Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

SLAM with Factor Graphs

September 12, 2022

Ravisankar Selvaraju
Samuel Salazar
Selvakumar Nachimuthu
Tharun Sethuraman

Advisors

Deebul nair

Team members

Ravisankar

Samuel

Tharun

Selvakumar

Introduction

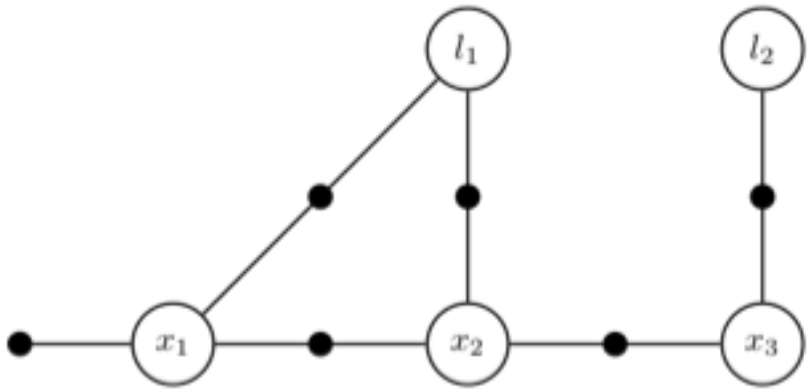
- **Project description:** Implementing Simultaneous Localization and Mapping (SLAM) using Factor Graphs
- **Project Goals:**
  - Creating a scientific library for factor graphs in GNU - GSL
  - Practical implementation of the developed library on a marker based localization problem
- **Modified Goals:**
  - Implementing factor graph based SLAM using GTSAM as ROS node
  - Implementation of Belief propagation in python
  - Implementation of Forney style factor graph and video presentation of it

What is SLAM?

- SLAM is a method used for autonomous vehicles that lets you build a map and localize your vehicle in that map at the same time. SLAM algorithms allow the vehicle to map out unknown environments.

What is a factor graph?

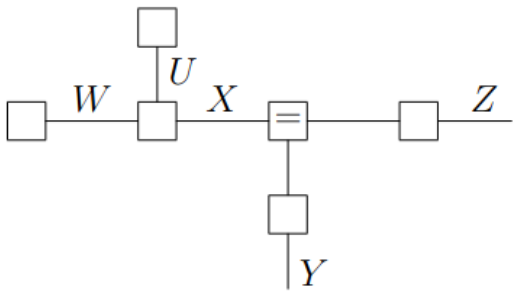
- Factor graphs are graphical model for inference that are well suited to modeling complex estimation problems, such as Simultaneous Localization and Mapping (SLAM) or Structure from Motion (SFM)



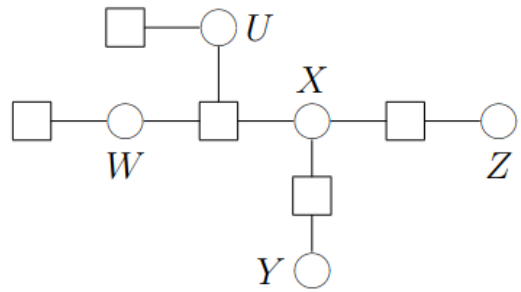
Different factor graph representation

$$p(u, w, x, y, z) = p(u)p(w)p(x|u, w)p(y|x)p(z|x).$$

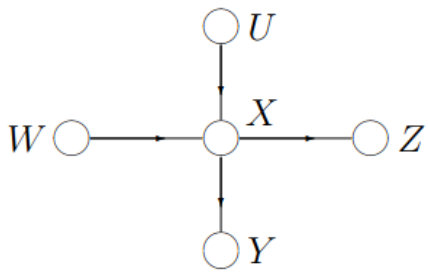
The different factor graph representations for the given equation is given below:



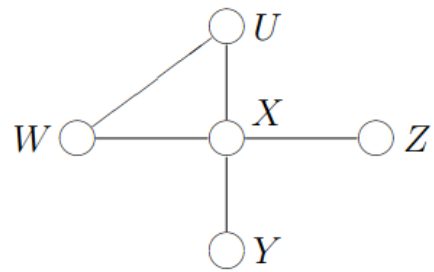
Forney-style factor graph.



Original factor graph [FKLW 1997].



Bayesian network.

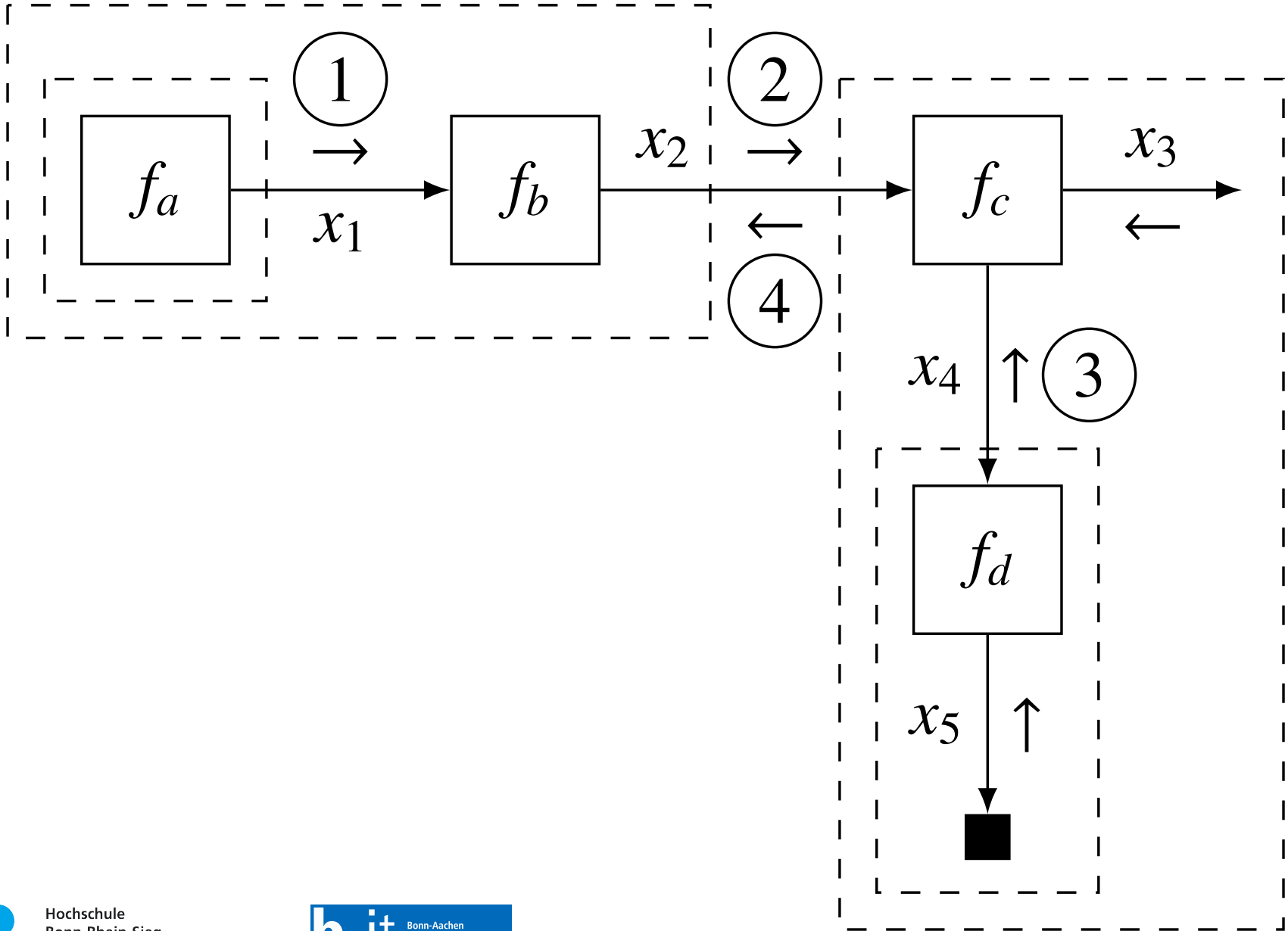


Markov random field.

Inferencing in factor graph

- Belief propagation or Sum-product algorithm
- Max-product algorithm
- Variational message passing
- Non-Linear optimization techniques (GTSAM)

Forney-Factor Graph



Summary (with reference to the above figure)

- Forney-style factor graph consists of variables as edges and factors as nodes
- Each factor is represented by a unique node (f_a, f_b, f_c, f_d)
- Each variables is represented by a unique edge ($x_1, ...x_5$) which meant for representing pose in our case
- Colored square block represents the clamping factor
- Joint probability distribution for the above shown figure can be represented as,

$$f(x_1, x_2, x_3, x_4, x_5) = f_a(x_1).f_b(x_1, x_2).f_c(x_2, x_3, x_4).f_d(x_4, x_5)$$

Equality node

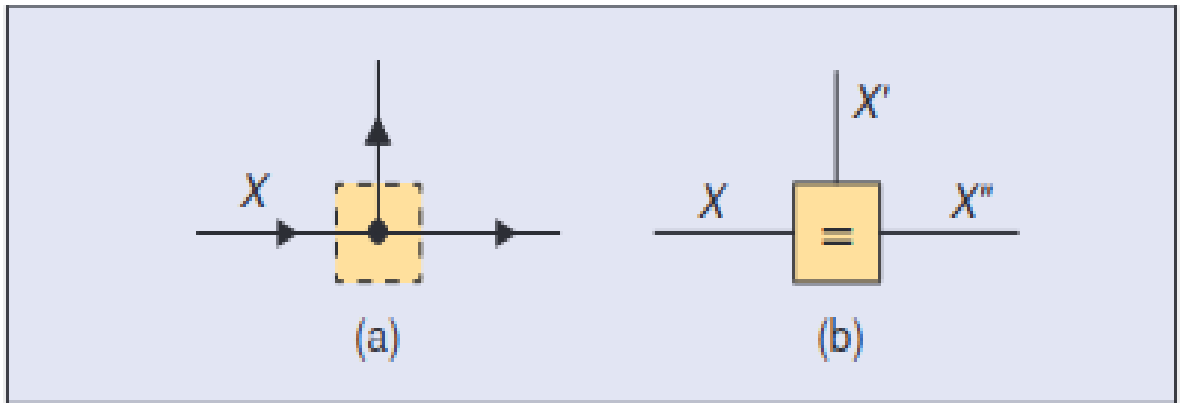


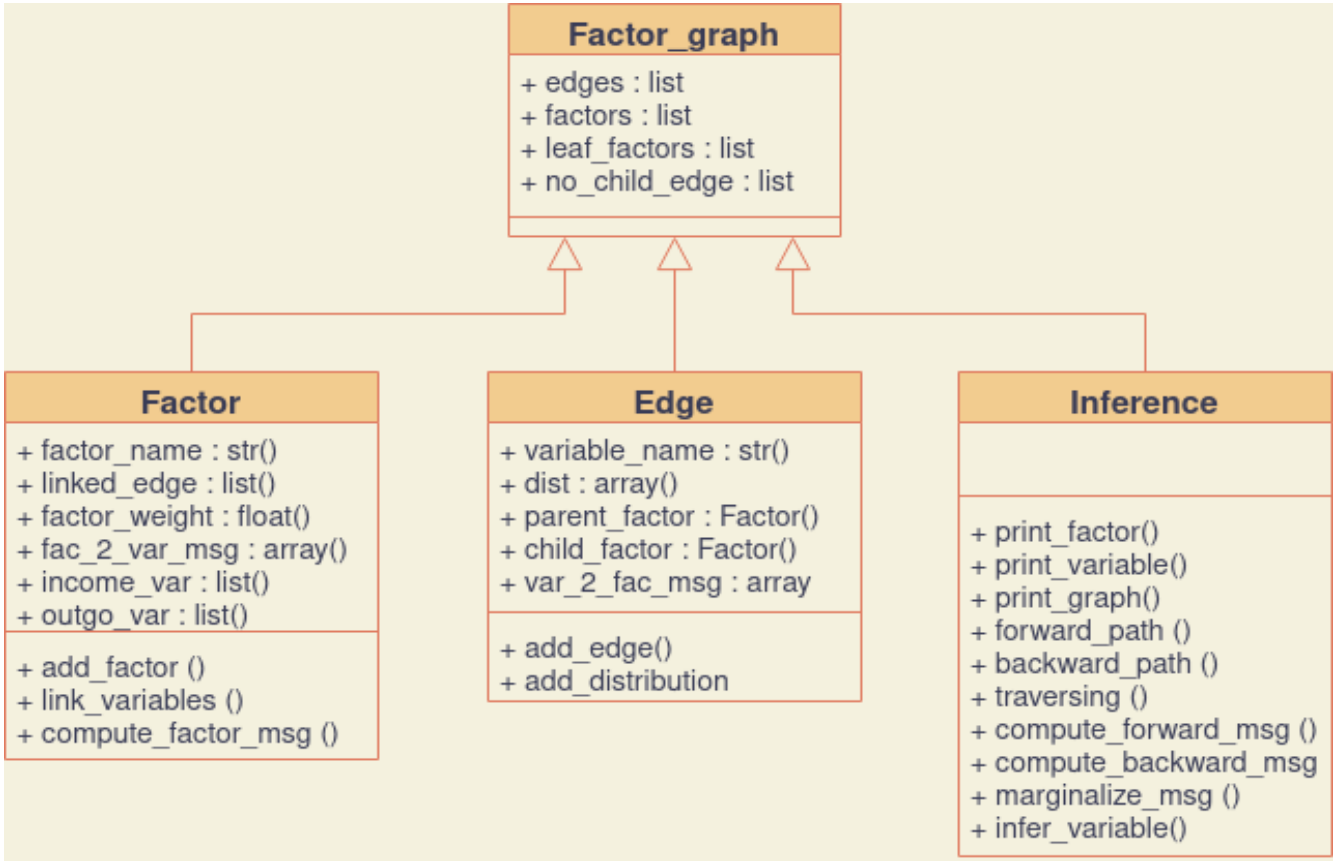
Figure 1: FFG with equality node

- No two variables should appear in more than two factor nodes
- It can be resolved by decomposing them into equality nodes

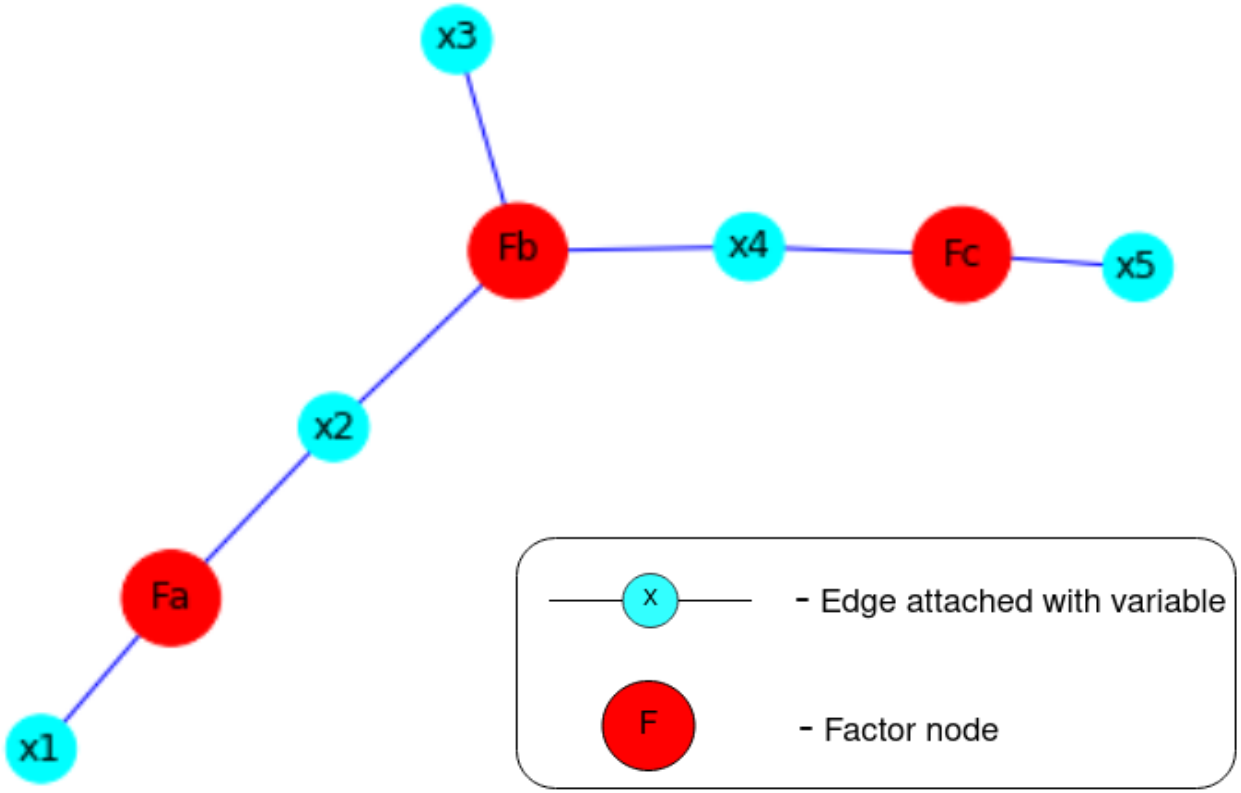
Why we need Forney-style Factor graph?

- Simplifies the process of message-passing between factors and variables
- Computation speed is high compared to the other factor graph implementations

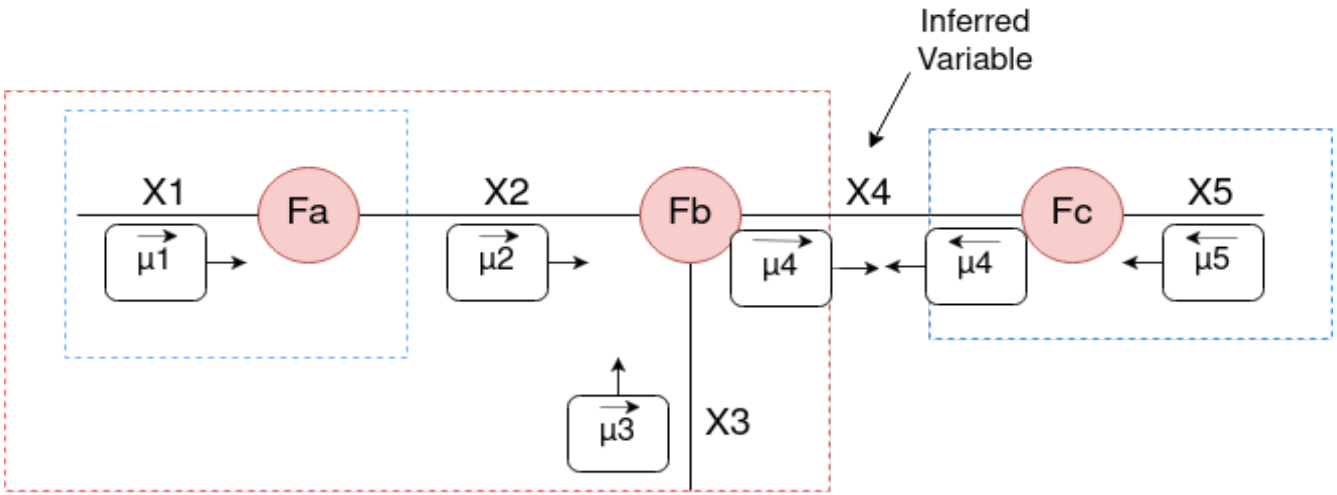
Forney Factor Graph UML Diagram



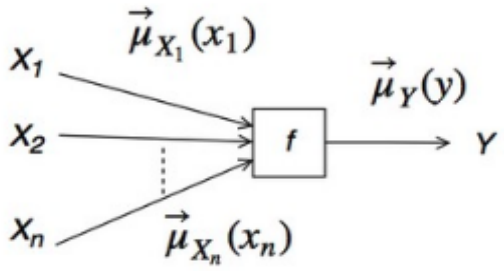
Sample Factor graph generation



Inferencing (Sum product algorithm



Outgoing Message



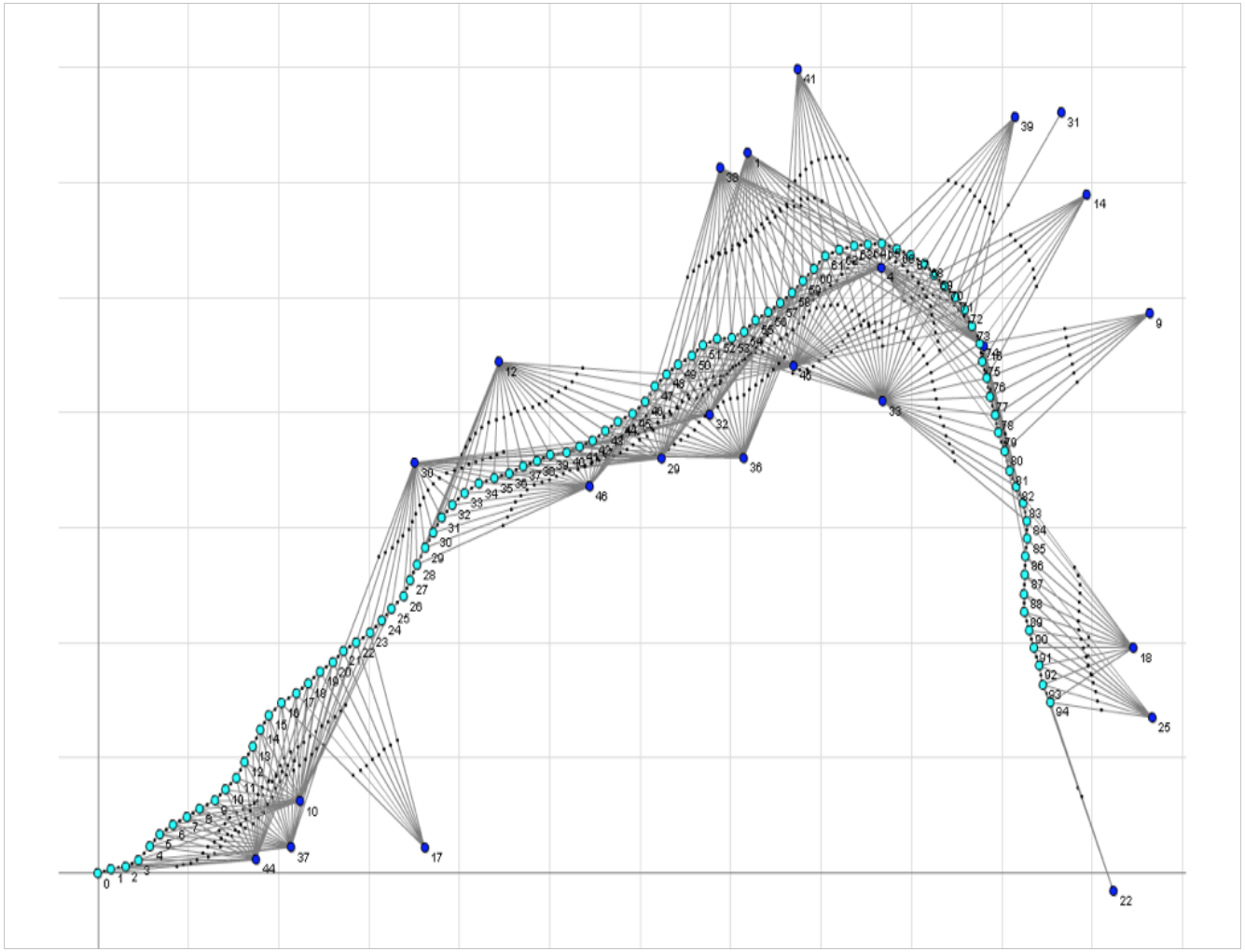
$$\mu_Y(y) = \sum_{x_1,\ldots x_n} \mu_{X_1}(x_1)\ldots\mu_{X_n}(x_n).\, f(y, x_1\ldots, x_n)$$

$$f(x_4) = \overrightarrow{\mu_4}.\overleftarrow{\mu_4}$$

Available factor graph libraries

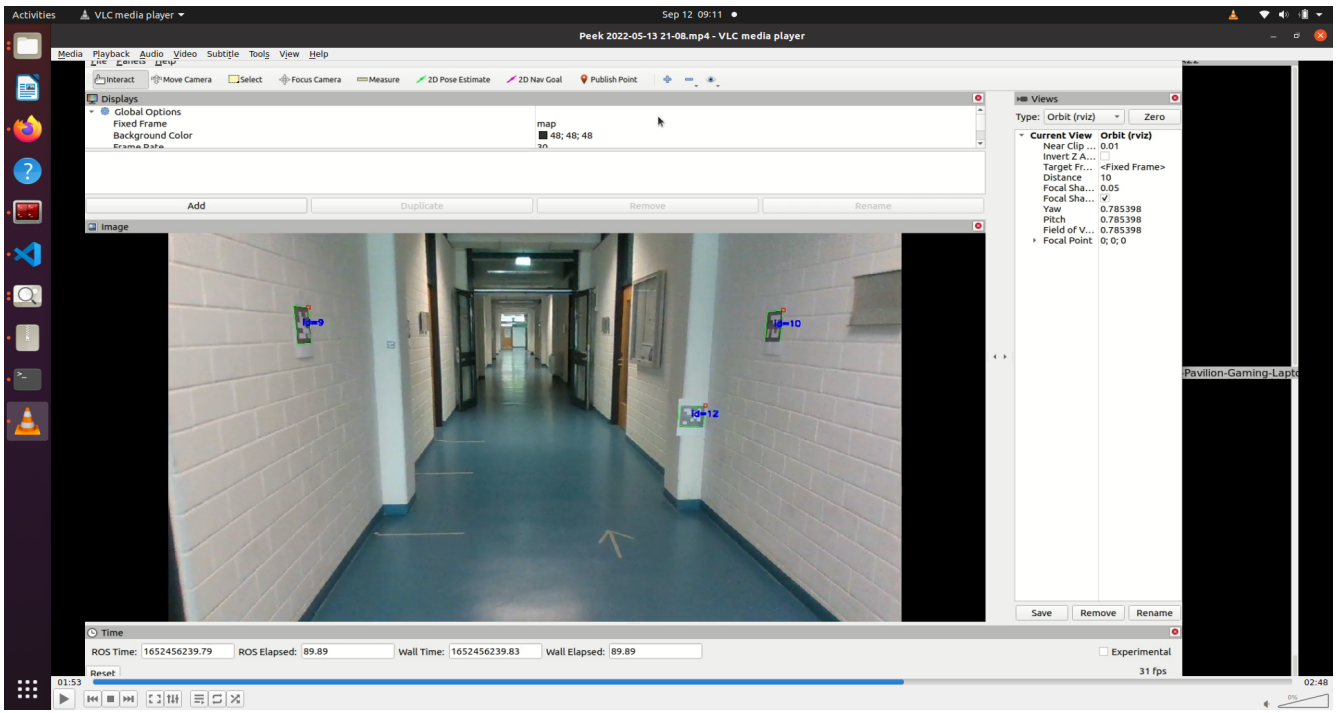
- GTSAM
- minisam
- g2o
- Ceres solver

SLAM using Factor graph

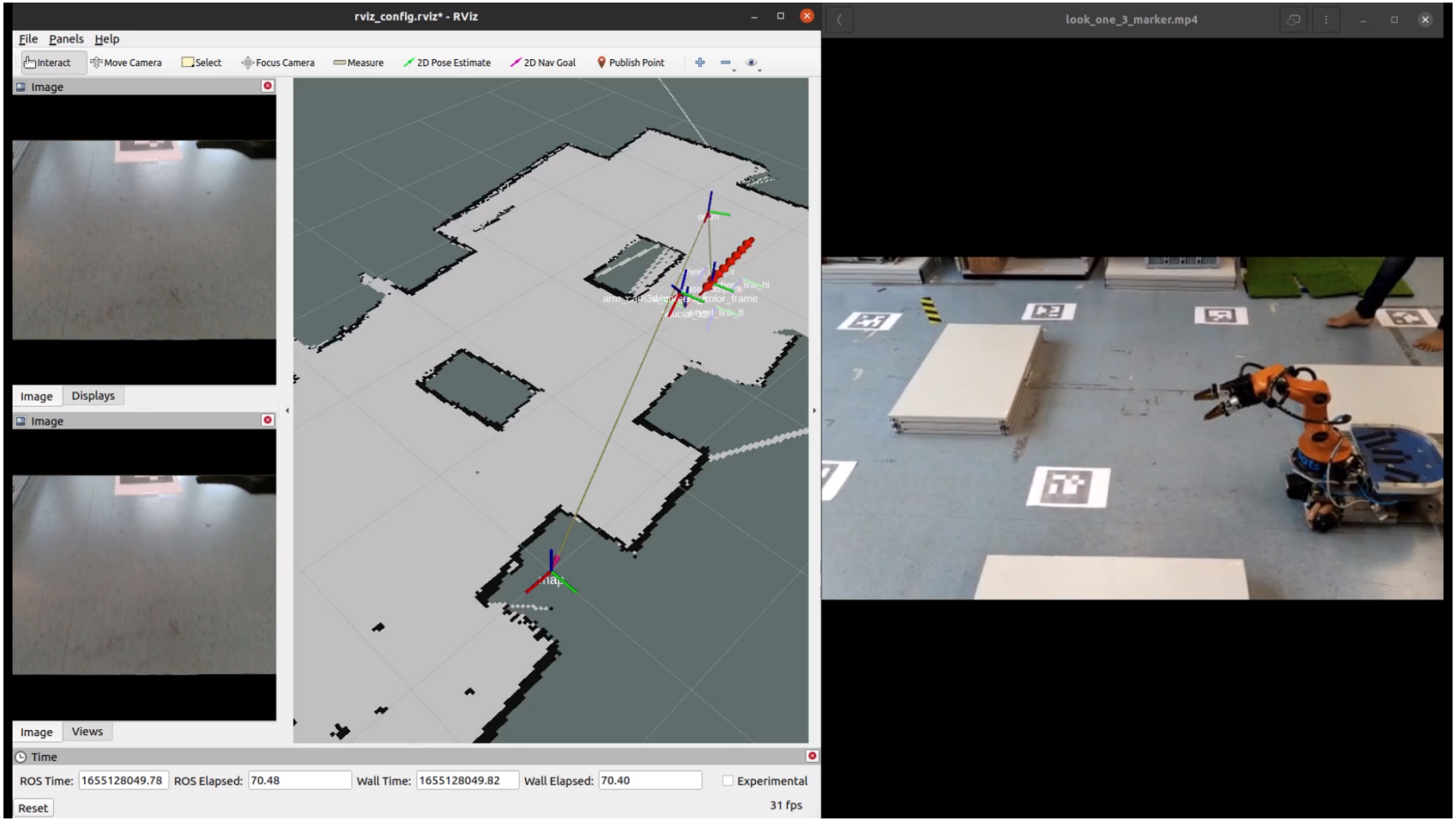


Aruco Markers as Landmarks

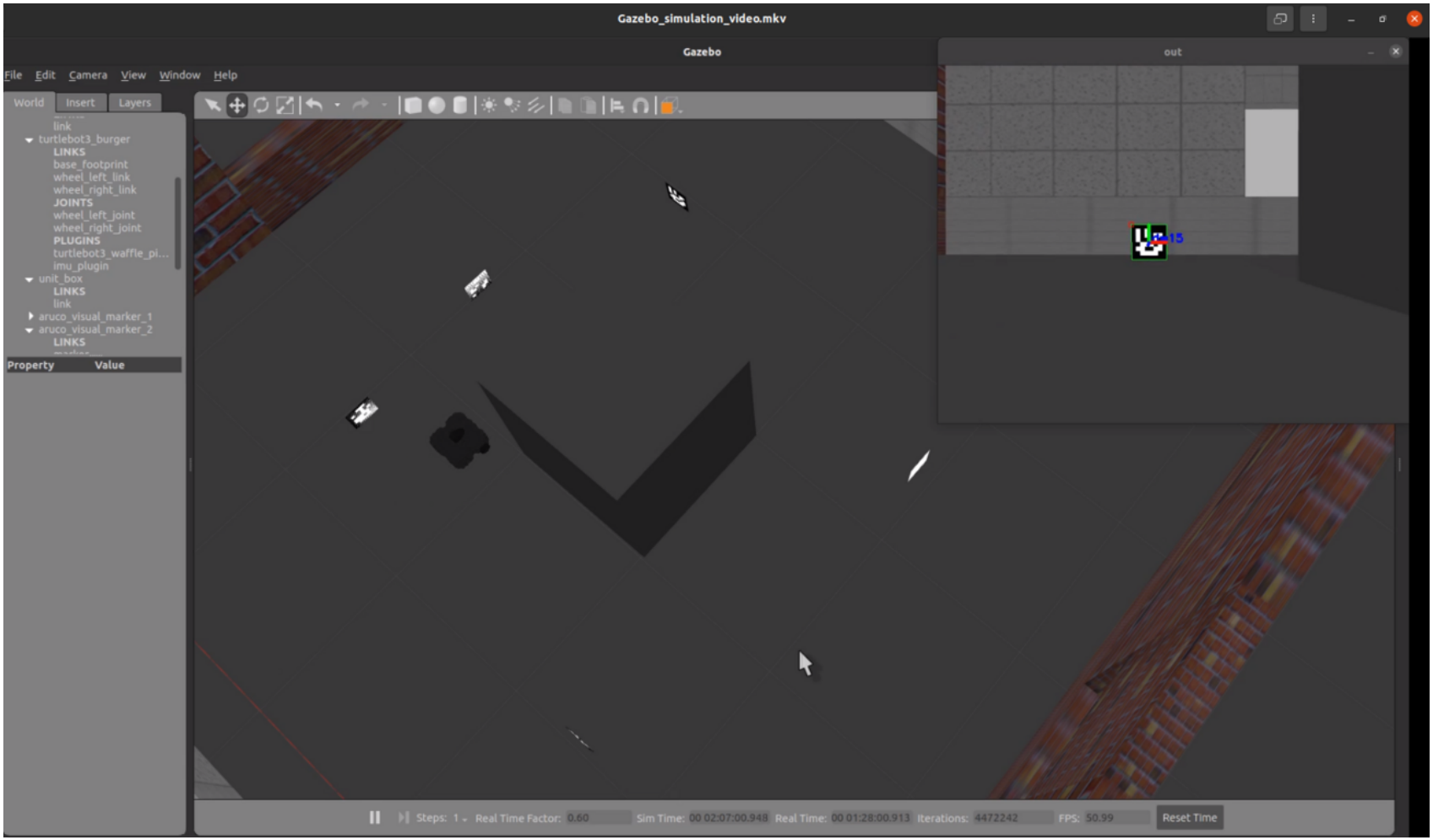
- Aruco markers are used as landmarks in the environment for SLAM
- Each Aruco marker has unique ID
- Markers are placed in the environment in such a way that one marker is always perceived by the camera



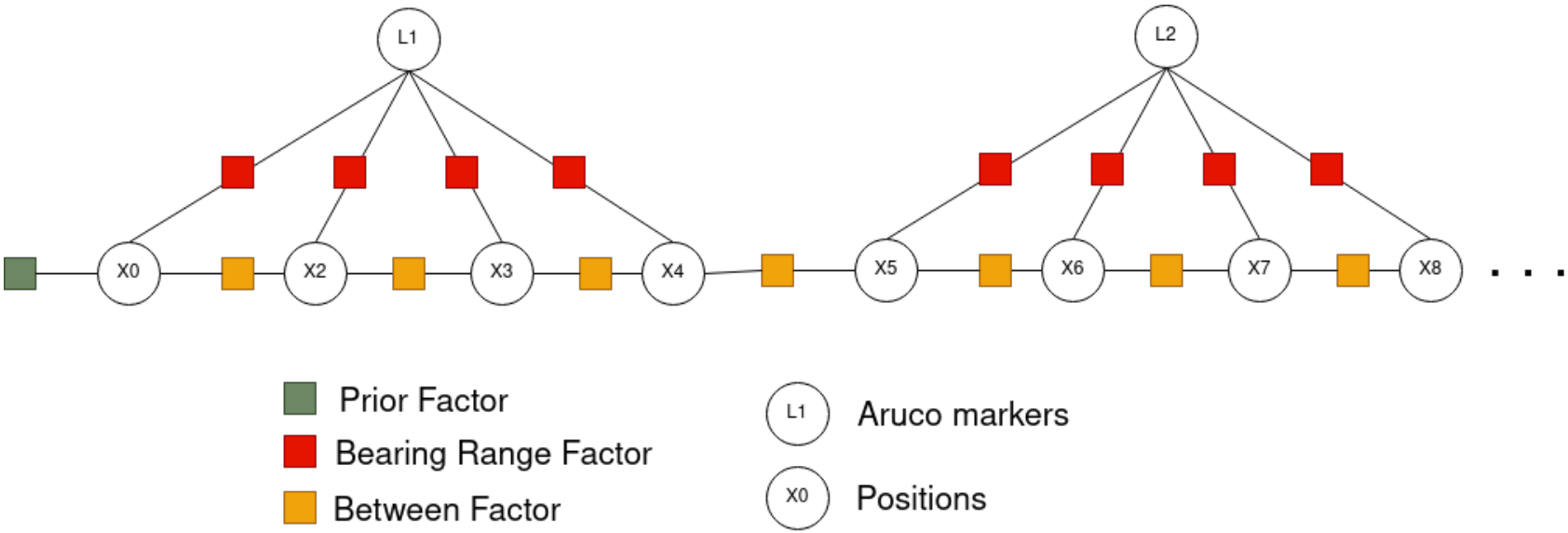
Data collection - Using Youbot



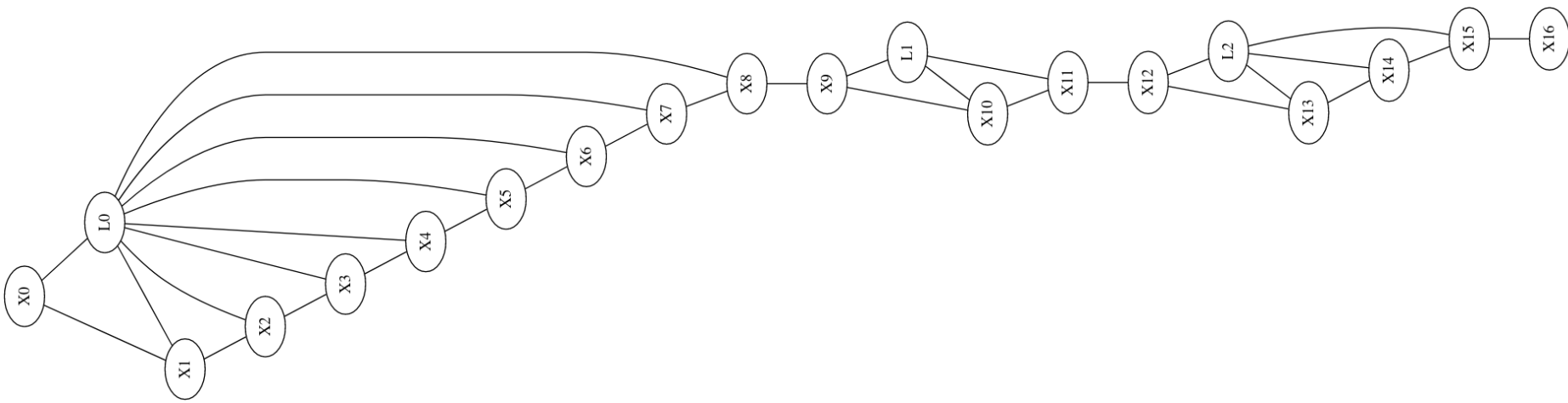
Data collection - Using Gazebo Simulation



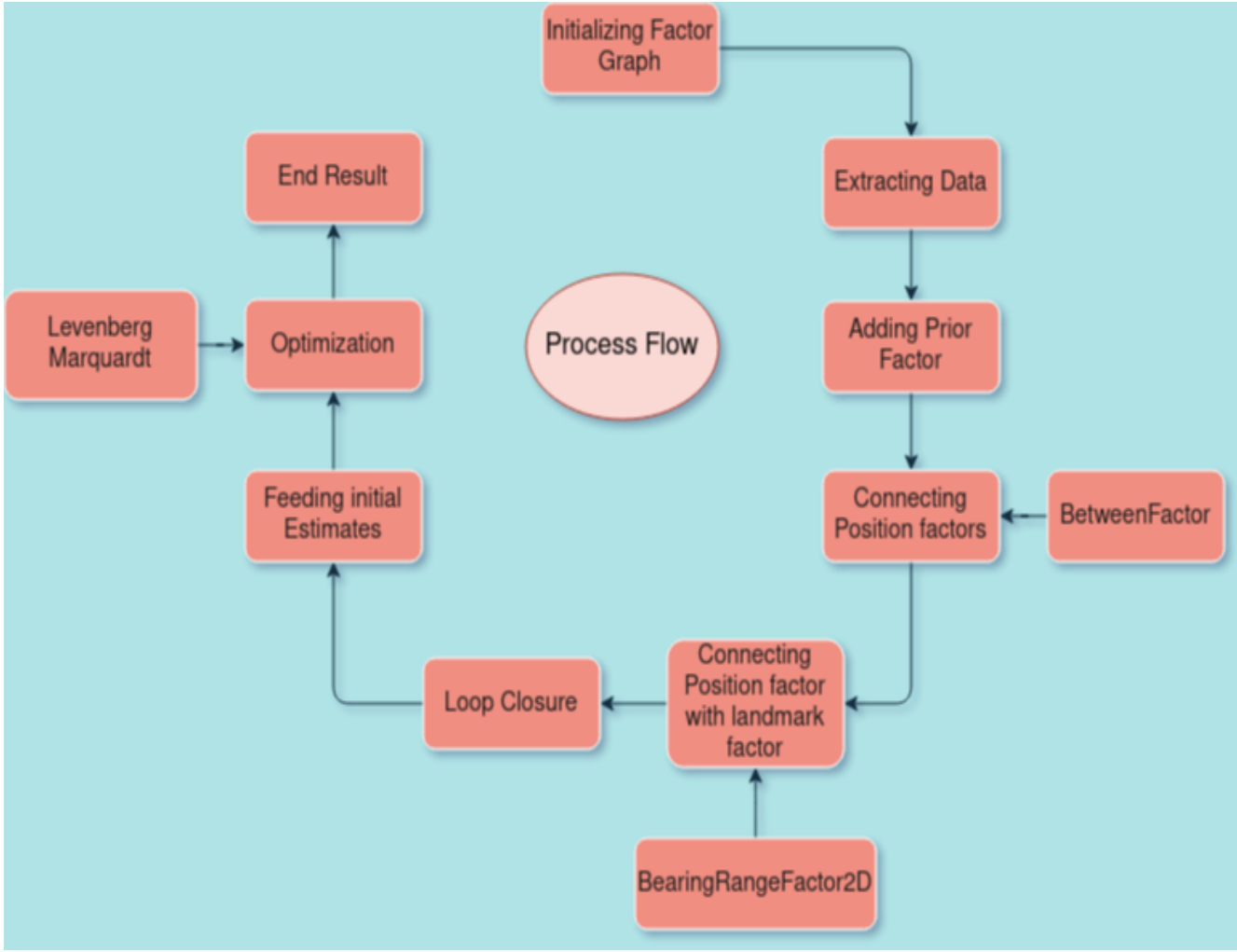
Standard planar SLAM representation



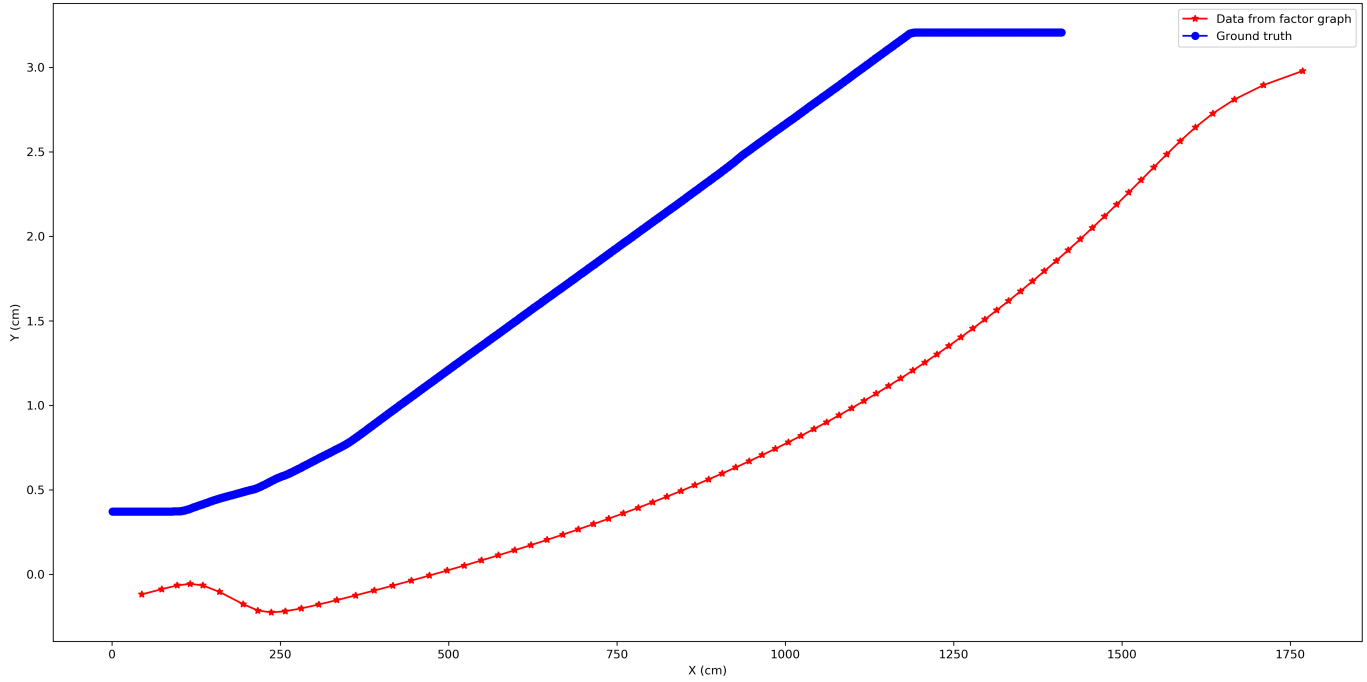
Factor graph representation of Youbot data



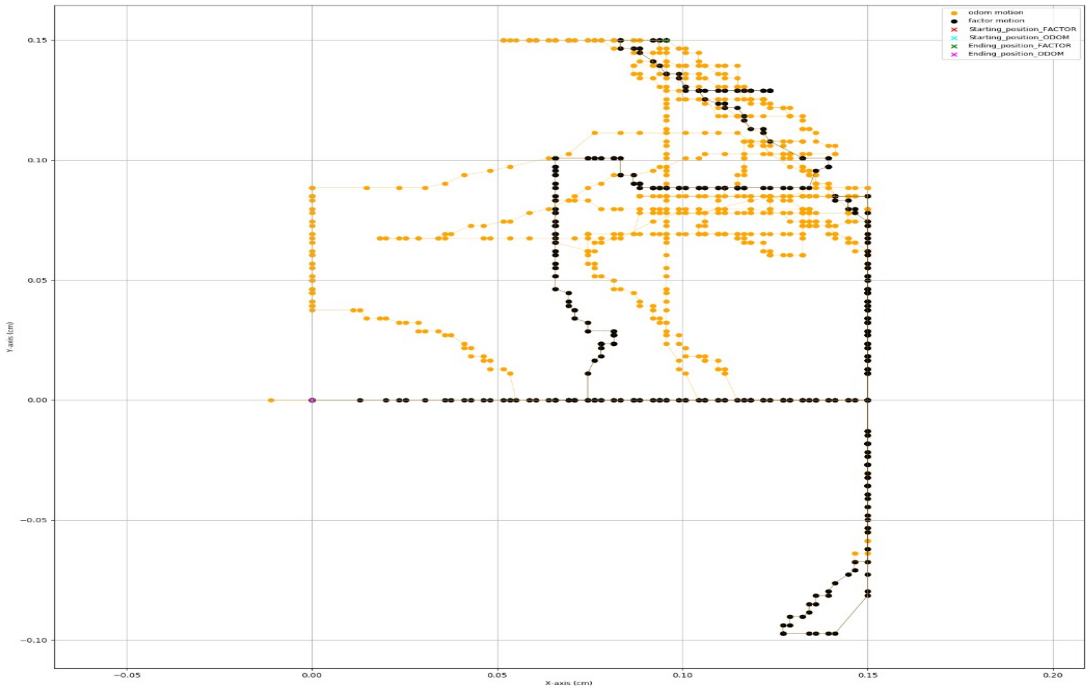
GTSAM implementation process flow



Factor data vs Ground truth using Youbot data (Previous result

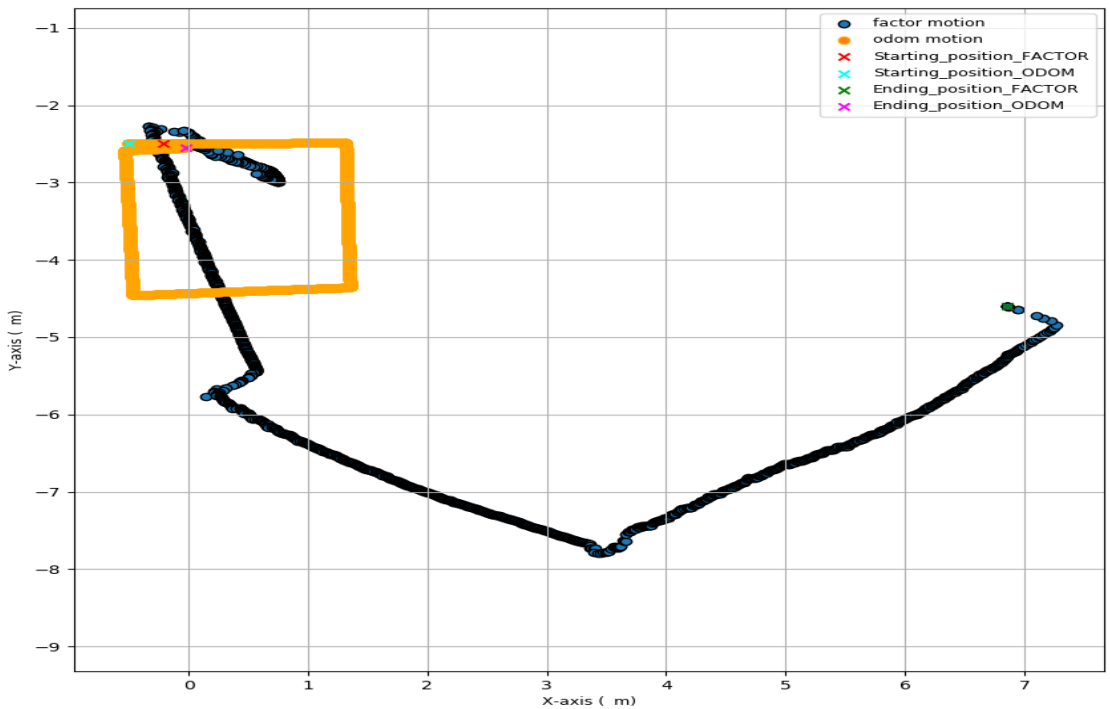


Results from Youbot data



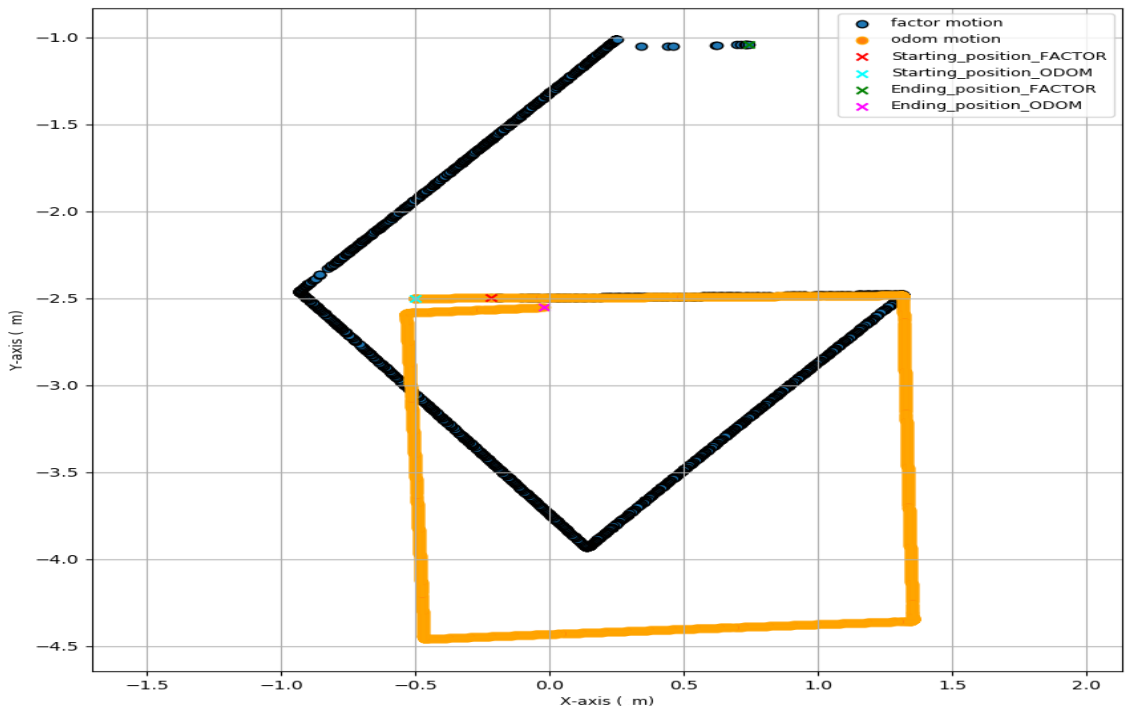
Results from simulated data

- Configuration :
  - Odometry noise: (0.02, 0.02, 0.01)
  - Measurement noise: (0.01, 0.02)
  - Prior noise: (0.03, 0.03, 0.01)
  - Loop closure: No



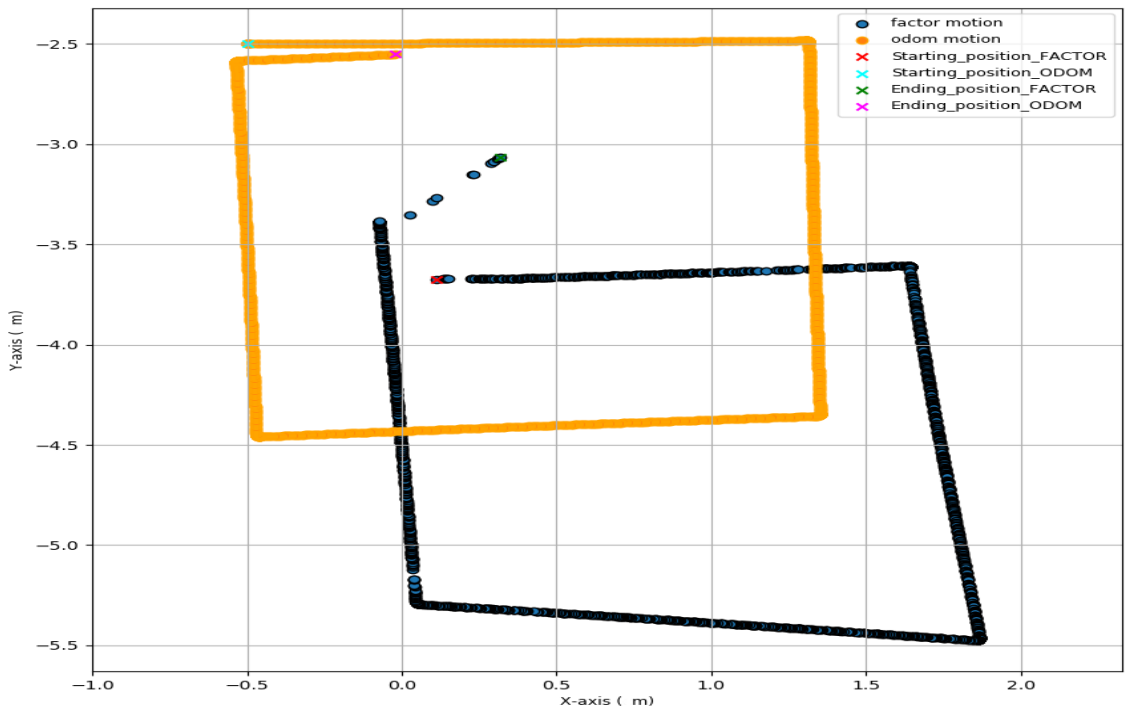
Results Contd.,

- Configuration :
  - Odometry noise: (0.0, 0.0, 0.0)
  - Measurement noise: (0.01, 0.02)
  - Prior noise: (0.03, 0.03, 0.01)
  - Loop closure: No



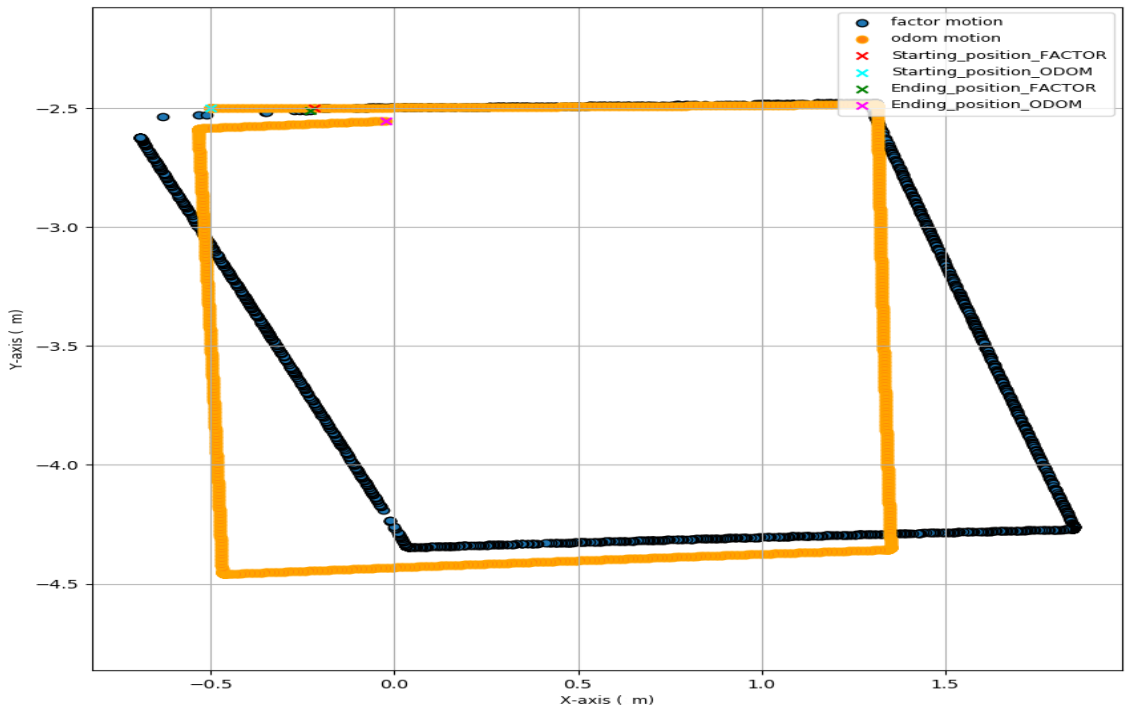
Results Contd.,

- Configuration :
  - Odometry noise: (0.0, 0.0, 0.0)
  - Measurement noise: (0.00, 0.00)
  - Prior noise: (0.03, 0.03, 0.01)
  - Loop closure: Yes, End factor connected to First factor



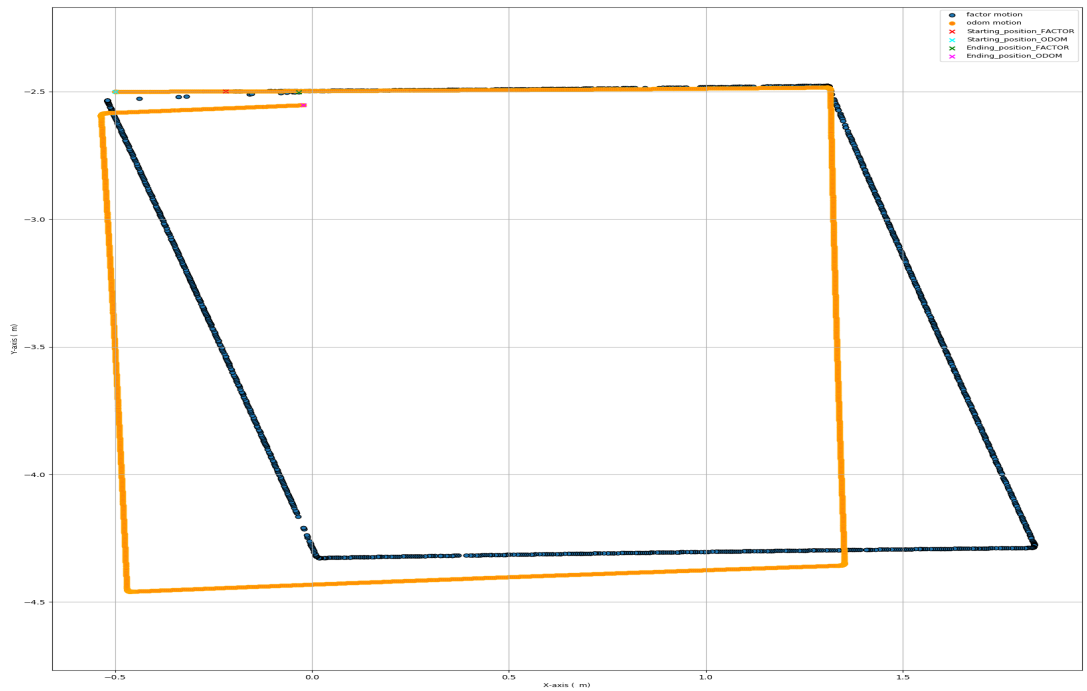
Results Contd.,

- Configuration :
  - Odometry noise: (0.0, 0.0, 0.0)
  - Measurement noise: (0.01, 0.02)
  - Prior noise: (0.03, 0.03, 0.01)
  - Loop closure: Yes, End factor connected to First factor



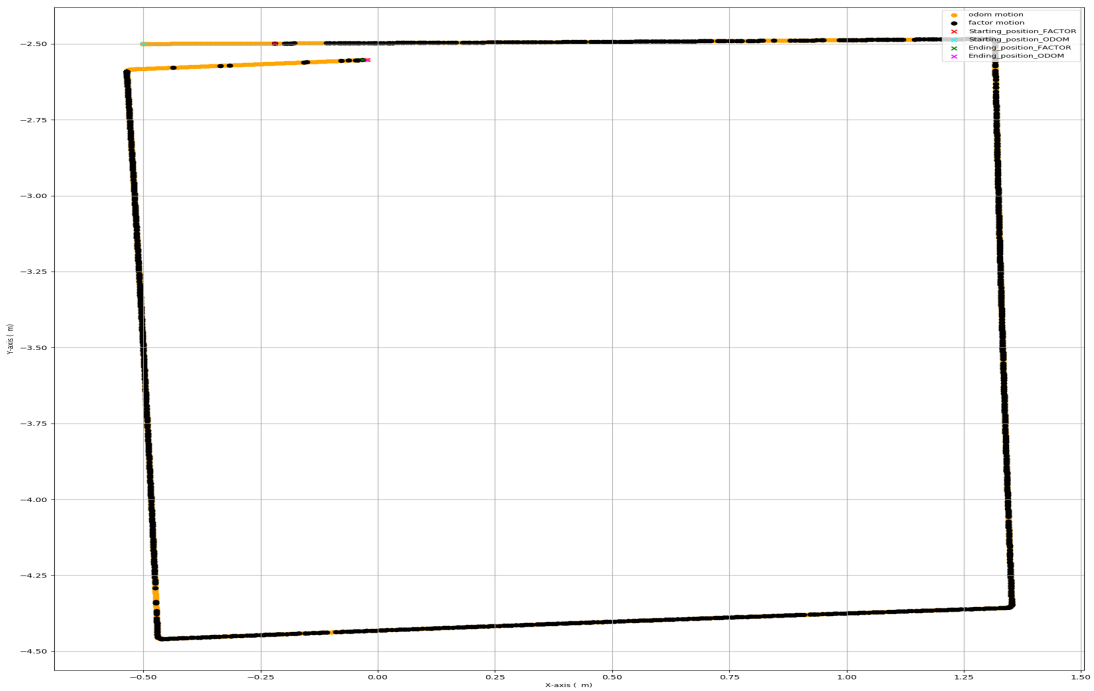
Results Contd.,

- Configuration :
  - Odometry noise: (0.0, 0.0, 0.0)
  - Measurement noise: (0.01, 0.02)
  - Prior noise: (0.0, 0.0, 0.0)
  - Loop closure: Yes, End factor connected to closest factor

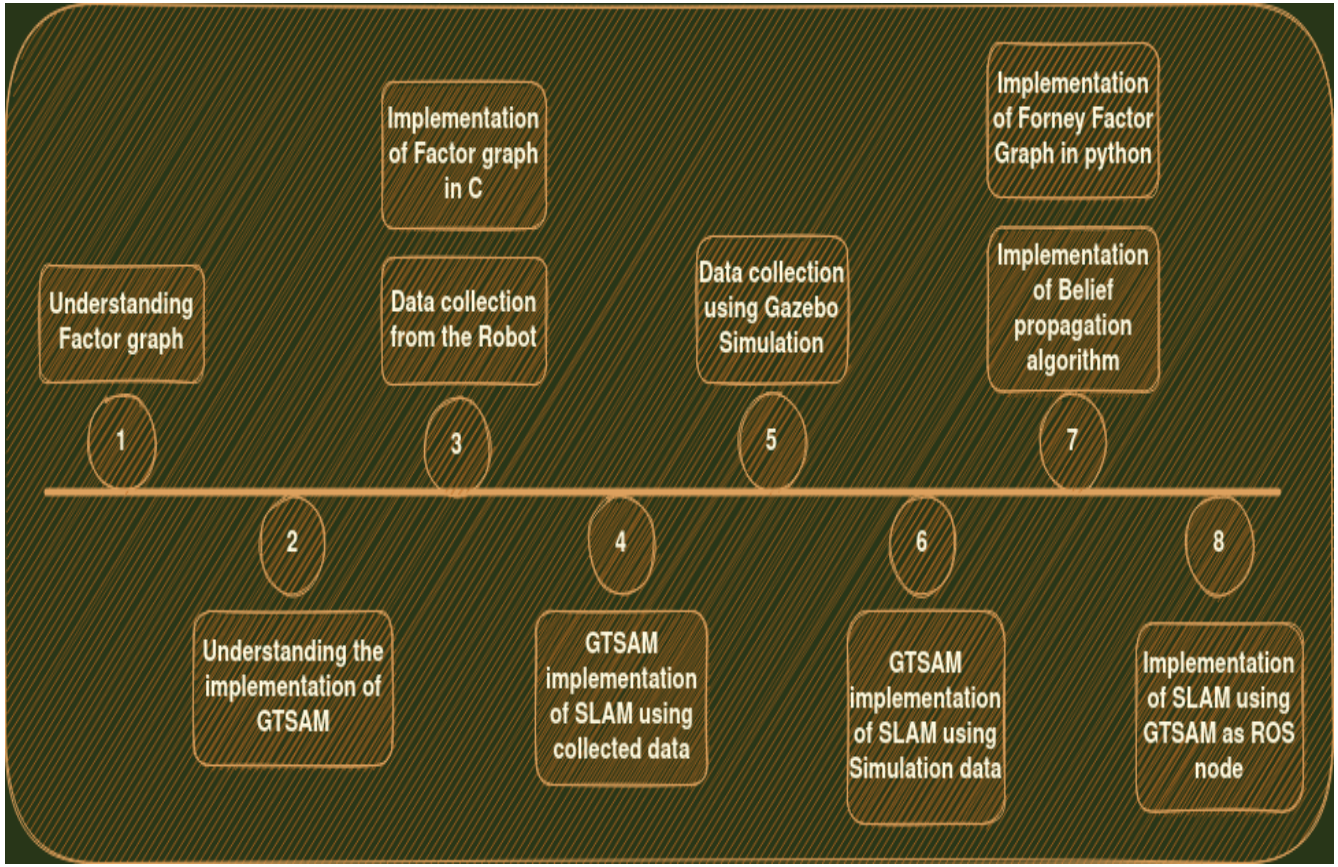


Results Contd.,

- Configuration :
  - Odometry noise: (0.0, 0.0, 0.0)
  - Measurement noise: (0.1, 0.2)
  - Prior noise: (0.0005, 0.0003, 0.0001)
  - Loop closure: Yes, End factor connected to closest factor



Project Workflow



Software development methodology

SCRUM process

- Goal/backlog setting for next sprint
- Retrospection of the past sprint
- Sprint Meetings
  - Meeting among developers to discuss what has been done/ongoing [10min]
  - Sprint meetings every three weeks with scrum master/coach [1hr]
    1. April 25
    2. May 16
    3. June 7
    4. June 27
    5. July 20

Means of communication

- **Github:** Task assigning, maintaining to-do lists, documentation of meetings
- **Webex/offline:** For conducting sprint meetings and technical discussions

Tools and Technologies

Languages

- C, C++
- Python