# Module Interface Specification for 2D Localizer

Aliyah Jimoh

April 17, 2025

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| 2025/03/19 | 1.0 | Initial Draft |

# 2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/AliyahJimoh/2D-Localizer/blob/main/docs/SRS/SRS.pdf

# Contents

# 3 Introduction

The following document details the Module Interface Specifications for 2D Localizer, a program that implements various sensors to help localize mobile robots on a 2D plane in enclosed environments.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/AliyahJimoh/2D-Localizer.

# 4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by 2D Localizer.

| **Data Type** | **Notation** | **Description** |
|---|---|---|
| character | char | a single symbol or digit |
| factor | Factor | a constraint in a factor graph that relates variables |
| factor graph | Graph | a collection of factors defining an optimization problem |
| integer | $\mathbb{Z}$ | a number without a fractional component in (-$\infty$, $\infty$) |
| natural number | $\mathbb{N}$ | a number without a fractional component in [1, $\infty$) |
| noise Model | Model | a model that defines uncertainty in a measurement |
| real | $\mathbb{R}$ | any number in (-$\infty$, $\infty$) |
| string | String | more than one symbol put together |
| values | Values | a container that stores variable estimates in a factor graph |

The specification of 2D Localizer uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, 2D Localizer uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

2D Localizer also uses data types from the Georgia Tech Smoothing and Mapping (GTSAM) library which is used for solving estimation problems using factor graphs. Factor graphs are a way to represent relationships between variables using "factors"(pieces of information gotten from sensors or motion)

# 5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

| **Level 1** | **Level 2** |
|---|---|
| Hardware-Hiding Module | |
| Behaviour-Hiding Module | GTSAM Module<br>Input Format Module<br>Output Module<br>Localization Module<br>Control Module<br>Accuracy Evaluation Module |
| Software Decision Module | Plotting Module |

Table 1: Module Hierarchy

# 6 MIS of Control Module

## 6.1 Module

main

## 6.2 Uses

- Input Format Module (Section 8)
- Localization Module (Section 9)
- Accuracy Evaluation Module (Section 10)
- Plotting Module (Section 12)
- Output Module (Section 11)

### 6.2.1 Exported Constants

None

### 6.2.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| main | - | - | - |

## 6.3 Semantics

### 6.3.1 State Variables

None

### 6.3.2 Environment Variables

- date_queue: A queue storing tuples of estimated pose data (time, x, y, theta).

### 6.3.3 Assumptions

None

### 6.3.4 Access Routine Semantics

main():

- transition: Modifying data_queue with each iteration of range measurements as the Plotting and Output modules get updated

*# Get Data*

input = InputData()

*# Start the Output Data*

data_queue = Queue()

process = Process(target=run_gui, args=(data_queue,))

process.start()

m = np.size($\tilde{\mathbf{D}}$, 0)

*# Getting estimated pose for each set of measurements*

for t in range(1,m):

    $\hat{\mathbf{x}}$:= localize($\mathbf{a}$, $T_{mf}$, $T_{rf}$, $\tilde{\mathbf{D}}[t, :]$)

    *# Computing FIM & CRLB*

    fim = compute_fim($\hat{\mathbf{x}}$, $\mathbf{a}$, variances($\boldsymbol{\sigma}^{\mathbf{2}}$))

    crlb = compute_crlb(fim) *# Will be printed*

    update_trajectory($\hat{\mathbf{x}}$)

    data_queue.put((t, $\hat{\mathbf{x}}$.x(), $\hat{\mathbf{x}}$.y(), $\hat{\mathbf{x}}$.theta()))

*# Plot on the map*

plot_localization_live($\mathbf{a}$, $T_{mf}$, map)

# 7 MIS of GTSAM Module

## 7.1 Module

gtsam_wrapper

## 7.2 Uses

None

## 7.3 Syntax

### 7.3.1 Exported Constants

None

### 7.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| Pose2 | $x : \mathbb{R}, y : \mathbb{R}, \theta : \mathbb{R}$ | $\mathbb{R}^3$ | - |
| Point2 | $x : \mathbb{R}, y : \mathbb{R}$ | $\mathbb{R}^2$ | - |
| symbol | char: char, int: $\mathbb{Z}$ | String | - |
| NonlinearFactorGraph | - | Graph | - |
| PriorFactorPose2 | $key : \mathbb{Z}, \textbf{pose} : \mathbb{R}^3, noise : Model$ | Factor | - |
| PriorFactorPoint2 | $key : \mathbb{Z}, \textbf{pose} : \mathbb{R}^2, noise : Model$ | Factor | - |
| RangeFactor2D | $key1 : \mathbb{Z}, key2 : \mathbb{Z}, d : \mathbb{R}, noise : Model$ | Factor | - |
| noiseModel_Isotropic_Sigma | $dim : \mathbb{Z}, \sigma : \mathbb{R}$ | Model | - |
| LevenbergMarquardtOptimizer | $graph : Graph, values : Values$ | Values | - |
| Values | - | Values | - |
| insert | $values : Values, key : \mathbb{Z}, value :$ Pose2 or Point2 | - | - |
| atPose2 | $result : Values, key : \mathbb{Z}$ | $\mathbb{R}^3$ | - |
| compose | $T_{mf} : \mathbb{R}^3, T_{rf} : \mathbb{R}^3$ | $\mathbb{R}^3$ | - |
| inverse | $T_{rf} : \mathbb{R}^3$ | $\mathbb{R}^3$ | - |

## 7.4 Semantics

### 7.4.1 State Variables

None

### 7.4.2 Environment Variables

None

### 7.4.3 Assumptions

None

### 7.4.4 Access Routine Semantics

Pose2($x, y, \theta$):

- output: $out := [x, y, \theta]$ (A 2D pose with orientation)
- exception: None

Point2($x, y$):

- output: $out := [x, y]$ (2D position)
- exception: None

symbol($char, int$):

- output: $out := x1(pose), a1, a2, a3(beacons)$
- exception: None

NonlinearFactorGraph():

- output: $out :=$ An empty factor graph
- exception: None

PriorFactorPose2($key, pose, noise\_model$):

- output: $out :=$ Factor (A prior factor on a 2D pose)
- exception: None

PriorFactorPoint2($key, point, noise\_model$):

- output: $out :=$ Factor (A prior factor on a 2D point)
- exception: None

RangeFactor2D($key_1, key_2, measured, noise\_model$):

- output: $out :=$ Factor (A range factor between two keys)
- exception: None

noiseModel_Isotropic_Sigma($dim, \sigma$):

- output: $out$ := Model (An isotropic noise model)
- exception: None

LevenbergMarquardtOptimizer($graph, values$):

- output: $out$ := Values (Optimized results from factor graph)
- exception: None

Values():

- output: $out$ := Values (An empty values container)
- exception: None

insert($Values, key, value$):

- transition: Adds point/pose into a Values variable according to its id (key)
- exception: None

atPose2($result, key$):

- output: $out := \hat{\mathbf{x}}$
- exception: None

compose($T_{mf}, T_{rf}$):

- output: $out := T_{mr}$ (The composition of two poses)
- exception: None

inverse($T_{rf}$):

- output: $out := T_{fr}$
- exception: None

# 8 MIS of Input Format Module

## 8.1 Module

input_format

## 8.2 Uses

- GTSAM Module (Section 7)

## 8.3 Syntax

### 8.3.1 Exported Constants

None

### 8.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| load_input | self | - | FileNotFoundError, ValueError |
| get_beacons | self | $\mathbb{R}^{N\times 2}$ | - |
| get_fmMap | self | $\mathbb{R}^3$ | - |
| get_fmRobots | self | $\mathbb{R}^3$ | - |
| get_map | self | String | - |
| get_ranges | self | $\mathbb{R}^N$ | - |
| get_variances | self | $\mathbb{R}^N$ | - |

## 8.4 Semantics

### 8.4.1 State Variables

- input_file: A string representing the path to the user input file (user_input.yaml).
- data: A dictionary storing parsed YAML input data.

### 8.4.2 Environment Variables

None

### 8.4.3 Assumptions

- The module will call on a pre-existing YAML file

#### 8.4.4 Access Routine Semantics

load_input():

- transition: Reads the YAML input file and stores it in 'self.data'.
- exception: FileNotFoundError if the input file is not detected and ValueError if the YAML file is formatted incorrectly

input.get_beacons():

- output: $out := \mathbf{a}$
- exception: None

get_fmMap():

- output: $out := T_{mf} = Pose2(\mathbb{R}^3)$
- exception: None

get_fmRobot():

- output: $out := T_{rf} = Pose2(\mathbb{R}^3)$
- exception: None

get_map():

- output: out:= String of picture's name
- exception: None

get_ranges():

- output: $out := \tilde{\mathbf{D}}$
- exception: None

get_variances():

- output: $out := \boldsymbol{\sigma^2}$
- exception: None

# 9 MIS of Localization Module

## 9.1 Module

localization

## 9.2 Uses

- GTSAM Module (Section 7)
- Input Format Module (Section 8)

## 9.3 Syntax

### 9.3.1 Exported Constants

None

### 9.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| localize | $\mathbf{a} : \mathbb{R}^{N\times 2}, \mathbf{T}_{mf} : \mathbb{R}^3, \mathbf{T}_{rf} : \mathbb{R}^3, \tilde{\mathbf{d}} : \mathbb{R}^N$ | $\mathbb{R}^3$ | - |

## 9.4 Semantics

### 9.4.1 State Variables

None

### 9.4.2 Environment Variables

None

### 9.4.3 Assumptions

- GTSAM is installed

### 9.4.4 Access Routine Semantics

localize($\mathbf{a}$, $\mathbf{T}_{mf}$, $\mathbf{T}_{rf}$, $\tilde{\mathbf{d}}$):

- output: $out := \hat{\mathbf{x}}$
- exception: None

# 10 MIS of Accuracy Evaluation Module

## 10.1 Module

accuracy

## 10.2 Uses

- Localization Module (Section 9)

## 10.3 Syntax

### 10.3.1 Exported Constants

None

### 10.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| compute_fim | $\hat{\mathbf{x}} : \mathbb{R}^3, \mathbf{a} : \mathbb{R}^{N\times 2}, \boldsymbol{\sigma^2} : \mathbb{R}^N$ | $\mathbb{R}^{2\times 2}$ | - |
| compute_crlb | $\boldsymbol{\mathcal{I}}(\hat{\mathbf{x}}) : \mathbb{R}^{2\times 2}$ | $\mathbb{R}^{2\times 2}$ | - |

## 10.4 Semantics

### 10.4.1 State Variables

None

### 10.4.2 Environment Variables

None

### 10.4.3 Assumptions

- Noise variances are positive

### 10.4.4 Access Routine Semantics

compute_fim($\hat{\mathbf{x}}, \mathbf{a}, \boldsymbol{\sigma^2}$):

- output: *out* := $\boldsymbol{\mathcal{I}}(\hat{\mathbf{x}})$ where $\boldsymbol{\mathcal{I}}(\hat{\mathbf{x}})$ is a $2 \times 2$ Fisher Information Matrix (FIM) of the estimated pose, computed as:

$$\mathcal{I}(\hat{\mathbf{x}}) = \sum_{j=1}^{N} \frac{1}{\sigma_j^2} \frac{(\hat{\mathbf{x}} - \mathbf{a}_j)(\hat{\mathbf{x}} - \mathbf{a}_j)^T}{\|\hat{\mathbf{x}} - \mathbf{a}_j\|^2}$$

- exception: None

compute_crlb($\boldsymbol{\mathcal{I}}(\hat{\mathbf{x}})$):

- output: *out* := A $2 \times 2$ CRLB matrix, computed as:

$$\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{I}}^{-1}$$

- exception: None

# 11 MIS of Output Module

## 11.1 Module

output

## 11.2 Uses

- Localization Module (Section 9)

## 11.3 Syntax

### 11.3.1 Exported Constants

None

### 11.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| update_table | - | - | - |
| run_gui | queue: Queue | - | - |

## 11.4 Semantics

### 11.4.1 State Variables

None

### 11.4.2 Environment Variables

- date_queue: A queue storing tuples of estimated pose data (time, x, y, theta).

### 11.4.3 Assumptions

- The function ‘run_gui()‘ is executed in a separate process to prevent a stalled execution.

### 11.4.4 Access Routine Semantics

update_table():

- transition: Retrieves the latest pose estimates from the queue and updates the Graphical User Interface (GUI) table.

run_gui(queue):

- transition: Initializes and runs the Tkinter GUI while continuously checking for pose updates.

# 12 MIS of Plotting Module

## 12.1 Module

plot

## 12.2 Uses

- Localization Module (Section 9)

## 12.3 Syntax

### 12.3.1 Exported Constants

None

### 12.3.2 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| plot_localization_live | $\mathbf{a} : R^{N\times 2}, \mathbf{T}_{mf} : R^3$, map: String | - | - |
| update_trajectory | $\hat{\mathbf{x}} : R^3$ | - | - |

## 12.4 Semantics

### 12.4.1 State Variables

- trajectory: A list storing estimated positions over time as $(x, y, \theta)$.

### 12.4.2 Environment Variables

None

### 12.4.3 Assumptions

- 'plot_localization_live()' is run in an interactive Matplotlib session.
- 'update_trajectory()' is only called when valid estimated poses exist.

### 12.4.4 Access Routine Semantics

plot_localization_live($\mathbf{a}, \mathbf{T}_{mf}$, map):

- transition: Initializes and continuously updates a real-time localization plot.

update_trajectory($\hat{\mathbf{x}}$):

- transition: Adds the latest estimated pose to the trajectory list for the map.

### 12.4.5 Local Functions

update(frame):

- transition: Retrieves the latest estimated pose from the trajectory and updates the visualization.

# References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.