Invisible Math

Kawin Nikomborirak

June 25, 2019

1 Principle

$$
\begin{aligned}
\underset{\mathbf{w}}{\operatorname{argmax}} f(\mathbf{w}) &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{y}_1 \dots \mathbf{y}_n | \mathbf{x}_1 \dots \mathbf{x}_n, \mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \log P(\mathbf{y}_1 \dots \mathbf{y}_n | \mathbf{x}_1 \dots \mathbf{x}_n, \mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \log \prod_{i=1}^{n} P(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \log P(\mathbf{y}_i | \mathbf{x}_i, \mathbf{w}) \\
\mathbf{y} &\in \mathcal{N}(0, e^{\mathbf{w}^T \mathbf{x}_i}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \log \frac{e^{-\frac{\mathbf{y}_i^2}{2e^{\mathbf{w}^T \mathbf{x}_i}}}}{\sqrt{2\pi e^{\mathbf{w}^T \mathbf{x}_i}}} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{1}{2} \sum_{i=1}^{n} \left(-\mathbf{y}_i^2 e^{-\mathbf{w}^T \mathbf{x}_i} - \mathbf{w}^T \mathbf{x}_i - \log 2\pi \right) \\
\nabla f(\mathbf{w}) &= \frac{1}{2} \sum_{i=1}^{n} \left(\mathbf{y}_i^2 e^{-\mathbf{w}^T \mathbf{x}_i} - 1 \right) \mathbf{x}_i
\end{aligned}
$$

- $e^{\mathbf{w}^T \mathbf{x}}$ is used as the variance because it is positive and monotonic
- $f(\mathbf{w})$ is concave

2 Algorithm

Use the EM algorithm. The above derivation takes y_i from g2o, and x being just 1, or some vector of 1s and some other info with it, such as brightness levels or detected features. The E step is g2o, and the M step is the above.

Since we have different types of edges, the vector $\mathbf{x}$ will have a one-hot encoding for the first four entries (odometry, tag, dummy, waypoint), as well as potentially lighting values and detected feature number. $\mathbf{x}$ is a 1-hot vector indicating the edge and measurement type

$$
\mathbf{x}_i = \begin{bmatrix} \text{odometry x?} \\ \text{odometry y?} \\ \text{odometry z?} \\ \text{odometry yaw?} \\ \text{odometry pitch?} \\ \text{odometry roll?} \\ \text{tag x?} \\ \text{tag y?} \\ \text{tag z?} \\ \text{tag yaw?} \\ \text{tag pitch?} \\ \text{tag roll?} \\ \text{dummy x?} \\ \text{dummy y?} \\ \text{dummy z?} \\ \text{dummy yaw?} \\ \text{dummy pitch?} \\ \text{dummy roll?} \end{bmatrix}^T
$$

and $\mathbf{y}$ is a vector containing the translational and YPR angle error of each edge.

3 Computing Importance for Dampened Edges

The importance of an edge is a function of its rotation. In our code, the rotation in the odometry frame of the dummy vertex is represented by a quaternion $\begin{bmatrix} a & b & c & d \end{bmatrix}$, where $\begin{bmatrix} a & b & c \end{bmatrix}$ forms the vector component and d the scalar.

Since the gravity vector from ARKit is stable, but the other basis are not, we must assign lower importance for rotations that are greatest affected by a change in yaw. The affect of a rotation is quantified by rotating the dummy vertex's quaternion about the odometry z-axis by θ, or 0.05 radians in our model. Letting q be the dummy vertex's rotation quaternion and p be the quaternion representing a change in θ yaw,

$$
\begin{aligned}
\phi &= \Delta \begin{bmatrix} \text{roll} & \text{pitch} & \text{yaw} \end{bmatrix} \\
&\approx p_{1,2,3} - q_{1,2,3}
\end{aligned}
$$

This works because for small rotations such as incremental rotations, the vector component of a quaternion is approximately the represented roll, pitch, and yaw. From there, we normalize ϕ and take the null space to establish a square matrix $\mathbf{b} = \begin{bmatrix} \phi & \phi_2 & \phi_3 \end{bmatrix}$, and weigh the rows by a diagonal matrix of the yaw For some reason, the bottom right block for the angular importance is

$$
\mathbf{b} \begin{bmatrix} \text{yaw importance} & & \\ & \text{pitch importance} & \\ & & \text{roll importance} \end{bmatrix} \mathbf{b}^T
$$