Point Cloud Registration with Applications in Distributed SLAM

João Bernardino
joao.m.s.bernardino@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

January 2021

Abstract

This extended abstract presents a point cloud registration algorithm and a simultaneous localization and mapping formulation developed to collectively map a GPS denied environment. Firstly, the SLAM problem is formulated in the vehicle body-fixed frame, with a filtering process devised to guarantee the convergence of the state estimates, including the positions of the landmarks, vehicle velocity and gyroscope bias. Since the landmark estimates are determined in the body-fixed frame, a Procrustes problem is solved at every iteration to build the inertial map. The point cloud registration problem is then addressed, with an innovative algorithm being developed. A theoretical approach is used to ensure the correct calibration and robustness of the algorithm in stochastic settings. Several branches of the registration algorithm are developed, culminating in two and three-dimensional versions of a SLAM tailored algorithm. The developed versions are tested individually, highlighting precision and runtime. A two-dimensional distributed SLAM mission simulation is performed in a Simulink environment, using a fictitious arena as the region to be mapped by two drones. Finally, and to assure the performance of the developed solutions in a real-world mission, a three-dimensional mapping mission is simulated using a real scan of a classroom.
Keywords: Distributed SLAM, Point Cloud Registration, Kalman Filter, Autonomous Vehicles, Range and Bearing Camera

1. Introduction

In recent years, the market for Unmanned Aerial Vehicles (UAVs) has experienced an unprecedented growth. Still somewhat perceived as toys, mainly due to their handling and size, these same features make them an ideal and cost-effective solution for a wide range of applications. From automated deliveries and infrastructure inspections, to weather forecasting and disaster humanitarian response, the potential for these vehicles to improve everyday life is still being discovered.

Although some of these implementations require just the use of a skilled drone operator, capable of getting close to what is being observed, applications such as cliff mapping to predict erosion involve the creation of precise georeferecend point clouds, to be compared year after year. Similarly, mining and agricultural implementations rely on these same point clouds to calculate the volume of extracted ore and plant growth, respectively. Accordingly, it is in the interest of these industries to streamline and automate as much of the mapping process as possible.

1.1. Related work

To create these clouds, naive approaches where the captured images are fused based on gyroscope and GPS data can be used. Nonetheless, these inertial sensors are subject to noise, with the latter relying on a good coverage and lack of obstacles. To perform any kind of mapping in GPS denied environments, more robust approaches, such as simultaneous localization and mapping (SLAM) are needed.

SLAM is the process of answering the question "Where am I and what surrounds me?" by constructing the map of a previously unknown environment and locating the vehicle therein. In many approaches, such as the ones presented in (Newman and Ho, 2005) and (Durrant-Whyte and Bailey, 2006), the SLAM problem is analysed in an inertial frame, with the vehicle position and orientation comprising the state estimates. Conversely, in (Lourenço et al., 2016), the SLAM problem is formulated in body-fixed frame, where the unobservable vehicle pose is removed from the state, and replaced by the observed relative positions of the landmarks. As stated in (Lourenço, 2019), this exchange aids state convergence, while maintaining enough information to determine the position and orientation of the vehicle. In the developed work, an analogous body-fixed frame formulation was derived and implemented, with a Procrustes problem solved at every iteration to construct the inertial map.

Notwithstanding, when mapping extensive regions, a single vehicle may have trouble going through the area of interest in a reasonable amount of time. A simple solution, the deployment of several robots, although natural, entails a new difficulty, the fusion of the data. Point cloud registration is the process of finding the best fitting rigid motion between two given point clouds. This process becomes increasingly harder if the point clouds only

partially overlap. A widespread solution, first developed in (Besl and McKay, 1992), is the iterative closest point (ICP). In this algorithm, the point clouds are assumed to have a rough initial alignment, and an iterative process is used to reduce the error between the two. Many flavours of the initial algorithm have been developed, all maintaining the disadvantage of not dealing with arbitrary different initial alignments. Another widespread technique, developed in (Fischler and Bolles, 1981), is the random sample consensus (RANSAC), whose main idea is to randomly select points from one cloud until an acceptable rigid motion, measured by the number of inliers is determined. Although robust enough to handle partially overlapping clouds, the randomized nature of the selection process means real time applications are difficult to achieve.

The main contribution of this work was the development of an algorithm to rapidly register partially overlapping point clouds. In this abstract, emphasis is given to the elaborated SLAM-tailored version, to be used in a distributed SLAM mission, upon an encounter of two vehicles.

1.2. Extended abstract organization and notation

This extended abstract is organized as follows: in Section 2, the distributed SLAM problem is formalized; the individual SLAM problem is formulated in Section 3, with the point cloud registration algorithms being developed in Sections 4 and 5; Sections 6 and 7 present simulations of a distributed SLAM mission in 2D and 3D, respectively; lastly, concluding remarks are given in Section 8.

The following symbol convention is used in this extended abstract: vectors are denoted in bold small letters and matrices in bold capital letters. The number of vehicles in a given mission is n_V. Superscripts are used to indicate the frame of reference a quantity is expressed in, with superscript ^{E} indicating an Earth-fixed inertial frame $\{E\}$, and superscripts ^{i}, indicating a vehicle body-fixed frame $\{B_i\}$, with $i = 1, 2, \ldots, n_V$. Accent $\check{}$ is used to indicate a measured quantity, subject to noise, and accent $\hat{}$ is used to represent estimates of true quantities. $\mathbf{I}_n$ is the identity matrix of size n and $\mathbf{0}_{n\times m}$ is a n by m matrix filled with zeros. If m is omitted, the matrix is square. Lastly, $\mathbf{S}[\mathbf{a}]$ is a skew symmetric matrix such that $\mathbf{S}[\mathbf{a}]\mathbf{b} = \mathbf{a} \times \mathbf{b}$, with $\times$ denoting the cross product.

2. Problem formulation

In this section, the distributed SLAM mission is theorized. The SLAM problem is first formulated with the point cloud registration problem being then formalized.

2.1. SLAM system

Consider a vehicle operating in a d dimensional environment, with $d \in \{2, 3\}$. Let the rotation matrix ${}^E\mathbf{R}_i(t) \in \mathcal{SO}(d)$ represent the rotation from $\{B_i\}$ to $\{E\}$, satisfying ${}^E\dot{\mathbf{R}}_i(t) = {}^E\mathbf{R}_i(t)\mathbf{S}[{}^i\boldsymbol{\omega}(t)]$, where ${}^i\boldsymbol{\omega}(t)$ is the angular velocity of the vehicle expressed in $\{B_i\}$. Furthermore, let ${}^E\mathbf{p}_{\mathrm{v}_i}$ be the vehicle position in the inertial frame.

The vehicle is equipped with a range and bearing camera, capturing, in any given moment, a noisy point set, ${}^i\check{\mathbf{X}}$, of the vehicle surroundings, ${}^i\mathbf{X} = \{{}^i\mathbf{x}_1, {}^i\mathbf{x}_2, \ldots, {}^i\mathbf{x}_n\}$, where ${}^i\mathbf{x}_j \in \mathbb{R}^d$. In this set, there are some static points with distinctive features, ${}^i\mathbf{p}_j \in \mathbb{R}^d$, hereby called landmarks. Let ${}^E\mathbf{p}_j(t)$ denote the inertial position of a landmark. Since the landmarks, are static, ${}^E\dot{\mathbf{p}}_j(t) = \mathbf{0}$. It follows that

$$ {}^i\mathbf{p}_j(t) = {}^i\mathbf{R}_E(t)\big({}^E\mathbf{p}_j(t) - {}^E\mathbf{p}_{\mathrm{v}_i}(t)\big), \tag{1}$$

and

$$ {}^i\dot{\mathbf{p}}_j(t) = -\mathbf{S}[{}^i\boldsymbol{\omega}(t)]\ {}^i\mathbf{p}_j(t) - {}^i\mathbf{v}(t), \tag{2}$$

where ${}^i\mathbf{v}(t)$ is the velocity of the vehicle, expressed in its own frame, assumed constant.

Each vehicle is also rigged with d_ω rate gyros, with $d_\omega = 1$ or $d_\omega = 3$ for the case where $d = 2$ or $d = 3$, respectively. These provide measurements of the angular velocity, ${}^i\check{\boldsymbol{\omega}}(t)$, affected by a constant bias, ${}^i\mathbf{b}_\omega(t)$. Hence,

$$ {}^i\check{\boldsymbol{\omega}}(t) = {}^i\boldsymbol{\omega}(t) + {}^i\mathbf{b}_\omega(t). \tag{3}$$

Taking this into account, and using the anti commutative property of the cross product, $\mathbf{S}[\mathbf{a}]\mathbf{b} = -\mathbf{S}[\mathbf{b}]\mathbf{a}$, it is possible to rewrite (2) as

$$ {}^i\dot{\mathbf{p}}_j(t) = -\mathbf{S}[{}^i\check{\boldsymbol{\omega}}(t)]\ {}^i\mathbf{p}_j(t) - \mathbf{S}[{}^i\mathbf{p}_j(t)]\ {}^i\mathbf{b}_\omega(t) - {}^i\mathbf{v}(t). \tag{4}$$

It is now assumed that m landmarks are present. Let the state variables be $\mathbf{z}(t) = \begin{bmatrix} {}^i\mathbf{p}_1^{\mathrm{T}} & \ldots & {}^i\mathbf{p}_j^{\mathrm{T}} & \ldots & {}^i\mathbf{p}_m^{\mathrm{T}} & {}^i\mathbf{v}^{\mathrm{T}}(t) & {}^i\mathbf{b}_\omega{}^{\mathrm{T}}(t) \end{bmatrix}^{\mathrm{T}}$. The system dynamics can be written as

$$ \begin{cases} \dot{\mathbf{z}}(t) = \mathbf{A}(t)\mathbf{z}(t) \\ \mathbf{y}(t) = \mathbf{C}(t)\mathbf{z}(t) \end{cases}, \tag{5}$$

where

$$ \mathbf{A}(t) = \begin{bmatrix} \mathbf{A}_\omega & \mathbf{A}_I & \mathbf{A}_p \\ \mathbf{0}_{2d\times md} & \mathbf{0}_{2d\times d} & \mathbf{0}_{2d\times d_\omega} \end{bmatrix}, \tag{6}$$

$$ \mathbf{A}_\omega = \mathrm{diag}(-\mathbf{S}[{}^i\check{\boldsymbol{\omega}}(t)],\ \ldots,\ -\mathbf{S}[{}^i\check{\boldsymbol{\omega}}(t)]), $$

$$ \mathbf{A}_I = \begin{bmatrix} \mathbf{I}_d,\ \ldots,\ \mathbf{I}_d \end{bmatrix}^{\mathrm{T}}, $$

and

$$ \mathbf{A}_p = \begin{bmatrix} -\mathbf{S}[{}^i\mathbf{p}_1(t)],\ \ldots,\ -\mathbf{S}[{}^i\mathbf{p}_m(t)] \end{bmatrix}^{\mathrm{T}}. $$

To construct the output matrix, $\mathbf{C}(t)$, a distinction must be made, as all the landmarks in the system may not be available at any given instant. From the m landmarks, let the first v be the visible ones. Considering that there are no measurements for the velocity and bias, it then follows that

$$ \mathbf{C}(t) = \begin{bmatrix} \mathbf{I}_{dv\times d} & \mathbf{0}_{dv\times(m-v)d} & \mathbf{0}_{dv\times d} & \mathbf{0}_{dv\times d_\omega} \end{bmatrix}. \tag{7}$$

The observability of the system above was studied and proved in detail in (Lourenço et al., 2016).

2.2. Point cloud registration

In the scope of a distributed SLAM mission, it is expected that once two vehicles meet, they exchange and fuse their maps, in order to increase the mapped region. Let ${}^{i}\mathbf{X}$ be of size n, and ${}^{j}\mathbf{X}$ be of size m. It is assumed, without loss of generality, that $n \geq m$. Since the vehicles recognize one another, it is guaranteed that there is a subset of both clouds, ${}^{i}\mathbf{X}_l = \{{}^{i}\mathbf{x}_1,\ {}^{i}\mathbf{x}_2, \ldots,\ {}^{i}\mathbf{x}_l\}$ and ${}^{j}\mathbf{X}_l = \{{}^{j}\mathbf{x}_1,\ {}^{j}\mathbf{x}_2, \ldots,\ {}^{j}\mathbf{x}_l\}$, that satisfies

$$ {}^{j}\mathbf{x}_g = {}^{j}\mathbf{R}_i\ {}^{i}\mathbf{x}_g + \underbrace{{}^{j}\mathbf{R}_E({}^{E}\mathbf{p}_{\mathrm{v}_i} - {}^{E}\mathbf{p}_{\mathrm{v}_j})}_{{}^{j}\mathbf{p}_{\mathrm{v}_i}} \tag{8} $$

where ${}^{j}\mathbf{R}_i$ is the rotation from $\{B_i\}$ to $\{B_j\}$, and ${}^{j}\mathbf{p}_{\mathrm{v}_i}$ is the position of vehicle i expressed in frame $\{B_j\}$. Recalling that the noisy sets are given by ${}^{i}\check{\mathbf{X}}$ and ${}^{j}\check{\mathbf{X}}$, the second problem addressed in this work is the discovery of the overlapping subsets, ${}^{i}\check{\mathbf{X}}_l$ and ${}^{j}\check{\mathbf{X}}_l$, and the calculation of the optimal rigid transformation, ${}^{j}\mathbf{R}_i^*$ and ${}^{j}\mathbf{p}_{\mathrm{v}_i}^*$, that minimizes the mean squared error, $e_{\mathbf{x}}$, given by

$$ e_{\mathbf{x}}({}^{j}\mathbf{R}_i^*,\ {}^{j}\mathbf{p}_{\mathrm{v}_i}^*) = \frac{1}{l}\sum_{g=1}^{l} \|{}^{j}\check{\mathbf{x}}_g - {}^{j}\mathbf{R}_i^*\ {}^{i}\check{\mathbf{x}}_g - {}^{j}\mathbf{p}_{\mathrm{v}_i}^*\|^2. \tag{9} $$

3. SLAM filter

In this section, the discrete dynamics are determined, and a Kalman filter is designed to ensure the convergence of the state estimates.

3.1. Discrete dynamics

Given the chosen time step T, the discrete time steps are given by $t_k = kT + t_0$. The discrete dynamics for the system defined in (4) can be written under the form

$$ \begin{cases} \mathbf{z}(k+1) = \mathbf{F}(k)\mathbf{z}(k) \\ \mathbf{y}(k) = \mathbf{H}(k)\mathbf{z}(k) \end{cases}, \tag{10} $$

where

$$ \mathbf{F}(k) = e^{\mathbf{A}(t_k)T}, \tag{11} $$

and

$$ \mathbf{H}(k) = \mathbf{C}(t_k). \tag{12} $$

For the two-dimensional case, $d_\omega = 1$, the measured angular velocity and gyroscope bias are reduced to scalar values, ${}^{i}\check{\omega}$ and ${}^{i}b_\omega$, inducing simplifications in the skew symmetric matrices, $\mathbf{S}[{}^{i}\check{\omega}]$ and $\mathbf{S}[{}^{i}\mathbf{p}_j(t)]$. Taking these into account, the matrix exponential given by (11) can be analytically calculated, resulting in

$$ \mathbf{F}(k) = \begin{bmatrix} \mathbf{F}_{\mathrm{diag}} & \mathbf{F}_I & \mathbf{F}_p \\ \mathbf{0}_{2\times 2m} & \mathbf{I}_2 & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2m} & \mathbf{0}_{1\times 2} & \mathbf{I}_1 \end{bmatrix}, \tag{13} $$

where

$$ \mathbf{F}_{\mathrm{diag}} = \mathrm{diag}(\mathbf{F}_\omega,\ \ldots,\ \mathbf{F}_\omega), \tag{14} $$

$$ \mathbf{F}_\omega(k) = \begin{bmatrix} \cos({}^{i}\check{\omega}(t_k)T) & \sin({}^{i}\check{\omega}(t_k)T) \\ -\sin({}^{i}\check{\omega}(t_k)T) & \cos({}^{i}\check{\omega}(t_k)T) \end{bmatrix}, $$

$$ \mathbf{F}_I = \begin{bmatrix} \mathbf{F}_v,\ \ldots,\ \mathbf{F}_v \end{bmatrix}^{\mathrm{T}}, $$

$$ \mathbf{F}_v = \begin{bmatrix} -\frac{\sin({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} & \frac{\cos({}^{i}\check{\omega}(t_k)T)-1}{{}^{i}\check{\omega}(t_k)} \\ \frac{1-\cos({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} & -\frac{\sin({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} \end{bmatrix}, $$

$$ \mathbf{F}_p = \begin{bmatrix} \mathbf{F}_1,\ \ldots,\ \mathbf{F}_j,\ \ldots,\ \mathbf{F}_m \end{bmatrix}^{\mathrm{T}}, $$

and

$$ \mathbf{F}_j = \begin{bmatrix} {}^{i}p_{j_1}(t_k)\frac{1-\cos({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} - {}^{i}p_{j_2}(t_k)\frac{\sin({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} \\ {}^{i}p_{j_2}(t_k)\frac{1-\cos({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} + {}^{i}p_{j_1}(t_k)\frac{\sin({}^{i}\check{\omega}(t_k)T)}{{}^{i}\check{\omega}(t_k)} \end{bmatrix}. $$

For the three-dimensional case, the Euler discretization was used to solve (11), as no analytical solution exists. Thus,

$$ \mathbf{F}(k) = \mathbf{I} + \mathbf{A}(t_k)T. \tag{15} $$

3.2. Kalman Filter

The system described in (10) assumes exact dynamics and no sensor noise. To model unknown disturbances in the system dynamics, a process disturbance, $\xi(k)$, is introduced. Likewise, to model noise in the measurements, a sensor noise, $\theta(k)$, is added. With this, (10) is rewritten as

$$ \begin{cases} \mathbf{z}(k+1) = \mathbf{F}(k)\mathbf{z}(k) + \xi(k) \\ \mathbf{y}(k) = \mathbf{H}(k)\mathbf{z}(k) + \theta(k) \end{cases}, \tag{16} $$

where

$$ \xi(k) \sim \mathcal{N}(0, \mathbf{Q}), \tag{17} $$

and

$$ \theta(k) \sim \mathcal{N}(0, \mathbf{R}), \tag{18} $$

with $\mathbf{Q}$ being a positive semi-definite and $\mathbf{R}$ a positive definite covariance matrix. The standard prediction and update steps, as derived in (Gelb, 1974) are applied so as to ensure the convergence of the estimates. A difficulty arises in the update step, as the new observations must be matched with the old estimates. Since the *a priori* estimates are expected to be close to the observed landmarks, the ICP algorithm was used to register one set of landmarks into the other, and solve the subsequent matching process.

Although a standard Kalman filter was used, it is worth remarking that, since the estimates of the landmarks are also present in the state matrix, this system falls outside the scope of the standard LTV systems, where $\mathbf{F}(k)$ is exact. Hence, this filtering process must be exercised with caution, as this non-linearity hinders optimality guarantees, as stated in (Lourenço et al., 2016).

3.3. Inertial map

Every iteration, each vehicle i captures, in its own frame, a point cloud of its surroundings. To relate clouds taken in successive steps and build the map of the environment in the inertial frame, it is necessary to determine how much the vehicle has moved between time instant t_k and t_{k+1}. An algorithm that effectively handles this problem was proposed in (Lourenço et al., 2016).

The landmark position estimates in $\mathbf{z}(k+1)$ can be represented in the inertial frame using

$$ {}^{E}\hat{\mathbf{p}}_j(k+1) = {}^{E}\hat{\mathbf{p}}_{\mathrm{v}_i}(k+1) + {}^{E}\hat{\mathbf{R}}_j(k+1)\, {}^{i}\hat{\mathbf{p}}_j(k+1). \quad (19) $$

In the previous Section, an approach to relate landmark estimates at t_k with the new observations at t_{k+1} was proposed. As such, one can write the difference, $\mathbf{e}_j$, between the new estimate ${}^{E}\hat{\mathbf{p}}_j(k+1)$, and the old estimate ${}^{E}\hat{\mathbf{p}}_j(k)$,

$$ \mathbf{e}_j = {}^{E}\hat{\mathbf{p}}_j(k) - {}^{E}\hat{\mathbf{p}}_{\mathrm{v}_i}(k+1) - {}^{E}\hat{\mathbf{R}}_j(k+1)\, {}^{i}\hat{\mathbf{p}}_j(k+1). \quad (20) $$

Since the landmarks are static, theoretically, ${}^{E}\hat{\mathbf{p}}_j(k) = {}^{E}\hat{\mathbf{p}}_j(k+1)$. Let $e_{\mathbf{p}}$ be the mean squared error of all m landmarks, defined as a function of the rotation estimate, ${}^{E}\hat{\mathbf{R}}_i(k+1)$, and vehicle position estimate, ${}^{E}\hat{\mathbf{p}}_{\mathrm{v}_i}(k+1)$,

$$ e_{\mathbf{p}}({}^{E}\hat{\mathbf{R}}_i(k+1), {}^{E}\hat{\mathbf{p}}_{\mathrm{v}_i}(k+1)) = \frac{1}{m}\sum_{j=1}^{m} \|\mathbf{e}_j\|^2. \quad (21) $$

The minimization of $e_{\mathbf{p}}$ is a problem with a closed form solution derived in (Umeyama, 1991). The optimal rotation matrix estimate, ${}^{E}\hat{\mathbf{R}}^{*}_i(k+1)$, is given by

$$ {}^{E}\hat{\mathbf{R}}^{*}_i(k+1) = \mathbf{U}\ \mathrm{diag}(1, |\mathbf{U}||\mathbf{V}|)\ \mathbf{V}, \quad (22) $$

where

$$ \mathbf{UDV} = \mathrm{svd}(\mathbf{B}^{\mathrm{T}}(k+1)), $$

$$ \mathbf{B}^{\mathrm{T}}(k+1) = (\mathbf{Y}(k) - \boldsymbol{\mu}_Y(k))\ (\mathbf{X}(k+1) - \boldsymbol{\mu}_X(k+1))^{\mathrm{T}}, $$

$$ \mathbf{Y}(k) = \left[{}^{E}\hat{\mathbf{p}}_1(k) \quad {}^{E}\hat{\mathbf{p}}_2(k) \ldots {}^{E}\hat{\mathbf{p}}_m(k)\right], $$

$$ \mathbf{X}(k+1) = \left[{}^{i}\hat{\mathbf{p}}_1(k+1) \quad {}^{i}\hat{\mathbf{p}}_2(k+1) \ldots {}^{i}\hat{\mathbf{p}}_m(k+1)\right], $$

$$ \boldsymbol{\mu}_Y(k) = \frac{1}{m}\sum_{j=1}^{m} {}^{E}\hat{\mathbf{p}}_j(k), $$

and

$$ \boldsymbol{\mu}_X(k+1) = \frac{1}{m}\sum_{j=1}^{m} {}^{i}\hat{\mathbf{p}}_j(k+1). $$

The optimal vehicle position estimate, ${}^{E}\hat{\mathbf{p}}^{*}_{\mathrm{v}_i}(k+1)$, then follows

$$ {}^{E}\hat{\mathbf{p}}^{*}_j(k+1) = \boldsymbol{\mu}_Y(k) - {}^{E}\hat{\mathbf{R}}^{*}_i(k+1)\boldsymbol{\mu}_X(k+1). \quad (23) $$

4. Two-dimensional registration algorithm

In this section, a SLAM-tailored registration algorithm is developed.

4.1. Noiseless setting

Let there be a vector formed by two points of subset ${}^{i}\mathbf{X}_l$, ${}^{i}\mathbf{x}_g$ and ${}^{i}\mathbf{x}_h$,

$$ {}^{i}\mathbf{x}_{gh} = {}^{i}\mathbf{x}_h - {}^{i}\mathbf{x}_g. \quad (24) $$

Similarly, let ${}^{j}\mathbf{x}_{gh}$ be the vector formed by the corresponding points of subset ${}^{j}\mathbf{X}_l$, ${}^{j}\mathbf{x}_g$ and ${}^{j}\mathbf{x}_h$,

$$ {}^{j}\mathbf{x}_{gh} = {}^{j}\mathbf{x}_h - {}^{j}\mathbf{x}_g. \quad (25) $$

Let ${}^{i}x_{gh}$, ${}^{j}x_{gh}$ be the modules of ${}^{i}\mathbf{x}_{gh}$, ${}^{j}\mathbf{x}_{gh}$ and ${}^{i}\beta_{\mathbf{x}_{gh}}$, ${}^{j}\beta_{\mathbf{x}_{gh}}$ be the angles between ${}^{i}\mathbf{x}_{gh}$, ${}^{j}\mathbf{x}_{gh}$ and the respective x axis. For a 2D case, it follows that

$$ {}^{j}x_{gh} = {}^{i}x_{gh}, \quad (26) $$

$$ {}^{j}\beta_{\mathbf{x}_{gh}} = {}^{i}\beta_{\mathbf{x}_{gh}} + {}^{j}\alpha_i, \quad (27) $$

where ${}^{j}\alpha_i$ is the rotation angle from $\{B_i\}$ to $\{B_j\}$. The complete rigid motion, defined in (8), can then be determined by

$$ {}^{j}\alpha_i = {}^{j}\beta_{\mathbf{x}_{gh}} - {}^{i}\beta_{\mathbf{x}_{gh}}, \quad (28) $$

and, with the rotation matrix defined by ${}^{j}\alpha_i$,

$$ {}^{j}\mathbf{p}_{\mathrm{v}_i} = {}^{j}\mathbf{x}_g - {}^{j}\mathbf{R}_i\, {}^{i}\mathbf{x}_g. \quad (29) $$

4.2. Rigid motion determination

Consider just the overlapping sets ${}^{i}\mathbf{X}_l$ and ${}^{j}\mathbf{X}_l$. Let a point ${}^{i}\mathbf{x}_g$ from ${}^{i}\mathbf{X}_l$ be chosen. Furthermore, let ${}^{i}\mathbf{V}_{\mathbf{x}_g}$ be the set of all the vectors that can be formed starting on ${}^{i}\mathbf{x}_g$ and ending in ${}^{i}\mathbf{x}_k \in {}^{i}\mathbf{X}_l \setminus {}^{i}\mathbf{x}_g$,

$$ {}^{i}\mathbf{V}_{\mathbf{x}_g} = \{{}^{i}\mathbf{x}_{g1}, {}^{i}\mathbf{x}_{g2}, \ldots, {}^{i}\mathbf{x}_{gl}\}. \quad (30) $$

Likewise, let the corresponding point ${}^{j}\mathbf{x}_g$ from ${}^{j}\mathbf{X}_l$ be selected. All the vectors starting on ${}^{j}\mathbf{x}_g$ and ending in ${}^{j}\mathbf{x}_k \in {}^{j}\mathbf{X}_l \setminus {}^{j}\mathbf{x}_g$ constitute the set ${}^{j}\mathbf{V}_{\mathbf{x}_g}$,

$$ {}^{j}\mathbf{V}_{\mathbf{x}_g} = \{{}^{j}\mathbf{x}_{g1}, {}^{j}\mathbf{x}_{g2}, \ldots, {}^{j}\mathbf{x}_{gl}\}. \quad (31) $$

Since the chosen anchors match, for every vector in ${}^{i}\mathbf{V}_{\mathbf{x}_g}$, there is at least one vector in ${}^{j}\mathbf{V}_{\mathbf{x}_g}$ with the same module. Hence,

$$ \forall\ {}^{i}\mathbf{x}_{gk} \in {}^{i}\mathbf{V}_{\mathbf{x}_g},\ \exists\ {}^{j}\mathbf{x}_{gk} \in {}^{j}\mathbf{V}_{\mathbf{x}_g} :\ {}^{i}x_{gk} = {}^{j}x_{gk}. \quad (32) $$

For every discovered pair of vectors with the same module, such as ${}^{i}\mathbf{x}_{gk}$ and ${}^{j}\mathbf{x}_{gk}$ the corresponding angle of rotation, given by (28), and vehicle position, given by (29) are calculated and stored as a tuple, ${}^{j}\mathbf{B}_i$,

$$ {}^{j}\mathbf{B}_i[{}^{i}\mathbf{x}_{gk}, {}^{j}\mathbf{x}_{gk}] = ({}^{j}\alpha_i, {}^{j}\mathbf{p}_{\mathrm{v}_i}). \quad (33) $$

All possible tuples are calculated from ${}^{i}\mathbf{V}_{\mathbf{x}_g}$ and ${}^{j}\mathbf{V}_{\mathbf{x}_g}$, forming set ${}^{j}\mathbf{S}_i[{}^{i}\mathbf{x}_g, {}^{j}\mathbf{x}_g]$. This set comprises all the possible point correspondences, with the correct ones encoding the equal, true rigid motion.

4.3. Stochastic Case

When the measured points are subject to noise, it is unreasonable to expect that the measured modules, rotations and vehicle positions will be exactly identical. Let each measured point be defined by a normal distribution,

$$ {}^{i}\breve{\mathbf{x}}_g \sim \mathcal{N}({}^{i}\mathbf{x}_g, \boldsymbol{\Sigma}_p), \quad (34) $$

where $\boldsymbol{\Sigma}_p = \sigma_p \mathbf{I}_2$. By the properties of normally distributed variables, it then follows that

$$ {}^{i}\breve{\mathbf{x}}_{gh} \sim \mathcal{N}({}^{i}\mathbf{x}_{gh}, 2\boldsymbol{\Sigma}_p). \quad (35) $$

Drawing the 95% confidence interval for the measured vector, one sees that the difference between two module measurements, ${}^{i}\check{x}_{gh}$, of the same vector ${}^{i}\mathbf{x}_{gh}$ is expected to be smaller than $7\sigma_p$.

The same reasoning is then applied to the measured angle between the vector and the x axis. This error can be observed in Fig. 1. Considering (35), it follows that

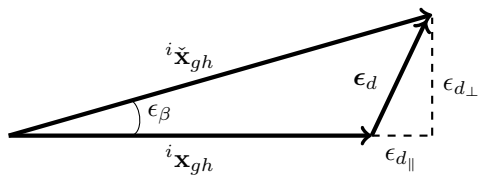

Figure 1: Angle between a measured vector and the true vector

$$\epsilon_{d_\perp} \sim \mathcal{N}(0, 2\sigma_p^2),$$

and since $\epsilon_\beta \approx \frac{\epsilon_{d_\perp}}{{}^{i}x_{gh}}$,

$${}^{i}\check{\beta}_{\mathbf{x}_{gh}} \sim \mathcal{N}\left({}^{i}\beta_{\mathbf{x}_{gh}}, \frac{2\sigma_p^2}{{}^{i}x_{gh}^2} \right). \tag{36}$$

The measured rotation angle is defined as

$${}^{j}\check{\alpha}_i = {}^{j}\check{\beta}_{\mathbf{x}_{gh}} - {}^{i}\check{\beta}_{\mathbf{x}_{gh}}. \tag{37}$$

Considering both (36) and (37),, it follows that

$${}^{j}\check{\alpha}_i \sim \mathcal{N}\left({}^{j}\alpha_i, \frac{4\sigma_p^2}{{}^{i}x_{gh}^2} \right). \tag{38}$$

Finally, an assumption is made that the distance between between points is at least 200 times the standard deviation σ_p. Hence,

$${}^{j}\check{\alpha}_i \sim \mathcal{N}({}^{j}\alpha_i, 1 \times 10^{-4}). \tag{39}$$

The maximum expected between two measurements of the same rotation angle is thus 0.04 rad.

Lastly, the measured vehicle position, ${}^{j}\check{\mathbf{p}}_{\mathrm{v}_i}$ is also affected by the point and rotation uncertainty. Analogous to (29), ${}^{j}\check{\mathbf{p}}_{\mathrm{v}_i}$ is defined as

$${}^{j}\check{\mathbf{p}}_{\mathrm{v}_i} = {}^{j}\check{\mathbf{x}}_g - {}^{j}\check{\mathbf{R}}_i \, {}^{i}\check{\mathbf{x}}_g. \tag{40}$$

Propagating the point noise and rotation angle noise defined in (34) and (38), and assuming a good spread of points in the map, *i.e.*, same order of magnitude of the points and formed vectors, one determines that the maximum expected distance between two measurements of the same vehicle position is $14\sigma_p$.

4.4. The SLAM-tailored algorithm

Now that the tolerances are defined, the algorithm designed for a distributed SLAM mission can be developed. Since the vehicles recognize each other, two common points in sets ${}^{i}\mathbf{X}$ and ${}^{j}\mathbf{X}$ are known *a priori*, that is the position of the vehicles in both reference frames. Let these points be ${}^{i}\check{\mathbf{x}}_f$ and ${}^{i}\check{\mathbf{x}}_g$ in frame $\{B_i\}$, and ${}^{j}\check{\mathbf{x}}_f$ and ${}^{j}\check{\mathbf{x}}_g$ in frame $\{B_j\}$. For a 2D case, this information alone is enough to calculate a good estimate of the rotation and translation, saved in tuple ${}^{j}\mathbf{B}_i[{}^{i}\check{\mathbf{x}}_{fg},\ {}^{j}\check{\mathbf{x}}_{fg}]$. Although an estimate is available, since the points are subject to noise, the more overlapping points are found, the more precise the calculated rigid motion will be.

Selecting one of the vehicles as the anchor, *e.g.*, ${}^{i}\check{\mathbf{x}}_g$, set ${}^{j}\mathbf{S}_i[{}^{i}\mathbf{x}_g,\ {}^{j}\mathbf{x}_g]$ is calculated. Every tuple ${}^{j}\mathbf{B}_i[\,]$ in this set is compared to the "correct" one, ${}^{j}\mathbf{B}_i[{}^{i}\check{\mathbf{x}}_{fg},\ {}^{j}\check{\mathbf{x}}_{fg}]$. If the defined rigid motions are within the tolerances defined in Section 4.3, the points forming the vectors are considered to be a match. After all the matching points are discovered, the algorithm developed in (Umeyama, 1991) is once again applied to determine the optimal rigid motion.

4.5. Time complexity

In the first step of the algorithm, vectors from set ${}^{i}\mathbf{X}$ are created and stored in a binary tree, organized by their module. Since ${}^{i}\mathbf{X}$ is composed of n points, the total number of vectors is $n-1$, and the creation of the binary tree is bounded by $\mathcal{O}(n \log n)$. Secondly, the vectors from ${}^{j}\mathbf{X}$ are generated and the binary tree is searched for matching modules, in order to create the tuples ${}^{j}\mathbf{B}_i[\,]$. Considering that ${}^{j}\mathbf{X}$ comprises m points, $m-1$ vectors are formed and the queries of the binary tree are bounded by $\mathcal{O}(m \log n)$.

As the common region, ${}^{i}\mathbf{X}_l$ and ${}^{j}\mathbf{X}_l$, is made up of l points, at least $l-1$ matching vectors are discovered. However it is unreasonable to assume that no symmetries are present and only these correct tuples are created. Given a point ${}^{i}\mathbf{x}_j \in {}^{i}\mathbf{X} \setminus {}^{i}\mathbf{x}_g$ and the formed vector ${}^{i}\mathbf{x}_{gj}$, the number of vectors ${}^{j}\mathbf{x}_{gk}$, where ${}^{j}\mathbf{x}_k \in {}^{j}\mathbf{X} \setminus \{{}^{j}\mathbf{x}_g,\ {}^{j}\mathbf{x}_j\}$, with the same module is assumed constant. As such, when the binary tree is queried looking for a given module, not only will the corresponding vector be found, if it exists, but also a constant number of false matches, C. Thus, when $m-1$ queries are made, the number of formed tuples will be $l - 1 + C(m-1)$. Since $l \leq m$, $\mathcal{O}(m)$ tuples are formed and compared to the "correct" one, ${}^{j}\mathbf{B}_i[{}^{i}\check{\mathbf{x}}_{fg},\ {}^{j}\check{\mathbf{x}}_{fg}]$. The total time complexity of the algorithm comes out as $\mathcal{O}((n+m)\log n + m)$.

4.6. Algorithm performance

To test the performance of the algorithm, a set of uniformly distributed points was generated. Part of the set was assumed to be measured by one vehicle, constituting set ${}^{i}\mathbf{X}$, and part by the other, composing set ${}^{j}\mathbf{X}$. Several overlapping percentages between both sets were experimented, ranging from 6 to 100%. Besides the overlaps, different sizes for the captured sets ${}^{i}\mathbf{X}$ and ${}^{j}\mathbf{X}$ were tried. Recalling that ${}^{i}\mathbf{X}$ is of size n and that ${}^{j}\mathbf{X}$ is of size m, the parameter $n = m$ was varied between 50 and 1000. The sets characterized by $n = 200$ and 25% overlap, expressed in an inertial frame, can be observed in Fig. 2.

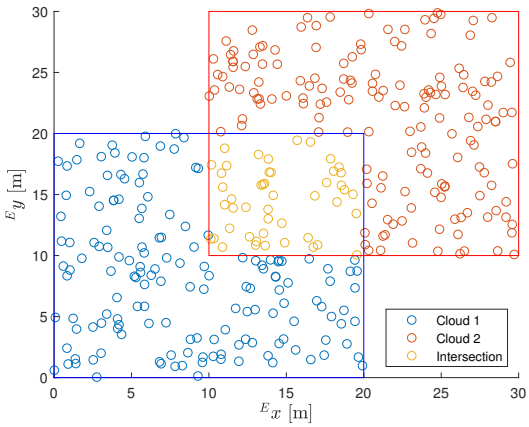

Figure 2: Initial point sets: 25% overlap

Each measured point is subject to a white noise, characterized by the covariance matrix $\Sigma_p = \sigma_p \mathbf{I}_2$. The captured sets were then rotated and translated, to simulate the vehicles capturing it in different frames of reference. The positions of the vehicles in both frames are known *a priori*.

Firstly, the accuracy of the algorithm was assessed. Fixating the number of points, $n = 200$, the average rotation error, defined by $\epsilon_\alpha = {}^j\check{\alpha}_i - {}^j\alpha_i$, and position error, defined by $\epsilon_\mathbf{t} = {}^j\check{\mathbf{p}}_{\mathrm{v}_i} - {}^j\mathbf{p}_{\mathrm{v}_i}$, were evaluated for different overlaps, over the course of several simulations. The results are depicted in Fig. 3. As expected, as the

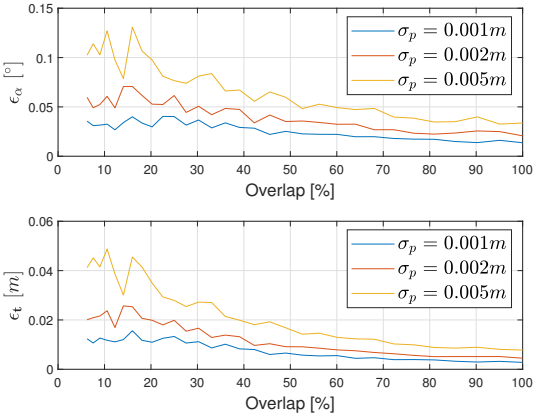

Figure 3: 2D SLAM-tailored algorithm: error metrics

overlap increases, the higher number of points allows for a decreased error in the retrieved rigid motion. Moreover, one can verify how the higher noise negatively affects the retrieved rigid motion, especially in the smaller overlaps. Nonetheless, the success of the algorithm is always guaranteed due to the vehicles recognizing one another.

Afterwards, tests were run to confirm the runtime performance of the algorithm. For that, the elapsed time was measured as the number of points, n, was varied from 50 to 1000. The results for some overlapping percentages, are presented in Fig. 4. The results confirm the

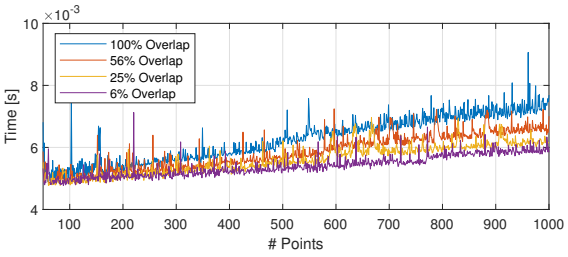

Figure 4: Runtime for different overlaps: linear scale

near linear behaviour of the algorithm and attest that as the overlap, and the number of correct tuples, decreases, the runtime of the algorithm also diminishes.

5. Three-dimensional registration algorithm

In this section, a similar formulation is used for the three-dimensional case.

5.1. Noiseless setting

In the three-dimensional case, three non-collinear correspondent points are needed to unambiguously define a rigid motion. Let these three non-collinear points be ${}^i\mathbf{x}_f$, ${}^i\mathbf{x}_g$, ${}^i\mathbf{x}_h$, and ${}^j\mathbf{x}_f$, ${}^j\mathbf{x}_g$, ${}^j\mathbf{x}_h$ in $\{B_i\}$ and $\{B_j\}$ respectively. The vectors ${}^i\mathbf{x}_{fh}$, ${}^j\mathbf{x}_{fh}$, ${}^i\mathbf{x}_{gh}$, and ${}^j\mathbf{x}_{gh}$ are formed. The TRIAD algorithm, first presented in (Black, 1964), is then used to determine the rotation matrix, ${}^j\mathbf{R}_i$, from these vector measurements. After said rotation is discovered, (29) is used to calculate the vehicle position.

5.2. Rigid motion determination

Once again, consider just the matching subsets ${}^i\mathbf{X}_l$ and ${}^j\mathbf{X}_l$. Let ${}^i\mathbf{x}_f$, ${}^i\mathbf{x}_g$ and and ${}^j\mathbf{x}_f$, ${}^j\mathbf{x}_g$ be the positions of the vehicles in $\{B_i\}$ and $\{B_j\}$ respectively. Since the vehicles see each other, these points are guaranteed to be in the overlapping region of the point clouds.

Using ${}^i\mathbf{x}_f$ and ${}^i\mathbf{x}_g$ as anchors, all the possible triplets are formed using the remaining points in ${}^i\mathbf{X}_l \backslash \{{}^i\mathbf{x}_f,\ {}^i\mathbf{x}_g\}$. These triplets are stored in set ${}^i\mathbf{T}_{\mathbf{x}_{fg}}$,

$$ {}^i\mathbf{T}_{\mathbf{x}_{fg}} = \{{}^i\mathbf{x}_{fg1},\ {}^i\mathbf{x}_{fg2}, \ldots,\ {}^i\mathbf{x}_{fgl}\}. \tag{41} $$

Analogously, the triplets in $\{B_j\}$ are formed and stored in set ${}^j\mathbf{T}_{\mathbf{x}_{fg}}$,

$$ {}^j\mathbf{T}_{\mathbf{x}_{fg}} = \{{}^j\mathbf{x}_{fg1},\ {}^j\mathbf{x}_{fg2}, \ldots,\ {}^j\mathbf{x}_{fgl}\}. \tag{42} $$

Since the rigid motion does not affect the distances between the points, it is guaranteed that for every triplet ${}^i\mathbf{x}_{fgh} \in {}^i\mathbf{T}_{\mathbf{x}_{fg}}$, there is a congruent triplet ${}^j\mathbf{x}_{fgh} \in {}^j\mathbf{T}_{\mathbf{x}_{fg}}$, meaning

$$ \forall\ {}^i\mathbf{x}_{fgh} \in {}^i\mathbf{T}_{\mathbf{x}_{fg}},\ \exists\ {}^j\mathbf{x}_{fgh} \in {}^j\mathbf{T}_{\mathbf{x}_{fg}} :\ {}^i x_{fh} = {}^j x_{fh} \wedge {}^i x_{gh} = {}^j x_{gh}. \tag{43} $$

For every triplet in ${}^j\mathbf{T}_{\mathbf{x}_{fg}}$, ${}^i\mathbf{T}_{\mathbf{x}_{fg}}$ is searched for a congruent triplet. Whenever a match is discovered, such as

with ${}^{i}\mathbf{x}_{fgh}$ and ${}^{j}\mathbf{x}_{fgh}$, the rotation matrix and vehicle position are determined and stored in a tuple, ${}^{j}\mathbf{B}_i$,

$$ {}^{j}\mathbf{B}_i[{}^{i}\mathbf{x}_{fgh},\ {}^{j}\mathbf{x}_{fgh}] = ({}^{j}\mathbf{R}_i,\ {}^{j}\mathbf{p}_{\mathrm{v}_i}). \tag{44} $$

All possible tuples from ${}^{i}\mathbf{T}_{\mathbf{x}_{fg}}$ and ${}^{j}\mathbf{T}_{\mathbf{x}_{fg}}$ are determined and stored in set ${}^{j}\mathbf{S}_i[{}^{i}\mathbf{x}_{fg},\ {}^{j}\mathbf{x}_{fg}]$. Some symmetries in the clouds may occur, leading to congruent triplets that don't necessarily match. It is assumed that the number of correct matches outnumbers any given symmetry, meaning that the correct rigid motion is the most common one present in ${}^{j}\mathbf{S}_i[{}^{i}\mathbf{x}_{fg},\ {}^{j}\mathbf{x}_{fg}]$, *i.e*, the mode.

5.3. Stochastic case

As before, it is irrational to expect that noisy points will yield exactly congruent triplets and equal rigid motions. Analogously to (34) and (35), each point and vector can be represented by a normal distribution centered in the true value. The covariance matrix is, however, updated to accommodate the extra dimension, $\Sigma_p = \sigma_p \mathbf{I}_3$.

Due to the extra dimension, the 95% confidence interval for the measured vector increases slightly, with the difference between two module measurements, ${}^{i}\check{x}_{gh}$, of the same vector now expected to be smaller than $8\sigma_p$.

Due to the complexity of the TRIAD algorithm, it is hard to analytically propagate the uncertainty, as done for the two-dimensional case. Monte Carlo simulations were run to determine how the point noise affects the retrieved matrix. Using the Frobenius norm to determine the distance between two matrices, one can write the error of the measured rotation as

$$ \epsilon_{\mathbf{R}} = \sqrt{\mathrm{tr}((\ {}^{j}\check{\mathbf{R}}_i - \ {}^{j}\mathbf{R}_i)(\ {}^{j}\check{\mathbf{R}}_i - \ {}^{j}\mathbf{R}_i)^{\mathrm{T}})}. \tag{45} $$

Over the course of 1000 simulations, a distance between the vehicle and an obstacle, Δ, of $200\sigma_p$ was assumed. In practice, this is equivalent to imposing a lower bound on the vector size. The 95% confidence interval for $\epsilon_{\mathbf{R}}$ proved to be $0 \leq \epsilon_{\mathbf{R}} \leq 0.05$. The "distance" between two measurements of the same rotation matrix is expected to be smaller than 0.1.

Similarly, using the same simulations, the distance between two measurements of the vehicle position was expected to be smaller than $8\sigma_p$.

5.4. The SLAM-tailored algorithm

For the 3D case, knowing the positions of the vehicles is not enough to determine an approximation of the rigid motion. Let ${}^{i}\mathbf{x}_f$, ${}^{i}\mathbf{x}_g$ be the captured positions of the vehicles in frame $\{B_i\}$, and ${}^{j}\mathbf{x}_f$, ${}^{j}\mathbf{x}_g$ be the same positions in frame $\{B_j\}$. Since these two points are guaranteed to be in the overlapping region, they will be the chosen anchors.

Firstly, the triplets containing ${}^{i}\mathbf{x}_f$, ${}^{i}\mathbf{x}_g$ and an extra point ${}^{i}\mathbf{x}_h \in \ {}^{i}\mathbf{X} \setminus \{{}^{i}\mathbf{x}_f,\ {}^{i}\mathbf{x}_g\}$ are formed. The triplets are organized in a height balanced tree by their module ${}^{i}x_{fh}$. In each node, a subtree is created, organized by the other module, ${}^{i}x_{gh}$.

After this step, all the triplets containing the points ${}^{j}\mathbf{x}_f$, ${}^{j}\mathbf{x}_g$ and ${}^{j}\mathbf{x}_h \in \ {}^{j}\mathbf{X} \setminus \{{}^{j}\mathbf{x}_f, {}^{j}\mathbf{x}_g\}$ are formed. Given a triplet ${}^{j}\mathbf{x}_{fgh}$, the main tree is queried, looking for matches of module ${}^{j}x_{fh}$. If a match is found, the respective subtree is searched for module ${}^{j}x_{gh}$. All these searches are conducted taking into account the previously defined tolerances. If congruent triplets are found, the tuple encoding the rigid motion is determined.

After all the tuples have been calculated, one has to determine the most common rigid motion, *i.e.*, the mode. A clustering algorithm is run, taking into account the tolerances stated in Section 5.3. The point correspondences forming the biggest cluster are assumed to be the overlapping sets, ${}^{i}\mathbf{X}_l$ and ${}^{j}\mathbf{X}_l$. The optimal rigid motion is then calculated using the algorithm developed in (Umeyama, 1991).

5.5. Time complexity

The algorithm starts by forming the $n-2$ triplets from set ${}^{i}\mathbf{X}$. The main tree insertion times is upper bounded by $\mathcal{O}(n \log n)$. The time spent inserting in subtrees is negligible, as not many triplets are expected to have equal modules ${}^{i}x_{fh}$. Afterwards, $m-2$ queries to the main tree are made, in order to find congruent triplets, translating to $\mathcal{O}(m \log n)$. Once again, the time spent in subtrees is negligible in comparison. Similar to the 2D situation, it is assumed that besides the $l-2$ congruent triplets from the overlapping region, $C(m-2)$ false matches appear in this triplet matching process. Hence, $\mathcal{O}(m)$ tuples are calculated. Lastly the complexity of the clustering process grows quadratically with the number of tuples, $\mathcal{O}(m^2)$. The total time complexity is thus $\mathcal{O}((m+n)\log n + m^2)$.

5.6. Algorithm performance

Similar tests to the two-dimensional were performed in this three-dimensional case. To validate the 3D algorithm, a similar method to the 2D simulation was employed, with the creation of random point clouds with different overlapping percentages. The matrix error, $\epsilon_{\mathbf{R}}$, defined in (45), and vehicle position error, defined by $\epsilon_{\mathbf{t}} = \ {}^{j}\check{\mathbf{p}}_{\mathrm{v}_i} - \ {}^{j}\mathbf{p}_{\mathrm{v}_i}$ were then evaluated for several random vehicle orientations. The results are presented in Fig. 5. The errors showed a proportional relation to the introduced noise and, as before, the increasing overlap reduced the error in the retrieved rigid motion. The algorithm also revealed a great success rate, even in an unusually high noise scenario such as $\sigma_p = 0.01\,\mathrm{m}$.

Secondly, the runtime of the algorithm was tested for the overlapping percentages of 10, 40 and 100%, with the number of points, $n = m$, ranging from 100 to 1000. The average runtime is presented in Fig. 6, in log scale. Looking at the time complexity derived in Section 5.5, it was expected that the algorithm runtime grows as the number of formed tuples increases. Fig. 6 clearly backs up this claim, with the increasing overlap slowing the registration process. Furthermore, the approximately straight lines show that the derived quadratic time complexity

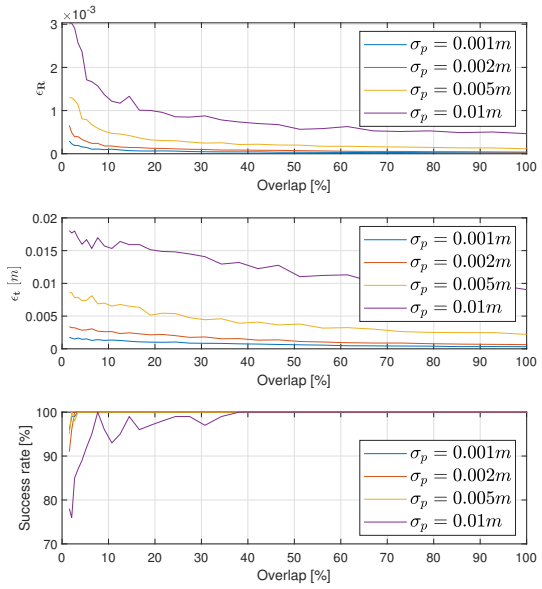

Figure 5: 3D SLAM-tailored algorithm: error metrics

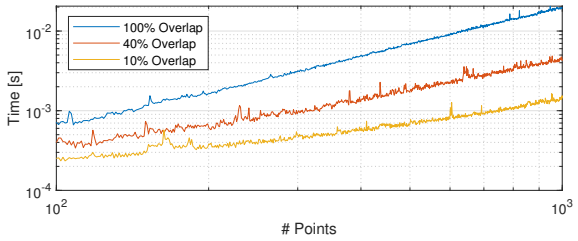

Figure 6: Runtime for different overlaps: log scale

is perhaps a pessimistic scenario, seeing that over the course of a tenfold increase in the number of points, the runtime did not increase by two orders of magnitude in any of the overlaps.

6. Two-dimensional simulation

For the 2D simulation, an artificial arena was created. Two types of trajectories were covered in the simulation: a uniform linear motion, and a uniform circular motion. The individual SLAM stage was maintained until the vehicles were within range of each other, at which point they exchange the necessary information to run the registration algorithm and fuse the maps. Some parameters were defined before the simulation was run:

$$\sigma_p = 0.01\,\mathrm{m}, \tag{46}$$

$$T = 0.1\,\mathrm{s}, \tag{47}$$

$${}^1\mathbf{v}(k) = {}^2\mathbf{v}(k) = \begin{bmatrix} 0 & 1 \end{bmatrix}^{\mathrm{T}}\ \mathrm{m\,s^{-1}}, \tag{48}$$

$${}^1b_\omega(k) = {}^2b_\omega(k) = \pi/100\,\mathrm{rad\,s^{-1}}, \tag{49}$$

with the initial state estimates initialized as

$${}^1\hat{\mathbf{v}}(0) = {}^2\hat{\mathbf{v}}(0) = \begin{bmatrix} 0 & 0.8 \end{bmatrix}^{\mathrm{T}}\ \mathrm{m\,s^{-1}}, \tag{50}$$

and

$${}^1\hat{b}_\omega(0) = {}^2\hat{b}_\omega(0) = 0\,\mathrm{rad\,s^{-1}}. \tag{51}$$

6.1. Individual SLAM

Some metrics are now presented to evaluate the accuracy of the implemented SLAM algorithm. Firstly, the convergence of the velocity and bias estimates was confirmed, as shown in Figs. 7 and 8 However, since the

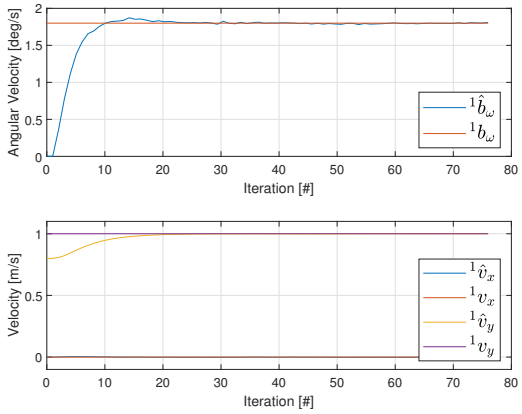

Figure 7: Bias and velocity estimates of vehicle 1

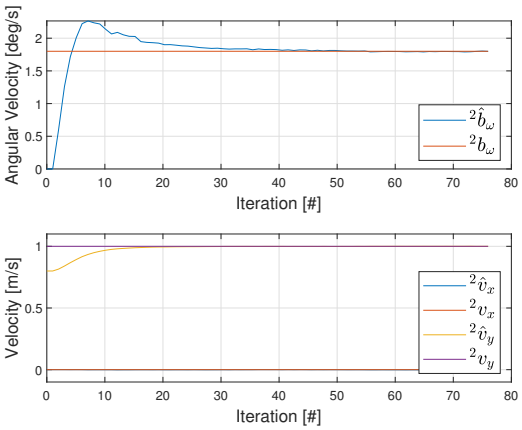

Figure 8: Bias and velocity estimates of vehicle 2

goal is to recreate the map of the surroundings, it is important to measure how accurate the retrieved plan is. For any iteration, the optimal rotation and vehicle position are determined using (22) and (23). Since the true rotation and vehicle position are readily available in a simulation environment, the rotation angle error ϵ_α, and vehicle position error, $\epsilon_{\mathbf{t}}$, were calculated. The results are presented in Fig. 9. The accuracy of the algorithm is verified, as both cases present, at all times, an angular error smaller than 1°, and a position error in the order of the centimetre. However it is possible to see that the angular error, and subsequent position error, do not converge to zero. This can be explained by the time it takes for the bias and velocity estimates to converge. Since

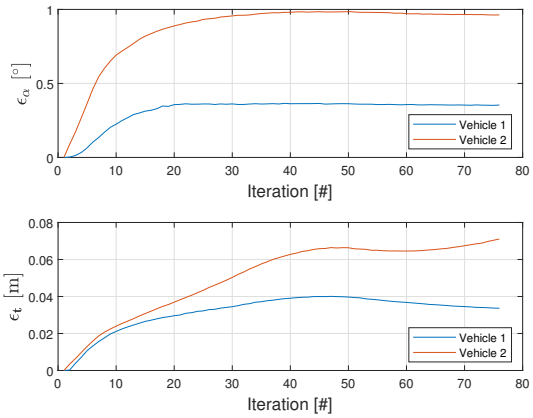

Figure 9: Rotation angle and vehicle position error for vehicles 1 and 2

the bias estimate is initialized as 0, in the first iterations this variable will converge, but an error is accumulated in the initial estimates of the landmarks positions. Since the map is built recursively, that is, relating the landmarks positions in instants t_k and t_{k+1}, (21) and (22), this error in the initial positions is propagated throughout the simulation. A solution to this initial convergence problem would be to perform a loop closure after the bias and velocity estimates have converged. By returning to a previously visited region of the map, it is possible to update the initial estimated landmark position using the newly observed landmark. If everything has converged, this new observation will reduce the uncertainty and decrease the angular and vehicle position error in the entire map.

6.2. The rendezvous

After the vehicles recognize each other, the registration algorithm was put in place in order to fuse the captured maps. Once again, since the positions of the vehicles are known, both the angular error, ϵ_α, and position error, $\epsilon_{\mathbf{t}}$, of this step were calculated. The former, ϵ_α, came at just 2.63×10^{-2}°, whereas the latter, $\epsilon_{\mathbf{t}}$, was just 3.94×10^{-3} m. Both these errors are in line with the averages obtained in Fig. 3.

7. Three-dimensional simulation

Analogously to the 2D simulation, a 3D mission was run using a scan of real-life classroom. The following parameters were used throughout this simulation:

$$\sigma_p = 0.01\,\mathrm{m} \tag{52}$$

$$T = 0.05\,\mathrm{s}, \tag{53}$$

$${}^1\mathbf{v}(k) = {}^2\mathbf{v}(k) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^\mathrm{T}\ \mathrm{m\,s^{-1}}, \tag{54}$$

$${}^1b_{\boldsymbol{\omega}}(k) = \begin{bmatrix} \pi/100 & -\pi/150 & \pi/120 \end{bmatrix}^\mathrm{T}\ \mathrm{rad\,s^{-1}}, \tag{55}$$

$${}^2b_{\boldsymbol{\omega}}(k) = \begin{bmatrix} -\pi/80 & -\pi/130 & \pi/140 \end{bmatrix}^\mathrm{T}\ \mathrm{rad\,s^{-1}}, \tag{56}$$

with the initial estimates set as

$${}^1\hat{\mathbf{v}}(0) = {}^2\hat{\mathbf{v}}(0) = \begin{bmatrix} 0 & 0.8 & 0 \end{bmatrix}^\mathrm{T}\ \mathrm{m\,s^{-1}}, \tag{57}$$

and

$${}^1\hat{\mathbf{b}}_\omega(0) = {}^2\hat{\mathbf{b}}_\omega(0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\mathrm{T}\ \mathrm{rad\,s^{-1}}. \tag{58}$$

7.1. Individual SLAM

As with the two-dimensional case, the convergence of the state estimates pertaining the vehicle was confirmed, as shown in Figs. 10 and 11. Regarding the retrieved

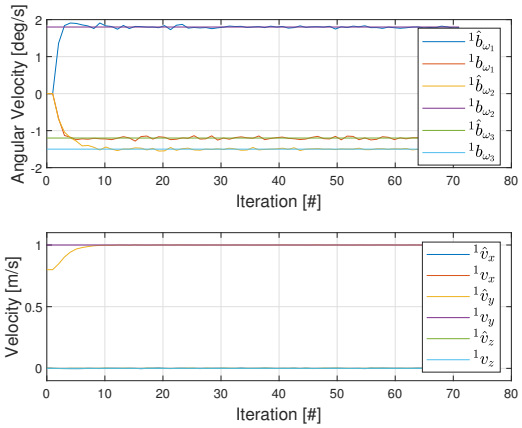

Figure 10: Bias and velocity estimates of vehicle 1

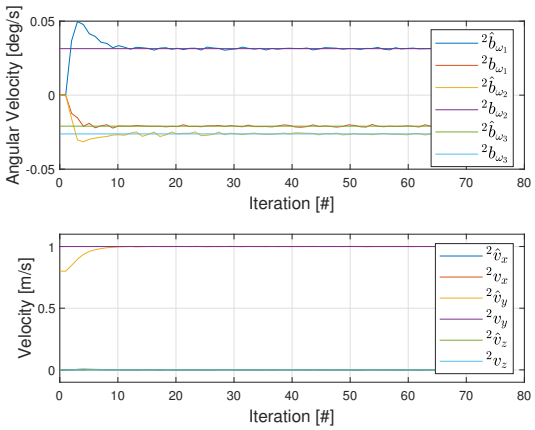

Figure 11: Bias and velocity estimates of vehicle 2

map, the optimal rotation and vehicle position estimates are once again given by (22) and (23), respectively. At any given iteration, these optimal estimates can be compared to their true values, known in a simulation environment. The matrix error, $\epsilon_{\mathbf{R}}$, and vehicle position error, $\epsilon_{\mathbf{t}}$ are now evaluated throughout the simulation.

Looking firstly at the rotation matrix error in Fig. 12, one can see that most of it is accumulated while the velocity and bias estimates are converging to their true values. Like the 2D case, the non convergence of the rotation error to zero can be explained by looking at the way the

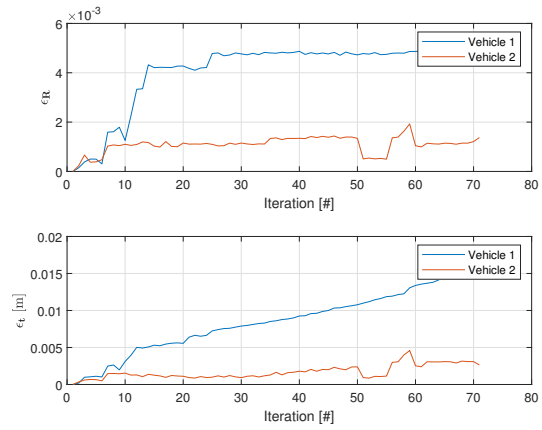

Figure 12: Rotation matrix and vehicle position error for vehicles 1 and 2

inertial map is constructed. In equations (21) through to (3.3), it was established that the optimal matrix estimate at t_{k+1}, ${}^{E}\hat{\mathbf{R}}_i^*(k+1)$, depends on the previous estimate, ${}^{E}\hat{\mathbf{R}}_i^*(k)$. The initial error in the landmark estimates is then propagated throughout the simulation, causing the retrieved map to be slightly rotated.

Since the optimal vehicle position is calculated from this rotation matrix, (23), error $\epsilon_{\mathbf{t}}$ cannot converge to zero either, even appearing to diverge in Fig. 12. This divergence is not worrisome, and can be explained by the presence of a constant $\epsilon_{\mathbf{R}}$. As the map being built is slightly rotated when compared to the real one, it is natural for the retrieved vehicle position to also be slightly rotated. As the vehicle moves away from the origin of the frame, this constant angular error is felt more in terms of absolute value, leading to the increase observed in Fig. 12.

7.2. The rendezvous

Lastly, the developed registration algorithm was used to register the maps collected by the two vehicles. The rotation matrix error, $\epsilon_{\mathbf{R}}$, came at just 8.97×10^{-4}, bettering the results obtained in Section 5.6. This improvement can be explained by having a physically larger overlapping region, while maintaining the standard deviation. As for the vehicle position error, $\epsilon_{\mathbf{t}}$, it also benefited from this smaller matrix error, coming at just 5.58×10^{-3} m. Both these results confirm the accuracy of the algorithm in a real-world scenario.

8. Conclusions

Concerning the individual SLAM stage, the formulation of the problem in the vehicle body-frame proved to be reliable, with a rapid convergence of all estimates in both the 2D and 3D case. Despite the non-linearity of the system, no problems were had in the use of the standard Kalman filter. The retrieved earth-fixed map also revealed to be precise, with the only inaccuracies arising from the initial instants, where the state estimates had yet to converge.

Regarding the developed registration algorithms, both the 2D and 3D algorithm showed impeccable precisions, with runtime performances allowing for future real-time implementations. Also noteworthy to mention that the SLAM missions were performed with a relatively high noise, $\sigma_p = 0.01$ m, when compared to the vehicle range of 3 m. The correct registration is thus a testament to the robustness of the algorithm.

In summary, the primary goal of the work was accomplished. Albeit in a simulation, environment, numerous steps were taken to ensure realistic scenarios. The developed registration algorithms delivered excellent performances in all the test cases, confirming themselves as worthy alternatives to industry standards such as the ICP.

References

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

Black, H. D. (1964). A passive system for determining the attitude of a satellite. *AIAA Journal*, 2(7):1350–1351.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395.

Gelb, A. (1974). *Applied Optimal Estimation*. MIT Press.

Lourenço, P., Guerreiro, B. J., Batista, P., Oliveira, P., and Silvestre, C. (2016). Simultaneous localization and mapping for aerial vehicles: a 3-d sensor-based gas filter. *Autonomous Robots*, 40(5):881–902.

Lourenço, P. (2019). *Globally Convergent Simultaneous Localization and Mapping: Design Techniques, Analysis, and Implementation*. PhD thesis, Universidade de Lisboa, Instituto Superior Técnico.

Newman, P. and Ho, K. (2005). Slam-loop closing with visually salient features. volume 2005, pages 635 – 642.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:376–380.