Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

SLAM with Factor Graphs

June 13, 2022

Ravisankar Selvaraju
Samuel Salazar
Selvakumar Nachimuthu
Tharun Sethuraman

Advisors

Deebul nair

Team members

Ravisankar

Samuel

Tharun

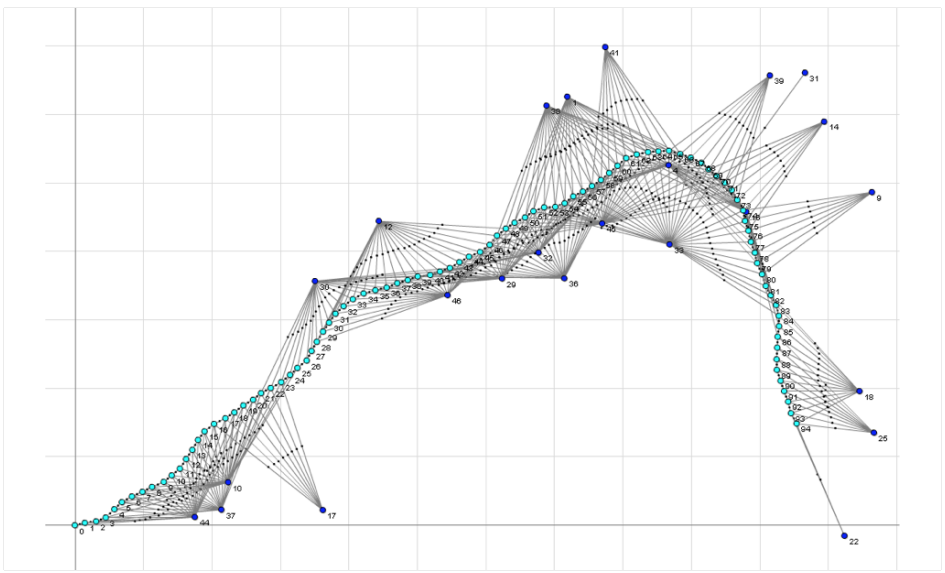
Selvakumar

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

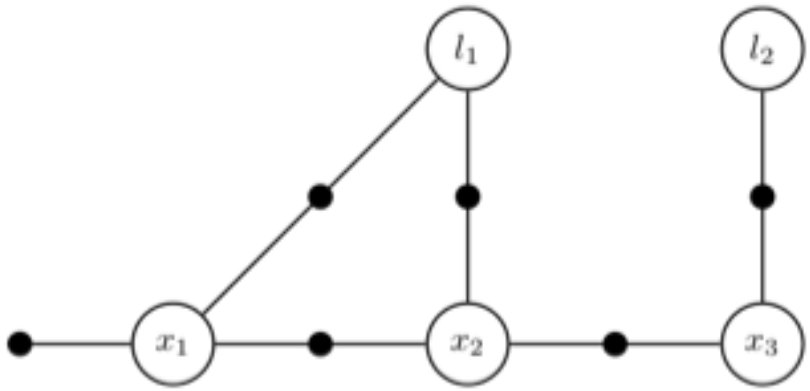


What is SLAM?

- 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)



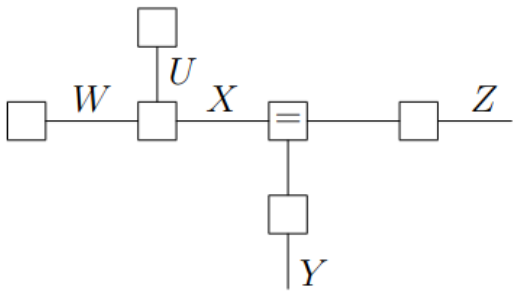
Why do we need a new factor graph library?

- The existing GTSAM library uses the conventional representation [FKLW 1997] of factor graph and it uses non-linear optimization techniques like (LM optimizer, Dogleg optimizer) to obtain the posterior
- This optimization is more complex when the size and constraints in the graph increases.
- The GTSAM library has its core as factor graph for Robotic application like SLAM and so it cannot be used to represent general problems
- The GTSAM implementation uses Non linear optimisation techniques as inferencing method but using belief propagation algorithm(Sum product algorithm) increases the computation speed of the optimisation process

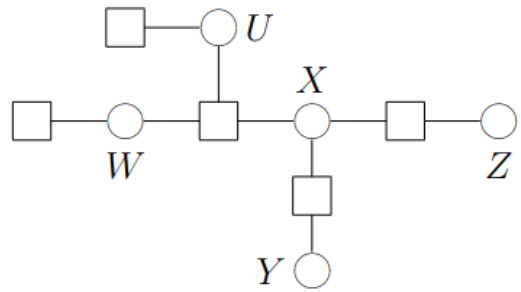
Different factor graph representation

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

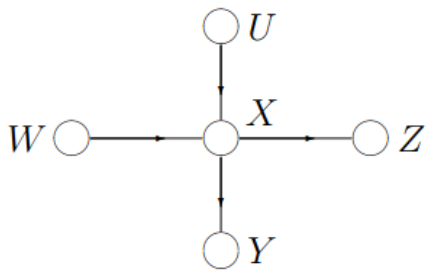
The different factor graph representations for the given equation is given below:



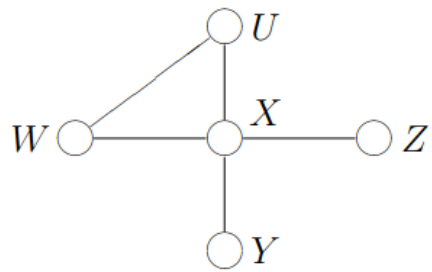
Forney-style factor graph.



Original factor graph [FKLW 1997].



Bayesian network.

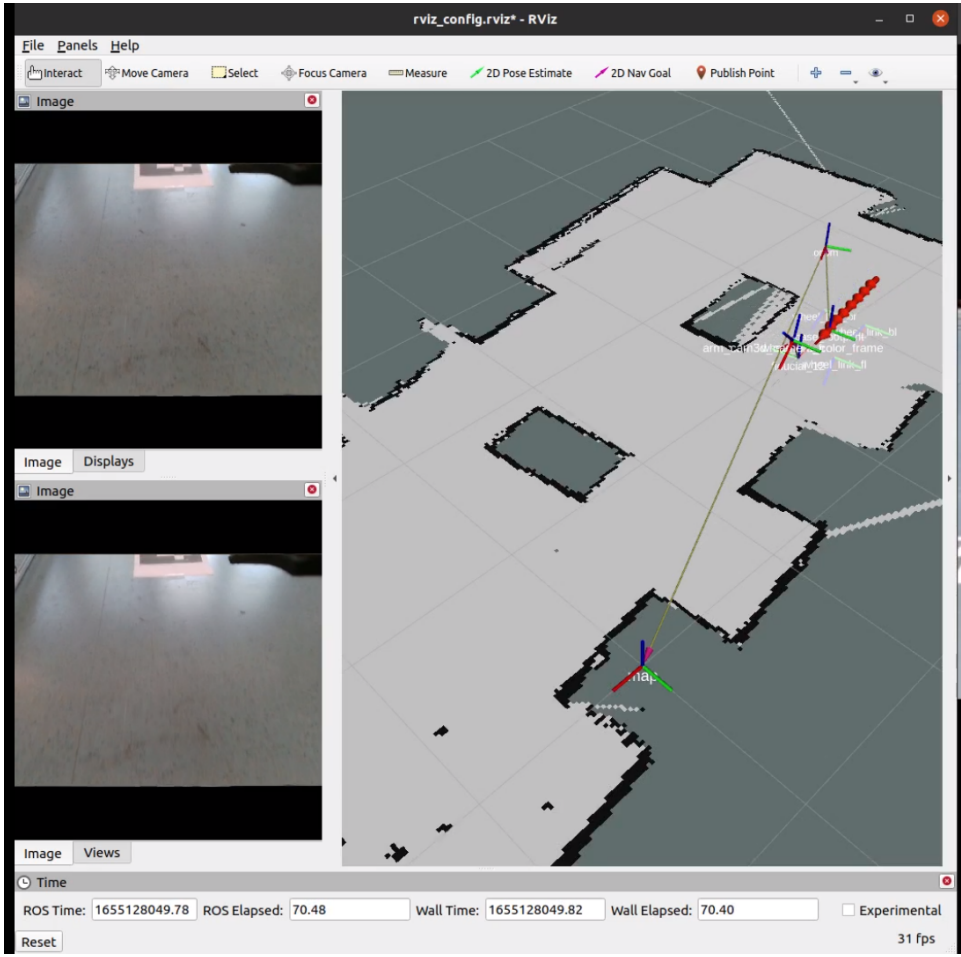


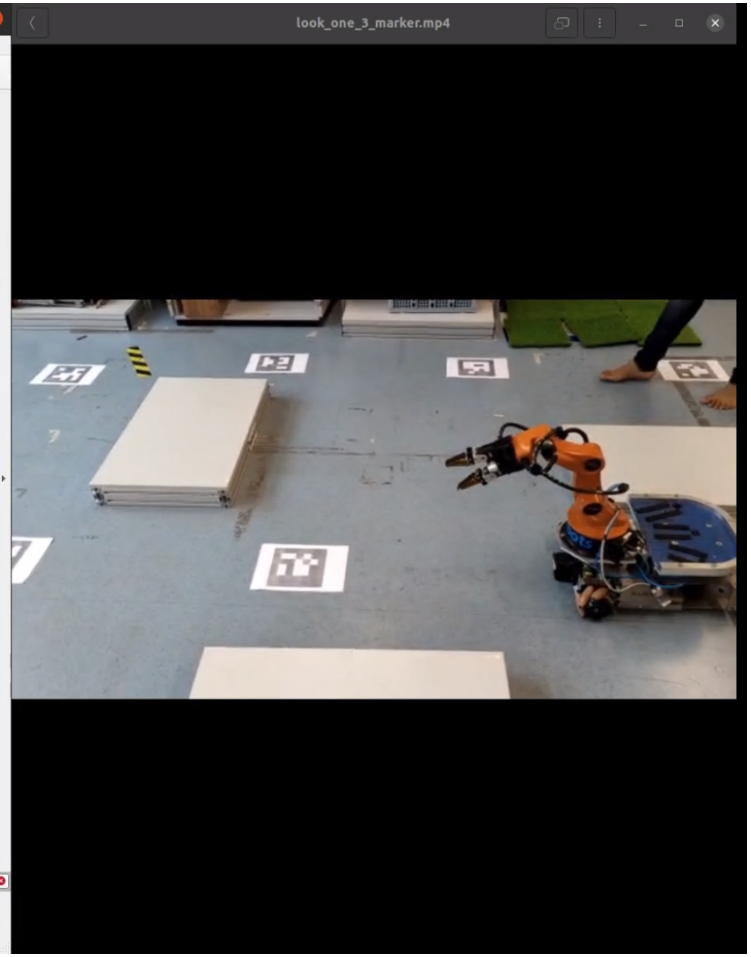
Markov random field.

Available factor graph libraries

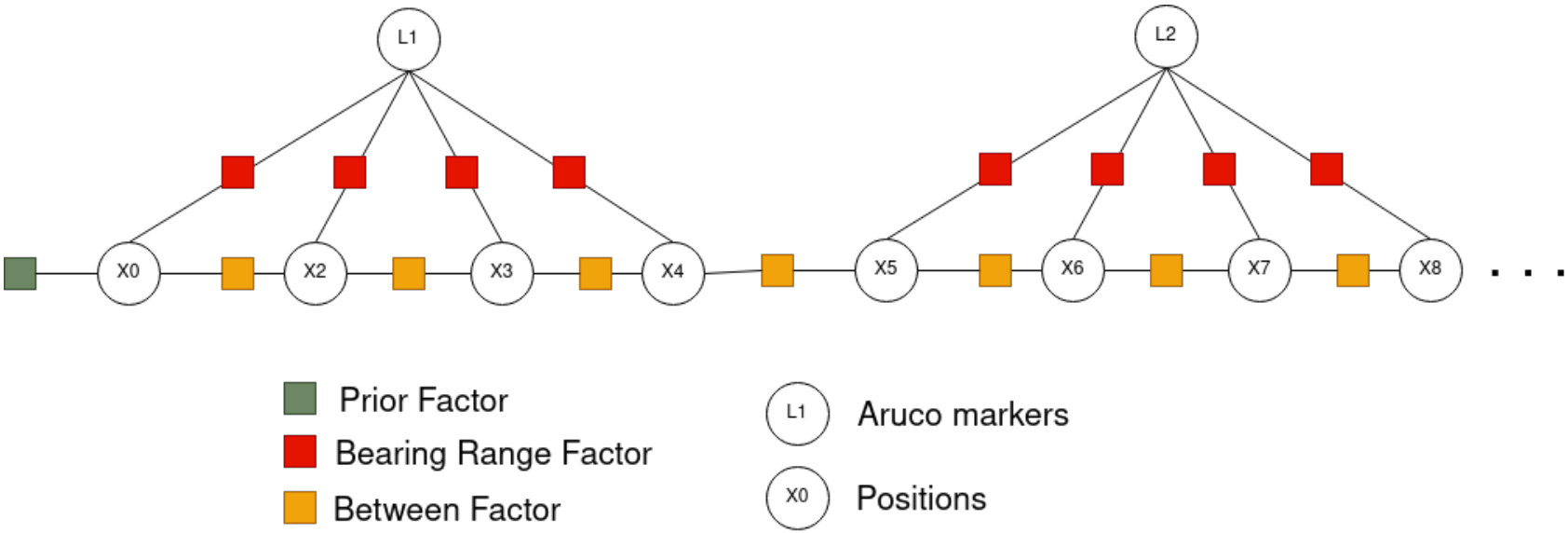
- GTSAM
- minisam
- g2o
- Ceres solver

Data collection setup

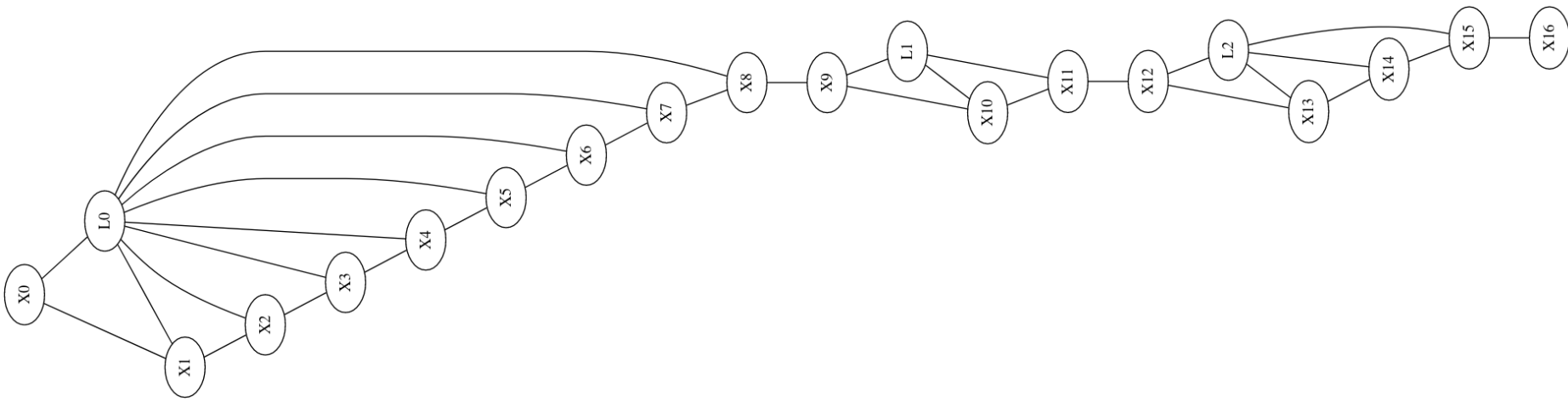




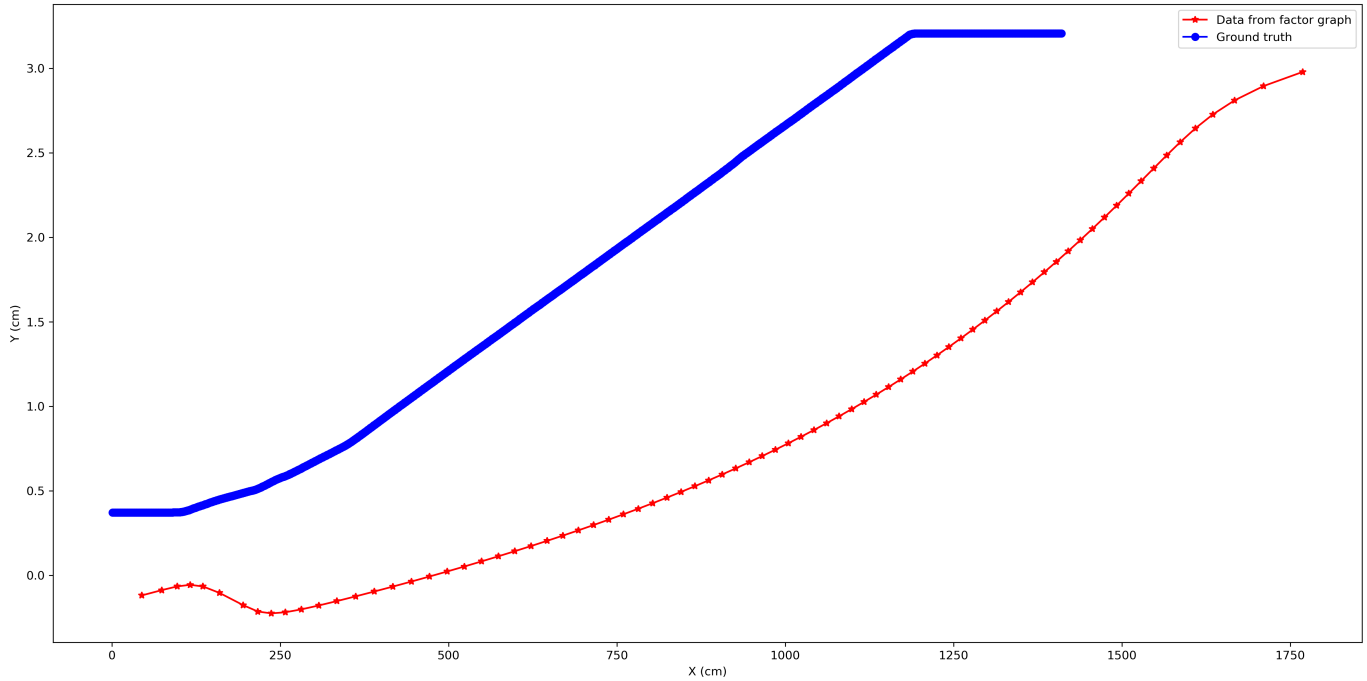
Standard planar SLAM representation



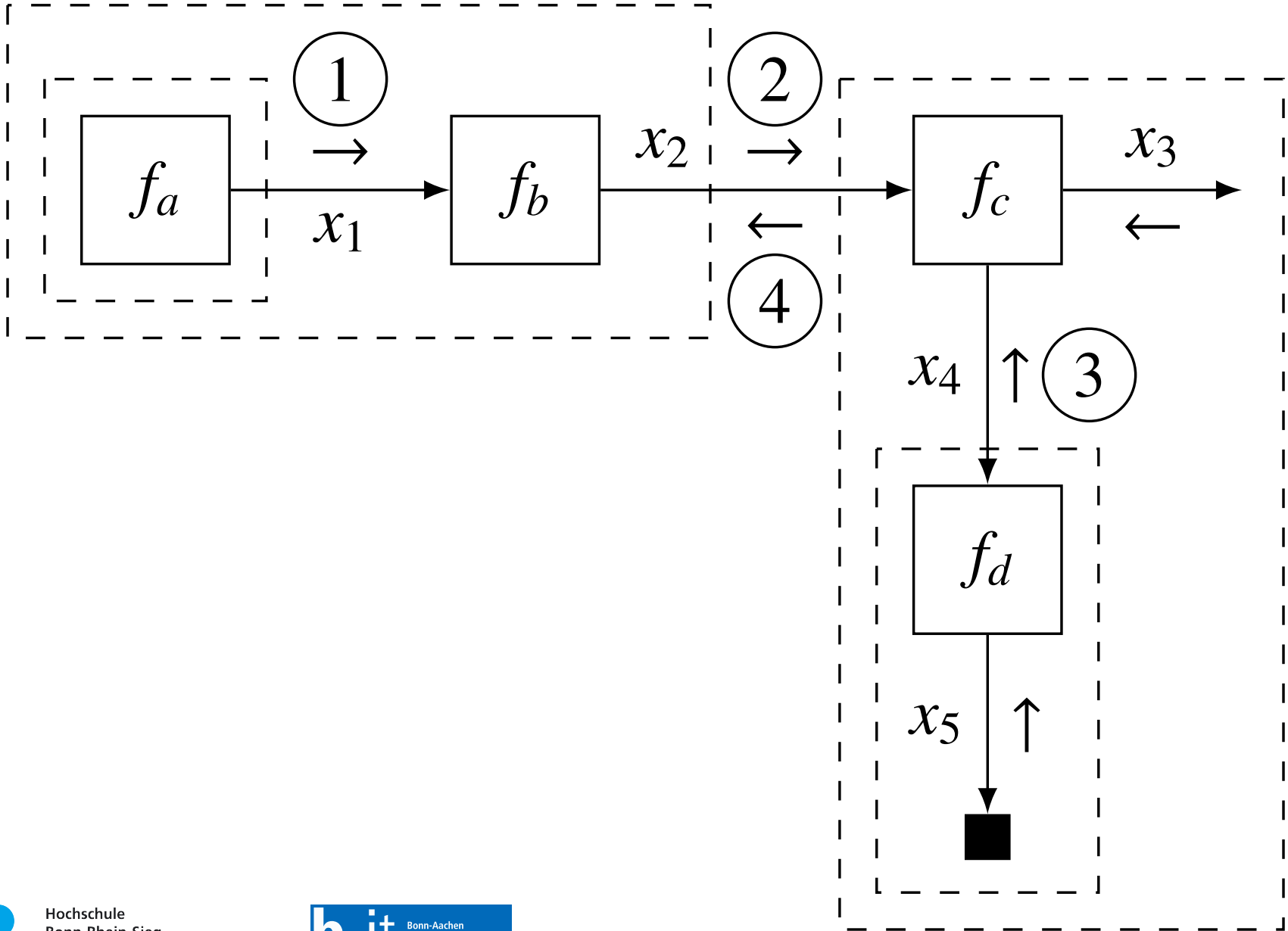
Our data planar SLAM representation



Graph from 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

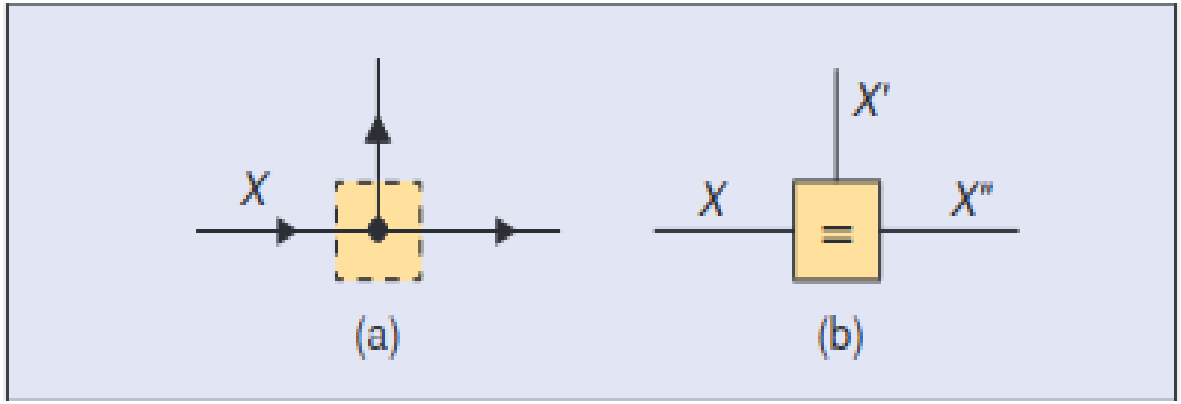


Figure 1: FFG with equality node

- 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

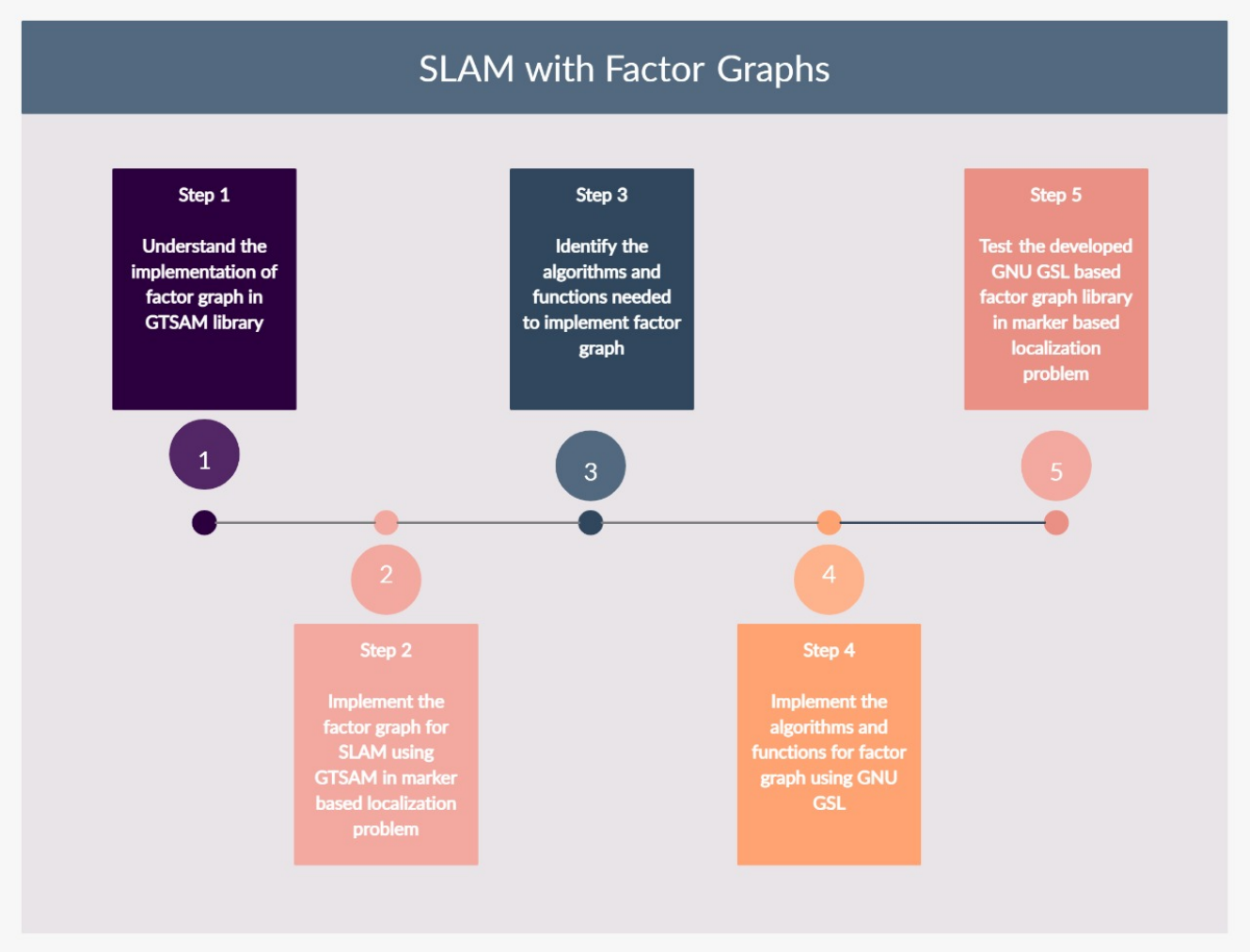
To-Do for implementing the library in C

- Creating a factor graph (data structure) resembling Forney-Factor Graphs
- Modify the created factor graph using functions from GNU-GSL
- Understand the sum-product (and required algorithms) algorithm by implementing a simple message passing across a graph
- Implement the same with the created factor graph

Libraries identified in C

- GNU-GSL
- apophenia
- GNU plot

Process Workflow



Goals yet to be achieved

- Creating a graph data structure for factor graphs using GNU-GSL libraries
- Perform the message passing algorithm across the variable and factors nodes of the created factor graph
- Creating a C library based on GNU-GSL which takes inputs (variables, factors, priors, and observations) and performs inference on such inputs by building a graph
- Generating a data set by detecting aruco markers in gazebo simulation to later use it to create factor graphs.

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

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