A Tutorial of Graph Optimization: Theory, Matlab Code and Applications

Teng Zhang and Liyang Liu

Centre for Autonomous Systems

University of Technology Sydney

May 16, 2017

Outline

- Concept of Factor Graph
- Code Framework
- Related Applications

Theory: Factor Graph

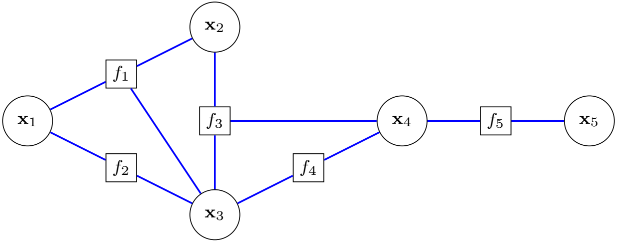


Figure: A Typical Factor Graph

Definition

A factor graph $G = (\mathcal{F}, \mathbf{X}, \mathcal{E})$ is a bipartite graph, consisting of factor nodes in $\mathcal{F} = \{f_i\}$, variable nodes $\mathbf{X} = \{\mathbf{x}_i\}$ and the edges in $\mathcal{E} = \{e_i\}$.

- $\mathbf{X} =$ denotes the set of variables to be estimated;
- $\mathbf{X}_i \subseteq \mathbf{X}$ denotes the set of variables $\mathbf{x}_j \in \mathbf{X}$ adjacent to $f_i \in \mathcal{F}$;
- $\mathcal{F}$ denotes the set of all functions $f_i(\cdot)$ where $f_i(\cdot)$ is a function of $\mathbf{X}_i$;
- the edge e_i connects a factor node f_i and all variable nodes in $\mathbf{X}_i$;

Theory: Factor Graph and MAP

The inference in the factor graph G refers to the optimization problem below

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \prod_i f_i(\mathbf{X}_i) \tag{1}$$

When $f_i(\mathbf{X}_i) = p(Z_i|\mathbf{X}_i)$ represents the probability density function of measurement Z_i given $\mathbf{X}_i$, $\mathbf{X}^*$ becomes the maximum a posteriori (MAP) estimate:

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} p(\mathbf{X}|Z), \tag{2}$$

For the Gaussian case,

$$f_i(\mathbf{X}_i) = p(Z_i|\mathbf{X}_i) \propto \exp(-\frac{1}{2}\|h_i(\mathbf{X}_i, Z_i)\|^2_{\mathbf{\Sigma}_i^{-1}}) \tag{3}$$

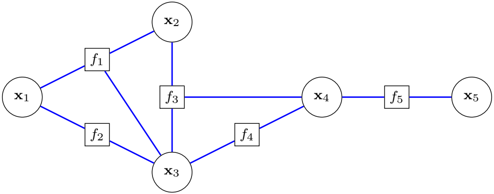
i.e.,

$$h_i(\mathbf{X}_i, Z_i) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_i) \tag{4}$$

Therefore, the maximum a posteriori (MAP) estimate $\mathbf{X}^*$ of the factor graph G corresponds to a nonlinear least squares optimization (5).

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|h_i(\mathbf{X}_i, Z_i)\|^2_{\mathbf{\Sigma}_i^{-1}}. \tag{5}$$

Theory: Nonlinear Least Squares



$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \| h_i(\mathbf{X}_i, Z_i) \|^2_{\boldsymbol{\Sigma}_i^{-1}}. \tag{6}$$

Example: for the factor f_1

- $\mathbf{X}_1 = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ are all involving variables
- $h_1(\cdot, \cdot)$ is called the error function.
- $\boldsymbol{\Sigma}_1$ is called the covariance matrix and $\boldsymbol{\Sigma}_1^{-1}$ is called the information matrix.
- Z_1 is the measurement

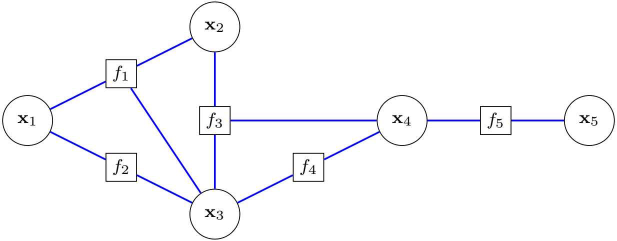
Theory: Representation

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_i \|h_i(\mathbf{X}_i, Z_i)\|^2_{\mathbf{\Sigma}_i^{-1}}. \tag{7}$$

Beyond a column vector

- The variable node $\mathbf{x}_i$ is regarded as a *structure*, **not required to be** a column vector. Each variable $\mathbf{x}_i$ belongs to a Node type and this Node type has to be associated with a special addition $\oplus$ used in update.
  - Example: $\mathbf{x}_1$ represents the robot pose at time-step 1, belonging to the Node type $\mathbb{SE}(3)$. The special addition of $\mathbb{SE}(3)$: $\mathbf{X}_1 \oplus \mathbf{e} = \exp(\mathbf{e})\mathbf{X}_1$ where $\mathbf{e} \in \mathbb{R}^6$.
- The measurement Z_i is regarded as a *structure*, **not required to be** a column vector.
  - Example: In pose graph, the measurement Z_i is the relative pose, represented by $\mathbb{SE}(3)$, not a column vector.
- The error function $h_i(\cdot,\cdot)$ **must be** a column vector.

How to creat a Graph



Creating a Factor Graph needs to ...

- add all factors f_i to the Graph with the measurement Z_i and the information matrix $\mathbf{\Sigma}_i^{-1}$;
- for the factors f_i, point out the node ID of all involving variables $\mathbf{x}_j$ and their order
- provide a reasonable initial guess for $\mathbf{X}$

How to define new Node and Factor

Defining a new Node type needs to ...

define the special addition $\oplus$ and give the related dimension

- For example, defining the Node type $\mathbb{SE}(3)$, we need to provide the expression of $\oplus$: $\mathbf{x}_1 \oplus \mathbf{e} = \exp(\mathbf{e})\mathbf{x}_1$ and point out the dimension is 6 due to $\mathbf{e} \in \mathbb{R}^6$.

Defining a new factor $f_i(\cdot)$ needs to ...

- point out all involving Node types and the order
- point out the type of measurement Z_i
- provide the expression of the error function $h_i(\mathbf{X}_i, Z_i)$ and the dimension of $h_i(\cdot,\cdot)$
- provide all small Jacobian matrices of $h_i(\mathbf{X}_i, Z_i)$ w.r.t., the involving variable node $\mathbf{x}_j$.
  - For example: $\mathbf{X}_i = (\mathbf{x}_1, \mathbf{x}_2)$. We need to provide two Jacobians,

$$\begin{aligned} \mathbf{H}_{i,1} &= \frac{\partial h_i(\mathbf{x}_1 \oplus \mathbf{e}_1, \mathbf{x}_2, Z_i)}{\partial \mathbf{e_1}}\Big|_{\mathbf{e}_1=\mathbf{0}} \\ \mathbf{H}_{i,2} &= \frac{\partial h_i(\mathbf{x}_1, \mathbf{x}_2 \oplus \mathbf{e}_2, Z_i)}{\partial \mathbf{e_2}}\Big|_{\mathbf{e}_2=\mathbf{0}} \end{aligned} \tag{8}$$

Matlab Code: Framework

Introduction

- **Data**: store the data to be processed
- **Factor**: the implementations of all factors and nodes
- **g2o_files**: core, provide the main framework of the nonlinear least squares
- **Math**: provide the mathematical forumulations like $\exp(\cdot)$,...
- **Geometry**: some operations on geometry such as triangulation
- **Examples**: provide some commonly used state estimation problems such as Bundle Adjustment

Matlab Code: Content and Features

Content

- **Node type**: $\mathbb{R}^3$, $\mathbb{SO}(2)$, $\mathbb{SE}(2)$, $\mathbb{SO}(3)$, $\mathbb{SE}(3)$ and some others.
- **Math**: including some commonly used mathematical operations such as $\exp(\cdot)$ and $\log(\cdot)$ on Lie Group.
- **Factor**: several vision factors, RGB-D factor, Parallax vision factor, IMU factor.

Features:

- **Fixed variable**: any variable $\mathbf{x}_i$ can be set as fixed.
- **Optimization algorithms**: Gauss-Newton, Levenberg-Marquart and Powell's Dogleg (**recommend**!)
- **Schur decomposition**
- **Incremental inference friendly**
- **Novel Factors**:
  IMU factor, Parallax Vision Factor (**may be the best in the East hemisphere**)
- **Novel Node type**: IMU state on real manifold, Parallax feature on manifold
- **High readability and easy to extend**
- **Necessary warnings**

Why we need this?

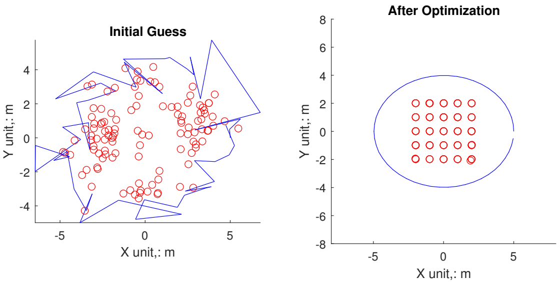
Applications

- 2D SLAM
- 3D SLAM
- Visual-Inertial Odometry
- Kinematics: Mirror Tracking
- Planning

Significance

- The first Matlab version of Graph-Optimization on manifold.
- Quickly validate a Graph-Optimization based algorithm.
- It is a good tutorial code for SLAM beginners.

Example


Initial Guess
4
2
0
-2
-4
Y unit,: m
-5
0
5
X unit,: m
After Optimization
8
6
4
2
0
-2
-4
-6
-8
Y unit,: m
-5
0
5
X unit,: m

End

Download

$https://github.com/UTS-CAS/Matlab-Graph-Optimization$

Thanks!