Simultaneous Localization and Mapping in UAV

Chandra R. Murthy, Akshay Kumar, Raksha S

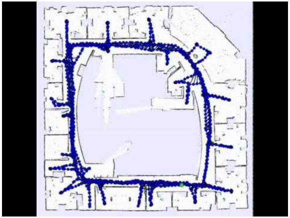
Table of Contents

Table of Contents

Introduction

What we want to do?

- Build a 2D/3D map of an environment using a UAV
- Localize the position of the UAV in the map



Introduction

What we need?

- A UAV
- Sensors such as LIDAR, camera etc
- A processing system

Why we want to do?

- Can be used in autonomous navigation and surveillance
- Search and rescue operations

Table of Contents

SLAM

- Front end processes the raw data to generate a set of spatial constraint between robot/UAV poses
- Back end finds the optimal robot/UAV poses given these spatial constraint

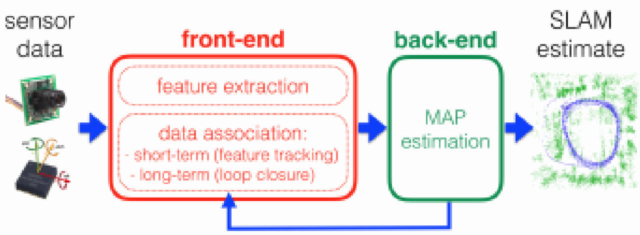


Table of Contents

Scan matching

Why Scan matching?

- Estimation of the relative roto-translation displacement between two robot poses
- Scan match between consecutive sensor messages and publish the estimated pose as geometry messages

Types of Scan matchers:

- Laser scan matcher (LSM)
- Canonical scan matcher (CSM)
- Polar scan matcher (PSM)

Scan matching with LSM

Handheld mapping system by TU Darmstadt:

Figure: Rescue arena

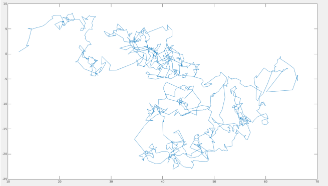


Figure: Odometry

- Data available at: `https://code.google.com/archive/p/tu-darmstadt-ros-pkg/downloads`

Loop closure

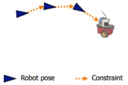
What is Loop closure?

- If the UAV visits previously seen areas, it generates a constraint between non-successive poses called loop closures

Why Loop closure?

- To understand the real topology of the environment
- To minimize the error in the map

Loop closures are detected using different approaches such as Bag of Words (BoW) and Tree of Words (ToW)



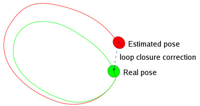


Table of Contents

Front-end Tools

- Google cartographer
- Fast Laser Interest Region Transform Library (FLIRTLib)
- Open karto
- Nav2d
- RTABMap

RTABMap

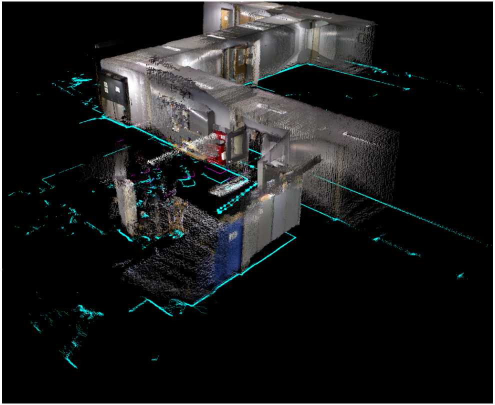


Figure: 3D map by RTABMap

RTABMap

- It stands for Real Time Appearance Based Mapping

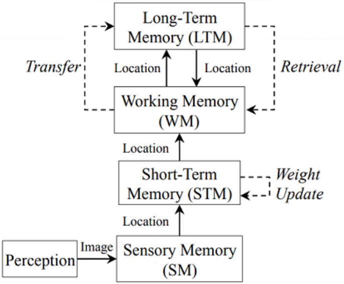


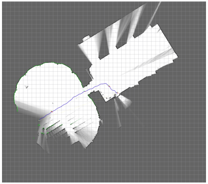
Figure: Memory management in RTABMap

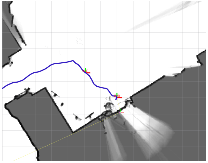
Google-Cartographer

- Provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D

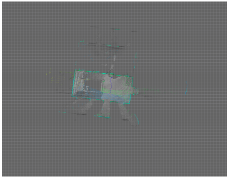
Figure: 2D SLAM by Google-Cartographer

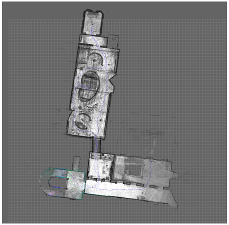
Google-Cartographer 2D SLAM

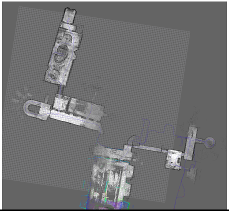




Google-Cartographer 3D SLAM







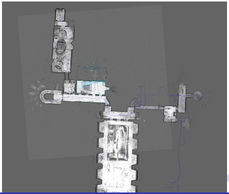


Table of Contents

Back-end

- Let $\mathbf{x_i} = [\mathbf{t_i}, \theta_\mathbf{i}]^\mathbf{T}$ be robot/UAV pose in global frame
- Let $\mathbf{z_{ij}} = [\mathbf{\Delta_{ij}}, \theta_\mathbf{ij}]^\mathbf{T}$ be spatial constraint between ith and jth pose
- Measurement model, $f(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{z_{ij}} + \mathbf{e_{ij}}$, $\mathbf{e_{ij}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega})$
- A network of spatial constraints between poses obtained from the front-end is optimized in the back-end to find global poses

Optimization and mapping

- Let $x = [x_1, x_2, ...x_n]^T$ be a vector of parameters which describes nodes
- Let $\delta_{\mathbf{ji}}$ be a constraint between nodes j and i
- Ω_{ji} represents the uncertainity in the δ_{ji}
- Given a constraint between node i and j, error is given by $e_{ji}(x) = f_{ji}(x) - \delta_{ji}$
- Assuming a Gaussian error, $F_{ji} \propto (f_{ji}(x) - \delta_{ji})^T \Omega_{ji}(f_{ji}(x) - \delta_{ji})$
- $F_{ji} = e_{ji}(x)^T \Omega_{ji} e_{ji}(x)$
- x*$= \underset{x}{\text{argmin}} F(x)$

Optimization techniques

Optimization techniques:

- General framework for Graph Optimization(G2O)
- Tree based netwORk Optimizer (TORO)
- Vertigo

G2O

Approximate the error function by its first order Taylor expansion around the initial guess $\hat{x}$

$$\mathbf{e_{ij}(\hat{x} + \Delta x_i, \hat{x} + \Delta x_j) = e_{ij}(\hat{x} + \Delta x)}$$

$$\mathbf{e_{ij}(\hat{x} + \Delta x_i, \hat{x} + \Delta x_j) \simeq e_{ij} + J_{ij}\Delta x}$$

$$\mathbf{F_{ij}(\hat{x} + \Delta x) = e_{ij}(x + \Delta x)^T \Omega_{ij} e_{ij}(\hat{x} + \Delta x)}$$

$$\mathbf{F_{ij}(\hat{x} + \Delta x) \simeq (e_{ij} + J_{ij}\Delta x)^T \Omega_{ij} (e_{ij} + J_{ij}\Delta x)}$$

$$\mathbf{F_{ij}(\hat{x} + \Delta x) \simeq e_{ij}^T \Omega_{ij} e_{ij} + 2e_{ij}^T \Omega_{ij} J_{ij} \Delta x + \Delta x^T \Omega_{ij} J_{ij} \Delta x}$$

$$\mathbf{F_{ij}(\hat{x} + \Delta x) \simeq c_{ij} + 2b_{ij}\Delta x + \Delta x^T H_{ij} \Delta x}$$

$$\mathbf{H\Delta x = -b}$$

$$\mathbf{x^* = \hat{x} + \Delta x^*}$$

G2O optimizer results

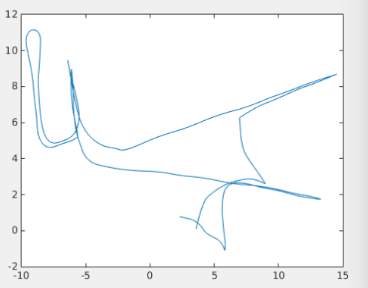


Figure: Non optimized graph

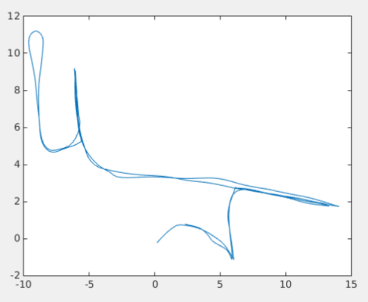


Figure: Optimized graph

TORO optimizer results

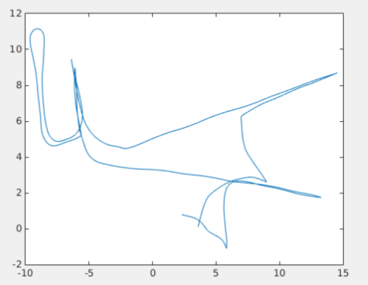


Figure: Non optimized graph

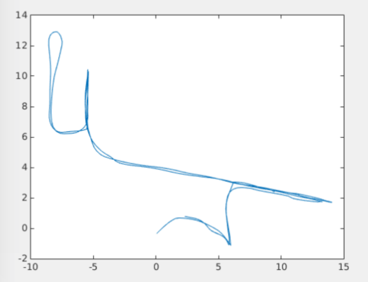


Figure: Optimized graph

Vertigo optimizer results

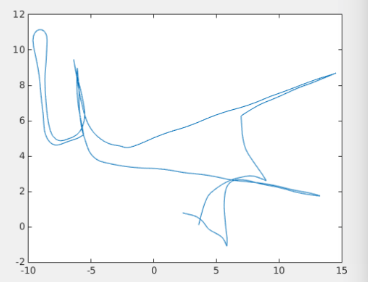


Figure: Non optimized graph

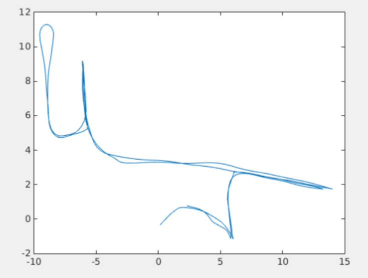


Figure: Optimized graph

Table of Contents

Real-time Image optimization results (G2O)

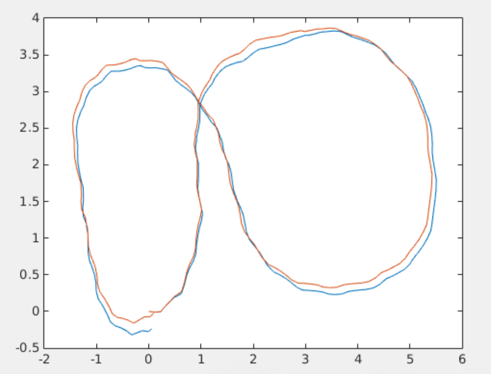


Table of Contents

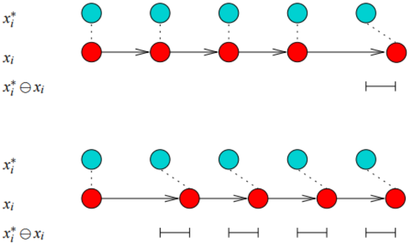
Error computation

Absolute pose error

- $E(x_{1:T}) = \sum_{t=1}^{T}[(x_t - x_t^*)]^2$ [1]

Relative pose error

- $E(x_{1:T}) = \sum_{t=1}^{T}[(x_{t+1} - x_t) - (x_{t+1}^* - x_t^*)]^2$



[1] "On Measuring the Accuracy of SLAM Algorithms" Rainer Kummerle

Table of Contents

Progress and future work

Progress:

- Front-end: Scan matching and loop closure detection
- Back-end: Optimization and mapping

Future work:

- Stochastic optimization for SLAM
- Compute relative pose error for different optimization techniques

Thank you