta x^T J_{ij}^T \Omega_{ij} e_{ij}\right)^T + \Delta x^T J_{ij}^T \Omega_{ij} J_{ij} \Delta x \\
&= e_{ij}^T \Omega_{ij} e_{ij} + e_{ij}^T \Omega_{ij} J_{ij} \Delta x + e_{ij}^T \Omega_{ij}^T J_{ij} \Delta x + \Delta x^T J_{ij}^T \Omega_{ij} J_{ij} \Delta x
\end{aligned}$$

Dikarenakan Ω_{ij} adalah metriks simetrik, maka $\Omega_{ij} = \Omega_{ij}^T$

$$F_{ij}(x + \Delta x) = \underbrace{e_{ij}^T \Omega_{ij} e_{ij}}_{c} + 2\underbrace{e_{ij}^T \Omega_{ij} J_{ij}}_{b} \Delta x + \Delta x^T \underbrace{J_{ij}^T \Omega_{ij} J_{ij}}_{H} \Delta x$$

Dapat diperoleh turunan pertama dari $F_{ij}(x + \Delta x)$,

$$\frac{\partial F_{ij}(x + \Delta x)}{\partial \Delta x} \approx 2b + 2H\Delta x = 0$$
$$\Delta x = -H^{-1} b$$
$$x \leftarrow x + \Delta x$$

$\hat{z}_{ij}$ dihitung dengan *inverse pose composition*,

$$T \in SE(n)$$
$$R \in SO(n)$$
$$T_i^{-1} = \begin{bmatrix} R_i^T & -R_i^T t_i \\ 0 & 1 \end{bmatrix}$$
$$T_j = \begin{bmatrix} R_j & t_j \\ 0 & 1 \end{bmatrix}$$

$$\hat{z}_{ij} = T_i^{-1} \cdot T_j$$
$$\hat{z}_{ij} = \begin{bmatrix} R_i^T R_j & R_i^T (t_j - t_i) \\ 0 & 1 \end{bmatrix} \triangleq \begin{bmatrix} R_i^T (t_j - t_i) \\ \theta_j - \theta_i \end{bmatrix} \rightarrow \text{vector space}$$

$$\text{vector space} = \begin{bmatrix} \Delta t \\ \Delta \theta \end{bmatrix}$$

Nilai e_{ij} diperoleh dengan cara yang sama seperti $\hat{z}_{ij}$, Sehingga $e_{ij} = z_{ij}^{-1} T_i^{-1} T_j$

$$e_{ij} = \begin{bmatrix} R_{ij}^T (R_i^T (t_j - t_i) - t_{ij}) \\ \theta_j - \theta_i - \theta_{ij} \end{bmatrix} \rightarrow \text{vector space}$$

Tentukan turunan parsial e_{ij} terhadap x_i dan x_j,

$$x_i = \begin{bmatrix} t_i \\ \theta_i \end{bmatrix};\ x_j = \begin{bmatrix} t_j \\ \theta_j \end{bmatrix}$$
$$\frac{\partial e_{ij}}{\partial x_i} = \begin{bmatrix} \frac{\partial \Delta t}{\partial t_i} & \frac{\partial \Delta t}{\partial \theta_i} \\ \frac{\partial \Delta \theta}{\partial t_i} & \frac{\partial \Delta \theta}{\partial \theta_i} \end{bmatrix},\ \frac{\partial e_{ij}}{\partial x_i} = \begin{bmatrix} \frac{\partial \Delta t}{\partial t_j} & \frac{\partial \Delta t}{\partial \theta_j} \\ \frac{\partial \Delta \theta}{\partial t_j} & \frac{\partial \Delta \theta}{\partial \theta_j} \end{bmatrix}$$
$$\frac{\partial e_{ij}}{\partial x_i} = \begin{bmatrix} -R_{ij}^T R_i^T & R_{ij}^T \frac{\partial R_i^T}{\partial \theta_i} (t_j - t_i) \\ 0 & -1 \end{bmatrix}$$
$$\frac{\partial e_{ij}}{\partial x_j} = \begin{bmatrix} R_{ij}^T R_i^T & 0 \\ 0 & 1 \end{bmatrix}$$

Jacobian matriks J_{ij} hanya berisikan nilai $\frac{\partial e_{ij}}{\partial x_i}$ pada baris ke-i, $\frac{\partial e_{ij}}{\partial x_j}$ pada baris ke-j, dan yang lainnya bernilai 0

$$J_{ij} = \begin{bmatrix} 0 & 0 & 0 & \dots & \frac{\partial e_{ij}}{\partial x_i} & \dots & \frac{\partial e_{ij}}{\partial x_j} & \dots & 0 & 0 & 0 \end{bmatrix}$$

$$\nabla_{ij} = J_{ij}^T \Omega_{ij} e_{ij}$$
$$H_{ij} = J_{ij}^T \Omega_{ij} J_{ij}$$

$$\nabla = \sum_{i,j} J_{ij}^T \Omega_{ij} e_{ij}$$
$$H = \sum_{i,j} J_{ij}^T \Omega_{ij} J_{ij}$$

Setelah ∇ dan H dapat diperoleh, permasalahan Non-Linear Least Square dapat diselesaikan dengan iterasi proses optimisasi menggunakan Gauss-Newton atau Levenberg-Marquardt hingga $\Delta F(x) < \text{tolerance}$.

$$\text{(GN) } \Delta x = -H^{-1} \nabla$$
$$\text{(LM) } \Delta x = -(H + \lambda \operatorname{diag}(H))^{-1} \nabla$$
$$x \leftarrow x + \Delta x$$

References :

- [SLAM][En] Errors and Jacobian Derivations for SLAM Part 1
- A Technical Walkthrough of the SLAM Back-end
- Graph-based SLAM using Pose Graphs (Cyrill Stachniss)