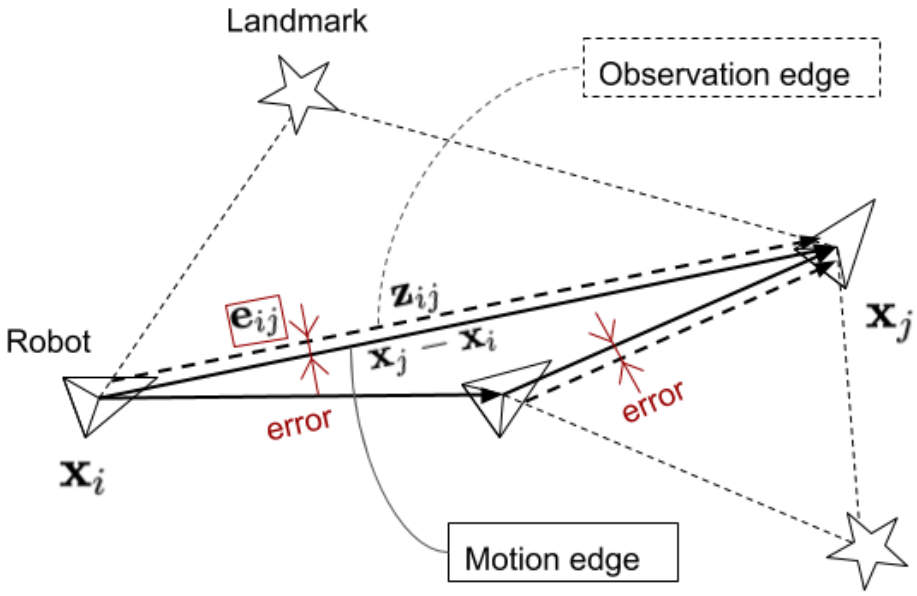
グラフベースSLAM

1　グラフベースSLAMとは

グラフベースSLAMとはオフラインSLAMの一種で、収集した全ての観測データを用いてロボットの過去も含めた経路とランドマークの位置を修正する手法である。本手法は時刻 t_i におけるロボットの姿勢を頂点、時刻 t_i での姿勢と時刻 t_j での姿勢の2点間のベクトルを辺とみなして、動作辺 (オドメトリなど内界センサにより得られた情報) と観測辺 (レーザーレンジファインダなどの外界センサにより得られた情報) との間の誤差を全て考慮したコスト関数を最小化するロボットの経路とランドマークの位置を推測する。



時刻 t_i と時刻 t_j 間での動作辺と観測辺との誤差 $\boldsymbol{e}_{ij}$ は、

$$\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_j - \boldsymbol{x}_i) - \boldsymbol{z}_{ij}$$

ここで $\boldsymbol{x}_i$ と $\boldsymbol{x}_j$ はそれぞれ時刻 t_i と時刻 t_j でのロボットの姿勢、 $\boldsymbol{z}_{ij}$ は時刻 t_i と時刻 t_j で同じランドマークを観測した際に得られる観測辺である。従ってコスト関数 $F(\boldsymbol{x}_{0:t})$ は、

$$F(\boldsymbol{x}_{0:t}) = \sum_{i,j} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)^{\mathrm{T}} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

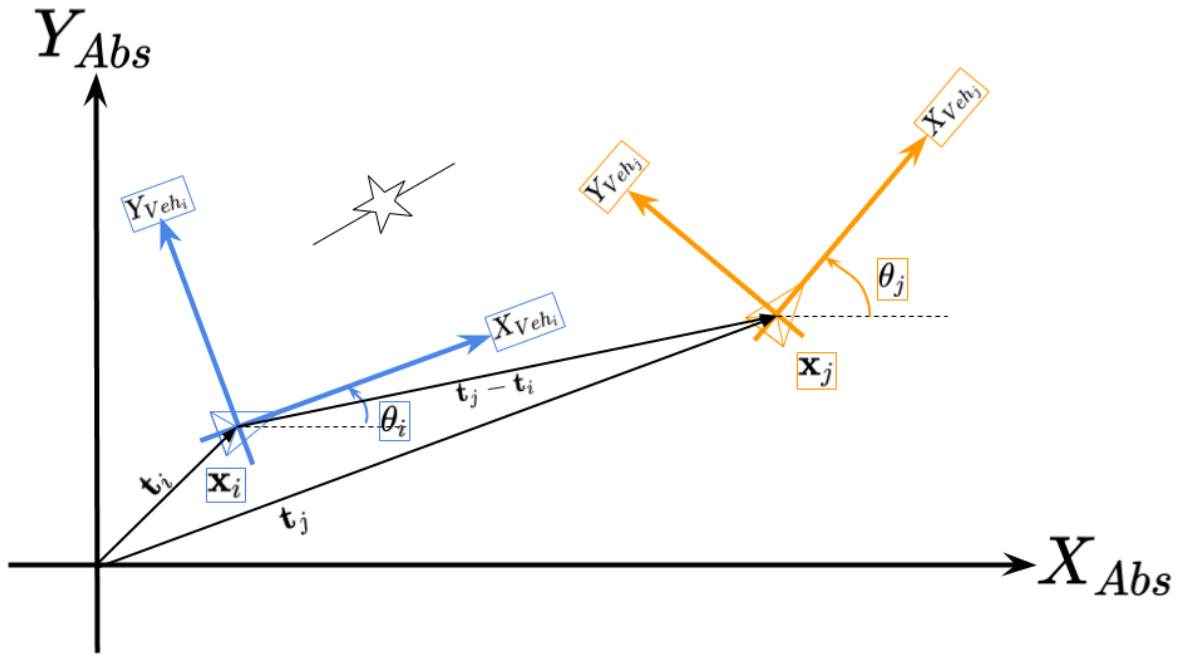
と、情報の正確さを表す Ω_{ij} で重み付けされた二乗誤差 $\boldsymbol{e}_{ij}^{\mathrm{T}} \boldsymbol{e}_{ij}$ (マハラノビス距離) の総和で表すことができる。グラフベースSLAMとは、このコスト関数を最小化するロボットの経路及びランドマークの位置を探す事である。

2　定義

本稿で使用する座標系と文字をいくつか定義する。

2.1　座標系

基準となる絶対座標系上に、ロボットの姿勢が 2 つ、ランドマークが 1 つあるとする。ロボットは自分を原点とした車両座標系を持ち、車両前方方向を X 軸、それに垂直な左手方向を Y 軸とする。また、ランドマーク自身も位置だけでなく各々の角度を持っているとする。しかしそのランドマークの角度の値自体は重要ではない (後述)。



2.2　ロボットの位置ベクトル: $\boldsymbol{t}_i$

ロボットの絶対座標系における x_i 座標と y_i 座標から成るベクトル。

$$\boldsymbol{t}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

2.3　ロボットの姿勢ベクトル: $\boldsymbol{x}_i$

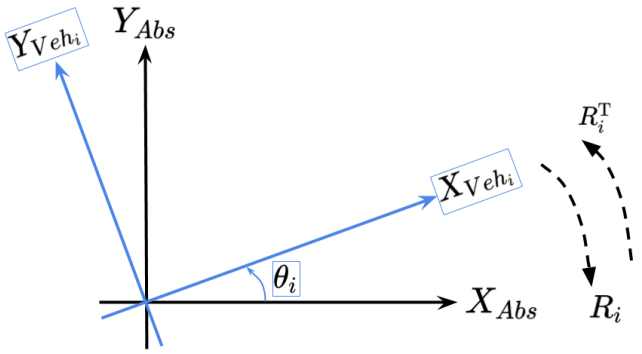
しかしながらロボットはもう一つ、絶対座標系の X 軸から車両座標系の X 軸までの角度であるヨー角度というパラメータを持っている。

$$\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{t}_i \\ \theta_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix}$$

2.4 回転行列: R_i

回転行列による座標変換は、座標系を時計回りに回転させる。特に R_i は、車両 $_i$ 座標を絶対座標に回転変換する。

$$R_i = \begin{pmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{pmatrix}$$



2.5 姿勢表現行列: X_i

本稿では、回転行列 R_i と位置ベクトル t_i から構成される姿勢表現行列によってロボットの姿勢を表すこととする。

$$X_i = \begin{pmatrix} R_i & \boldsymbol{t}_i \\ 00 & 1 \end{pmatrix} = \begin{pmatrix} cos\theta_i & -sin\theta_i & x_i \\ sin\theta_i & cos\theta_i & y_i \\ 0 & 0 & 1 \end{pmatrix}$$

$$Abs \leftarrow Veh_i \qquad Abs$$
$$X_i = \begin{pmatrix} R_i & \mathbf{t}_i \\ 00 & 1 \end{pmatrix}$$

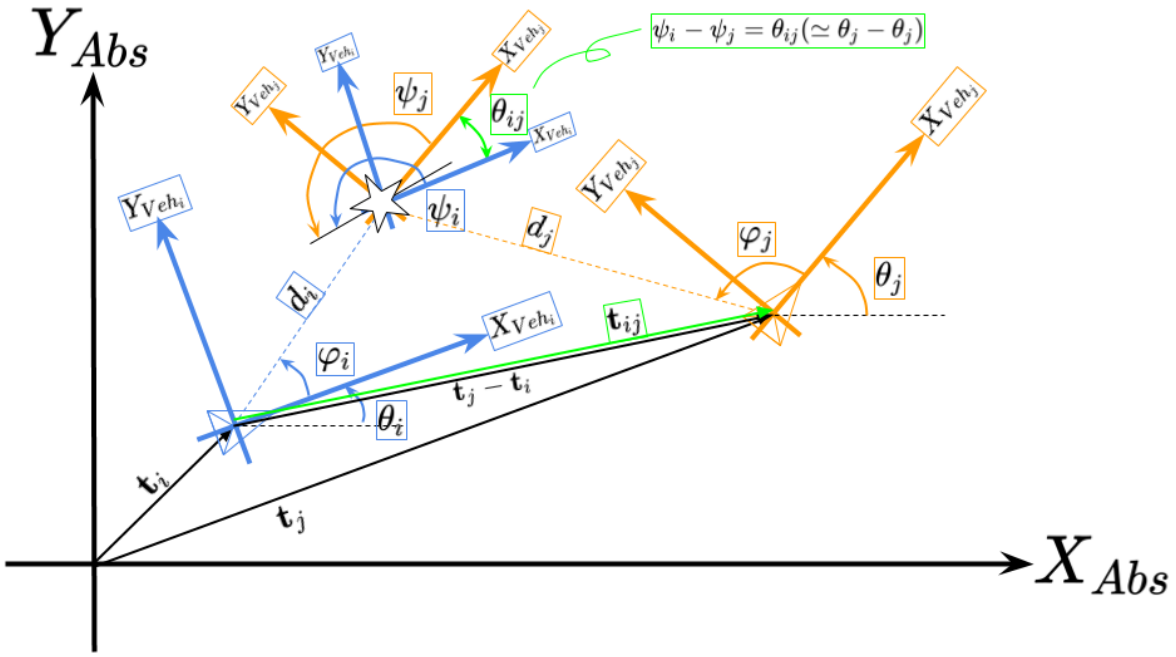
姿勢行列が自身の車両座標系から絶対座標系に座標変換するのに対し、姿勢表現行列の逆行列は絶対座標系から自身の車両座標系に座標変換することを意味する。特に X_i^{T} は絶対座標系から車両 $_i$ 座標系に変換する。

$$X_i^{-1} = \begin{pmatrix} R_i^{\mathrm{T}} & -R_i^{\mathrm{T}}\boldsymbol{t}_i \\ 00 & 1 \end{pmatrix}$$

$$Veh_i \leftarrow Abs \qquad Abs$$
$$X_i^{-1} = \begin{pmatrix} R_i^{\mathrm{T}} & -R_i^{\mathrm{T}}\mathbf{t}_i \\ 00 & 1 \end{pmatrix}$$

2.6　センサーモデル

ロボットに積載されている外界センサは、ロボットからランドマークへの距離 d と角度 φ の 2 つの情報を計測する。時刻 t_i と時刻 t_j において同じランドマークを観測した際、それぞれの時刻で観測されたランドマークの外見の差分を取ることによって、ランドマーク自身の角度の相対的変化 $\psi_i - \psi_j$ を計算することができる。ロボットは ψ_i と ψ_j の絶対的な値は知ることはできず、差分を取ることでその相対値のみを知ることができる。下記で定義する観測表現行列 Z_{ij} の中にある θ_{ij} は、この各時刻で観測されたランドマーク自身の角度の差分 (相対角度) $\psi_i - \psi_j$ の事を表している。

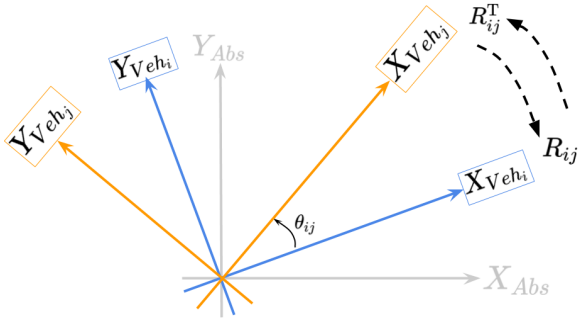


2.7　観測表現行列: Z_{ij}

違う地点から同じランドマークを観測すると、観測されたランドマークを基準にその 2 点間の相対的なロボットの姿勢の差を計算できる。時刻 t_i と時刻 t_j において同じランドマークを観測したと仮定すると観測表現行列 Z_{ij} は、

$$Z_{ij} = \begin{pmatrix} R_{ij} & \mathbf{t}_{ij} \\ 00 & 1 \end{pmatrix} = \begin{pmatrix} cos\theta_{ij} & -sin\theta_{ij} & x_{ij} \\ sin\theta_{ij} & cos\theta_{ij} & y_{ij} \\ 0 & 0 & 1 \end{pmatrix}$$

$$Z_{ij} = \begin{pmatrix} R_{ij} & \mathbf{t}_{ij} \\ 00 & 1 \end{pmatrix}$$



観測表現行列 Z_{ij} の原点は車両 $_i$ 座標系上にあるので、 $\mathbf{t}_{ij}$ と θ_{ij} はそれぞれ車両 $_i$ 座標系から車両 $_j$ 座標系への距離と角度を示している。

3　最適化

動作辺と観測辺との誤差から計算されたコスト関数は、ガウス・ニュートン法などの最小二乗法により最適化する事ができる。

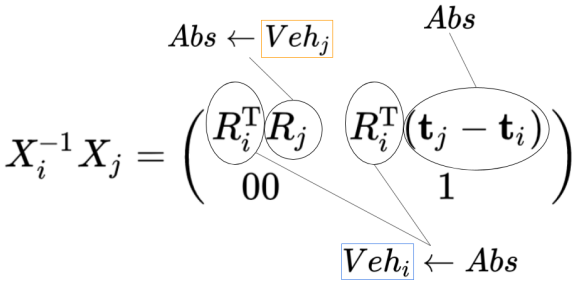
3.1　誤差関数とコスト関数

理想的な環境においては、観測表現行列 Z_{ij} に含まれている $\boldsymbol{t}_{ij}$ と $\theta_{ij}(=\psi_i-\psi_j)$ は、各姿勢間の差 $\boldsymbol{t}_j-\boldsymbol{t}_i$ と $\theta_j-\theta_i$ にそれぞれ一致するはずである。しかし実世界では、これらの値はセンサ誤差によって差が出てくる。この差を表したのが誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ である。

$$\boldsymbol{e}_{ij}(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_j-\boldsymbol{x}_i)-\boldsymbol{z}_{ij}$$

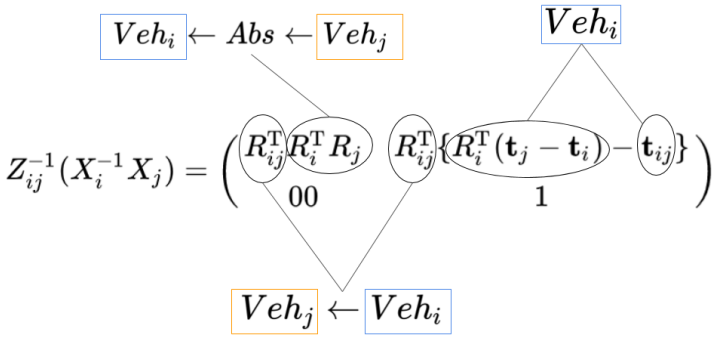
この誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ は、前節で導入した姿勢表現行列 X_i, X_j と、観測表現行列 Z_{ij} で表現することができる。まず、時刻 t_i と時刻 t_j での姿勢間の差を表す動作辺 (オドメトリなど内界センサにより得られた情報) は、時刻 t_i での姿勢表現行列の逆行列 X_i^{-1} を時刻 t_j での姿勢表現行列 X_j にかける事で表現することができる。

$$\begin{aligned}X_i^{-1}X_j&=\begin{pmatrix}R_i^{\mathrm{T}} & -R_i^{\mathrm{T}}\boldsymbol{t}_i\\ 00 & 1\end{pmatrix}\begin{pmatrix}R_j & \boldsymbol{t}_j\\ 00 & 1\end{pmatrix}\\&=\begin{pmatrix}R_i^{\mathrm{T}}R_j & R_i^{\mathrm{T}}(\boldsymbol{t}_j-\boldsymbol{t}_i)\\ 00 & 1\end{pmatrix}\end{aligned}$$



次に、動作辺 (オドメトリなど内界センサにより得られた情報) とそれに対応する観測辺 (レーザーレンジファインダなどの外界センサにより得られた情報) との間の誤差 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ は、観測表現行列の逆行列 Z_{ij}^{-1} をかける事で表現することができる。

$$\begin{aligned}Z_{ij}^{-1}(X_i^{-1}X_j)&=\begin{pmatrix}R_{ij}^{\mathrm{T}} & -R_{ij}^{\mathrm{T}}\boldsymbol{t}_{ij}\\ 00 & 1\end{pmatrix}\begin{pmatrix}R_i^{\mathrm{T}}R_j & R_i^{\mathrm{T}}(\boldsymbol{t}_j-\boldsymbol{t}_i)\\ 00 & 1\end{pmatrix}\\&=\begin{pmatrix}R_{ij}^{\mathrm{T}}R_i^{\mathrm{T}}R_j & R_{ij}^{\mathrm{T}}\{R_i^{\mathrm{T}}(\boldsymbol{t}_j-\boldsymbol{t}_i)-\boldsymbol{t}_{ij}\}\\ 00 & 1\end{pmatrix}\end{aligned}$$



誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ はこの行列 $Z_{ij}^{-1}(X_i^{-1}X_j)$ の平行移動要素 $R_{ij}^{\mathrm{T}}\{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\}$ と回転要素 $R_{ij}^{\mathrm{T}}R_i^{\mathrm{T}}R_j$ (の角度) によって構成されるので、

$$\begin{aligned}\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \begin{pmatrix} R_{ij}^{\mathrm{T}}\{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} \\ angle(R_{ij}^{\mathrm{T}}R_i^{\mathrm{T}}R_j) \end{pmatrix} \\ &= \begin{pmatrix} R_{ij}^{\mathrm{T}}\{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} \\ (\theta_j - \theta_i) - \theta_{ij} \end{pmatrix}\end{aligned}$$

グラフベース SLAM の目標は、重み付けされた二乗誤差 (マハラノビス距離) と最小二乗法 (ガウス・ニュートン法) を用いて、この動作辺 (オドメトリなど内界センサにより得られた情報) とそれに対応する観測辺 (レーザーレンジファインダなどの外界センサにより得られた情報) との間の誤差 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ を最小にすることである。この時のコスト関数は、全ての観測データにおける重み付け二乗誤差の総和である。

$$\begin{aligned}F(\boldsymbol{x}_{0:t}) &= \sum_{i,j} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)^{\mathrm{T}}\Omega_{ij}\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})\end{aligned}$$

3.2 線形化

このコスト関数を最小化するのにガウス・ニュートンを用いる。まず、誤差関数をロボットの初期経路 $\boldsymbol{x}_{0:t}$ 周りでテイラー展開し、1 次近似する。

$$\begin{aligned}F(\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}) &= \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}) \\ &\simeq (\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}) + J_{0:t}\Delta\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}(\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}) + J_{0:t}\Delta\boldsymbol{x}_{0:t}) \\ &= \{\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}} + (J_{0:t}\Delta\boldsymbol{x}_{0:t})^{\mathrm{T}}\}\Omega_{0:t}(\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}) + J_{0:t}\Delta\boldsymbol{x}_{0:t}) \\ &= \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}) + (J_{0:t}\Delta\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}) \\ &\quad + \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}(J_{0:t}\Delta\boldsymbol{x}_{0:t}) + (J_{0:t}\Delta\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}(J_{0:t}\Delta\boldsymbol{x}_{0:t}) \\ &= F(\boldsymbol{x}_{0:t}) + \{(\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}))^{\mathrm{T}}(J_{0:t}\Delta\boldsymbol{x}_{0:t})\}^{\mathrm{T}} + \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}^{\mathrm{T}}J_{0:t}^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t}\end{aligned}$$

$\Omega_{0:t}$ は対称行列、また $\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t}$ はスカラーなので、

$$\begin{aligned}F(\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}) &\simeq F(\boldsymbol{x}_{0:t}) + (\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t})^{\mathrm{T}} + \boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}^{\mathrm{T}}J_{0:t}^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t} \\ &= F(\boldsymbol{x}_{0:t}) + 2\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t})^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}^{\mathrm{T}}J_{0:t}^{\mathrm{T}}\Omega_{0:t}J_{0:t}\Delta\boldsymbol{x}_{0:t} \\ &= F(\boldsymbol{x}_{0:t}) + 2\boldsymbol{b}_{0:t}^{\mathrm{T}}\Delta\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}^{\mathrm{T}}H_{0:t}\Delta\boldsymbol{x}_{0:t}\end{aligned}$$

ここで

$$\boldsymbol{b}_{0:t} = J_{0:t}^{\mathrm{T}}\Omega_{0:t}\boldsymbol{e}_{0:t}(\boldsymbol{x}_{0:t}), \quad H_{0:t} = J_{0:t}^{\mathrm{T}}\Omega_{0:t}J_{0:t}$$

3.3 経路更新

$\boldsymbol{x}_{0:t}$ を定数、$\Delta\boldsymbol{x}_{0:t}$ を変数とみなすと、$F(\boldsymbol{x}_{0:t}+\Delta\boldsymbol{x}_{0:t})$ を $\Delta\boldsymbol{x}_{0:t}$ で微分することによって、$F(\boldsymbol{x}_{0:t}+\Delta\boldsymbol{x}_{0:t})$ を最も減らす方向の $\Delta\boldsymbol{x}_{0:t}$ を算出することができる。

$$\frac{\partial F(\boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t})}{\partial\Delta\boldsymbol{x}_{0:t}} \simeq 2\boldsymbol{b}_{0:t} + (H_{0:t} + H_{0:t}^{\mathrm{T}})\Delta\boldsymbol{x}_{0:t} = 0$$

$H_{0:t}$ も対称行列なので (なぜなら $\Omega_{0:t}$ が対称行列であるため)、

$$\begin{aligned}2\boldsymbol{b}_{0:t} + 2H_{0:t}\Delta\boldsymbol{x}_{0:t} &= 0 \\ \Delta\boldsymbol{x}_{0:t} &= -H_{0:t}^{-1}\boldsymbol{b}_{0:t}\end{aligned}$$

このオフセット $\Delta\boldsymbol{x}_{0:t}$ を初期経路 $\boldsymbol{x}_{0:t}$ に足し合わせることによって、最適化後のロボットの推測経路を計算することができる。

$$\boldsymbol{x}'_{0:t} = \boldsymbol{x}_{0:t} + \Delta\boldsymbol{x}_{0:t}$$

最後に、前回の計算結果を用いて $H_{0:t}$ と $\boldsymbol{b}_{0:t}$ を再計算し、$\Delta\boldsymbol{x}_{0:t}$ を加算して経路の更新するという上記の流れを、収束するまで**繰り返す**。

3.4 系全体の情報行列

各辺の情報は、加算によって系全体の情報に反映される。系全体の情報に加算される各辺の情報 H_{ij} と $\boldsymbol{b}_{ij}$ を計算するために、誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ のヤコビアン行列 J_{ij} を 2 つの部分行列に分ける。誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ はロボットの各姿勢 $\boldsymbol{x}_i$ と $\boldsymbol{x}_j$ にしか依存しないので、2 つに分ける部分行列の片方は $\boldsymbol{x}_i$ に関するもので、もう片方は $\boldsymbol{x}_i$ に関するものを表す。

$$J_{ij} = \frac{\partial \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial (\boldsymbol{x}_i, \boldsymbol{x}_j)} = \begin{pmatrix} \frac{\partial e_{ij_x}}{\partial x_i} & \frac{\partial e_{ij_x}}{\partial y_i} & \frac{\partial e_{ij_x}}{\partial \theta_i} & \frac{\partial e_{ij_x}}{\partial x_j} & \frac{\partial e_{ij_x}}{\partial y_j} & \frac{\partial e_{ij_x}}{\partial \theta_j} \\ \frac{\partial e_{ij_y}}{\partial x_i} & \frac{\partial e_{ij_y}}{\partial y_i} & \frac{\partial e_{ij_y}}{\partial \theta_i} & \frac{\partial e_{ij_y}}{\partial x_j} & \frac{\partial e_{ij_y}}{\partial y_j} & \frac{\partial e_{ij_y}}{\partial \theta_j} \\ \frac{\partial e_{ij_\theta}}{\partial x_i} & \frac{\partial e_{ij_\theta}}{\partial y_i} & \frac{\partial e_{ij_\theta}}{\partial \theta_i} & \frac{\partial e_{ij_\theta}}{\partial x_j} & \frac{\partial e_{ij_\theta}}{\partial y_j} & \frac{\partial e_{ij_\theta}}{\partial \theta_j} \end{pmatrix} = \begin{pmatrix} A_{ij} & B_{ij} \end{pmatrix}$$

ここで A_{ij} と B_{ij} は、誤差関数 $\boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ をそれぞれ $\boldsymbol{x}_i$ と $\boldsymbol{x}_j$ で微分した部分行列であり、

$$\begin{aligned} A_{ij} &= \begin{pmatrix} \frac{\partial e_{ij_x}}{\partial x_i} & \frac{\partial e_{ij_x}}{\partial y_i} & \frac{\partial e_{ij_x}}{\partial \theta_i} \\ \frac{\partial e_{ij_y}}{\partial x_i} & \frac{\partial e_{ij_y}}{\partial y_i} & \frac{\partial e_{ij_y}}{\partial \theta_i} \\ \frac{\partial e_{ij_\theta}}{\partial x_i} & \frac{\partial e_{ij_\theta}}{\partial y_i} & \frac{\partial e_{ij_\theta}}{\partial \theta_i} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial \boldsymbol{t}_i} R_{ij}^{\mathrm{T}} \{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} & \frac{\partial}{\partial \theta_i} R_{ij}^{\mathrm{T}} \{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} \\ \frac{\partial}{\partial \boldsymbol{t}_i} \{(\theta_j - \theta_i) - \theta_{ij}\} & \frac{\partial}{\partial \theta_i} \{(\theta_j - \theta_i) - \theta_{ij}\} \end{pmatrix} \\ &= \begin{pmatrix} -R_{ij}^{\mathrm{T}} R_i^{\mathrm{T}} & R_{ij}^{\mathrm{T}} \frac{\partial R_i^{\mathrm{T}}}{\partial \theta_i} (\boldsymbol{t}_j - \boldsymbol{t}_i) \\ 00 & -1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} B_{ij} &= \begin{pmatrix} \frac{\partial e_{ij_x}}{\partial x_j} & \frac{\partial e_{ij_x}}{\partial y_j} & \frac{\partial e_{ij_x}}{\partial \theta_j} \\ \frac{\partial e_{ij_y}}{\partial x_j} & \frac{\partial e_{ij_y}}{\partial y_j} & \frac{\partial e_{ij_y}}{\partial \theta_j} \\ \frac{\partial e_{ij_\theta}}{\partial x_j} & \frac{\partial e_{ij_\theta}}{\partial y_j} & \frac{\partial e_{ij_\theta}}{\partial \theta_j} \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial \boldsymbol{t}_j} R_{ij}^{\mathrm{T}} \{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} & \frac{\partial}{\partial \theta_j} R_{ij}^{\mathrm{T}} \{R_i^{\mathrm{T}}(\boldsymbol{t}_j - \boldsymbol{t}_i) - \boldsymbol{t}_{ij}\} \\ \frac{\partial}{\partial \boldsymbol{t}_j} \{(\theta_j - \theta_i) - \theta_{ij}\} & \frac{\partial}{\partial \theta_j} \{(\theta_j - \theta_i) - \theta_{ij}\} \end{pmatrix} \\ &= \begin{pmatrix} R_{ij}^{\mathrm{T}} R_i^{\mathrm{T}} & \mathbf{0} \\ 00 & 1 \end{pmatrix} \end{aligned}$$

部分行列 H_{ij} と部分ベクトル $\boldsymbol{b}_{ij}$ は A_{ij} と B_{ij} を用いて下記の様に書き直すことができる。

$$H_{ij} = \begin{pmatrix} A_{ij}^{\mathrm{T}} \\ B_{ij}^{\mathrm{T}} \end{pmatrix} \Omega_{ij} \begin{pmatrix} A_{ij} & B_{ij} \end{pmatrix} = \begin{pmatrix} A_{ij}^{\mathrm{T}} \Omega_{ij} A_{ij} & A_{ij}^{\mathrm{T}} \Omega_{ij} B_{ij} \\ B_{ij}^{\mathrm{T}} \Omega_{ij} A_{ij} & B_{ij}^{\mathrm{T}} \Omega_{ij} B_{ij} \end{pmatrix}$$

$$\boldsymbol{b}_{ij} = \begin{pmatrix} A_{ij}^{\mathrm{T}} \\ B_{ij}^{\mathrm{T}} \end{pmatrix} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{pmatrix} A_{ij}^{\mathrm{T}} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ B_{ij}^{\mathrm{T}} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{pmatrix}$$

従って系全体の情報行列 $H_{0:t}$ と系全体の情報ベクトル $\boldsymbol{b}_{0:t}$ は、これらの部分行列 H_{ij} と部分ベクトル $\boldsymbol{b}_{ij}$ をそれぞれ対応する箇所に加算することで得ることができる。

$$\begin{aligned} H_{0:t_{[ii]}} &+= A_{ij}^{\mathrm{T}} \Omega_{ij} A_{ij} & H_{0:t_{[ij]}} &+= A_{ij}^{\mathrm{T}} \Omega_{ij} B_{ij} \\ H_{0:t_{[ji]}} &+= B_{ij}^{\mathrm{T}} \Omega_{ij} A_{ij} & H_{0:t_{[jj]}} &+= B_{ij}^{\mathrm{T}} \Omega_{ij} B_{ij} \end{aligned}$$

$$\begin{aligned} \boldsymbol{b}_{0:t_{[i]}} &+= A_{ij}^{\mathrm{T}} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \boldsymbol{b}_{0:t_{[j]}} &+= B_{ij}^{\mathrm{T}} \Omega_{ij} \boldsymbol{e}_{ij}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

4　ソースコード

実装したソースコードの一部分を下記に示す。$\boldsymbol{x} = (x, y, \theta)^{\mathrm{T}}$ である。

4.1　誤差関数とコスト関数

```
# Local information matrix 'Omega' (from a dataset file)
Omega = edge_ij.info_matrix # 3 by 3 matrix

# Pose representation matrix 'X_i' and
# Rotation matrix 'R_i' on Vehicle_i coordinate
X_i = vec2mat(node_i) # 3 by 3 matrix
R_i = X_i[0:2, 0:2]   # 2 by 2 matrix

# Pose representation matrix 'X_j' on Vehicle_j coordinate
X_j = vec2mat(node_j) # 3 by 3 matrix

# Observation representation matrix 'Z_ij' and
# Rotation matrix 'R_ij' on Vehicle_j coordinate
Z_ij = vec2mat(edge_ij.mean) # 3 by 3 matrix
R_ij = Z_ij[0:2, 0:2]        # 2 by 2 matrix

# Error between edges 'e'
e = mat2vec(Z_ij.I * X_i.I * X_j) # 3 by 1 matrix
```

4.2　線形化

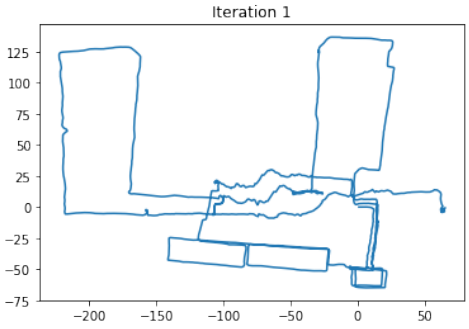
```
# Differentail of 'R_i' ... d(R_i)/d(yaw_i)
dR_dyaw_i = np.mat([
    [-s_i, -c_i], # [-sin(yaw_i), -cos(yaw_i)],
    [c_i,  -s_i]  # [ cos(yaw_i), -sin(yaw_i)]
])
# Robot position vector 't_i', 't_j'
t_i = node_i[0:2, 0] # 2*1 matrix, [x_i, y_i]
t_j = node_j[0:2, 0] # 2*1 matrix, [x_j, y_j]

# Separated Jacobian matrix 'A_ij' which is regarding to 'x_i'
A = np.mat(np.zeros((3, 3)))                    # 3 by 3 matrix with all zeros
A[0:2, 0:2] = -R_ij.T * R_i.T                   # Top left 2 by 2 elements
A[0:2, 2:3] = R_ij.T * dR_dyaw_i.T * (t_j - t_i) # Top right 2 by 1 elements
A[2:3, 0:3] = np.mat([0, 0, -1])                # Bottom 1 by 3 elements

# Separated Jacobian matrix 'B_ij' which is regarding to 'x_j'
B = np.mat(np.zeros((3, 3)))    # 3 by 3 matrix with all zeros
B[0:2, 0:2] = R_ij.T * R_i.T    # Top left 2 by 2 elements
B[0:2, 2:3] = np.mat([0, 0]).T  # Top right 2 by 1 elements
B[2:3, 0:3] = np.mat([0, 0, 1]) # Bottom 1 by 3 elements

# Information sub-matrix of the system 'H_ii', 'H_ij', 'H_ji', 'H_jj'
H_ii = A.T * Omega * A;     H_ij = A.T * Omega * B
H_ji = B.T * Omega * A;     H_jj = B.T * Omega * B

# Adding the sub-matrix into the information matrix of the system 'H'
self.H[i_idx[0]:i_idx[1], i_idx[0]:i_idx[1]]+=H_ii;
self.H[i_idx[0]:i_idx[1], j_idx[0]:j_idx[1]]+=H_ij
self.H[j_idx[0]:j_idx[1], i_idx[0]:i_idx[1]]+=H_ji;
self.H[j_idx[0]:j_idx[1], j_idx[0]:j_idx[1]]+=H_jj

# Information sub-vector of the system 'b_i','b_j'
b_i = A.T * Omega * e
b_j = B.T * Omega * e

# Adding the sub-vector into the information vector of the system 'b'
self.b[i_idx[0]:i_idx[1]] += b_i
self.b[j_idx[0]:j_idx[1]] += b_j
```

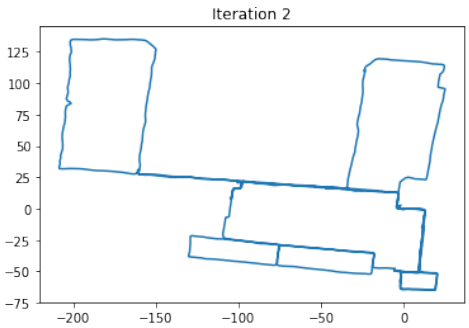
4.3 経路更新

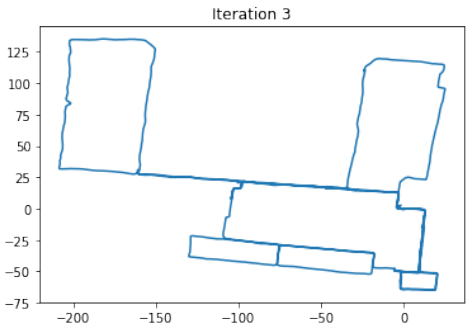
```
# Add an Identity matrix to fix the first pose, `x0` and `y0`, as the origin
H[0:3, 0:3] += np.eye(3)

# Make sparse matrix of `H`
H_sparse = scipy.sparse.csc_matrix(H) # 3`n_node` by 3`n_node` matrix

# `H`^-1
H_sparse_inv = scipy.sparse.linalg.splu(H_sparse)

# `dx` = -`H`^-1 * `b`
dx = -H_sparse_inv.solve(self.b) # 3`n_node` by 1 matrix

# Reshape
dx = dx.reshape([3, self.n_node], order='F')  # 3 by `n_node` matrix

# Update
for i in range(self.n_node):
    self.node[i].pose += dx[:, i]
```

4.4 結果

前章 3.3 を繰り返した内の 3 回分を結果を以下に示す。







同じ経路のはずにも関わらず違う経路だと認識されていた箇所が、元通りになっている事がわかる。

References

[1] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM", IEEE Intelligent Transportation Systems Magazine, 2010.

[2] GitHub - deleji/graph-slam: 一个二⊠平面的激光⊠⊠化例子, https://github.com/deleji/graph-slam

[3] Udacity - Artificial Intelligence for Robotics: Implementing SLAM, https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373