Graph-based SLAM: Theory & Practice

XIANG GAO, 2014.3

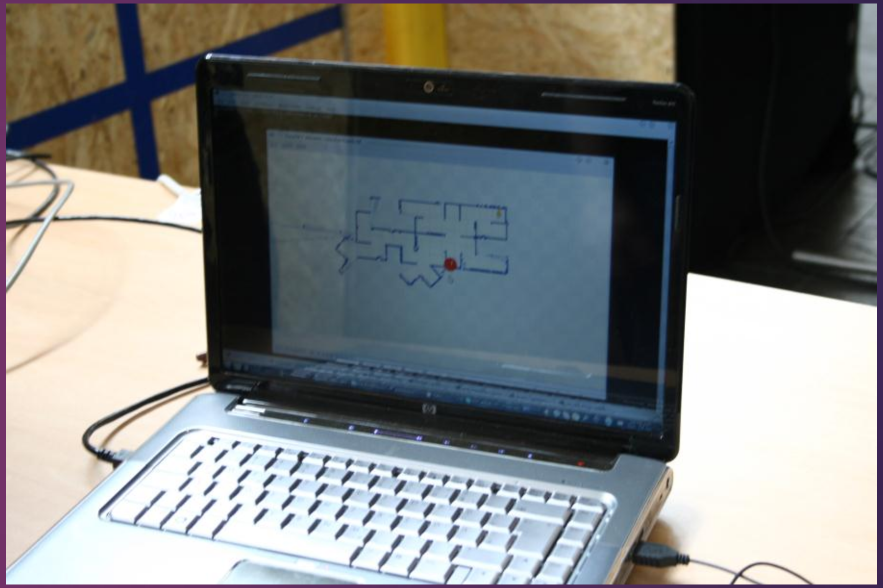
Contents

Contents

- Basic Theory
  - Definition
  - Historical Perspective
- Graph-based Optimization
- Implementation using g2o

Basic Theory

- SLAM: Simultaneous Location and Mapping



- Applications:

mobile robot (indoor/outdoor), auto-driver, underwater robot, aerial robot, etc.

Basic Theory

- Definition of SLAM:
  - Given a set of measurement about robot and observations from its sensors, to estimate the robot's location and the environments model around it.

Basic Theory

- Math model

Location: $x_p^i, i = 1, \dots, n$

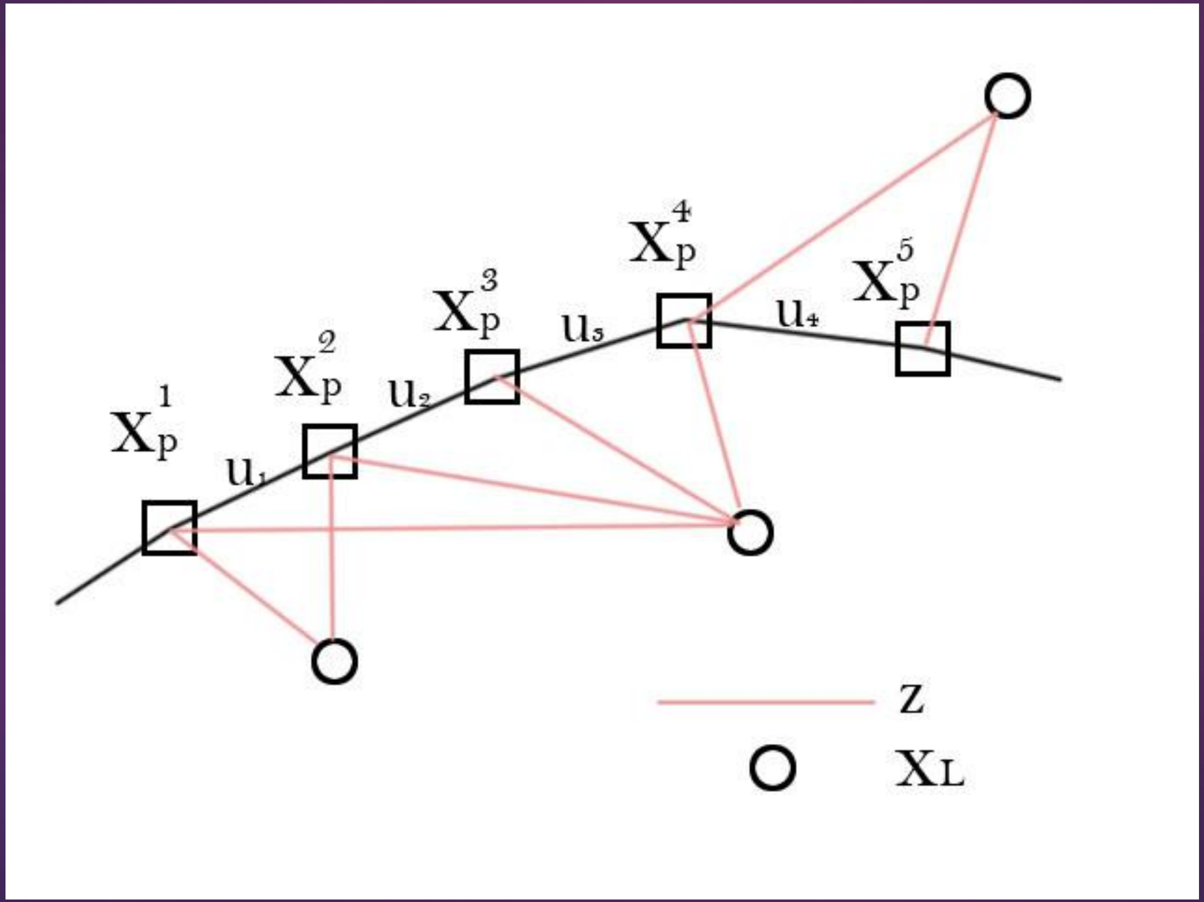
Landmarks: $x_L^i, i = 1, \dots, m$

Motion: $x_p^{i+1} = f(x_p^i, u_i) + w_i$

Observations: $z_{i,j} = h(x_p^i, x_L^j) + v_{i,j}$

Basic Theory

- Math model



Basic Theory

- A little explanation

Motion: $$x_p^{i+1} = f(x_p^i, u_i) + w_i$$

Observations: $$z_{i,j} = h(x_p^i, x_L^j) + v_{i,j}$$

- In 2D SLAM x_p can be set as $x_p = (x, y, \theta)$, and in 3D will be $x_p = (x, y, z, q_x, q_y, q_z, q_w)$
- The observation model is related to sensor and landmark model.

Basic Theory

- A little explanation

Motion: $x_p^{i+1} = f(x_p^i, u_i) + w_i$

Observations: $z_{i,j} = h(x_p^i, x_L^j) + v_{i,j}$

- If we use 3D landmarks and camera as sensors, then:

$$x_L = (x, y, z), z = (u, v)$$

$$z_{i,j} = (u, v)_{i,j}^T = CR_i x_L^j + v_{i,j}$$

Basic Theory

- A little explanation

Motion: $$x_p^{i+1} = f(x_p^i, u_i) + w_i$$

Observations: $$z_{i,j} = h(x_p^i, x_L^j) + v_{i,j}$$

- Before we solving SLAM, the only things we know are $u_i, z_{i,j}$ which are recorded by odometers and sensors with noise. In fact, $z_{i,j}$ is not known perfectly without correct DA (data association). We need to get x_p, x_L.

Basic Theory

- History perspective.
  - 1986, R. Smith, M. Self, P. Cheeseman, a series of path breaking works.
  - Two branches
    - Filters: KF, EKF, PF, RBPF, FastSLAM, DCSLAM, etc.

      Marginalize out past poses and summaries the information gained over time with a probability distribution.
    - Global optimization

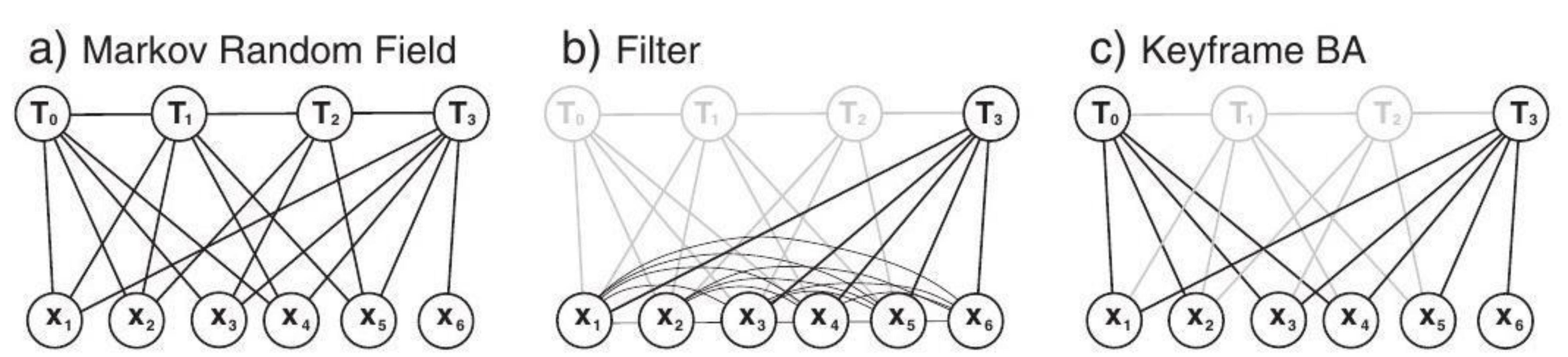
      Since (F Lu, E Milios, 1997).

      Retain optimization over all observations using approach of global bundle adjustment (BA). Also known as full-SLAM.

      Considered computational expensive in the past, however the sparse structure is realized in 21st centery.

Basic Theory

- Difference between filter and BA approaches



From (H Strasdat, et al. 2012)

Basic Theory

- Theory of BA
- Known:

From sensor: $z_{i,j}, u_i$
And Initial Guess: $\overline{x}_p^i, \overline{x}_L^j$

- Then, we can estimate errors from motion and observation equations:

$$e_p^i = \overline{x}_p^{i+1} - f(\overline{x}_p^i, u_i), e_L^{i,j} = z_{i,j} - h\left(\overline{x}_p^i, \overline{x}_L^j\right)$$

Basic Theory

- Theory of BA
- Minimize these errors which show the inconsistency between guess values and measurements.

$$\min \quad \varphi = \sum_{i} (e_p^i)^2 + \sum_{i,j} (e_L^{i,j})^2$$

- Of course it is no linear problem and is not convex. BA assumes that local area of cost function can be linearized, thus we can get derivative.
- Since we have the initial guesses, what we need to find is a gradient decent direction (Jacobian matrix, and Hessian matrix if we use Newton methods).

Basic Theory

- Theory of BA

$$J = \frac{\nabla \varphi}{\nabla x}, H = \frac{\nabla^2 \varphi}{\nabla x \nabla x^T}$$

- BA is basically a least-square problem.
- In the past, these two matrices are considered too complicated to solve. But these days, people find that they invariably have sparse structure when solving SLAM, for we can only see a part of landmarks in one observation.
- So, we can make use of sparse algebra like Sparse Cholesky Decomposition, etc.

Basic Theory

- Theory of BA

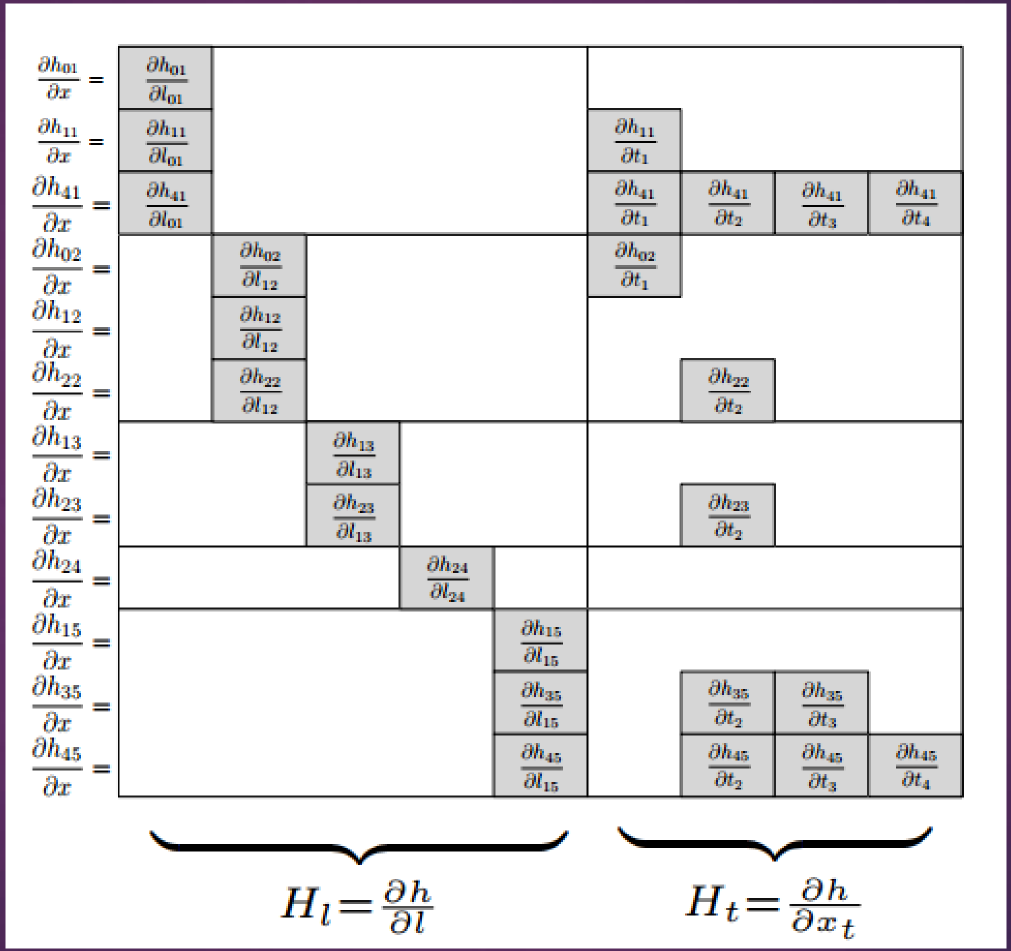


Figure from (Sibley D et al. 2009)

Contents

- Basic Theory
  - Definition
  - Historical Perspective
- Graph-based Optimization
- Implementation using g2o

Graph-based Optimization

- Graph-based Optimization is a institutive representation of BA which show the problem in a "graph", aka a number of vertices and edges.

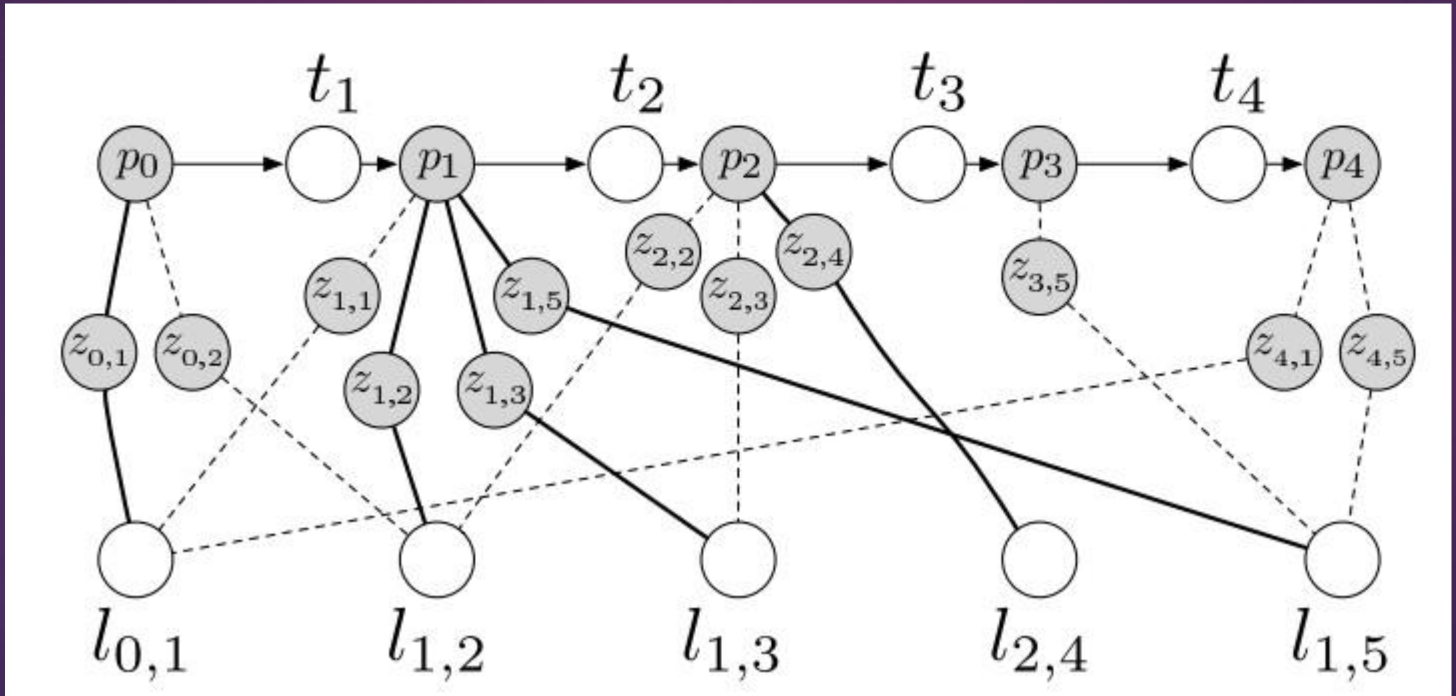


Figure from (Sibley D et al. 2009)

Graph-based Optimization

- Graph:

$$G = \{V, E\}$$
$$V = \{x_p^i, x_L^j\}$$
$$E = \{e_i\}$$

- Vertex: Each vertex is a optimizing variable, namely, location and landmarks (mapping).
- Edge: Each edge is just a constraint of the opt. problem, which bind two (or more) vertices together with an observation.

Graph-based Optimization

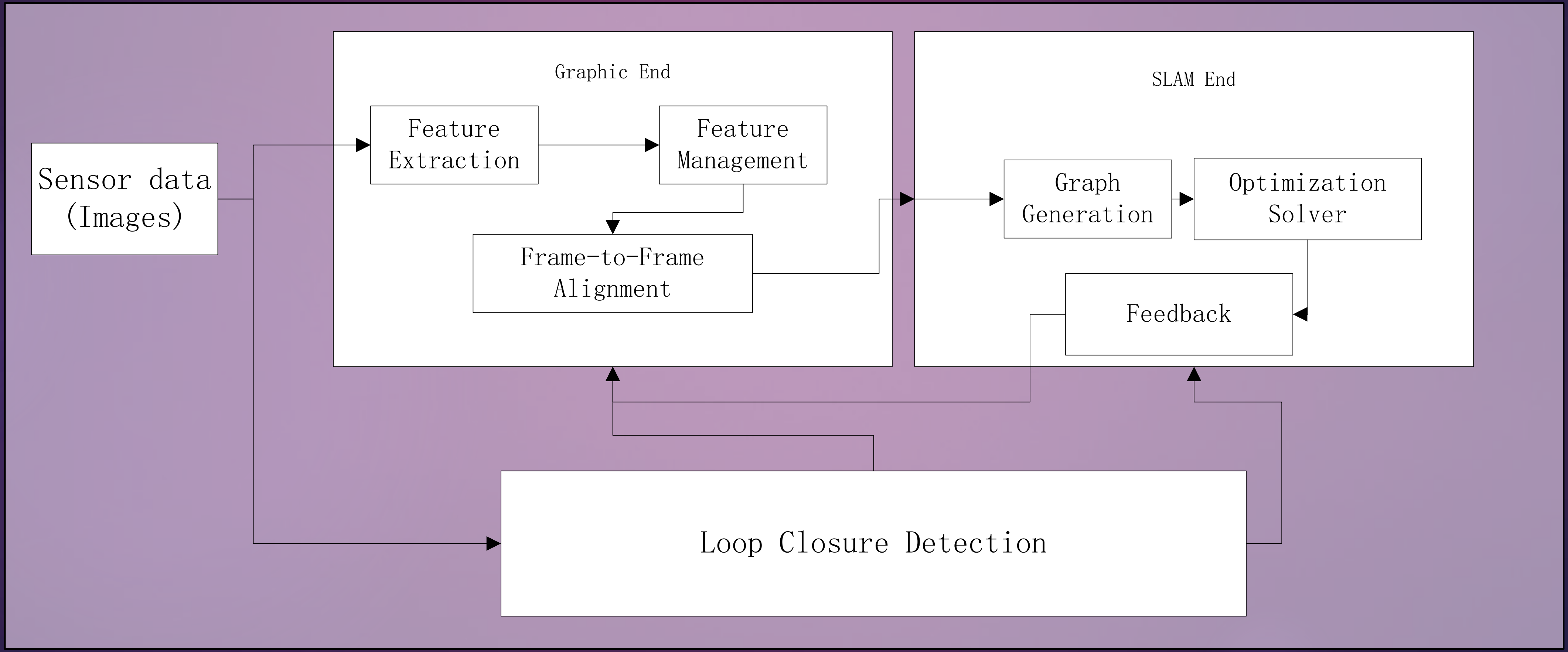
- Some comments:
  1. In Graph-based Opt., both odometer and sensor observation constraints are treated equally, i.e., edges of the graph, which is fairly different from the filter approaches.
  2. We may assign numerical or analytical Jacobian and Hessian to each edge. Also we can choose appropriate covariance matrix for it.
  3. Correctly solving this optimization problem is based on correct DA. The general Graph-based Optimization solver is sensitive to abnormal values, but it can be robustified by choosing "robust kernel".

Contents

- Basic Theory
  - Definition
  - Historical Perspective
- Graph-based Optimization
- Implementation using g2o

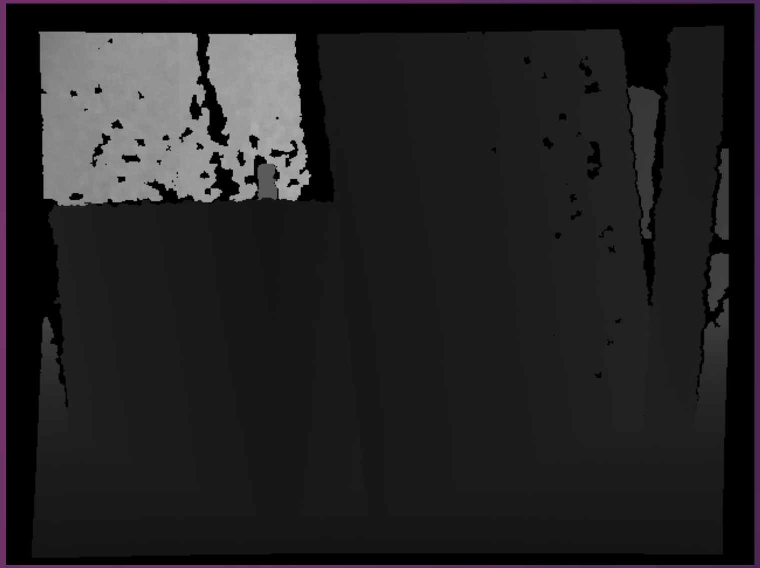
Implementation

- To implement a whole SLAM system is quite complicated, which mainly contains the following components.



Implementation

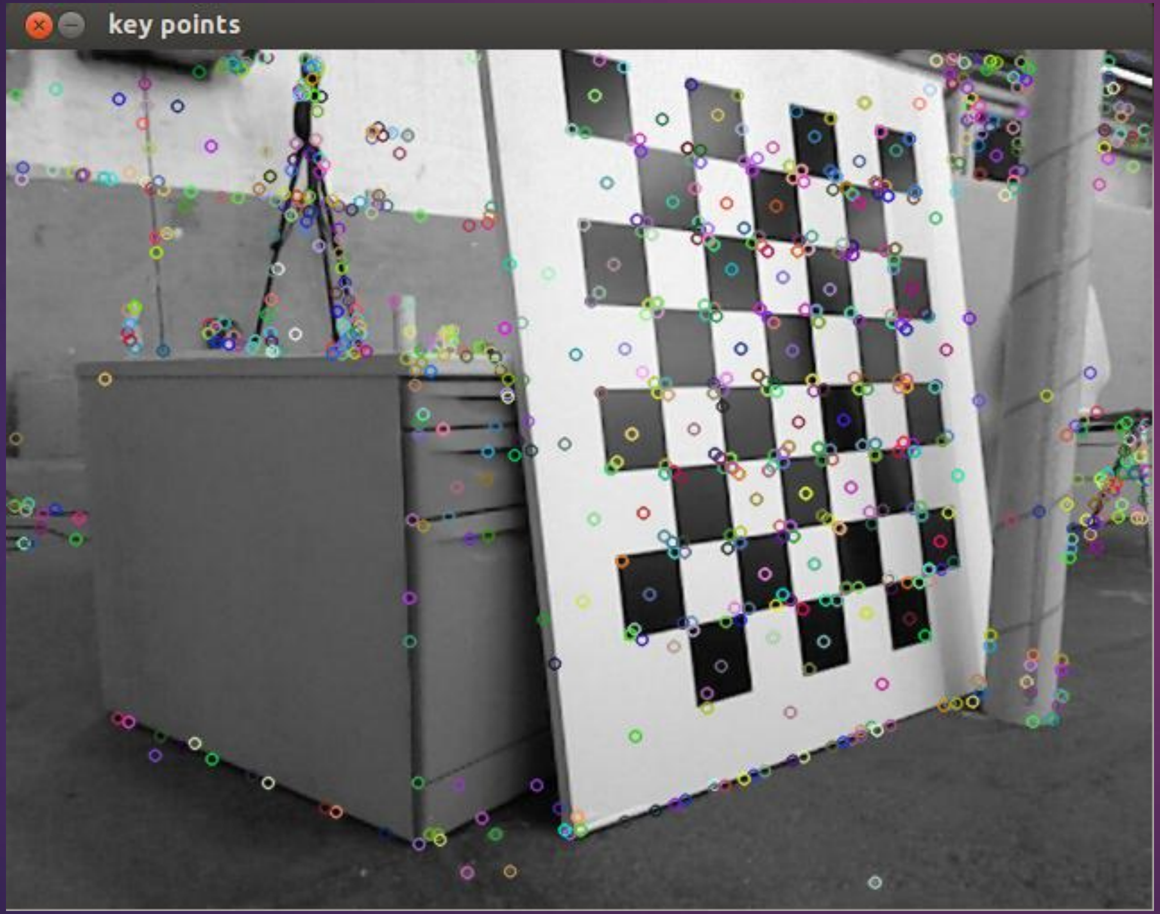
- Sensor data: Kinect-style cameras



- Dataset: http://vision.in.tum.de/data/datasets/rgbd-dataset

Implementation

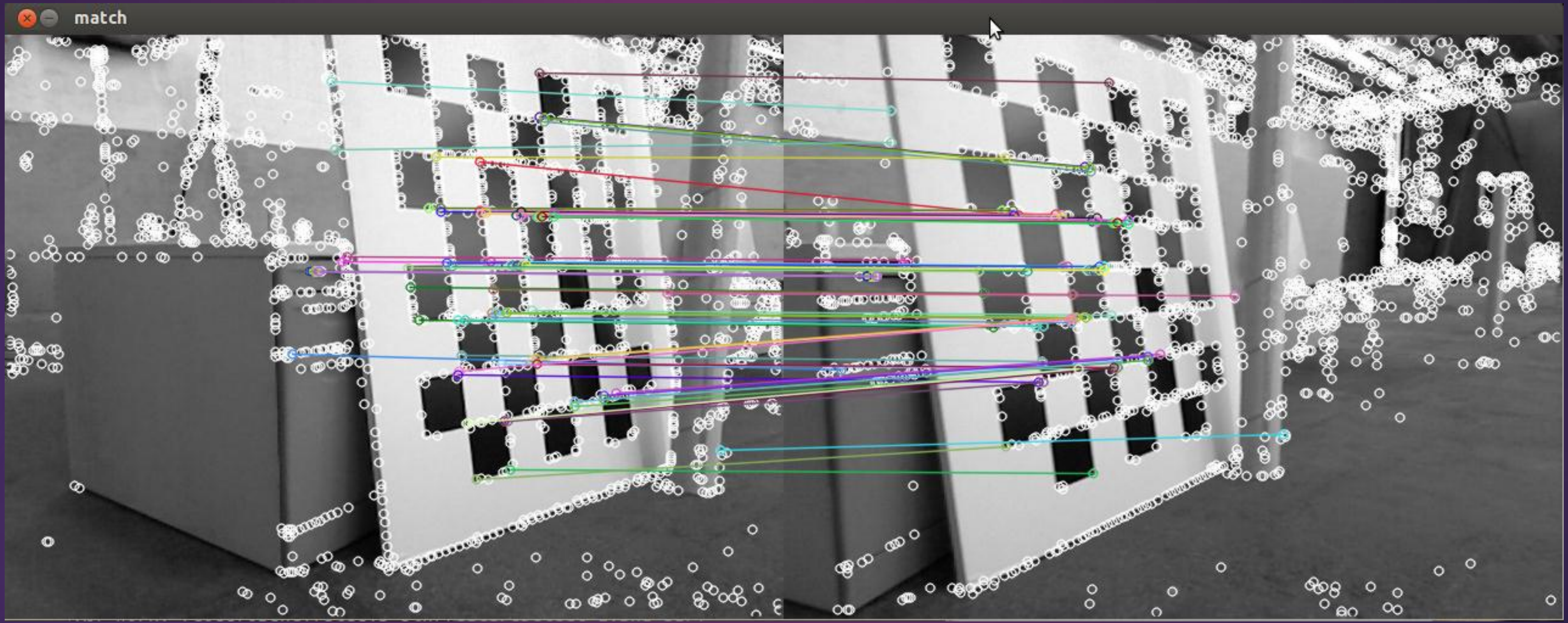
- Feature Extraction: SIFT/FAST/SURF etc.



For each feature, we will get a “descriptor” which will be used as its ID.

Implementation

- Feature Match

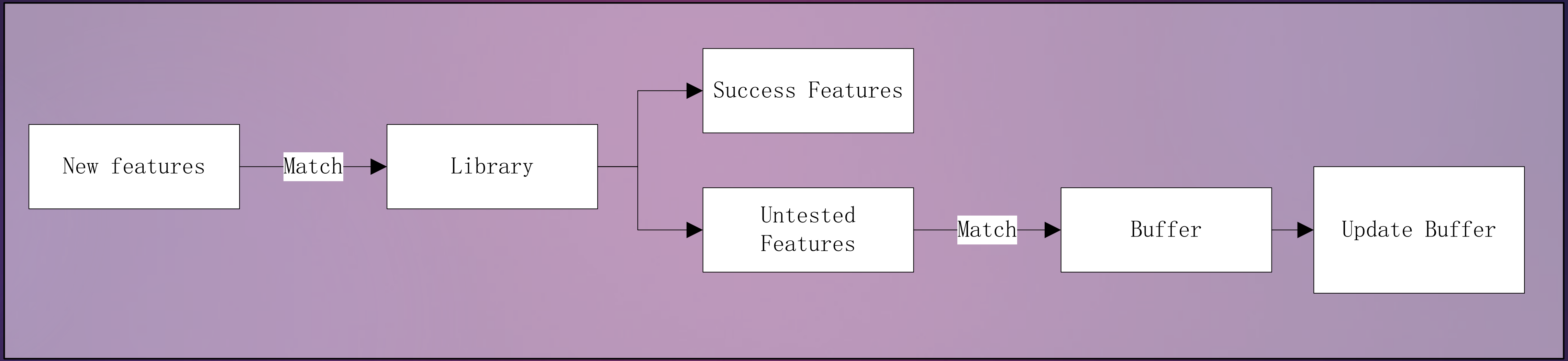


- Some wrong matches, can be corrected by RANSAC.

Implementation

- Feature Management: basically has 2 ideas.

1. Feature Library.

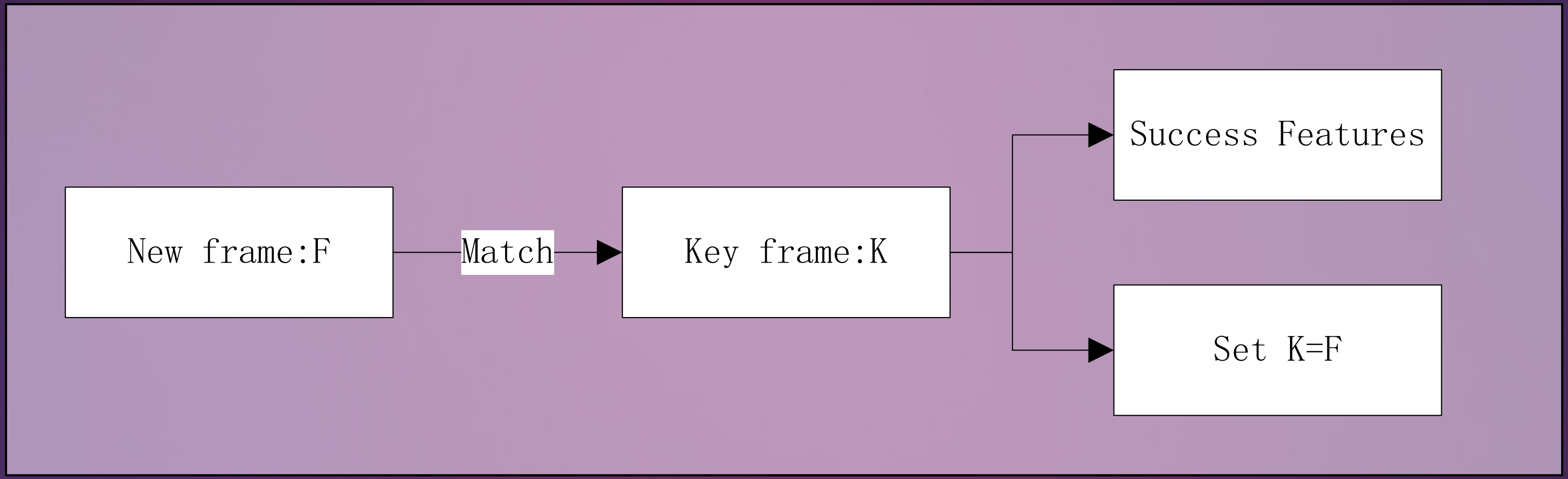


- Easy to close loop since previous observations are saved in Library.
- The library will quickly be too large.

Implementation

- Feature Management: basically has 2 ideas.

2. Key frame.

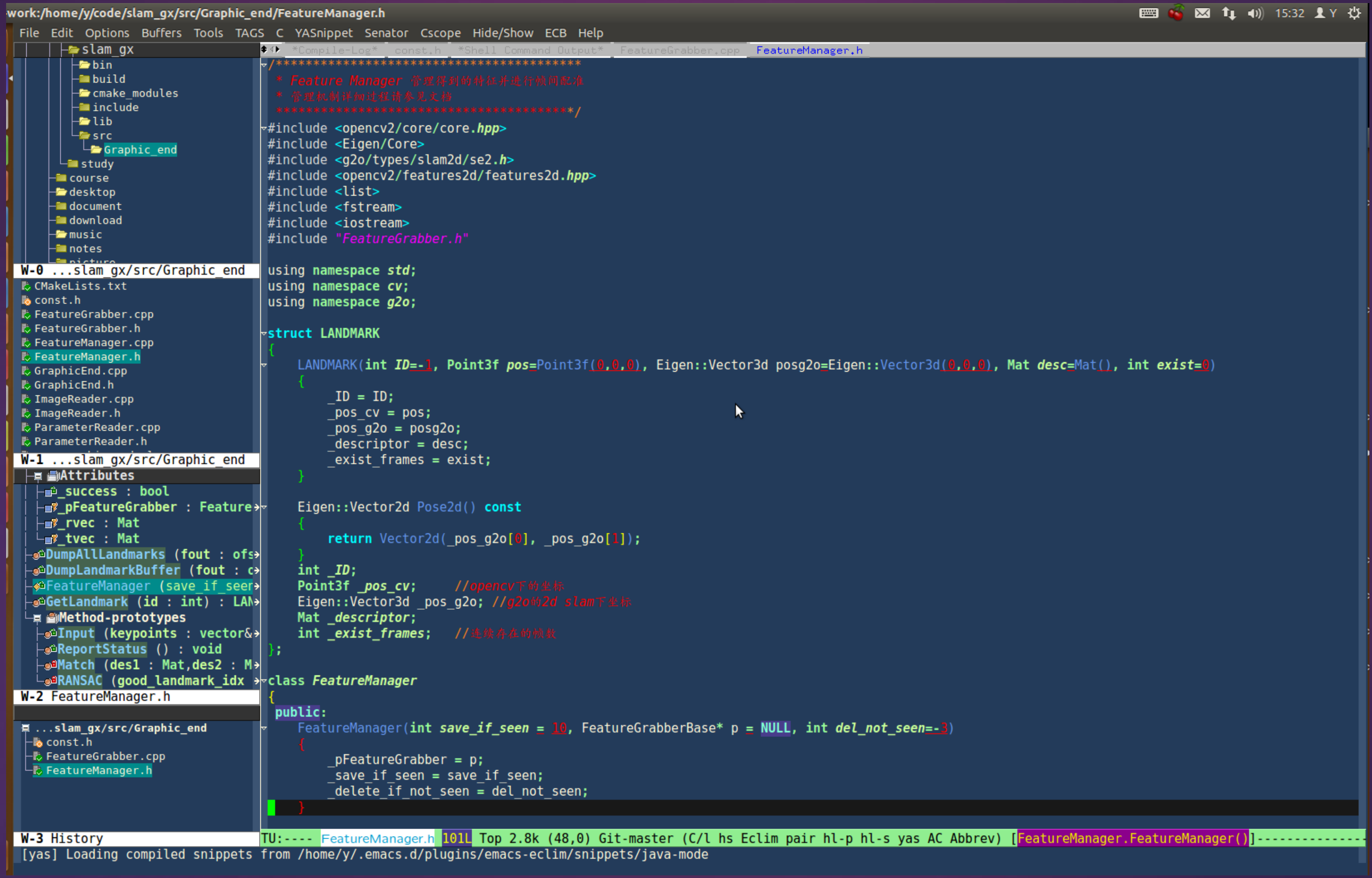


- Easy to implement. Only compare to key frame.
- Hard to do loop closure.

Implementation

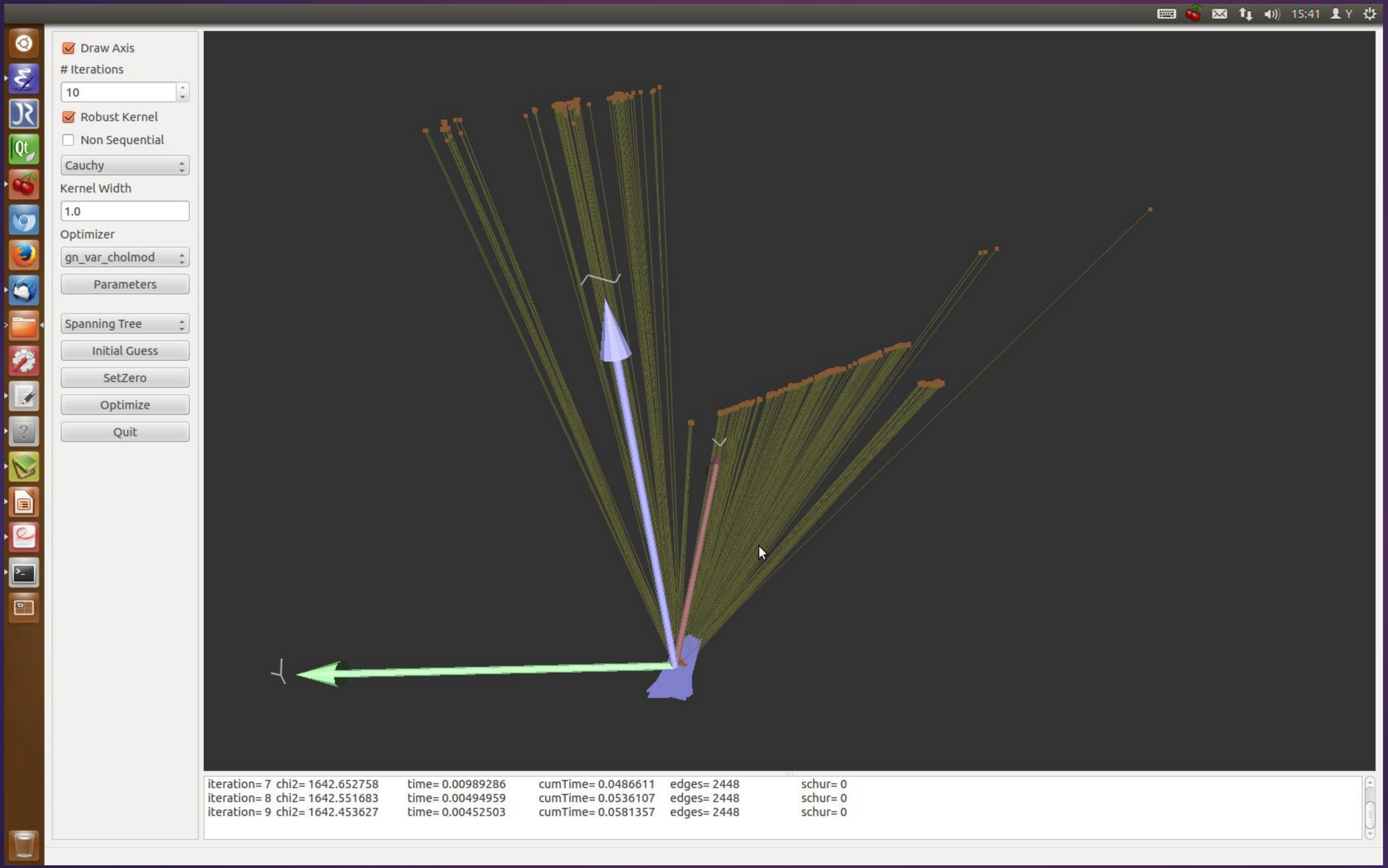
- SLAM End.
  - g2o: A general framework for graph optimization. (K Rainer, ICRA 2012)
  - C++ coded.
  - What we need to do is to put vertices and edges into the solver and wait for the answer.

Implementation



My project: https://github.com/gaoxiang12/slam_gx.git

Implementation



2-D visual SLAM.

Result from 1~50 frames.

The robot changed its head to right hand about 17.53 degrees.

Implementation

- Future work
  - Perform loop closure.
  - output the final result in point cloud format.
  - Try to give semantic labels to the results.

Thank you for your attention!