CMSC828T Vision, Planning And Control In Aerial Robotics

FACTOR GRAPH BASED FILTERING USING GTSAM





2D Robot World



 \mathbb{Y}

- World with landmarks
- Landmarks are unique and numbered
- Each landmark l_k is denoted by,

$$l_k = \begin{bmatrix} l_{k,x} \\ l_{k,y} \end{bmatrix}$$



Robot Start

X





Robot Start



Robot Odometry says this is origin



Robot State

 \mathbb{Y}



Robot State is given by, $x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ Because this varies with respect to time,

We will write the state as,

$$\boldsymbol{x_t} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_t$$





Sensor



Robot has a Wide FOV Camera/Lidar of 120°





 \mathbb{K}

Measurement Model







Measurement Model



- At x_t the robot observes l_k landmarks where k denotes individual landmark IDs
- Sensor is not perfect hence you see the landmark with a probability of $p_{obs} = 0.95$
- The measurement at time t with respect to landmark l_k obtained by the robot is given by

$$m_{t,k} = \begin{bmatrix} m_{t,k,x} \\ m_{t,k,y} \end{bmatrix}$$

• $m_{t,k}$ is noisy and can be thought of as drawn from $\mathcal{N}(\widehat{m}_{t,k}, \Sigma_m)$







Robot at t = 1



- Odometry is denoted by o_t^{t+1} between times t and t+1
- It is obtained by some wheel encoder and can be thought of as drawn from $\mathcal{N}(\hat{o}_t^{t+1}, \Sigma_o)$





For multiple steps







SLAM Problem



Given Initial pose x_1 , odometry o_t^{t+1} and landmark measurements $m_{t,k}$ Obtain landmark locations l_k and robot pose x_t SLAM stands for "Simultaneous Localization and Mapping"





SLAM as a Bayes Net







SLAM as a Bayes Net: Graph







SLAM as a Bayes Net: Math

Motion Model:

$$x_t = f_t(x_{t-1}, u_t) + w_i \Leftrightarrow P(x_t | x_{t-1}, u_t) \propto e^{-\frac{1}{2} \|f_t(x_{t-1}, u_t) - x_t\|_{\Lambda_t}^2}$$

Measurement Model:

$$m_i = h_i(x_{t,i}, l_{k,i}) + v_i \Leftrightarrow P(m_i | x_{t,i}, l_{k,i}) \propto e^{-\frac{1}{2} \|h_i(x_{t,i}, l_{k,i}) - m_i\|_{\Sigma_i}^2}$$

MAP to maximize:

$$P(X,L,M) = P(x_0) \prod_{t=1}^{T} P(x_t | x_{t-1}, u_t) \prod_{i=1}^{M} P(m_i | x_{t,i}, l_{k,i})$$







SLAM as a Factor Graph: Graph







SLAM as a Factor Graph: Math

Prior:

 $\phi_0(x_0) \propto P(x_0)$

Motion Model:

$$\psi_{t-1,t}(x_{t-1},x_t) \propto P(x_t|x_{t-1},u_t)$$

Measurement Model:

$$\psi_{t,k}(x_t, l_k) \propto P(m_{t,k} | x_t, l_k)$$

Value of the graph to maximize:

$$P(\Theta) \propto \prod_{t=0}^{T} \phi_t(\theta_t) \prod_{\{i,j\}, i < j} \psi_{ij}(\theta_i, \theta_j)$$

$$\Theta \triangleq (X, L)$$







SLAM as Non-Linear Least Squares

• Maximum A Posteriori (MAP) estimation

$$f(\Theta) = \prod_{i} f_{i}(\Theta_{i}), \Theta \triangleq (X, L) \forall f_{i}(\Theta_{i}) \propto e^{-\frac{1}{2} \|h_{i}(\Theta_{i}) - m_{i}\|_{\Sigma_{i}}^{2}}$$

$$\Theta^* = \operatorname*{argmax}_{\Theta} f(\Theta)$$

• Negative Log Likelihood (NLL)

$$\underset{\Theta}{\operatorname{argmin}}(-\log f(\Theta)) = \underset{\Theta}{\operatorname{argmin}}\left(\frac{1}{2}\sum_{i} \|h_{i}(\Theta_{i}) - m_{i}\|_{\Sigma_{i}}^{2}\right)$$





Numerical Optimization 101







Convex Function







Concave Function



Intuitively: Concave = -Convex or vice versa

Concave function

$$x^* = \operatorname*{argmax}_{x} f(x)$$





Non-Convex Function



Neither convex nor concave Intuitively has a lot of local optimum





Convex Optimization 101







Steepest Descent

 $x^{t+1} = x^t - \tau \nabla f(x^t)$ τ is called the step-size $\nabla f(x)$ is the local gradient at x







Step-size Restrictions

 $\tau < \frac{2}{\alpha} \text{ for } f(x) = \frac{\alpha}{2} x^2$ $\nabla f(x) = \alpha x$ $x^{t+1} = x^t - \tau \alpha x^t$

Sort of like PD controller

Too high au will cause you to diverge

Too low τ will take forever to converge



Convex function





Lipschitz Constant

 $\|\nabla f(x) - \nabla f(y)\| \le M \|x - y\|$

Here *M* is the Lipschitz constant or intuitively *M* represents a function of maximum curvature

If a Hessian exists: $M \ge \|\nabla^2 f(x)\|$

The step-size restriction becomes

$$\tau < \frac{2}{M}$$

It is generally hard to obtain a value of M

There are methods to find "best" τ for each step and are called Line Search Methods





Recall Condition Number

 κ denotes how sensitive the function is to noise or in other words how circular are the iso-contours









The "Best" Pre-conditioner



When $P = H^{-\frac{1}{2}}, P^{T}HP = H^{-\frac{1}{2}}HH^{-\frac{1}{2}} = I$







Newton's Method

$$\underset{x}{\operatorname{argmin} f(x)} \qquad \underbrace{x = \mathbf{H}^{-\frac{1}{2}}y}_{y} \qquad \operatorname{argmin} f\left(\mathbf{H}^{-\frac{1}{2}}y\right)$$

Gradient Step becomes $y^{k+1} = y^k - \mathbf{H}^{-\frac{1}{2}} \nabla f\left(\mathbf{H}^{-\frac{1}{2}} y^k\right)$

Changing back variables to *x* we get

 $x^{t+1} = x^t - \mathbf{H}^{-1} \nabla f(x^t)$

 $-\mathbf{H}^{-1}\nabla f(x^t)$ is called the Newton direction





Gauss Newton Method

Modification of Newton's method to find minimum of a sum of squared function values Let the function we are minimizing be

$$F(x) = \frac{1}{2} \sum_{i=1}^{m} f_i(x)^2 = \frac{1}{2} \|f(x)\|^2 = \frac{1}{2} f(x)^T f(x)$$

Our problem setup is as follows: $\operatorname{argmin} F(x)$

The gradient vector g is obtained as follows,

$$g_j = \sum_{i=1}^m f_i \frac{\partial f_i}{\partial x_j}$$

To obtain the Hessian we need to differentiate the gradient elements with respect to x_k

x

$$H_{jk} = \sum_{i=1}^{m} \left(\frac{\partial f_i}{\partial x_j} \frac{\partial f_i}{\partial x_k} + f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k} \right)$$





Gauss Newton Method

$$H_{jk} = \sum_{i=1}^{m} \left(\frac{\partial f_i}{\partial x_j} \frac{\partial f_i}{\partial x_k} + f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k} \right)$$

Now, ignore all the second-order derivative terms (second term) in the above expression

$$H_{jk} \approx \sum_{i=1}^{m} J_{ij} J_{ik}$$

Where $J_{ij} = \frac{\partial f_i}{\partial x_j}$ are the components of the Jacobian Matrix **J** Now, $g = \mathbf{J}^T f$ and $\mathbf{H} \approx \mathbf{J}^T \mathbf{J}$ The understanding for Cause Newton method become

The update equations for Gauss Newton method become

$$x^{t+1} = x^t - (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T f$$

Why is this better than Newton's Method with the update rule $x^{t+1} = x^t - \mathbf{H}^{-1}\nabla f(x^t) = x^t - \mathbf{H}^{-1}\mathbf{J}^T f$ Complicated Hessian computation is avoided





Levenberg-Marquardt (LM) Method

Also called "Damped Least Squares" The update rule is

$$x^{t+1} = x^t - \widehat{\mathbf{H}}^{-1} \nabla f(x^t)$$

Here $\widehat{\mathbf{H}}$ is modified Hessian

 $\widehat{\mathbf{H}} = \mathbf{H} + \lambda \operatorname{diag}(\mathbf{H})$

LM method blends the Steepest Descent method and Newton's method Recall steepest descent method update rule is

$$x^{t+1} = x^t - \tau \nabla f(x^t)$$

And Newton's method update rule is

$$x^{t+1} = x^t - \mathbf{H}^{-1} \nabla f(x^t)$$

- Steepest descent works well when we are far from minima and Newton's method which assumes local quadratic approximation works well near the minima as the quadratic approximation is good
- In the LM update rule when λ gets small the rule approaches Newton's method and when λ is large LM approaches steepest descent





Dogleg Method

Chooses between steepest descent (Cauchy) step and Gauss-Newton step

Let $h = x^{t+1} - x^t$ be the update rule such that $x^{t+1} = x^t + h$

Cauchy step is given by $h_C = -\tau \mathbf{J}^T f$

Gauss-Newton step is given by $h_{GN} = -(\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T f$

Dogleg uses a region of trust Δ around the linearization point to choose between Cauchy and GN steps

$$h_{dl} = h_C + \lambda (d_{GN} - d_C)$$

Here $\lambda \in [0,1]$ is the largest value in such that $||h_{dl}|| \leq \Delta$

```
If J is nearly singular then h_{dl} = h_C
```

Update rule is: $x^{t+1} = x^t + h_{dl}$



Ordering

- Selecting the correct column ordering matters since it decides the sparsity of information matrix
- Use COLAMD to find the best ordering just based on information matrix
- COLAMD stands for "COLumn Approximate Minimum Degree permutation"



COMPUTER SCIENCE



References

- GTSAM4 Tutorial Slides: <u>https://www.cc.gatech.edu/grads/j/jdong37/files/gtsam-tutorial.pdf</u>
- Frank Dallert's Hands-On GTSAM Tutorial: <u>https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf</u>
- Tom Goldstein's amazing optimization slides: <u>https://www.cs.umd.edu/~tomg/course/764_2017/L7_grad_descent.pdf</u>
- Boyd's Optimization book: <u>https://web.stanford.edu/~boyd/cvxbook/</u>
- Simple Optimization: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network
- Matlab's Optimization: <u>https://www.mathworks.com/help/optim/ug/equation-solving-algorithms.html#f51887</u>
- LM Optimizer: <u>https://www.cs.nyu.edu/~roweis/notes/lm.pdf</u>
- Dogleg Optimizer: <u>http://ceres-solver.org/nnls_solving.html</u>
- COLAMD: <u>https://www.mathworks.com/help/matlab/ref/colamd.html</u>



