Data-Driven Covariance Estimation for the Iterative Closest Point Algorithm

Mémoire

David Landry

Maîtrise en informatique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Data-Driven Covariance Estimation for the Iterative Closest Point Algorithm

Mémoire

David Landry

Sous la direction de:

Philippe Giguère
François Pomerleau

Résumé

Les nuages de points en trois dimensions sont un format de données très commun en robotique mobile. Ils sont souvent produits par des capteurs spécialisés de type lidar. Les nuages de points générés par ces capteurs sont utilisés dans des tâches impliquant de l'estimation d'état, telles que la cartographie ou la localisation. Les algorithmes de recalage de nuages de points, notamment l'algorithme ICP (*Iterative Closest Point*), nous permettent de prendre des mesures d'égo-motion nécessaires à ces tâches. La fusion des recalages dans des chaînes existantes d'estimation d'état dépend d'une évaluation précise de leur incertitude. Cependant, les méthodes existantes d'estimation de l'incertitude se prêtent mal aux données en trois dimensions. Ce mémoire vise à estimer l'incertitude de recalages 3D issus d'Iterative Closest Point (ICP). Premièrement, il pose des fondations théoriques desquelles nous pouvons articuler une estimation de la covariance. Notamment, il révise l'algorithme ICP, avec une attention spéciale sur les parties qui sont importantes pour l'estimation de la covariance. Ensuite, un article scientifique inséré présente CELLO-3D, notre algorithme d'estimation de la covariance d'ICP. L'article inséré contient une validation expérimentale complète du nouvel algorithme. Il montre que notre algorithme performe mieux que les méthodes existantes dans une grande variété d'environnements. Finalement, ce mémoire est conclu par des expérimentations supplémentaires, qui sont complémentaires à l'article.

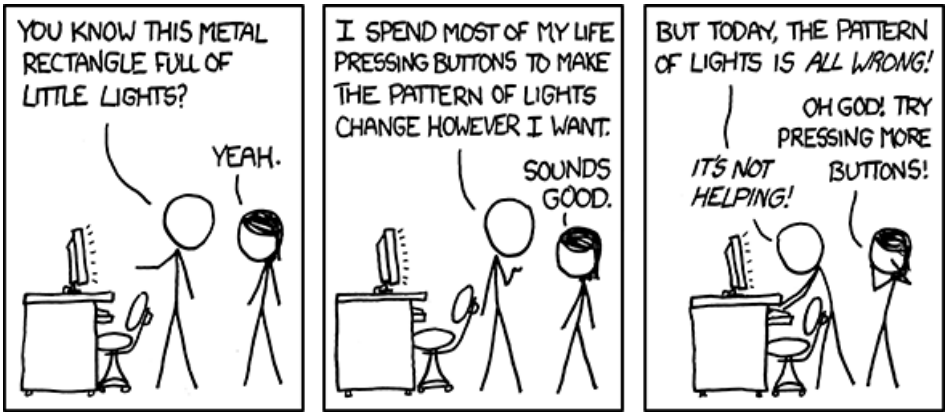
Abstract

Three-dimensional point clouds are an ubiquitous data format in robotics. They are produced by specialized sensors such as lidars or depth cameras. The point clouds generated by those sensors are used for state estimation tasks like mapping and localization. Point cloud registration algorithms, such as Iterative Closest Point (ICP), allow us to make ego-motion measurements necessary to those tasks. The fusion of ICP registrations in existing state estimation frameworks relies on an accurate estimation of their uncertainty. Unfortunately, existing covariance estimation methods often scale poorly to the 3D case. This thesis aims to estimate the uncertainty of ICP registrations for 3D point clouds. First, it poses theoretical foundations from which we can articulate a covariance estimation method. It reviews the ICP algorithm, with a special focus on the parts of it that are pertinent to covariance estimation. Then, an inserted article introduces CELLO-3D, our data-driven covariance estimation method for ICP. The article contains a thorough experimental validation of the new algorithm. The latter is shown to perform better than existing covariance estimation techniques in a wide variety of environments. Finally, this thesis comprises supplementary experiments, which complement the article.

Contents

List of Figures

List of Tables



Source: `www.xkcd.com/722/`

Remerciements

Merci à Maman, Papa, Élise, Catherine, Pascal, Jean-Michel, Myriam, Antoine. Vous êtes la fondation d'où je peux bâtir ma vie d'adulte. Merci à Philippe d'avoir ouvert autant de portes : ce sont toutes ces opportunités qui ont rendu mon projet possible. Merci à François pour le flegme et les petits coups de gouvernail. Merci à Philippe d'être le canard en plastique le plus sage de l'histoire. Merci à Gari pour la camaraderie et les nombreux lifts. Merci à Philippe pour le poulet frit. Merci à Jeff, Seb, Steffen, Quan, Simon-Pierre, Mathieu, Johann, Julia et tous les membres du lab. Grâce à vous, c'était un plaisir de prendre le bus chaque matin. Merci à Julien et toute l'équipe du VAUL pour m'avoir montré qu'en gang, on peut faire tellement plus.

Foreword

Chapter 4 of this work consists in an inserted paper. It was submitted to the IEEE International Conference on Robotics and Automation (ICRA) on September 15th, 2018. ICRA is a world-leading robotics conference, with an h5-index of 75 as of this year.[1] It was *accepted* for publication, but is not published at this time of writing these lines. In the meantime, the paper is available as a preprint in the *arXiv* repository with ID 1810.01470. I am the main author of the publication, and as such I was responsible of its redaction and the underlying experiments. My two co-authors, François Pomerleau and Philippe Giguère, are also the co-supervisors of my masters. They provided guidance for this endeavour and assisted me in the redaction of the paper. François also designed some of the figures. The paper is inserted almost exactly as submitted. The section numbers and figure numbers were modified to better integrate to the rest of this document. Some footnotes were added to better connect the paper with the rest of this work. The layout was modified to fit the one of a standard masters thesis at *Université Laval*. Some notation mistakes were corrected after we became aware of them.

[1] `https://scholar.google.com/citations?view_op=top_venues`

Introduction

As robots leave factory floors and laboratories, they need to navigate in environments that are complex and challenging [1]. Of particular importance is the need for robots to localize in unknown environments with accuracy. Various sensors are used to this end, such as cameras or radars. Lidar sensors draw particular attention because of their high level of precision and robustness to lighting change. This type of sensor is increasingly commoditized, as production scales up to meet the demand of the automotive industry.[2] Interestingly, this better cost efficiency makes the use of 3D lidars possible where only 2D sensors could be afforded in the past.

Lidar sensors produce point clouds at a regular interval, e.g. ten scans per second. Point clouds are a set of 3D points that live in the frame of reference of the sensor. Lidars points represent the location of objects that reflected laser beams. Those scans provide a rich and accurate representation of the surrounding environment.

The process of aligning two scans into one another is called *point cloud registration*. It is illustrated in Figure 1. The important feature of point cloud registration is that it measures the displacement from the location of first point cloud to the second. We refer to this displacement, a roto-translation in space, as the registration transformation. Knowing this transformation allows the robot to localize inside the environment, or build a map of it.

The Iterative Closest Point (ICP) algorithm [2], [3] is one such point cloud registration algorithm. It estimates a registration transformation using a two steps procedure. First, it associates the closest points in a pair of point clouds. It then finds a transformation that minimizes a distance between those pairs. This procedure is repeated iteratively until convergence. Since its introduction in the early 1990's, ICP has been widely used by the robotics community. It now has abundant variants [4]

[2] `https://arstechnica.com/cars/2018/01/driving-around-without-a-driver-lidar-technology-explained/3/`

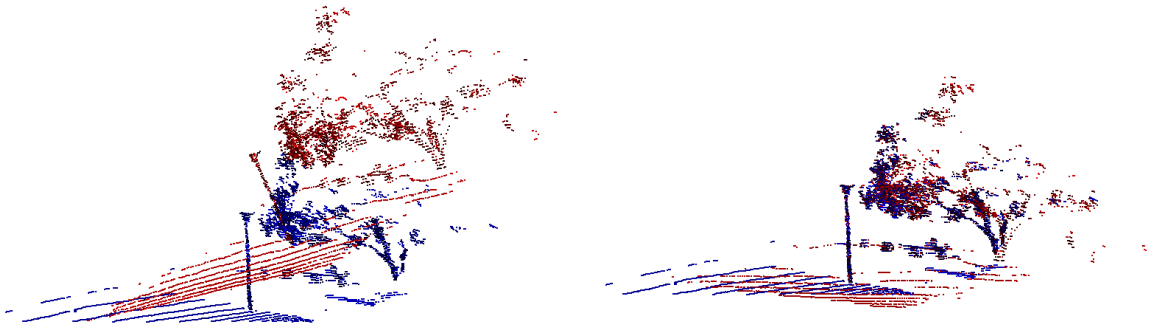


Figure 1 – Point cloud registration is the process of aligning two point clouds into one another. By measuring the registration transformation applied on the point clouds in doing so, a robot can measure its displacement in space.

that aim to make it more accurate and robust.

Mobile robotics being a science of uncertainty, it is important to know the precision and accuracy [5] of any measurement. Since the very introduction of ICP, estimating its uncertainty (in the form of a covariance) has been identified as an interesting research avenue [3]. This problem has attracted interest recently [6]–[9] because the existing solutions do not seem to scale well to 3D point clouds [10]. Consequently, we pose our research questions as follows: *How to improve our knowledge of the uncertainty of ICP measurements? How to estimate it at speeds that allow an online use?*

This thesis contains an inserted article, and is articulated around that publication. Chapters 1, 2 and 3 provide a more complete theoretical background. Their purpose is to bring the reader up to speed in terms of the ICP algorithm. They draw particular attention to the parts of ICP that are important to covariance estimation. They explain the theoretical context in which the article was written, and justify our formulation. More precisely, Chapter 1 introduces the $\mathrm{SE}(3)$ Lie group and its companion Lie algebra, which give us mathematical tools to deal with roto-translations in 3D. It is followed by Chapter 2, that studies the ICP algorithm in more details. It poses ICP as a chain of five well-defined blocks, and explains the effect of each of these blocks on the uncertainty of ICP. Chapter 3 considers this uncertainty. First, it clarifies the key terms *uncertainty* and *covariance*. This helps the discussions of the inserted article. Then, Chapter 3 also tackles the surprisingly challenging process of sampling ICP registrations to estimate their distribution. The inserted article [11], in Chap-

ter 4, is the heart of this work. It uses the groundwork made previously to elaborate a covariance estimation algorithm for 3D ICP. We put the algorithm through a complete experimental validation, thus demonstrating its usefulness. Finally, Chapter 5 extends on the submitted article by providing supplementary experiments. These experiments improve our knowledge of the capabilities of the introduced algorithm, and point at research directions which could improve ICP covariance estimation in the future.

Chapter 1

Mathematical background

Mobile robotics is a very tangible domain that requires us to think in terms of displacements in space. This chapter introduces mathematical formalism which allows us to do so. We present the Special Euclidian group $\mathrm{SE}(3)$ and its companion Lie algebra $\mathfrak{se}(3)$. We show how to use them to represent rigid transformations in 3D space. Then, we also introduce how to manipulate this representation to think about rigid transformations in a probabilistic manner. Finally, we use this algebraic representation to compound rigid transformations and keep track of their cumulative uncertainty.

1.1 Special Euclidian group

Roto-translations in 3D are typically represented as a 4×4 matrix

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{bmatrix}, \tag{1.1}$$

where $\boldsymbol{R} \in \mathrm{SO}(3)$ is a rotation matrix, and $\boldsymbol{t} \in \mathbb{R}^3$ is a translation vector. The notation $\boldsymbol{0}$ represents a null vector, which has three dimensions in this case. The matrix $\boldsymbol{T}$ is a member of the Special Euclidian group $\mathrm{SE}(3)$, which is the space of all roto-translations. $\mathrm{SE}(3)$ is a Lie group. Consequently, it has an associated Lie algebra $\mathfrak{se}(3)$. Indeed, every member of $\mathrm{SE}(3)$ has an associated vector representation that lives in $\mathfrak{se}(3)$. The vector representation has six dimensions. The rotations themselves also form a Lie group. The matrix $\boldsymbol{R}$ is in fact an element of the Special Orthogonal group $\mathrm{SO}(3)$, which comprises all 3D rotations. Consequently, any operation defined on a Lie group applies both to full transformations $\boldsymbol{T}$, but also simple rotations $\boldsymbol{R}$.

The mapping of roto-translations to $\mathfrak{se}(3)$ is useful because it allows us to think those transformations in an almost linear fashion [12]. Also, the Lie algebra representation is optimal in that it has as many terms as 3D roto-translations have degrees of freedom. There are no redundant terms. This makes optimization easier, because no regularization is needed to make a valid roto-translation from a Lie algebra vector. This is an advantage over quaternions (which have to be normalized) or full transformation matrices (which need an orthonormal $\boldsymbol{R}$), for instance.

The caveat is that we have to be careful about singularities when mapping from a group to its algebra, and vice versa. Our general strategy is to use the numerical stability [12, p. 267] of the $\mathrm{SE}(3)$ group representation when possible, and use the linear model $\mathfrak{se}(3)$ when it is useful. This way, we get the best of both worlds.

1.1.1 Exponential map

To go from a Lie algebra to its group, we use the exponential map [12]. For $\mathrm{SE}(3)$, we pose $\exp(\cdot)$ such that

$$\boldsymbol{T} = \exp(\boldsymbol{\xi}) \qquad \text{and} \qquad \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\omega} \end{bmatrix}. \tag{1.2}$$

Here, the vector $\boldsymbol{\xi} \in \mathfrak{se}(3)$ is decomposed in a translation part $\boldsymbol{u} \in \mathbb{R}^3$ and an axis-angle rotation $\boldsymbol{\omega} \in \mathbb{R}^3$. Given that the angle of rotation is $\theta = \sqrt{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}$, we have a closed-form expression for this exponent:

$$\exp(\boldsymbol{\xi}) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{V}\boldsymbol{u} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{1.3}$$

such that

$$\boldsymbol{R} = \boldsymbol{I} + \Big(\frac{\sin\theta}{\theta}\Big)\boldsymbol{\omega}_\times + \Big(\frac{1-\cos\theta}{\theta^2}\Big)\boldsymbol{\omega}_\times^2 \tag{1.4}$$

and

$$\boldsymbol{V} = \boldsymbol{I} + \Big(\frac{1-\cos\theta}{\theta^2}\Big)\boldsymbol{\omega}_\times + \Big(\frac{\theta-\sin\theta}{\theta^3}\Big)\boldsymbol{\omega}_\times^2. \tag{1.5}$$

Here, the $\times$ operator indicates the cross product matrix of a vector such that

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{1.6}$$

This closed-form expression is derived using a Taylor-expansion [13].

1.1.2 Log map

We may also wish to do the reverse mapping, i.e., go from a transformation matrix $\boldsymbol{T}$ to a Lie algebra representation $\boldsymbol{\xi}$. With that in mind, we first use the Rodriguez formula

$$\theta = \arccos\left(\frac{\operatorname{tr}(\boldsymbol{R}) - 1}{2}\right) \tag{1.7}$$

to find the angle of rotation θ. Then, the rotation part $\boldsymbol{\omega}$ can be recovered with

$$\ln(\boldsymbol{R})_\times = \boldsymbol{\omega}_\times = \frac{\theta}{2\sin\theta}(\boldsymbol{R} - \boldsymbol{R}^\top). \tag{1.8}$$

For the translation part $\boldsymbol{t}$, we simply reuse Equation 1.5 such that

$$\boldsymbol{t} = \boldsymbol{V}^{-1}\boldsymbol{u}. \tag{1.9}$$

We restate that one should be careful about singularities when using $\exp(\cdot)$ and $\log(\cdot)$. More concretely, when the angle of rotation is small we should replace the components of Equation 1.3 and Equation 1.8 with their Taylor series, for the sake of numerical stability [13]. The Taylor series are developed in Appendix C.

1.2 Normal distributions over rigid transformations

One of the main benefits of representing rigid transformations using Lie groups and Lie algebras is that the linearity of $\mathfrak{se}(3)$ allows us to express uncertain transformations easily [13]. Suppose an uncertain transformation

$$\boldsymbol{T} = \exp(\boldsymbol{\xi})\bar{\boldsymbol{T}} \tag{1.10}$$

where $\bar{\boldsymbol{T}} \in \mathrm{SE}(3)$ is the mean transformation, and $\boldsymbol{\xi} \in \mathfrak{se}(3)$ is a small perturbation applied to it. We say that $\boldsymbol{T}$ is normally distributed according to a covariance $\boldsymbol{Y}$ when we have

$$\boldsymbol{T} \sim \mathcal{N}(\bar{\boldsymbol{T}}, \boldsymbol{Y}), \tag{1.11}$$

which is shorthand for

$$\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Y}). \tag{1.12}$$

The above equations apply to any Lie group, but for 3D transformations they imply that the covariance $\boldsymbol{\Upsilon} \in \mathbb{R}^{6\times6}$. Furthermore, transposing the anatomy of the $\boldsymbol{\xi}$ vector from Equation 1.2, we can subdivide the covariance with

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\Upsilon}_{uu} & \boldsymbol{\Upsilon}_{u\omega} \\ \boldsymbol{\Upsilon}_{\omega u} & \boldsymbol{\Upsilon}_{\omega\omega} \end{bmatrix}. \tag{1.13}$$

A method for applying a transformation $\boldsymbol{A} \in \mathrm{SE}(3)$ on a distribution emerges quite naturally from this context. Applying $\boldsymbol{A}$ on both sides of Equation 1.10, we get

$$\boldsymbol{A}\boldsymbol{T} = \boldsymbol{A}\exp(\boldsymbol{\xi})\bar{\boldsymbol{T}} \tag{1.14}$$

$$= \exp(\mathrm{Adj}_{\boldsymbol{A}}\,\boldsymbol{\xi})\boldsymbol{A}\bar{\boldsymbol{T}}. \tag{1.15}$$

Note the use of the adjoint $\mathrm{Adj}_{\boldsymbol{A}} \in \mathbb{R}^{6\times6}$ here, which by definition is the matrix that allows us to commute a transformation from one side of the exponential map to another. From there, we can show [13] that the transformed distribution is in fact

$$\boldsymbol{A}\boldsymbol{T} \sim \mathcal{N}(\boldsymbol{A}\bar{\boldsymbol{T}}, \mathrm{Adj}_{\boldsymbol{A}} \cdot \boldsymbol{\Upsilon} \cdot \mathrm{Adj}_{\boldsymbol{A}}^{\top}). \tag{1.16}$$

Instructions to compute $\mathrm{Adj}_{\boldsymbol{A}}$, as well as other useful Lie algebras identities, are available in Barfoot [12] and Eade [13].

1.2.1 Computing moments

This formulation creates an interesting problem when it comes to computing the moments of samples of registration transformations. Indeed, the mean $\bar{\boldsymbol{T}}$ found in Equation 1.10 cannot be computed in closed form given n samples $\boldsymbol{T}_{0\ldots n}$. Instead, we rely on an iterative approach. At iteration $j = 0$ we set $\bar{\boldsymbol{T}} \leftarrow \boldsymbol{T}_0$ (or any other element of the sample). Using this initial guess, we can iteratively compute new estimates for the mean and the covariance. Given

$$\boldsymbol{\xi}_i = \log(\boldsymbol{T}_i\bar{\boldsymbol{T}}_j^{-1}), \tag{1.17}$$

we have

$$\bar{\boldsymbol{T}}_{j+1} = \exp\Big(\frac{1}{n}\sum_{i=0}^{n}\boldsymbol{\xi}_i\Big)\bar{\boldsymbol{T}}_j \tag{1.18}$$

$$\boldsymbol{\Upsilon}_j = \frac{1}{n}\sum_{i=0}^{n}\boldsymbol{\xi}_i\boldsymbol{\xi}_i^{\top}. \tag{1.19}$$

This procedure usually takes three to four iterations to converge [13]. Afterwards, $\bar{\boldsymbol{T}}_j$ and $\boldsymbol{\Upsilon}_j$ are estimates of the first and second moment of the samples $\boldsymbol{T}_{0..n}$. We can make the estimate of the covariance unbiased by replacing $\frac{1}{n}$ by $\frac{1}{n-1}$ in Equation 1.19. The effect of this replacement is minimal in the present context, since we typically use samples of size $n > 1000$. Our large sample sizes are required by the large dimensionality of $\mathrm{SE}(3)$.

1.3 Compounding uncertain poses

For this work, we also wish to understand the compounding of uncertain poses in $\mathrm{SE}(3)$. Given

$$\boldsymbol{T} = \boldsymbol{T}_1 \boldsymbol{T}_2 \tag{1.20}$$

with $\boldsymbol{T}_1 \sim \mathcal{N}(\bar{\boldsymbol{T}}_1, \boldsymbol{\Upsilon}_1)$ and $\boldsymbol{T}_2 \sim \mathcal{N}(\bar{\boldsymbol{T}}_2, \boldsymbol{\Upsilon}_2)$ a pair of uncertain poses, we want to compute the cumulative uncertainty of $\boldsymbol{T}$. This proves useful, for instance, to track the uncertainty of odometry. To compute this uncertainty, we need to compound the smaller uncertainties $\boldsymbol{\Upsilon}_i$ of the individual localization estimates, in order to better represent the larger uncertainty of the integrated result $\boldsymbol{T}$.

There are approximations up to the fourth order for this compounding [12]. The resulting expressions are actually quite daunting, and are left out in the interest of space. Here, we simply denote important properties of the compounding formulas. Two equations are provided in Barfoot [12], a second-order approximation and a fourth-order approximation. The second-order approximation is mathematically much simpler. It has the drawback that it does not capture all the "leak" from rotational components into additional degrees of freedom. In other words, using the second approximation, a rotational uncertainty may not provoke a translational uncertainty at the next time step (even though it should). The fourth-order approximation captures this phenomenon better, at the cost of being more complicated to express.

In both cases, we have to remember that we only have approximations at hand. The compounding error will grow as uncertainty grows. It grows much more slowly for the fourth-order approximation than for the second-order one [12, Fig. 7.7]. Still, these approximations should be used to keep track of small transforms when possible, in order to minimize the approximation errors. Knowing this, it is wise to store many small uncertain transformations instead of a single larger one.

Chapter 2

Iterative Closest Point algorithm

ICP is a point cloud registration algorithm that has applications in various fields, such as medical imagery and computer vision. For mobile robotics applications, which are of interest to us, it is commonly used to register point clouds from lidar sensors. With two lidar scans of the same environment (but from different point of views), we can find a rigid transformation between two reference frames. This, in turn, is very useful for a robot to keep track of its position. The present chapter describes the ICP algorithm in broad terms. It lists the components of an ICP pipeline, and the effect of these components on the uncertainty. Finally, we discuss the objective function of ICP, and how it creates an optimization landscape. The optimization landscape is a metaphor for the shape of the objective function as it is optimized. This notion will be useful to discuss the covariance of ICP later on.

2.1 Defining ICP

ICP is attributed to Besl *et al.* [2] and Zhang [14]. It is a procedure to find the transformation ${}^{A}_{B}\boldsymbol{T}$ that brings a reading point cloud ${}^{B}\boldsymbol{Q}$ in the frame of reference point cloud ${}^{A}\boldsymbol{P}$. It does so by first associating every point in $\boldsymbol{Q}$ with its nearest neighbor in $\boldsymbol{P}$. Then, some metric of the distance between the associated points is minimized. This reassociation/minimization procedure is repeated iteratively until convergence. Figure 1 illustrates a successful ICP registration.

Like any non-convex algorithm, ICP is sensitive to local minima. To alleviate this, we want to provide a good initial estimate ${}^{A}_{B}\check{\boldsymbol{T}}$ before initiating the registration, i.e., something better than identity. This estimate could be the result of the odometry of the wheels when moving the sensor from A to B, for instance.

2.2 ICP pipeline

A large collection of variants of ICP was developed over the years [4]. They are formalized into a chain with well-defined blocks. These blocks, as well as an example choice for each of them, are shown in Figure 2.1. The figure contains the five main blocks that comprise an ICP toolchain.

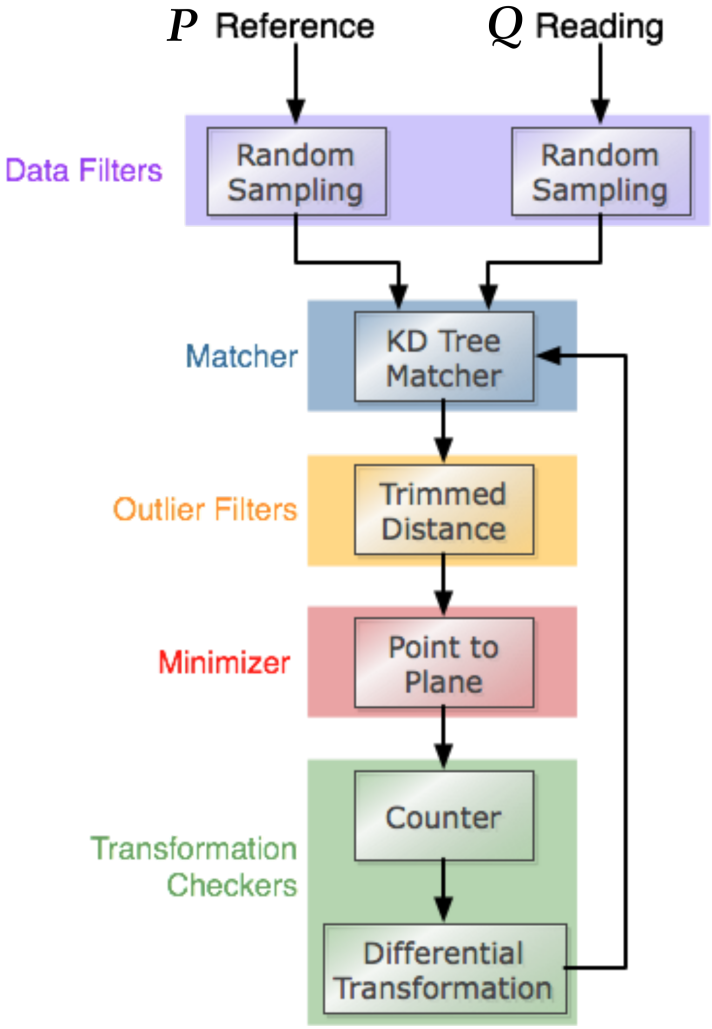


Figure 2.1 – Example of ICP toolchain configuration. The reading $\boldsymbol{Q}$ and reference $\boldsymbol{P}$ point clouds are input into the pipeline at the top. Some of the choices made here can largely influence the uncertainty of ICP. Source: `http://github.com/ethz-asl/libpointmatcher`.

Firstly, we select the data filters, which are the preprocessing operations applied to the point clouds before they are used for registration. In the current case, *Random sampling* indicates that a random subsampling of the points is used. In practice, data filters are also the opportunity to augment the points with supplementary data.

For instance, a data filter could supplement every point with information about its neighborhood, such as the estimated plane formed by the neighborhood or the density of points in it. This supplementary data can then be used in the subsequent steps of the registration.

The second step in the pipeline is the point matcher. It has the responsibility of finding the nearest neighbors between the reference $\boldsymbol{P}$ and the reading $\boldsymbol{Q}$ point clouds. The k-dimensional tree (k-d tree) is popular for this, because it allows for nearest neighbors searches in a sub-linear time (with respect to the number of points) [15].

Next, the outlier filters have the responsibility of filtering spurious point associations. They aim to remove associations between points that are unrelated, e.g., points from a house associated with the points from a tree. With this objective, outlier filters often remove the pairs of points for which the error metric is too large. Choosing what "too large" means in this context is an active research area [16]. Outlier filtering is very important to the accuracy of ICP. Indeed, it makes it a lot more robust, and thus less susceptible to suffer from local minima. A typical choice of outlier filter is *trimmed distance*, which keeps a percentage of associations that have the smallest error according to the minimizer.

Minimizers represent the choice of error metric between the pairs of associated points. For instance, using the point-to-point error metric minimizes the Euclidian distance between the associated points. Other possible choices include plane-to-plane, point-to-Gaussian or Gaussian-to-Gaussian. In Figure 2.1, the choice of error metric is point-to-plane. With this metric, ICP minimizes the distance between a point and the *estimated plane* around the other point. Note that changing the minimizer changes the closed-form equations that are minimized during the iterative process. Consequently, it modifies the function that ICP optimizes (the objective function).

Finally, transformation checkers are simply convergence criteria for the iterative process. In Figure 2.1, we have both a counter and a differential transformation checker. The counter puts an upper bound on the number of iterations. The differential transformation checker detects convergence by looking at the delta between successive iterations. If the delta becomes small enough, we consider that convergence is attained and the iterative process is stopped.

Of all the steps in an ICP toolchain, the choice of minimizer (error metric) is the one that has the most importance to uncertainty estimation. That is because it dramati-

cally affects the shape of the objective function. In turn, this changes the optimization landscape of ICP, and affects where the algorithm is likely to converge. The objective function of ICP is studied in more details in Section 2.3.

To a lesser extent, the transformation checkers also affect the uncertainty of ICP. This is illustrated in Figure 2.2. The dots in the figure represent 5000 ICP registrations at different moments of the optimization process. The subject of registration is a pair of identical cubes, modulo a Gaussian noise on the individual points. The Gaussian noise simulates the measurement noise that we find on real lidar sensors. For visualization purposes, we only illustrate the $\boldsymbol{u}$ part of $\boldsymbol{\xi}$ here, which corresponds to the translation of the registration transformation.

From this figure, we conclude that transformation checkers that are too aggressive yield registrations transformations that are more scattered. That is because ICP did not have enough iterations to converge correctly. More scatter in the $\boldsymbol{\xi}$ vectors translates in a larger covariance (see Equation 1.19). We do not study the impact of the transformation checkers in detail, because their effect on the uncertainty is minimal if they are configured correctly. On the contrary, the effect of the choice of error metric on the uncertainty is, in a way, unavoidable. There is no objectively "correct" way to select an error metric for ICP, which explains the wide variety of metrics still in use today.

The effect of the remaining blocks of the toolchain (Data Filters, Matcher, Outlier Filters) on the uncertainty was not controlled for this work. The data filters and the matcher are expected to have a minimal impact, though validating this experimentally is potential future work. For the outlier filters, the application domain dictates what the correct choice is. We are interested by the covariance of ICP given that choice. Consequently, we made a choice that we thought was reasonable (trimmed distance, keep 70 %), and maintained this configuration throughout the experiments.

2.3 ICP objective function

The choice of ICP pipeline affects the objective function. Consequently, once we establish that pipeline, we can explore the objective function of ICP in more details. From an optimization point of view, the problem of ICP can be formulated as finding

$$\arg\min_{\boldsymbol{\xi}} J\big(\exp(\boldsymbol{\xi})\check{\boldsymbol{T}}\big), \tag{2.1}$$

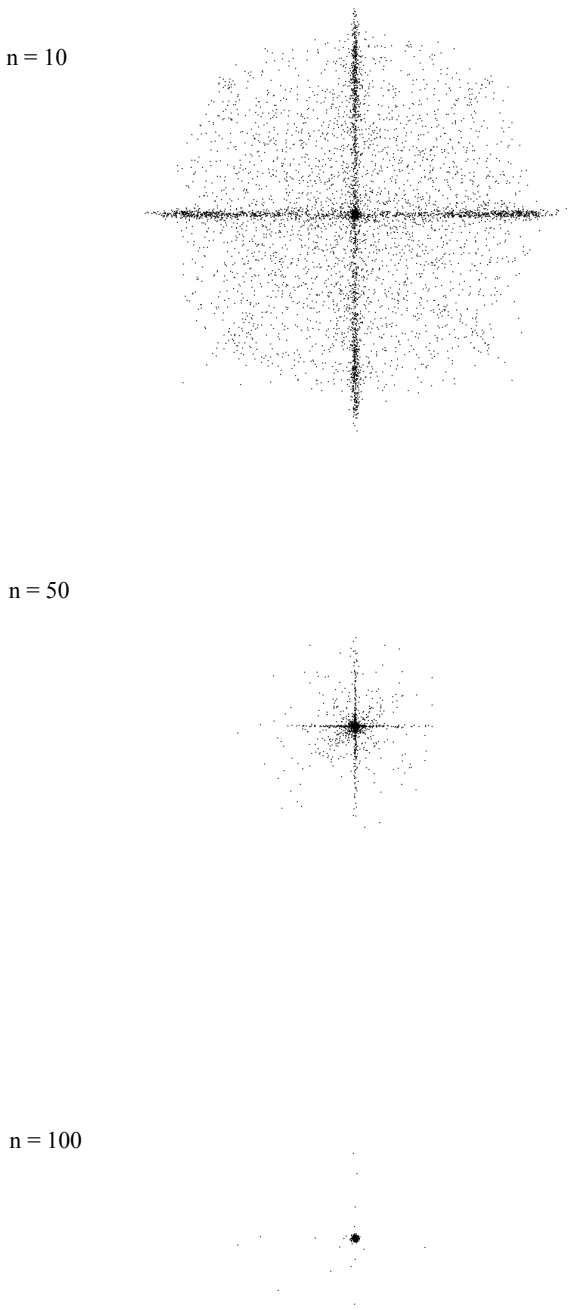


Figure 2.2 – The translation $\boldsymbol{u} \in \mathbb{R}^3$ to which ICP converges when the maximum number of iterations n varies. The point cloud of a cube is registered against itself here. The progression of 5000 ICP registration is illustrated. If n is too low, ICP will stop before convergence. This increases the spread of ICP registration transformations, which means the uncertainty is increased.

where $J(\cdot)$ is the objective function of ICP. The behavior of this function changes given the point clouds being registered, or the configuration of the ICP pipeline. We can think of that objective function as being the sum of the error metric for every pair of associated points (modulo the outliers).

In that framework, $\boldsymbol{\xi}$ is the perturbation applied on the initial estimate $\breve{\boldsymbol{T}}$ that best registers the point clouds. As an experiment, we can manipulate $\boldsymbol{\xi}$ to plot the value of $J(\cdot)$ for a given rigid transformation. Figure 2.3 illustrates the value of $J(\cdot)$ when registering a simple pair of cubes. To the left, we see the pair of cubes being registered. Currently, the blue cube is ill-aligned with the red one. The points of the blue cube are, in average, far from their closest neighbor in the red cube. Consequently, the value of the objective function is high. To minimize it, ICP must find a $\boldsymbol{\xi}$ that better aligns both cubes, and thus minimizes the value of $J(\cdot)$. To the right, we plot the value of the objective function for different values of $\boldsymbol{\xi}$. The perturbation is varied in two axes, the value of $\boldsymbol{u}_x$ and $\boldsymbol{u}_y$.

This plot gives us an overall portrait of the objective function landscape, over two dimensions out of six. We could expect to find a discrete minimum for $J(\cdot)$ in this situation. However, the data shows a cross shape. This shape is explained by the outlier filter, which we set to a 70 % *trimmed distance* filter (see Section 2.2). If the transformation is perfectly aligned in the x axis, then the outlier filter is free to remove errors in the y axis, letting ICP move more freely in that axis. The reverse is also true: if the cubes are perfectly aligned in the y axis, there is some freedom in the x axis. This leeway in both x and y creates the cross shape.

In practice, this modified shape of the objective function means that ICP could converge more slowly to the ground truth. Experiments show that the registration transformations live longer on the axes of the cross before converging to the ground truth in the middle. In fact, we already observed this "slow down on the cross" effect in Figure 2.2. The cross of the objective function shows up in the configuration of the registration transformations themselves. In a way, Figure 2.2 and Figure 2.3 are looking at the same phenomenon from two different point of views.

The duality between those figures shows that there is an intimate relationship between ICP's objective function and its uncertainty. A family of uncertainty prediction algorithms, called the *closed-form* estimates, use the derivative of the objective function directly to make their predictions [17]. Closed-form algorithms cannot cope with point reassociations well, and thus must make supplementary assumptions

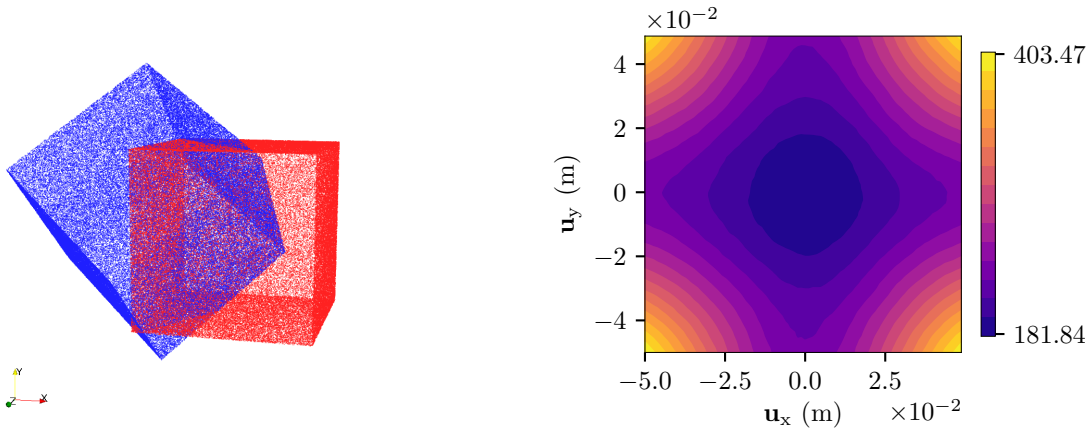


Figure 2.3 – *Left*: A pair of noiseless cubes being registered. *Right*: ICP objective function when registering the pair. A high value of the objective function means that the sum of errors for every pair of associated points is high for a given roto-translation. Two dimensions out of six are plotted: the translation in x and the translation in y ($\boldsymbol{u}_x$ and $\boldsymbol{u}_y$).

about the association of points. They either assume that the pairing does not change, or that the changes in pairing affect the objective function very little [7]. Chapter 4, the inserted article, gives a more complete overview of these algorithms, and the consequences of ignoring point reassociation for uncertainty estimation.

Coming back to the objective function, we are not limited to plotting the x and y translation axes. Figure 2.4 shows the same objective function, but this time plotted along a translation and a rotation axis. The figure shows that the rotation has a more important impact on the objective function than the translation for this pair of point clouds. Consequently, we consider that the rotation is more constraining than the translation in this case. Notice the difference in the scale of the objective function score in this figure, comparatively to that of Figure 2.3.

With those experiments in mind, we can progress towards real pairs of point clouds. To this purpose, we use pairs of point clouds from the *Challenging data sets for point cloud registration algorithms* [18]. This dataset contains dense point clouds from six locations, ranging from structured to unstructured and indoor to outdoor. It was chosen because it provides ground truth data accurate to the millimeter. For the present experiment, we pick point clouds from the *Gazebo* and *Hauptgebaude* locations. Figure 2.5 shows photographs of these locations. The value of J for these locations is shown in Figure 2.6. Since *Hauptgebaude* (on the right) is an almost featureless hall-

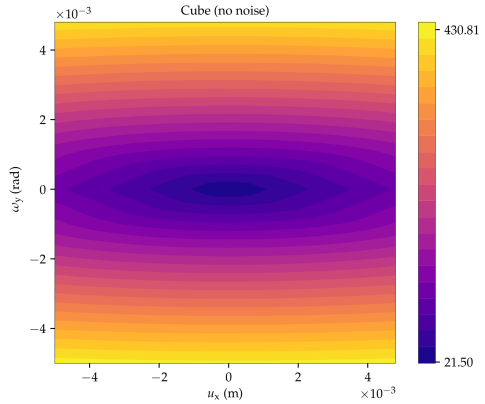


Figure 2.4 – ICP objective function when registering a pair of noiseless cubes. The vertical axis shows the impact of a rotation on the objective function (ω_z). In this case, the rotation is more constraining than the translation.

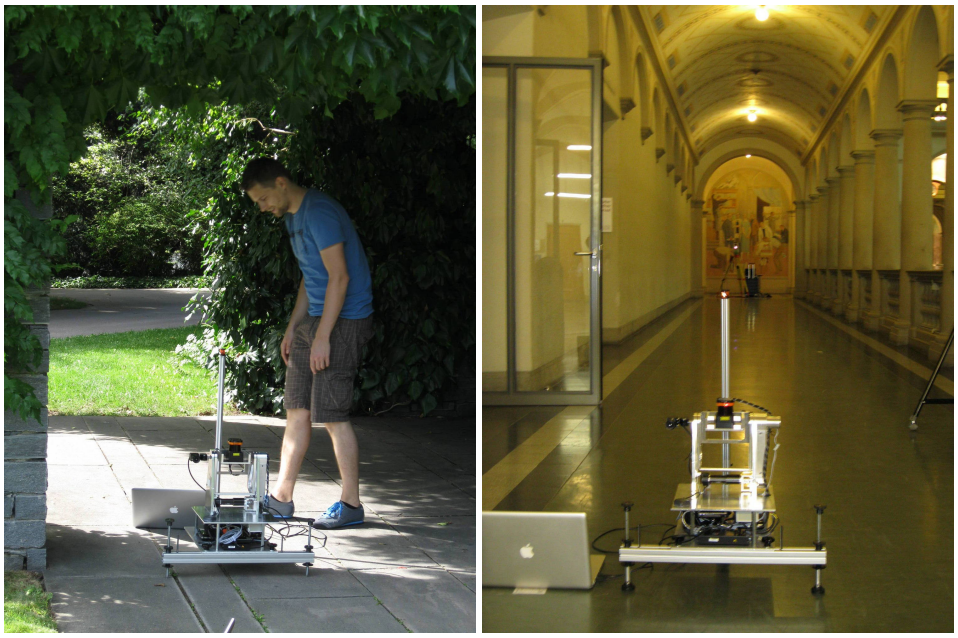


Figure 2.5 – Photographs of the locations point cloud registration datsets. *Left:* Gazebo Summer. *Right: Hauptgebaude*. Source: `https://projects.asl.ethz.ch/datasets/doku.php?id=laserregistration:laserregistration`.

way, the ICP registration is underconstrained in the x axis, which is clearly seen on the objective function. In practice, this means that ICP will have a large uncertainty in that axis. On the contrary, the *Gazebo Summer* pair is well constrained on both axes. Consequently, its objective function is a smooth descent towards a global minimum. Detecting underconstrainted directions is perhaps the most difficult part of designing an uncertainty prediction algorithm.

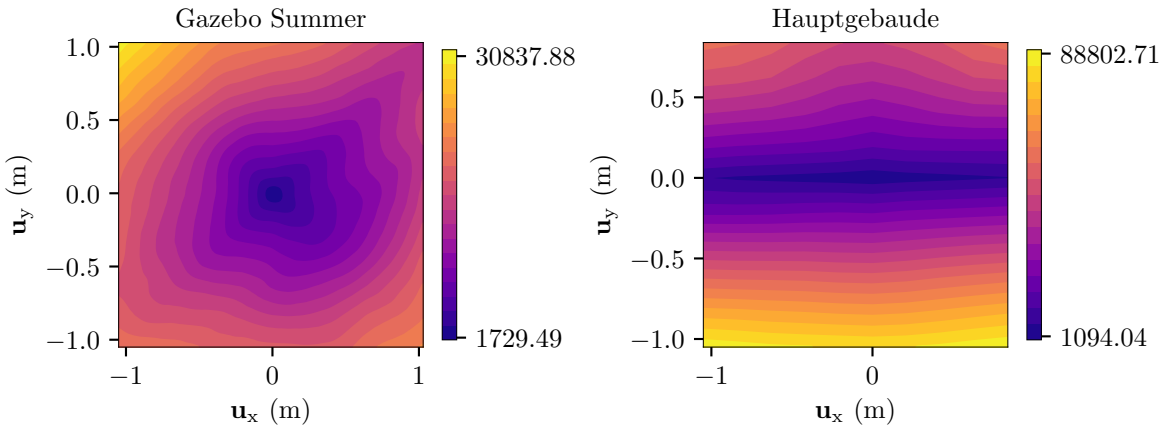


Figure 2.6 – Value of the objective function of ICP for pairs of point clouds from real datasets. The value is plotted for two point cloud pairs: on the *left* clouds 15 and 16 from *Gazebo Summer*, on the *right* clouds 15 and 16 from *Hauptgebaude*.

One particularity of the objective function of ICP is that it is made noisy by the re-association of points. Indeed, at every iteration, ICP pairs every point with their nearest neighbor. The optimal pairing changes constantly as the estimated transformation $\widehat{\boldsymbol{T}}$ is being optimized. This is visible on Figure 2.7. On it, the changes in point associations (the transition of the "plateaus" in the figure) are clearly visible. Notice the very small scale (in the order of the *micro*-meter) at which these transitions happen. It was necessary to zoom down to this level, because the optimal pairing changes very frequently when $\widehat{\boldsymbol{T}}$ changes.

We already showed the tight coupling between uncertainty and objective function. One can wonder if modelling the noisiness of Figure 2.7 is necessary to a correct estimation of the covariance. This question is explored in the inserted article.

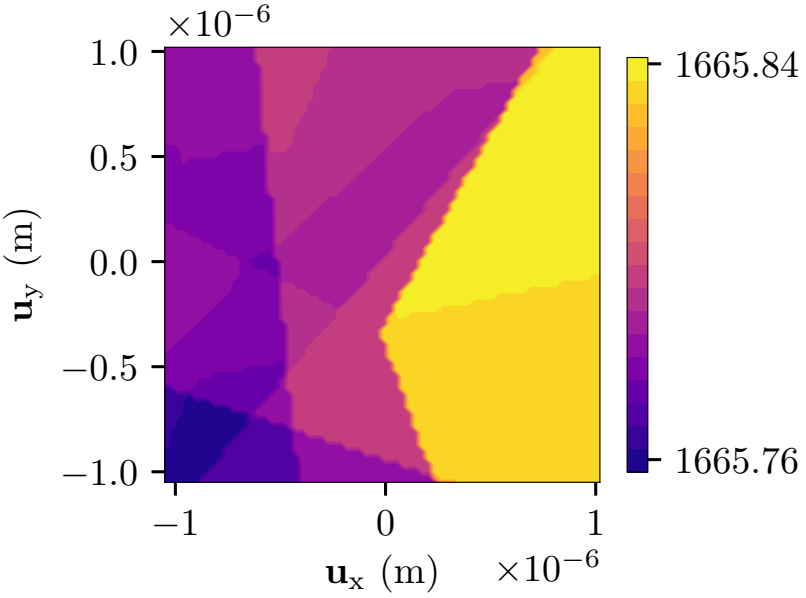


Figure 2.7 – Zoom in on the objective function of ICP for a pair from the *Plain* dataset (point clouds 15 and 16). At a small scale, the objective function is not smooth, because of the reassociation of nearest neighbors.

Chapter 3

From uncertainty to covariance

The motivation of this work is to better integrate ICP in probabilistic state estimation frameworks such as Kalman Filters or Simultaneous Localization and Mapping (SLAM). We plan to achieve this by studying ICP's uncertainty. One might wonder how uncertainty relates to *covariance*, which is in the title of this work. The purpose of this section is to pose ICP in a more formal probabilistic framework. It better identifies the random variables at stake and how they relate to the results we have seen so far. This will clarify the difference between uncertainty and covariance, and help us discuss those concepts more clearly.

3.1 ICP as a random variable

Strictly speaking, ICP is a function of a reading point cloud $\boldsymbol{Q}$, a reference point cloud $\boldsymbol{P}$, and an initial guess $\breve{\boldsymbol{T}}$ of the transformation between $\boldsymbol{P}$ and $\boldsymbol{Q}$:

$$\text{icp}(\boldsymbol{P}, \boldsymbol{Q}, \breve{\boldsymbol{T}}) = \widehat{\boldsymbol{T}}. \tag{3.1}$$

It yields an estimate $\widehat{\boldsymbol{T}} \in \text{SE}(3)$ of the rigid transformation between the frames of the point clouds. Consequently, ICP is not an inherently random process. To see the uncertainty, we must remember the experimental context. First, the point clouds themselves are uncertain: lidar sensors indeed suffer from noise. Also, the point associations are uncertain, because both scans $\boldsymbol{P}$ and $\boldsymbol{Q}$ did not sample the exact same locations of the environment. In underconstrained environments such as *Hauptgebaude* (see Figure 2.6), the objective function is not always able to steer ICP towards good point associations. Consequently, the structure of the environment is also a source of uncertainty for ICP. Finally, the initial guess $\breve{\boldsymbol{T}}$ often stems from odometry, and as such is uncertain.

We summarize all those sources of uncertainty in two distributions. First, we pose the distribution $\mathcal{O}$ such that $\check{\boldsymbol{T}} \sim \mathcal{O}$. This $\mathcal{O}$ is meant to represent the uncertainty from the odometry. Finally, we pose the uncertain output $\mathcal{I}$ of ICP such that $\widehat{\boldsymbol{T}} \sim \mathcal{I}$. If the chosen ICP toolchain is robust, and the input point clouds $\boldsymbol{P}$ and $\boldsymbol{Q}$ are easily "registerable", ICP will converge to a distribution $\mathcal{I}$ that is both accurate and precise. Otherwise the size of $\mathcal{I}$—the uncertainty—becomes larger.

The distributions $\mathcal{O}$ and $\mathcal{I}$ have shapes that vary wildly from one experiment to another. Still, an integration to known tools such as a Kalman Filter is deemed beneficial. Consequently, we must describe $\mathcal{O}$ and $\mathcal{I}$ in terms that allow for common tools to be used. This is why we pose them as normal distributions for this work, and focus our efforts towards estimating the *covariance* of $\mathcal{I}$. We described how to mathematically express normal distributions over $\mathrm{SE}(3)$ in Section 1.2.

The Gaussianity assumption is actually oversimplifying. Figure 3.1 shows the distribution $\mathcal{I}$ for two point cloud pairs. We observe results that are often clustered and non-symmetric. Whether the normal distribution supposition *can* be correctly made is left for future work. For now, we try to find normal distributions that represent $\mathcal{I}$ as well as possible.

In summary, this work considers that the uncertainty of ICP stems from various factors, such as the structure of the environment, the sensor noise and the uncertainty of the initial estimate. The distributions $\mathcal{O}$ and $\mathcal{I}$ encapsulate this uncertainty. To make our computations tractable, we make the simplifying assumption that $\mathcal{O}$ and $\mathcal{I}$ are normal distributions. This formulation is justified because it makes the integration to existing states estimation toolchains seamless.

3.2 Sampling ICP

To build an intuition of the distribution $\mathcal{I}$, we sample ICP a large number of times for various pairs of point clouds. This approach was recently applied to 3D ICP [9]. The high dimensionality of the problem (6 dimensions) forces us to have a large number of samples in order to represent the distribution of registration transformations correctly. In the larger scheme of things, sampling ICP allows us to generate training examples for data-driven algorithms. The plan here is to perform the computationally expensive sampling on some point cloud pairs, and then train a covariance estimation model that would automatically generate predictions.

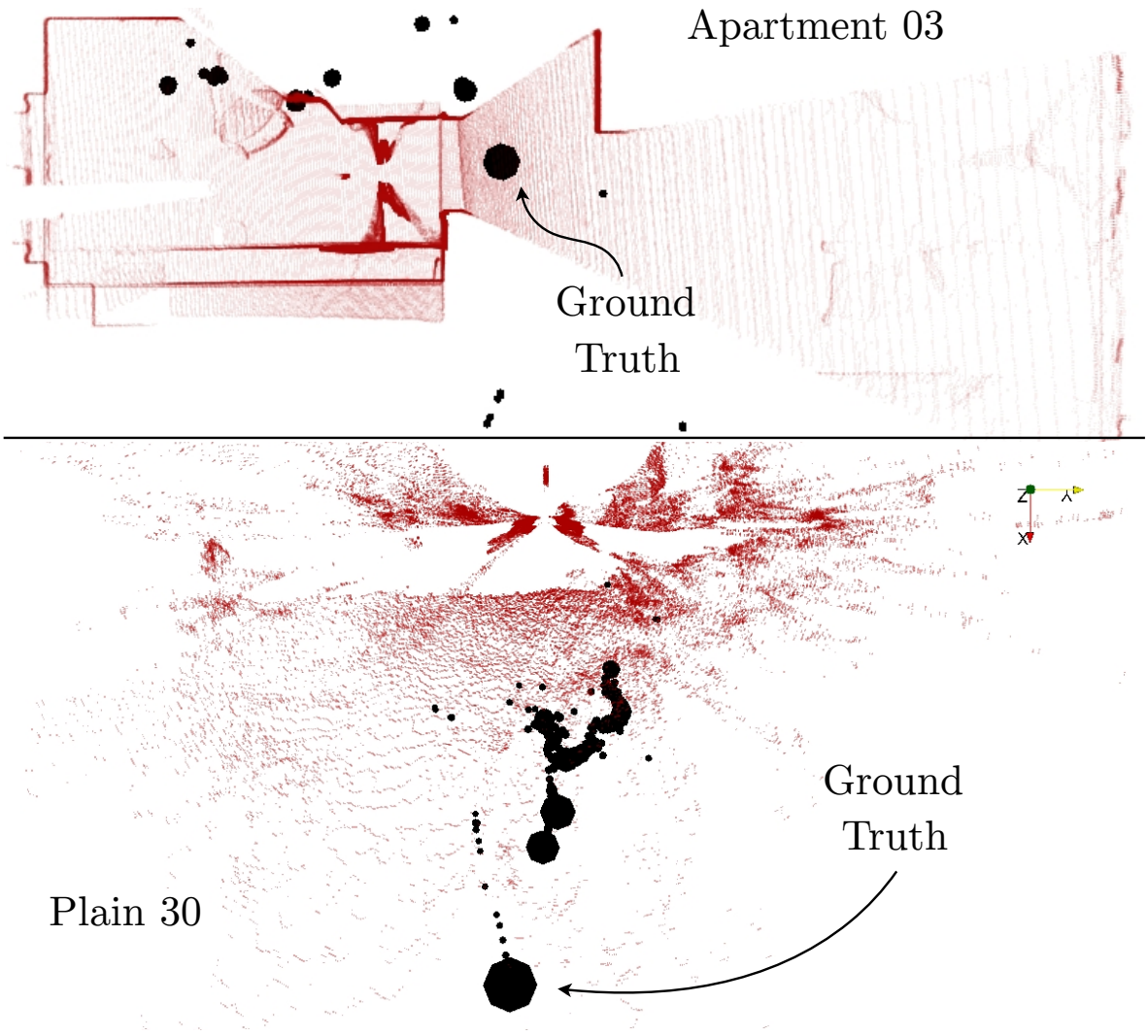


Figure 3.1 – Sampled registration transformations for the *Apartment* 03 (top) and *Plain* 30 (bottom) locations. The registered point clouds are displayed in red. The black spheres represent the density of registration transformations at that location. Once again, the $\boldsymbol{u}$ part of the full perturbations $\boldsymbol{\xi}$ is shown. To better cover $\mathrm{SE}(3)$, 5000 initial transformations were used. One can observe that the distributions are hardly normally centered on their ground truth.

We can see an example of such a sampling by looking once again at Figure 3.1. To estimate the covariance of ICP, we must filter outliers out of this sampling. This outlier filtering should not be confused with the outlier filtering performed *during* an ICP registration (see Section 2.2).

Indeed, existing covariance estimation algorithms make the assumption that ICP "*converged to the attraction region of the true solution*" [6], [17]. We know that this notion of attraction region is loosely defined, because of what we learned about the objective function in Section 2.3. There, we saw that the objective function of ICP is littered with small local minima (see Figure 2.7). Consequently, the attraction region of the true solution cannot be a clearly defined region, where the gradient constantly points towards a global minimum. We speculate that the attraction region of the global minimum intuitively corresponds to the larger scale behavior of the objective function, as seen for instance in Figure 2.6. On these objective functions, if we ignore the noise that happens at very small scales, we can intuitively identify a region that always converges towards a global minimum. We are not aware of a way to identify this region automatically. Consequently, we have to resort to heuristics to distinguish the transformations that converged inside of it from the ones that converged outside. That is, we want and heuristic to separate convergent samples from divergent samples.

We use an example pair of point clouds to illustrate the potential impact of outlier filtering. Figure 3.2 shows the distribution of ICP registration transforms when registering an *Apartment* point cloud pair. The red ellipsoid is the $\boldsymbol{\Upsilon}_{uu}$ part of the sampled covariance. It was obtained by computing Equation 1.19 directly, on all the samples. The shape of the ellipsoid seems roughly valid. However, a closer inspection reveals that 97% of the registration transformations live immediately around the ground truth. With that in mind, we may wish to consider that the results far from the large sphere are outliers. This is equivalent to considering that, for this pair of point clouds, the covariance of ICP is only the covariance of the centermost samples.

Figure 3.3 shows the distribution of registration transformations when we filter those outliers away. Notice the dramatic change in scale of the figure. Once the 3 % of outliers are filtered away, the covariance is of the order of the millimeter. Without outlier filtering, it is in the order of a meter. This example justifies the exploration of heuristics that identify the registration transformations that converged correctly.

For this work, we explored many different heuristics. We present heuristics based

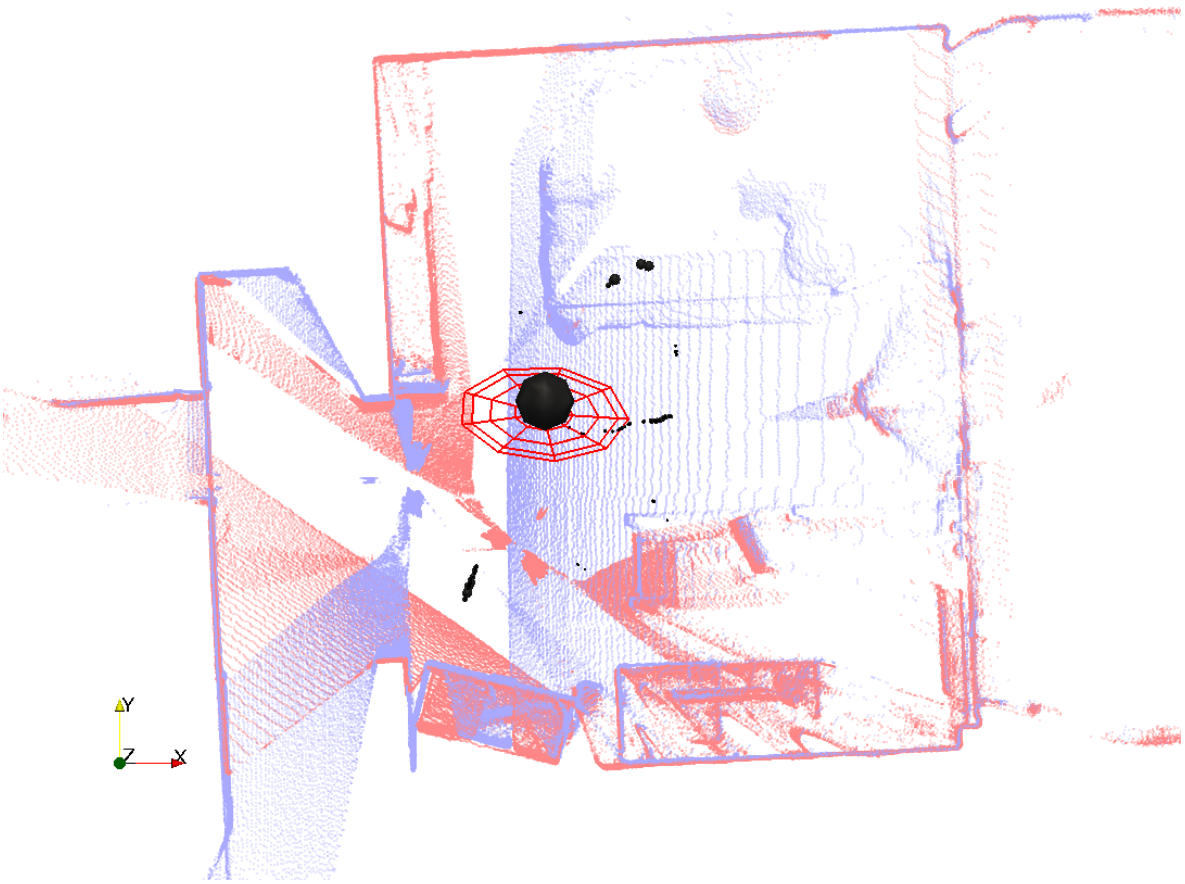


Figure 3.2 – Distribution of ICP registrations when registering a pair of point clouds from the *Apartment* dataset. The red point cloud is being registered against the blue point cloud. Simultaneously, we plot the $\boldsymbol{u}$ part of 5000 ICP registration results. The size of the black spheres indicates the density of ICP registration transformations at this location. The red ellipsoid represents the covariance of the registration transformations if no outlier filtering is applied.

on a fixed distance threshold, a density threshold, and clustering similar to Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [19]. The advantages and disadvantages of every heuristic are discussed.

3.2.1 Fixed distance threshold

A typical criterion to detect the convergence of ICP is a fixed distance threshold. To detect if a registration transformation is an outlier, we compute the perturbation

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\omega} \end{bmatrix} = \log(\bar{\boldsymbol{T}}^{-1}\hat{\boldsymbol{T}}). \tag{3.2}$$

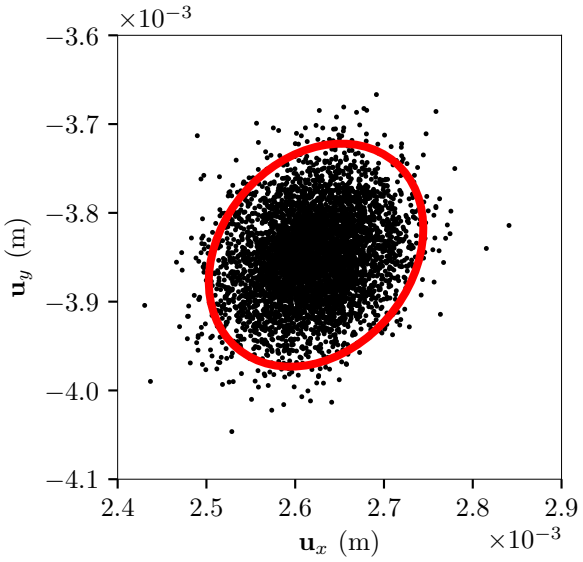


Figure 3.3 – Zoom in on the distribution of registration transformations when registering point clouds 7 and 10 from the *Apartment* dataset. The data is projected on the xy plane. Every black dot represents a registration transformation. The red ellipse is a 95 % confidence interval for the distribution. On this figure, 3 % of the data was filtered to eliminate outliers.

Then we have two options: we can filter away all the perturbations that have

$$\|\boldsymbol{\xi}\| > \tau_{\xi} \tag{3.3}$$

or filter the registration transforms that have one of the following:

$$\|\boldsymbol{u}\| > \tau_{\boldsymbol{u}} \tag{3.4}$$

$$\|\boldsymbol{\omega}\| > \tau_{\boldsymbol{\omega}}. \tag{3.5}$$

This method has the advantage of being simple and easy to compute. Its main disadvantage is that it imposes an upper bound on the covariance. Indeed, this method imposes a maximum norm for the $\boldsymbol{\xi}$ vector, which means that the covariance matrix also has a maximum norm (see Equation 1.19). The effect of this upper bound on the covariance matrix is felt especially when the registration transformations are arranged like in Figure 4.1. In this figure, the registration transformations have a very elongated distribution. If we were to consider that all the registrations with $\|\boldsymbol{u}\| > \tau_{\boldsymbol{u}}$ have diverged, we would loose much of that elongated aspect in the final covariance. This incapacity to capture elongated distribution disqualifies the fixed distance threshold as a method for filtering divergent registration transformations.

3.2.2 Density threshold

The density threshold method supposes that registration transforms converge into the attraction region of the true solution much more frequently than they diverge elsewhere. Hence, after all the iterations of ICP, there should be a greater density of registration transformations in the region of the true solution. This justifies the use of the density of points as a heuristic to detect outlier.

To detect outliers using a density threshold, we first compute a large sample of ICP registrations. For every transformation sampled, we find the k nearest neighbors. Every neighbor i has a distance

$$\delta_i = \|\boldsymbol{\xi}_i - \boldsymbol{\xi}\| \tag{3.6}$$

from the sample of interest $\boldsymbol{\xi}$. We estimate the density ρ by computing

$$\rho = \frac{k}{\max \delta_i}. \tag{3.7}$$

Afterwards, we simply filter away all the registration transformations $\boldsymbol{\xi}$ that have an estimated density ρ smaller than parameter τ_ρ.

This method is relatively easy to compute, although it is more computationally expensive than the fixed distance threshold. The computational complexity lies in the nearest neighbors search. It can be mitigated by first computing a k-d tree on the sample of registration transformations.

Since this heuristic does not depend on the distance from the ground truth, it does not impose an upper bound on the covariance of the filtered results. It has the disadvantage of having a parameter that is not intuitive to set (τ_ρ). Experimentation is required to find principled values for this parameter. Furthermore, good values for τ_ρ change with the number of samples, and the overall behavior or the registration for the pair of point clouds. Figure 3.4 illustrates this well. It plots the trace of the covariance matrix as the threshold varies. The trace of the covariance matrix is used here as a measurement of the size of the covariance. The results would be similar using other metrics, such as the Frobenius norm. On the figure, the *Apartment* has registration transformations that are very well clustered around the ground truth. Even with very aggressive filtering parameters, 97 % of the registration transformations remain in the distribution. The covariance becomes very compact (but does *not* reach zero) even though it contains the large majority of the results. In fact, the small covariance that is near zero on that figure corresponds to the cluster of results

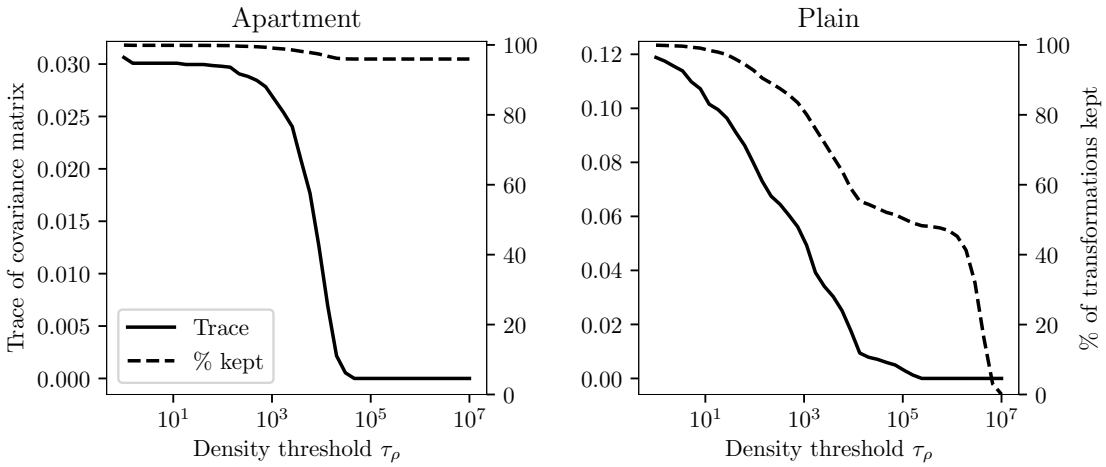


Figure 3.4 – Trace of the covariance of ICP registrations against the value of the parameter τ_ρ used to filter outliers. On the *left*, a point cloud pair (clouds 7 and 10) from the *Apartment* dataset is used. On the *right*, a point cloud pair (clouds 8 and 11) form the *Plain* dataset is used. The dashed lines are the proportion of registration results that remain after filtering.

shown in Figure 3.3. This is a situation where τ_ρ is relatively easy to choose. Any value that reaches a plateau in terms of size of covariance, but still keeps most of the registration results inside the distribution, is a good choice.

The *Plain* dataset is more challenging. The registration transformations from this sampling are more spread out. Consequently, the covariance has a larger size in the beginning. The size of the covariance diminishes progressively as the density threshold is increased, but so does the proportion of registration results that remain. The filter designer must choose a value of the parameter that filters as much outliers as possible, while keeping a healthy number of samples inside the distribution. We are looking for values that minimize the size of the covariance, but do not take it to zero. For this case, values of τ_ρ between 10^5 and 10^6 seem appropriate. Anything beyond that incurs the risk of over-filtering the data.

3.2.3 DBSCAN like clustering

The last heuristic to determine what registration transformations converged close to the ground truth is inspired by the DBSCAN clustering algorithm [19]. This algorithm progressively enlarge clusters by running nearest neighbors searches. The outlier filtering algorithm we used for this work is a simplified version of DBSCAN. Indeed, DBSCAN can accommodate an arbitrary number of clusters. However, for the current work we suppose there is only one cluster: the cluster of convergent

registration transformations. Consequently, we can simplify DBSCAN to improve performance.

Algorithm 1 describes this simplification precisely. It begins with a call to FINDSEED which initializes the border set $\mathcal{B}$. FINDSEED is a function that, given a set of registration transformations $\mathcal{S}$ and a ground truth value $\bar{\boldsymbol{T}}$, finds the registration transformation in $\mathcal{S}$ that best initializes the clustering. One potential implementation of FINDSEED is simply to find the transformation in $\mathcal{S}$ that is closest to the ground truth. Then, the algorithm iterates on the border set $\mathcal{B}$. For every $\boldsymbol{p}$ point in the border set, a nearest neighbors query is executed. If at least k neighbors are inside a radius τ_r, then the point $\boldsymbol{p}$ is considered a member of the cluster. It is added to the set $\mathcal{C}$. Its neighbors are added to $\mathcal{B}$ and will be considered next. In the end, the set $\mathcal{C}$ contains the points that are considered to have converged inside the attraction region of the ground truth.

Algorithm 1 Simplified DBSCAN

$\mathcal{B} \leftarrow \{\text{FINDSEED}(\mathcal{S}, \bar{\boldsymbol{T}})\}$ ▷ $\mathcal{B}$ the border set
$\mathcal{C} \leftarrow \{\}$ ▷ $\mathcal{C}$ the set of points inside the cluster
while $|\mathcal{B}| > 0$ **do**
  $\boldsymbol{p} \leftarrow \text{POP}(\mathcal{B})$
  nns, distances $\leftarrow \text{KNN}(\boldsymbol{p}, \mathcal{S}, k)$
  if MAX(distances) $< \tau_r$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{p}\}$ ▷ $\boldsymbol{p}$ is a core point
    $\mathcal{B} \leftarrow \mathcal{B} \cup (\text{nns} \setminus \text{nns} \cap \mathcal{C})$ ▷ Add potentially new core points in $\mathcal{B}$
  end if
end while
return $\mathcal{C}$

This algorithm needs two parameters to function: the number of neighbors k and the maximum radius τ_r. Schubert *et al.* [19] present a rule of thumb where k should be about two times the dimensionality of the problem, and then τ_r should be adjusted experimentally. Figure 3.5 shows how the covariance varies when we manipulate τ_r. The point cloud pair from the *Apartment* dataset is almost insensitive to this parameter. A very wide range of values causes it to keep about 97 % of the data, and have a rather small covariance. The *Plain* dataset, once again, is more challenging. Increasing τ_r causes nearly discrete jumps in the size of the covariance. Every discrete jump is explained by a cluster of registration transformations being included in the distribution of convergent transformations. Consequently, appropriate values of the parameter are values that contain only one cluster: the cluster that is closest

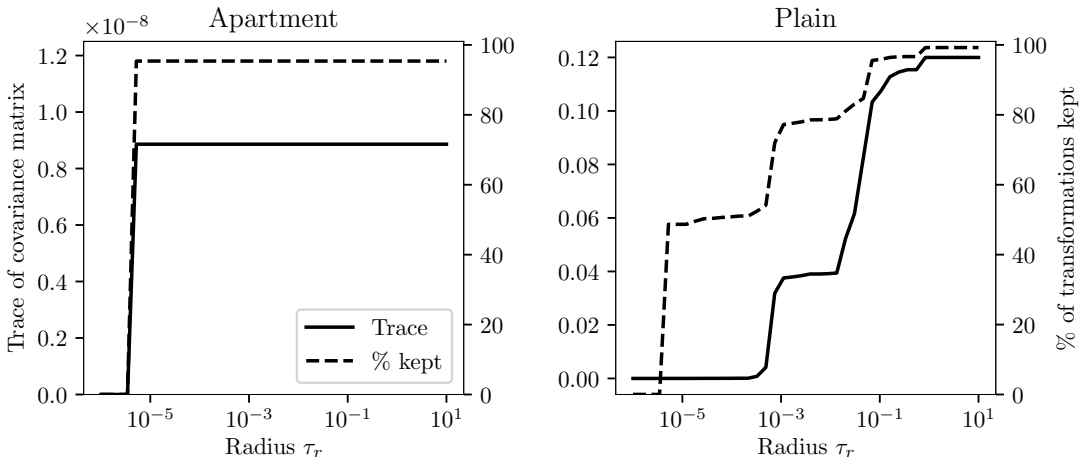


Figure 3.5 – Trace of the covariance of ICP registration transformations when the DBSCAN clustering radius τ_r varies. The covariances are computed using point clouds from the *Apartment* (7-10) and *Plain* (8-11) datasets.

to ground truth. In other words, τ_r should be as small as possible while including a healthy proportion of the registration transformations inside the distribution of convergent transformations. In that spirit, values of τ_r between 10^{-5} and 10^{-4} seem appropriate for that pair of point clouds.

This clustering algorithm similar to DBSCAN is a good heuristic to detect convergent registration transformations. Like the density threshold filter, it does not impose an upper bound to the covariance. Furthermore, it corresponds the intuition we have about the nature of the attraction region of the ground truth. We think of it as a region in the objective function where the gradient steadily descends towards the ground truth. Consequently, we expect that all the convergent registration transformations will be relatively close to one another. We expect the presence of gaps between the clusters of results associated to the various global minima. The DBSCAN heuristic algorithm relies on the presence of these gaps to work properly.

As of now, we consider that the DBSCAN based heuristic is the best way to filter away divergent transformations from a sample of ICP registrations. The fixed threshold is disqualified, because it imposes an upper bound on the norm of the covariance. Furthermore, DBSCAN is better than the density filtering in that it is better able to use the spatial relationship between registration transformations in the sample. One good illustration of the superiority of the DBSCAN like filtering strategy over the density threshold filtering is seen in the lower right part of Figure 4.4.

There are two clusters in those registration transformations: the inner line and the outer ring. Both clusters have a good density: it would be difficult to discriminate between those two using a density based filtering. However, the presence of a large gap between both clusters makes it easy for our DBSCAN clustering to separate the middle line (which is convergent) from the outer ring (which is divergent).

With such a way to filter away divergent registrations, we can now compute a covariance from a sample of registration transformations. We now have all the tools we need to work on a data-driven covariance prediction algorithm.

Chapter 4

CELLO-3D: Estimating the Covariance of ICP in the Real World

Résumé

La fusion de recalages d'ICP avec les outils existants d'estimation d'état repose sur une estimation précise de leur intertitude. Cet article étudie l'estimation de cette incertitude sous la forme d'une covariance. Premièrement, nous étudions les limites de méthodes d'estimation de covariance existantes sur des jeux de données 3D. Ensuite, nous estimons la covariance d'ICP avec une approche basée sur les données, utilisant plus de 5 100 000 recalages calculés sur 1020 paires de nuages de points 3D tirés d'environnements réels. Nous évaluons la solution sur une large collection d'environnements, allant de structurés à non-structurés et intérieurs à extérieurs. La capacité de notre algorithme à prédire des covariances est évaluée, ainsi que l'utilité des prédictions pour faire l'estimation de l'incertitude sur des trajectoires. La méthode proposée fait des prédictions qui sont plus précises que les solutions analytiques existantes. Ces prédictions sont consistantes avec des trajectoires observées expérimentalement.

Abstract

The fusion of ICP registrations in existing state estimation frameworks relies on an accurate estimation of their uncertainty. In this paper, we study the estimation of this uncertainty in the form of a covariance. First, we scrutinize the limitations of existing closed-form covariance estimation algorithms over 3D datasets. Then, we set out to estimate the covariance of ICP registrations through a data-driven approach, with over 5 100 000 registrations on 1020 pairs from real 3D point clouds. We assess our solution upon a wide spectrum of environments, ranging from structured to unstructured and indoor to outdoor. The capacity of our algorithm to predict covariances is accurately assessed, as well as the usefulness of these estimations for uncertainty estimation over trajectories. The proposed method estimates covariances better than existing closed-form solutions, and makes predictions that are consistent with observed trajectories.

4.1 Introduction

The ICP algorithm [2], [3] is ubiquitous in mobile robotics for the tasks of localization and mapping. It estimates the rigid transformation between the reference frames of two point clouds, by iteratively pairing closest points in both point clouds and minimizing a distance between those pairs. This is equivalent to optimizing an objective function that maps rigid transformations to a scalar optimization score for a pair of point clouds. There is an abundance of ICP variants [4], each of which yields slightly different transformations due to their different objective functions. One notable variation is the choice of error metric between each pair of points, where common choices of metric are point-to-point [2] and point-to-plane [3]. The registration process is subject to a number of sources of uncertainty and error, because of a bad adequation between the objective function and the desired result. Chief among them is the presence of local minima in the objective function. Other causes of uncertainty comprise noise from the range sensor, and underconstrained environments such as featureless hallways [17].

The fusion of ICP measurements in existing state estimation frameworks (e.g. SLAM) relies on an appropriate estimation of the uncertainty of ICP, expressed as a covariance [14]. In this context, ICP is modeled as a function of input point clouds and an initial estimate which yields a registration transformation that is normally distributed. Optimistic covariance estimates can lead to inconsistency and navigation failures, whereas pessimistic ones inhibit efficient state estimation. Figure 4.1 illustrates the process of estimating the covariance of a registration. The registration process shown takes place in a hallway, and is consequently loosely constrained in one axis. In this figure, optimistic covariance estimation frameworks might miss this underconstrainedness and encompass only the central samples.

This paper focuses on the problem of estimating the covariance of ICP in such real 3D environments. To this effect, we first provide an experimental explanation as to why current covariance estimation algorithms may perform poorly in that context. Furthermore, we present CELLO-3D, a data-driven approach to estimating the uncertainty of 3D ICP that works with any error metric.

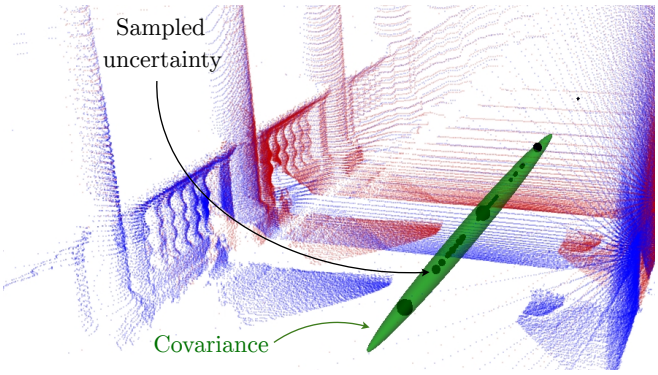


Figure 4.1 – A reading point cloud (red) is being registered against a reference point cloud (blue). This paper studies the estimation of covariances like the green ellipsoid to allow integration of ICP in state estimation toolchains. The result of ICP was sampled, and the black balls indicate the density of registration transformations at a particular location. There are three larger clusters which correspond to local minimas in the objective function caused by the regularly spaced pillars. We project the translation part of a covariance in $\mathbb{R}^3$ for illustration purposes, but in general the covariances of a 6 degrees of freedom phenomenon is studied here.

4.2 Related works

There are many approaches to estimating the covariance of the ICP algorithm, each of which must balance quality of prediction and computation time. On one end of the spectrum, Monte-Carlo (also called brute force) algorithms such as [9], [20] provide an accurate estimate ICP's covariance. They consist in sampling a large number of ICP registration transformations, and using the covariance of the sampled results as the covariance estimation. If a model of the environment is available, brute-force algorithms can take sensor noise into account by simulating many scans of the environment given some noise model. However, Monte-Carlo algorithms cannot be used online due to their high computational cost. This limits their practicality in mobile robotics applications.

Another important category of covariance estimation algorithms rely on the objective function's Hessian [17], [20]–[24]. These closed-form methods are motivated by the need for a covariance estimation that can be used online. Their underlying assumption is that the objective function $J(\boldsymbol{T})$ used in ICP can be linearized around the point of convergence. This allows the use of linear regression theory to derive a covariance from the Hessian of J. If J is analytically differentiable, then the Hessian can be computed directly [20], [23]. Otherwise, it can also be approximated numerically by sampling [21], [22]. This approach accounts for errors that are due to the

environment structure, but not for errors due to sensor noise. However, modelling the effect of sensor noise on the objective function J proves to be crucial for covariance estimation algorithm. Censi [17] addresses this by using the implicit function theorem. His model of the covariance contains the Hessian of J, but also the effect of the sensor noise on it. It is successfully evaluated on 2D datasets. The equations for the 3D case are derived in [24], but Censi's algorithm is considered to be largely optimistic in that situation [10]. More elaborate noise models alleviate this difficulty, but only for specific sensors [8].

Closed-form approaches have the shortcoming of not taking point reassociation into account. Therefore, they must assume that 1) ICP converged to a loosely defined region of attraction of the "true" solution [17], and 2) the reassociation of points that occur in that region have a negligible influence on the objective function. Bonnabel *et al.* [7] show that for point-to-point ICP variants, the second assumption is broken. However, they present a proof that Censi's method is accurate using the point-to-plane ICP variant in a noiseless context. They do so by demonstrating that changes in J due to point reassociations are small enough for the covariance estimation methods to remain valid under certain conditions. In spite of this, Mendes *et al.* [10] indicates that this covariance is still optimistic in a noisy experimental context.

A data-driven alternative to covariance estimation emerged from the Covariance Estimation and Learning through Likelihood Optimization (CELLO) framework [6]. It is a general covariance estimation strategy that projects point clouds in a descriptor space, then estimate the covariance within this space. It uses a machine learning algorithm that first estimates a distance metric between the predictors, and then uses this metric to weigh the learning examples during online inference. This procedure can be done with ground-truth data [6], but also without it [25] exploiting expectation-maximization. In the presence of ground-truth data, expectation-maximization should be avoided to simplify the machine learning process. The general strategy of CELLO was successfully applied to the estimation of the reliability of visual features for visual-inertial navigation [26], [27]. Peretroukhin *et al.* [27] used the prediction space to generate a noise model for visual landmarks, which in turn was used to predict the ego-motion of the sensor. For an application to 3D ICP, these data-driven approaches are challenging in that extracting relevant features from 3D point clouds is still an open problem. Hand-crafted feature designers must tread carefully between the expressiveness and the generality of the descriptor for this approach to be viable. Consequently, this method was never assessed in a 3D ICP

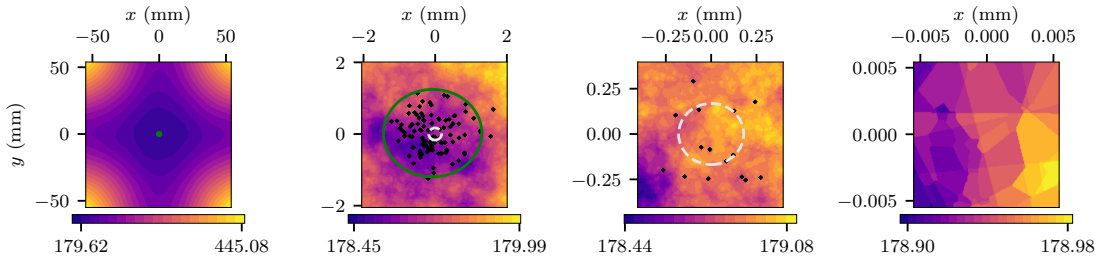


Figure 4.2 – The objective function of point-to-plane ICP around the ground truth when registering a simple cube. The black dots are sampled registration transformations, with their associated translation projected on the xy plane. The green circle is a 3σ covariance ellipse of the distribution of registration transformations. The white dashed circle is a similar representation, but of Censi's covariance estimate. *Rightmost frame*: the transitions from one plateau to another are explained by point reassociations. At a scale smaller than the scale of the covariance of the registration transformations, the error landscape is dominated by point reassociations.

context to the best of our knowledge. Finally, Liu *et al.* [28] circumvent the explicit design of features by training a deep neural network directly on the sensor data. Their method successfully estimates the covariance of 2D visual odometry. Unfortunately, the use of deep neural networks on 3D point clouds is still an active research area, and further work is required to apply this method to such data.

4.3 Shortcomings of closed-form covariance estimation algorithm

As discussed earlier, Bonnabel *et al.* [7] point out that closed-form covariance estimation methods are potentially ill-founded if point reassociations occur at a scale that is smaller than that of the covariance to be estimated. Our own analysis on 3D simulated data shows that it is likely that point reassociations happen at a scale this small. For example, Figure 4.2 shows the objective function $J(\boldsymbol{T})$ observed when registering a pair of $1 \times 1 \times 1$ m cube shaped point clouds. The points lie on the surface of the cube and have a $\sigma = 0.01$ m noise applied on them on every axis. At a larger scale, this objective function $J(\boldsymbol{T})$ corresponds to our intuition, with a seemingly-smooth slope towards large global minimum. At a smaller scale, however, it is composed of a large number of "plateaus", each of them corresponding to one fixed association of points between the reading and the reference. This litters the objective function with local minima to which ICP is sensitive. In turn, a larger covariance of ICP results is observed.

There is a mathematical explanation for the optimism of Censi's algorithm for the point-to-point variant of ICP in Bonnabel *et al.* [7]. We do not know of such a proof the point-to-plane case. Furthermore, the results in Bonnabel *et al.* [7] were encouraging about the validity of Censi's algorithm in the point-to-plane case. They show that the change in the objective function provoked by point reassociations is bounded under certain conditions, proving the correctness of Censi's estimate. On the contrary, our own experiments show that caution is required even in the point-to-plane case. Indeed, Figure 4.3 shows that sensor noise significantly impacts the empirically sampled covariance of ICP as it grows. On the other hand, Censi's covariance estimate grows slowly as the estimate of the sensor noise grows. To use Censi's estimate in that experimental context, we would need to inflate our estimation of the sensor noise to values that are beyond a meaningful range.

Consequently, we argue that a viable covariance estimation algorithm for ICP should take into account the effect of noise. More precisely, it should model both the direct effect of the sensor noise on the objective function of ICP, but also the point reassociations that it provokes around ground truth. Monte-Carlo based approaches circumvent those difficulties by incorporating the effect of noise directly. The complexity we observe in the 3D registration process motivates our shift from analytical to data-driven solutions. Our general approach is to implement the CELLO framework [6] for 3D ICP and work towards learning covariance models from training data generated through a sampling process. We aim at getting the best of both worlds: the accuracy of brute-force methods and the rapid inference of machine learning approaches.

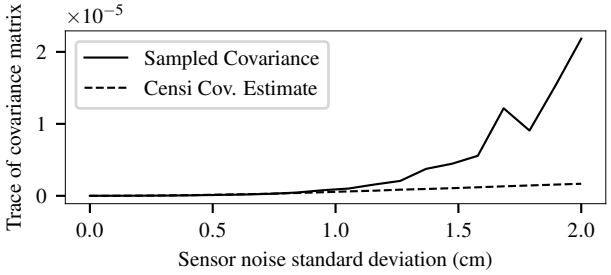


Figure 4.3 – Trace of ICP covariance estimation against sensor noise. Censi's covariance estimate increases slowly as the sensor noise model grows. The sampled covariance of ICP increases dramatically with sensor noise, even on simulated datasets. This is attributed to the point reassociation provoked by sensor noise.

4.4 3D covariance estimation of ICP

Casting CELLO onto a 3D registration problem requires a primer on notations. A rigid transformation ${}^a_b\boldsymbol{T} \in \mathrm{SE}(3)$ allows to express a point cloud ${}^b\boldsymbol{P} \in \mathbb{R}^{3\times n}$ in the coordinate system b in a second coordinate system a. Using the Lie algebra, we can express the matrix $\boldsymbol{T}$ as a vector $\boldsymbol{\xi} \in \mathfrak{se}(3)$ using $\log(\boldsymbol{T})$ and reverse the process using $\exp(\boldsymbol{\xi})$. The vector $\boldsymbol{\xi}$ is split into a translation $\boldsymbol{u} \in \mathbb{R}^3$ and an angle-axis rotation $\boldsymbol{\omega} \in \mathbb{R}^3$. This allows us us to express the uncertainty on a rigid transformation as a covariance matrix $\boldsymbol{\Upsilon} \in \mathbb{R}^{6\times 6}$, such that

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\omega} \end{bmatrix} \tag{4.1}$$

and

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\Upsilon}_{uu} & \boldsymbol{\Upsilon}_{u\omega} \\ \boldsymbol{\Upsilon}_{\omega u} & \boldsymbol{\Upsilon}_{\omega\omega} \end{bmatrix}. \tag{4.2}$$

Generally speaking, we need to rely on prior information, for which we use the notation $\widetilde{(\cdot)}$, to produce an estimate $\widehat{(\cdot)}$ of a true quantity $\bar{(\cdot)}$. For example, having access to a reference point cloud $\boldsymbol{Q} \in \mathbb{R}^{3\times m}$, it is possible to produce a transformation estimate $\widehat{\boldsymbol{T}}$ that reduces the alignment error between $\boldsymbol{P}$ and $\boldsymbol{Q}$ by relying on a prior transformation $\breve{\boldsymbol{T}}$ (see Figure 4.4). A typical solution to this registration problem is the ICP algorithm:

$$\widehat{\boldsymbol{T}} = \mathrm{icp}(\boldsymbol{P}, \boldsymbol{Q}, \breve{\boldsymbol{T}}). \tag{4.3}$$

The initial transformation $\breve{\boldsymbol{T}}$ can be seen as an element randomly selected from a distribution of transformations $\mathcal{O}$, for which the shape typically depend on odometry computation. Similarly, the estimated value $\widehat{\boldsymbol{T}}$ comes from a distribution of transformations $\mathcal{I}$ which has a complex shape that depends on both point clouds, $\mathcal{O}$ and the configuration of $\mathrm{icp}(\cdot)$. Estimating the covariance $\boldsymbol{\Upsilon}$ of ICP corresponds to making the (oversimplifying but tractable) assumption that the $\mathcal{I}$ is normally distributed

$$\mathcal{I} \approx \mathcal{N}(\bar{\boldsymbol{T}}, \boldsymbol{\Upsilon}), \tag{4.4}$$

where the right hand side is shorthand for

$$\breve{\boldsymbol{T}} = \exp(\boldsymbol{\xi})\bar{\boldsymbol{T}} \tag{4.5}$$

with

$$\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Upsilon}). \tag{4.6}$$

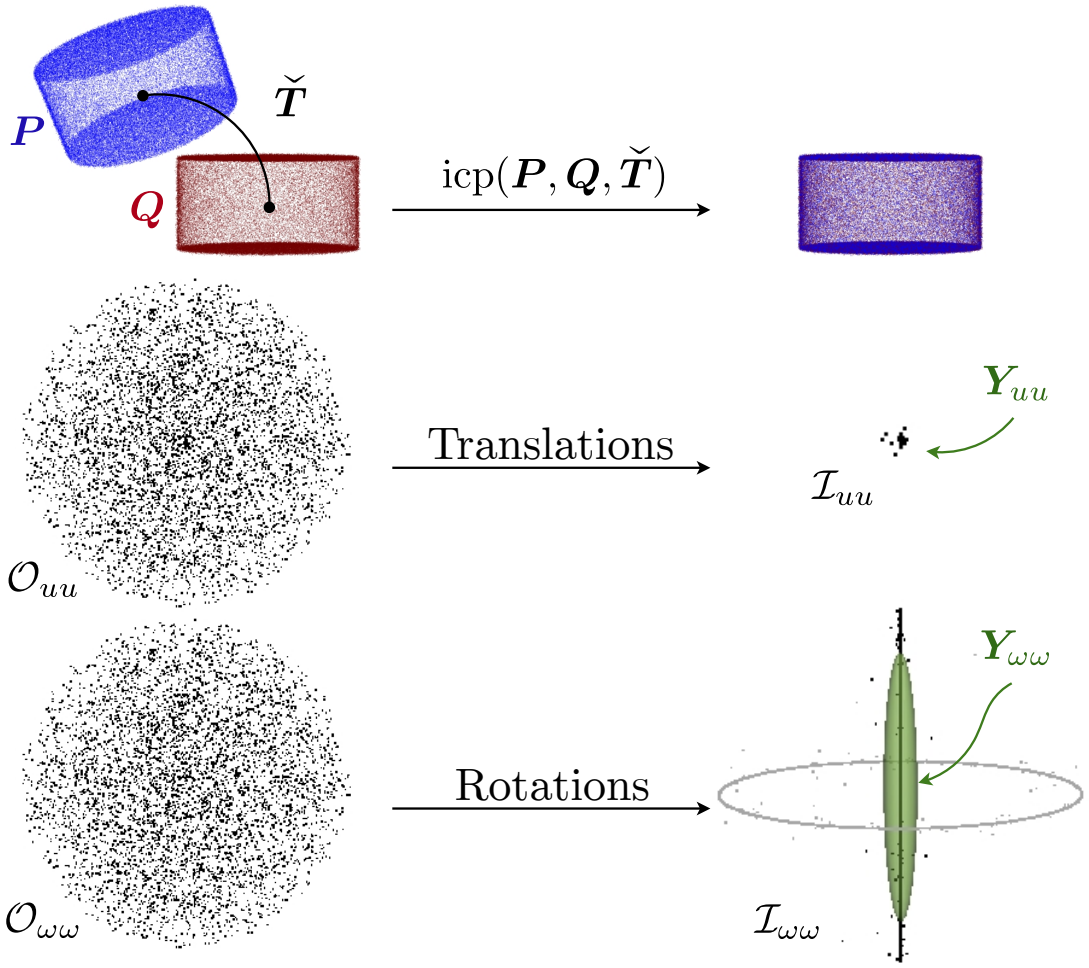


Figure 4.4 – Overview of different variables used to estimate the covariance of ICP. *Top row:* A single example of two cylinders being registered together using ICP and converging to well aligned point clouds $\boldsymbol{P}$ (blue) and $\boldsymbol{Q}$ (red). *Middle row:* A view of the translation components $\boldsymbol{u}$ of a set of initiation transformation $\mathcal{O}$ before ICP (left) and after ICP (right). *Bottom row:* Same view, but for the rotational components $\boldsymbol{\omega}$. For the rotations, the spread along the vertical axis is explained by the cylinder being unconstrained around one axis. The presence of points in the outer ring is explained by $\boldsymbol{P}$ first turning upside down and *then* being unconstrained around the same axis.

Note the use of $\bar{\boldsymbol{T}}$ in Equation 4.4, which supposes that ICP is unbiased. Figure 4.4 shows an example of those symbols, with two cylinders being registered starting with a wrongful initial alignment $\check{\boldsymbol{T}}$. The distribution $\mathcal{O}$ was manually configured and sampled to feed multiple $\check{\boldsymbol{T}}$ to ICP. The 5000 resulting transformations $\widehat{\boldsymbol{T}}$ form a distribution with covariance $\boldsymbol{Y}$ that approximates $\mathcal{I}$. The covariance approximates the translations well, but some points in rotations were considered divergent and filtered away.

4.4.1 Covariance prediction

Vega-Brown *et al.* [6] propose a data-driven framework for covariance estimation, in which the estimation function $\widehat{\boldsymbol{F}}$ is posed as a weighted average of training examples. The first step in such an approach is to collect a training dataset $\mathcal{D} = \{(\boldsymbol{d}_0, \boldsymbol{Y}_0), ..., (\boldsymbol{d}_n, \boldsymbol{Y}_n)\}$ composed of point cloud descriptors $\boldsymbol{d}_k$ and sampled covariances $\boldsymbol{Y}_k$. Our formulation differs from the original CELLO framework, which has error vectors in the place of the covariance matrices $\boldsymbol{Y}_k$.[1] This was rendered necessary due to the limited availability of 3D point cloud pairs with associated ground truth: it allows us to extract more knowledge from existing point cloud pairs.

The descriptors $\boldsymbol{d}_k$ are computed by a function $\boldsymbol{g}({}^a\boldsymbol{P}, {}^a_b\boldsymbol{T}\,{}^b\boldsymbol{Q})$ that extract features from a registered point cloud pair. One can then predict a covariance $\widehat{\boldsymbol{F}}(\boldsymbol{d})$ for an unseen example $\boldsymbol{d}$ using

$$\widehat{\boldsymbol{F}}(\boldsymbol{d}) = \frac{1}{\sum_k s\Big(\rho(\boldsymbol{d}, \boldsymbol{d}_k)\Big)} \sum_k s\Big(\rho(\boldsymbol{d}, \boldsymbol{d}_k)\Big) \boldsymbol{Y}_k. \tag{4.7}$$

The function ρ is a distance between a pair of point cloud descriptors which is defined as

$$\rho(\boldsymbol{d}, \boldsymbol{d}') = (\boldsymbol{d} - \boldsymbol{d}')^\top \boldsymbol{\Theta}^\top \boldsymbol{\Theta} (\boldsymbol{d} - \boldsymbol{d}') \tag{4.8}$$

where $\boldsymbol{\Theta}$ is an upper triangular matrix. The weighing function $s(x) = e^{-x}$ is chosen here, but any decreasing positive function is appropriate [6]. Large datasets could motivate a choice of s that completely ignores examples with large distances to make runtime predictions more efficient [27]. For the training of $\boldsymbol{\Theta}$, the loss for an individual covariance prediction $\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)$ is

$$\mathcal{L}(\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)|\boldsymbol{\Theta}) = \det\Big(\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)\Big) + \operatorname{tr}\Big(\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\boldsymbol{Y}_k\Big), \tag{4.9}$$

along with a regularization term [6].

4.4.2 3D point cloud descriptor

It is important for a point cloud descriptor $\boldsymbol{d}_k$ to contain relevant information to the prediction of the covariance for $\boldsymbol{P}$, $\boldsymbol{Q}$, while being small enough to be amenable to machine learning algorithms. Thus, the descriptor extraction function $\boldsymbol{g}(\cdot)$ should capture the geometry of the scene, in a way that translates our assumption that this

[1] Both formulations are mathematically equivalent: see Appendix B.

geometry is an important factor of the covariance. One consequence of this is that the extracted descriptors should *not* be rotation invariant, so as to capture the correct orientation of the covariance. For instance, the covariance of ICP in a featureless hallway should be aligned with its walls, as depicted in Figure 4.1. There is a wide variety of descriptors that could be used for 3D point clouds [29], but a more thorough evaluation of the existing descriptors is a question that is left for future work.

ICP—at least in its robust version—acts mainly on the overlapping region of point clouds. Consequently, descriptors should only be extracted from this overlapping region. In our approach, g first extracts a single point cloud $\boldsymbol{S}$ containing the subset of points from both $\boldsymbol{P}$ and $\boldsymbol{Q}$ that are overlapping, after registration. Then, descriptors are extracted from $\boldsymbol{S}$. The extraction pipeline separates the space into a fixed-size grid, from which all local descriptors are extracted.

First, two descriptors capture the overall "planarity" p and "cylindricality" c of the entire voxel, from the average values of these two metrics defined in [30] computed at each point within the voxel. Then, the orientation of the estimated surface normals vectors for every point in a voxel are summarized in a 9-histogram $\{h_1, ..., h_9\}$ [31]. The local descriptor of voxel i, j, k is $\boldsymbol{v}_{ijk} = \{p, c, h_1, ..., h_9\}$. The full descriptor $\boldsymbol{d}$ for the overlaping point cloud $\boldsymbol{S}$ is the concatenation of the local descriptors for every voxel. This way, our descriptor preserves a certain amount of global information, namely the spatial distribution of the local features.

4.4.3 Covariance sampling

We employ a brute-force approach similar to [9] to estimate the covariance of ICP for training. We sample the result of ICP for every point cloud pair in our dataset. Every sample uses an initial estimate $\breve{\boldsymbol{T}}$ drawn from $\mathcal{O}$. The distribution of results is then used to compute a sampled covariance $\boldsymbol{\Upsilon}_k$ for the point cloud pairs through

$$\boldsymbol{\Upsilon}_k = \frac{1}{n-1} \sum_i \boldsymbol{\xi}_i \boldsymbol{\xi}_i^\top \tag{4.10}$$

with $\boldsymbol{\xi}_i = \log(\bar{\boldsymbol{T}}_k^{-1} \hat{\boldsymbol{T}}_i)$ the n perturbations of the sampled transformations. In some cases, like in the lower row of Figure 4.4, ICP converges to many clusters. If we suppose that we estimate the covariance of ICP *when it converges*, it is ultimately up to the designer to decide what converged from what did not. We use the DBSCAN clustering algorithm [19] on the set of $\boldsymbol{\xi}_i$, and keep only the points in the cluster which is closest to ground truth. This filtering method has the benefit of avoiding unrepre-

sentative sampled covariances $\boldsymbol{Y}$ in the training dataset. It does so *without* imposing an upper bound to the covariance of the samples. One more look at Figure 4.4 illustrates the results of this procedure. The central line of samples is explained by the fact that rotations around the z axis are not constrained on this cylinder. The outer ring corresponds to situations where ICP converged upside-down, and then spun freely around the z axis. Our filtering strategy is able to remove results that converged incorrectly (i.e., the outer ring) while capturing the information about an underconstrained axis (i.e., the central line).

4.5 Experiments

4.5.1 Training datasets

To be realistic, we used datasets that are representative of a wide variety of environments that a mobile robot can encounter. Consequently, we used a subset of the *Challenging data sets for point cloud registration algorithms* [18]. It comprises point clouds taken in environments ranging from structured to unstructured, and indoor to outdoor. Every *Challenging* dataset contains a sequence of l point clouds $\boldsymbol{P}_i$ as well as ground truth positions $\bar{\boldsymbol{T}}_i$ for them. To generate a learning dataset $\mathcal{D} = \{(\boldsymbol{d}_0, \boldsymbol{Y}_0), ..., (\boldsymbol{d}_n, \boldsymbol{Y}_n)\}$, we considered pairs of point clouds $\boldsymbol{P}_i, \boldsymbol{P}_j$ for all i, j such that $i < j < l$ and $j - i \leqslant 4$.

The descriptors $\boldsymbol{d}_k$ were generated using $\boldsymbol{g}(\cdot)$. We used a grid of $4 \times 4 \times 4$ spanning 25 m in the x and y axes (parallel to the ground plane) and 10 m in the z axis (perpendicular to the ground plane) [29]. This grid was chosen because it encapsulates the typical spatial extent of a point cloud from the *Challenging* datasets.

Each sampled covariance $\boldsymbol{Y}_k$ was computed from 5000 registrations. Every registration had a $\mathcal{O}$ that was centered at the ground truth and a covariance of $a\boldsymbol{I}$. We set $a = 0.05$ to simulate a reasonable odometry scenario. A typical ICP registration pipeline was used, featuring the point-to-plane error metric. Table 4.1 describes the full registration pipeline, in terms of the framework layed down in [32]. As discussed in Section 4.4.3, the outliers were filtered from the registration transformations to enforce the assumption that ICP converged. Point cloud pairs where ICP failed consistently were removed from the training dataset. To do so we identified the pairs where ICP converged on average more than 1m or 1rad away from ground truth. In total, this work uses the data from about 5 100 000 registrations on 1020

Table 4.1 – ICP pipeline used for the sampled covariance computation

Pipeline Element	Configuration
Point cloud filters	Maximum density, random subsampling
Point matcher	k-d tree, 3 nearest neighbors
Outlier filter	Trimmed distance (keep the closest 70% of associations)
Error minimizer	point-to-plane
Transformation checkers	Max. 80 iterations

pairs. These registrations were performed on Compute Canada's computing clusters using a total of approximately 5 CPU-Years.

Data augmentation was performed to extract more knowledge from the computationally expensive sampling of the $\boldsymbol{Y}_k$. By applying a transformation $\boldsymbol{T}$ on the point clouds before sending them in $\boldsymbol{g}(\cdot)$, we obtained new descriptors $\boldsymbol{d}_{\text{aug}}$. The sampled covariances were transformed similarily using the adjoint representation of $\boldsymbol{T}$ such that $\boldsymbol{Y}_{\text{aug}} = \text{Adj}_{\boldsymbol{T}} \cdot \boldsymbol{Y}_k \cdot \text{Adj}_{\boldsymbol{T}}^{\top}$. For our application we chose to perform data augmentation only by rotating the frames of reference around the z axis of the reference point cloud. The z axis was chosen to preserve the 2.5D aspect of the dataset, while giving some rotation invariance to our algorithm. Other rotations were expected to create examples that are not close to the registration pairs encountered online, such as examples where the trees of a forest are sideways. A 6-DOF robot would warrant a more complete data augmentation.

4.5.2 ICP trajectories computation

Once the models were trained, we evaluated our covariance estimation algorithm for state estimation on indoor and outdoor trajectories. In that spirit, we computed an ICP odometry from our trajectory datasets. Since the point cloud pairs are in a sequence of length l, we compounded their ICP registration transformations using

$$\widehat{\boldsymbol{T}}_{\text{F}} = {}_1^0\widehat{\boldsymbol{T}}\,{}_2^1\widehat{\boldsymbol{T}} \ldots {}_l^{l-1}\widehat{\boldsymbol{T}} \tag{4.11}$$

where $\widehat{\boldsymbol{T}}_{\text{F}}$ is the final pose estimate of the odometry. The distribution of initial transformations was $\mathcal{O} \approx \mathcal{N}({}_{i+1}^{i}\bar{\boldsymbol{T}}, a\boldsymbol{I})$ with $a = 0.05$. In a second step, we computed covariance estimation $\widehat{\boldsymbol{Y}}_k$ for each $\widehat{\boldsymbol{T}}_k$ using a covariance estimation model $\widehat{\boldsymbol{F}}$. Finally, we compounded the covariances with the 4$^{\text{th}}$ order approximation in Barfoot *et al.*

[33] to obtain a final covariance estimate $\widehat{\boldsymbol{Y}}_{\mathrm{F}}$. This setup allowed us to assess the quality of the covariance estimation in context, over trajectories.

4.6 Results

We trained on one trajectory dataset, while testing on one or many others that had the same type of environment/structure. Testing on separate (but similar) datasets was done to obtain a fair evaluation, while avoiding overly optimistic result due to overfitting. This correspondence is detailed in Table 4.2. The weighting matrix $\boldsymbol{\Theta}$ was trained by Stochastic Gradient Descent (SGD) using the loss from Equation 4.9. The batch size was 4 and the learning rate was 1×10^{-5} The learning was stopped after 100 iterations or until convergence was reached.

4.6.1 Single-Pair Covariance Prediction

First, we validated the quality of our approach, for single pairs of point clouds. As a quality metric, we used the Kullback-Leibler (KL) divergence from our estimated distribution to the sampled distribution (covariance). This metric measures the amount of information lost when using the covariance $\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)$ instead of $\boldsymbol{Y}_k$ to express the distribution of ICP results $\mathcal{I}$. Results reported in Table 4.2 are the average KL divergence over all pairs within a testing group. For comparison, we also computed the average KL divergence from a baseline covariance $\boldsymbol{Y}_{\text{base}} = \frac{1}{n}\sum_k \boldsymbol{Y}_k$, using the $\boldsymbol{Y}_k$ of the training dataset. We made similar computations with a Censi estimate $\boldsymbol{Y}_{\text{Censi}}$, for the same point cloud pairs. One should keep in mind that the mere prediction of the scale of the ICP covariance has been historically challenging. As hinted by Figure 4.3, Censi's covariance estimate have large divergences, since they are orders of magnitude smaller than the sampled covariances. At worst, our approach makes predictions that have the correct order of magnitude, as seen in Table 4.2.

Trajectory in self-similar environments will have point clouds (and covariances) that are similar to one another. For these self-similar environment, gains over the baseline are expected to be modest. In this situation, we observe that our predictor's parameters $\boldsymbol{\Theta}$ converges to nearly uniform weights after training. Consequently, CELLO-3D treats training examples nearly equally, and outputs (something close to) their mean. This phenomenon is clearly visible in Table 4.2, where gains over the baseline are limited for *Wood Autumn* and *Wood Summer*. For the *Gazebo* environments, they also exhibit a certain degree of self-similarity, as measured by the

Table 4.2 – Loss of the CELLO algorithm in various training scenarios

Dataset	Trained on	N. Pairs	Avg. KL Divergence		
			Baseline	Ours	Censi
Apartment	Hauptgebaude & Stairs	1190	34.1	**26.6**	9.19×10^7
Hauptgebaude	Apartment & Stairs	938	34.0	**26.7**	2.06×10^8
Stairs	Apartment & Hauptgebaude	798	33.7	**27.0**	7.65×10^7
Gazebo Summer	Gazebo Winter	826	20.8	**19.8**	2.55×10^6
Gazebo Winter	Gazebo Summer	798	20.3	**18.9**	2.25×10^6
Wood Autumn	Wood Summer	812	13.2	**11.5**	4.94×10^6
Wood Summer	Wood Autumn	966	13.6	**11.3**	3.52×10^7

low KL-divergence of the baseline. Again, gains for our approach are modest there. However, for the three indoor environments (*Apartment*, *Hauptgebaude*, and *Stairs*), we can see that they have the highest baseline KL-divergence. This indicates that the sampled covariance varies largely throughout the trajectory, in comparison to the average (baseline) one. As expected, it is where our algorithm makes the strongest gains. These gains for indoor environments are also explained by their structured nature, well-suited for our descriptors.

4.6.2 Consistency over Trajectories

We evaluated the consistency of the estimated covariances when computing ICP odometry trajectories in diverse locations. This represents the fundamental situation where our predictor is used within a state-estimation algorithm. To visualize the error on the covariance in all dimensions at once, we resort to computing the Mahalanobis distance D_M between $\widehat{\boldsymbol{T}}_\mathrm{F}$ and the ground truth ${}^{l}_{0}\boldsymbol{T}$

$$D_\mathrm{M} = \sqrt{\boldsymbol{\xi}^\top \widehat{\boldsymbol{Y}}_\mathrm{F}^{-1} \boldsymbol{\xi}}, \tag{4.12}$$

in which $\boldsymbol{\xi} = \log({}^{0}_{l}\boldsymbol{T}^{-1}\, \widehat{\boldsymbol{T}}_\mathrm{F})$. The distance D_M can be thought of as the number of standard deviations between a sample and the mean, with a value zero meaning that $\widehat{\boldsymbol{T}}_\mathrm{F}$ was exactly on the ground truth. Table 4.3 shows the average D_M of 100 trajectories for every covariance model. It also lists the average Mahalanobis distances of the translation $\boldsymbol{u}$ against $\boldsymbol{Y}_{\boldsymbol{uu}}$ and rotation $\boldsymbol{\omega}$ against $\boldsymbol{Y}_{\boldsymbol{\omega\omega}}$ for the trajectories. The average values of D_M indicate that our algorithm is consistent overall. We consider that an average D_M above 3 indicates an optimistic covariance estimate, while an average below 1.5 indicates a pessimistic estimate. In that sense, the covariance es-

Table 4.3 – Final odometry error and consistency of CELLO-3D

Trajectory	Length (m)	Translation $\|\boldsymbol{u}\|$ (m)	Translation D_{M}	Rotation $\|\boldsymbol{\omega}\|$ (rad)	Rotation D_{M}	D_{M}
Apartment	22	0.115	0.274	0.0331	0.160	0.540
Hauptgebaude	24	0.168	0.467	0.00910	0.346	1.15
Stairs	12	0.0664	0.0998	0.0127	0.307	0.592
Gazebo Summer	14	0.0396	0.278	0.0165	0.278	0.491
Gazebo Winter	15	0.0311	2.000	0.0144	2.90	2.50
Wood Autumn	18	0.217	0.205	0.0178	0.394	0.405
Wood Summer	21	0.332	0.208	0.0299	0.533	0.762

timation algorithm is pessimistic for the *Stairs* or *Apartment* datasets. However, no extreme values were found demonstrating a functional solution.

In Figure 4.5, we take a closer look at the behaviour of CELLO-3D over short trajectories, in the *Wood Summer* (unstructured) and *Gazebo Winter* (semi-structured) environments. The figure compares the compounded sampled covariances $\boldsymbol{Y}_k$ with our estimated covariances $\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)$. In there, we sampled 20 ICP odometry trajectories to compare against the covariance predictions at each step (the grey ellipses). The green dots represent the final $\widehat{\boldsymbol{T}}_{\mathrm{F}}$ for each trajectory, with the green ellipses being $\widehat{\boldsymbol{Y}}_{\mathrm{F}}$. This figure shows that our covariance estimates are consistent with the compounded trajectories, within 2 sigma. Note that the CELLO-3D uncertainty estimate grows more steadily and uniformly than that of the sampled covariances. This indicates that, while our algorithm does not seem to model the input descriptors in very sharp detail, it is able to extract a consistent knowledge from the training dataset.

4.7 Conclusion

In this work, we presented CELLO-3D, an online covariance estimation algorithm for ICP that works well in 3D. CELLO-3D uses the covariances of a learning dataset to predict the covariance of similar point cloud pairs at runtime. It was successfully validated on individual pairs of point clouds and over trajectories, on challenging datasets. It provides also a better uncertainty estimate when compared to existing solutions. Our predicted covariances are neither too optimistic nor too pessimistic, and represent well sampled particles over trajectories of several meters.

Throughout the course of this work, some challenges became visible with the tran-

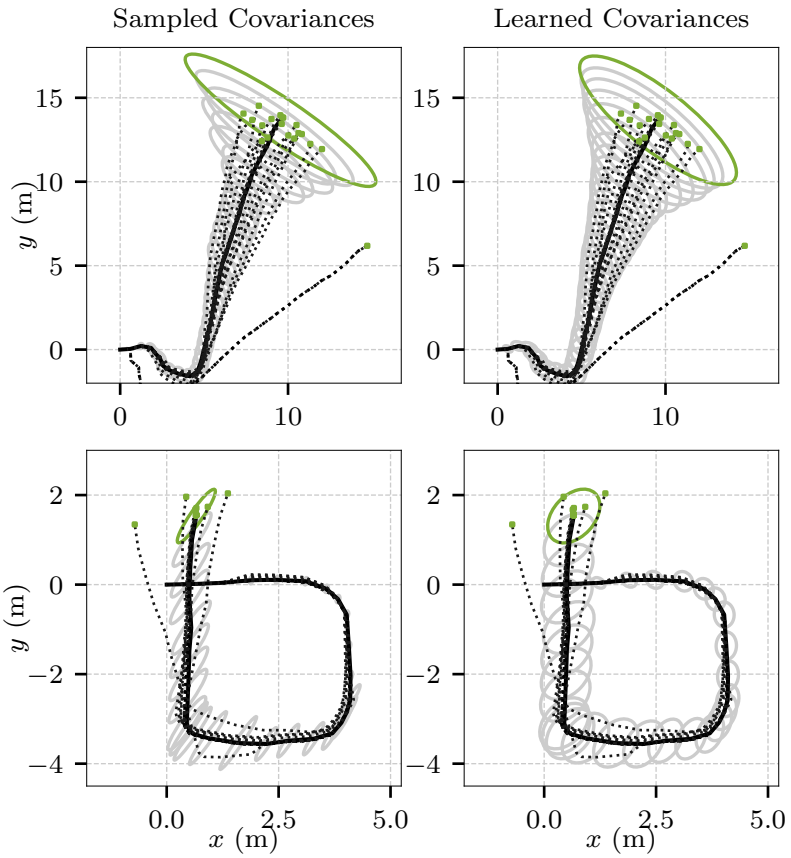


Figure 4.5 – Comparison of covariance estimations for *Wood Summer* (top) and *Gazebo Winter* (bottom). Every frame has 20 ICP odometry trajectories, although some are not visible because they are overlapping. The ground truth trajectory is shown as a thick black line. The green dots represent the final poses $\widehat{\boldsymbol{T}}_{\mathrm{F}}$, while the green ellipses represent $\widehat{\boldsymbol{Y}}_{\mathrm{F}}$. The data is projected on the ground plane for the sake of visualization.

sition from 2D to 3D. Due to the curse of dimensionality, generating a dataset for covariance in 3D requires more samples than in 2D. With this larger number of samples, we noticed that approximating $\mathcal{I}$ as a normal distribution is prone to larger estimation errors in $\mathrm{SE}(3)$, as we observe mainly multimodal distributions (see Figure 4.1 and Figure 4.4). Moreover, the original CELLO framework proposed the use of weak descriptors to alleviate the difficulties of descriptor design. However, the quality of the input features is found to be critical to the success of covariance predictions. Inspired by [28], we intend to meet this challenge using Deep Neural Networks (DNNs) directly on 3D point clouds.

References (Chapter 4)

[2] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[3] C. Yang and G. Medioni, "Object modelling by registration of multiple range images", in *Image and Vision Computing*, vol. 10, 1991, pp. 145–155.

[4] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics", *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[6] W. Vega-Brown, A. Bachrach, A. Bry, J. Kelly, and N. Roy, "CELLO: A fast algorithm for Covariance Estimation", in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 3160–3167.

[7] S. Bonnabel, M. Barczyk, and F. Goulette, "On the covariance of ICP-based scan-matching techniques", in *Proceedings of the American Control Conference*, IEEE, Jul. 2016, pp. 5498–5503.

[8] M. Barczyk and S. Bonnabel, "Towards Realistic Covariance Estimation of ICP-based Kinect V1 Scan Matching : the 1D Case", in *Proceedings of the American Control Conference*, 2017, pp. 4833–4838.

[9] T. M. Iversen, A. G. Buch, and D. Kraft, "Prediction of ICP pose uncertainties using Monte Carlo simulation with synthetic depth images", in *IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 4640–4647.

[10] E. Mendes, P. Koch, and S. Lacroix, "ICP-based pose-graph SLAM", in *International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2016, pp. 195–200.

[14] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces", *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[17] A. Censi, "An accurate closed-form estimate of ICP's covariance", in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3167–3172.

[18] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms", *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.

[19] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited", *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 1–21, 2017.

[20] O. Bengtsson and A.-J. Baerveldt, "Robot localization based on scan-matching — estimating the covariance matrix for the IDC algorithm", *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 29–40, 2003.

[21] P Biber and W Strasser, "The normal distributions transform: a new approach to laser scan matching", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 2743–2748.

[22] J. Nieto, T. Bailey, and E. Nebot, "Scan-SLAM: Combining EKF-SLAM and scan correlation", *Springer Tracts in Advanced Robotics*, vol. 25, pp. 167–178, 2006.

[23] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM", *International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[24] S. M. Prakhya, Liu Bingbing, Yan Rui, and Weisi Lin, "A closed-form estimate of 3D ICP covariance", in *Proceedings of the 14th IAPR International Conference on Machine Vision Applications, MVA 2015*, IEEE, 2015, pp. 526–529.

[25] W. Vega-Brown and N. Roy, "CELLO-EM: Adaptive sensor models without ground truth", in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, Nov. 2013, pp. 1907–1914.

[26] V. Peretroukhin, L. Clement, M. Giamou, and J. Kelly, "PROBE: Predictive robust estimation for visual-inertial navigation", in *IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 3668–3675. arXiv: 1708.00174.

[27] V. Peretroukhin, W. Vega-Brown, N. Roy, and J. Kelly, "PROBE-GK: Predictive robust estimation using generalized kernels", in *IEEE International Conference on Robotics and Automation*, Jun. 2016, pp. 817–824.

[28] K. Liu, K. Ok, W. Vega-brown, and N. Roy, "Deep Inference for Covariance Estimation : Learning Gaussian Noise Models for State Estimation", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1436–1443.

[29] M. Bosse and R. Zlot, "Place Recognition Using Regional Point Descriptors for 3D Mapping", in *Field and Service Robotics*, Springer Berlin Heidelberg, 2010, pp. 195–204.

[30] J. Demantke, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3d lidar point clouds", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. Part 5, W12, 2011.

[31] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, vol. 3, 2009, pp. 23–28.

[32] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets", *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, Feb. 2013.

[33] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems", *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

Chapter 5

Supplementary experiments

The process of writing the CELLO-3D paper (Chapter 4) involved a careful selection of the presented experiments. The selection was difficult because of the space constraints of the conference paper format. This chapter presents two experiments (their motivation, protocol, results and discussion) that had to be left out of the paper. The first experiment allows us to peek at the behavior of the CELLO-3D training. The second experiment tests the behavior of our algorithm on a new dataset, which gives us a better idea of the generalization capabilities of CELLO-3D. Together, these supplementary experiments give a better portrait of the introduced algorithm, and give us research directions to improve it in the future.

5.1 Learning introspection

When implementing CELLO for a particular use case, such as what we did in CELLO-3D, the greatest experimental challenge is the choice of descriptor [28]. It is not possible to use a large vector of weak features because the model scales poorly with the size of the descriptor. Consequently, the descriptor design requires some degree of domain knowledge, and experimentation. One way to approach this is to iteratively improve on the descriptor as the lidar sensor is used on new environments. For this iteration to be possible, however, the descriptor designer must have appropriate feedback on the quality of the descriptor. The training loss is a first tool that can be used, but a more complete portrait of the performance of the descriptor could prove useful. In this spirit, we set out to perform introspection on the learning process and gain more information on its behavior. The knowledge gained here can then be used to improve the descriptors. Better descriptors will then translate into better

covariance estimation for the model.

One interesting tool that emerged from this process is the activation matrix. An example of activation matrix is shown in Figure 5.1. Every row in that matrix corresponds to one training example. Every column corresponds to a validation example, i.e., a descriptor unknown to the model that was used for covariance prediction. The "pixels" represent the weight of the training example when predicting the covariance of the unknown descriptor. The weights are computed using the $\mathbf{\Theta}$ matrix that was obtained after training, once again using the distance metric posed in Equation 4.8. The lighter the pixel is, the more activated (influential) the example was when making a prediction. Identifying which training examples are influential to the learning process is an approach that is gaining traction in the machine learning community [34].

We can use this graph to validate our assumptions about the descriptors, and infer new knowledge about the model. In Figure 5.1, the training dataset and the validation dataset contain roughly the same trajectory (taken at different times). Both trajectories were recorded in the *Gazebo* location of the *Challenging* dataset. The first sequence of point clouds was recorded in the summer, the second in the winter. Consequently, we expect the descriptors of the first point clouds to be similar in both datasets, i.e., have a good activation. The block-diagonal aspect of Figure 5.1 confirms this. It shows that the model learned to differentiate the environment in the different parts of the trajectory.

The same phenomenon is observed in the *Wood Summer* and *Wood Autumn* datasets. Figure 5.2 shows that same block diagonal pattern, to a lesser degree. The dark area at the bottom right indicates that the trajectory recorded in the summer went on for longer than the trajectory recorded in the winter. Consequently, the model does not use the last examples of *Wood Summer* when predicting the covariance of *Wood Autumn*.

This type of figure is insightful when we train the model over a larger variety of environments. Figure 5.3 shows the activation matrix of a model trained using structured and unstructured environments. The examples from every location were split in a training dataset (the first 70 % of examples) and a validation dataset (the remaining 30 %). That way, every location is represented in the validation rows.

A first characteristic of that matrix is its strong block diagonal aspect. It indicates

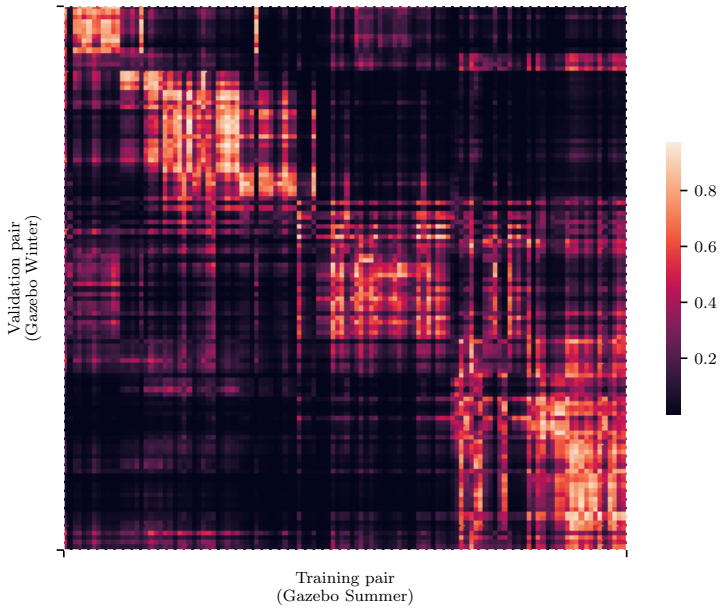


Figure 5.1 – An example of activation matrix, when predicting the covariances of *Gazebo Winter* using examples from *Gazebo Summer*.

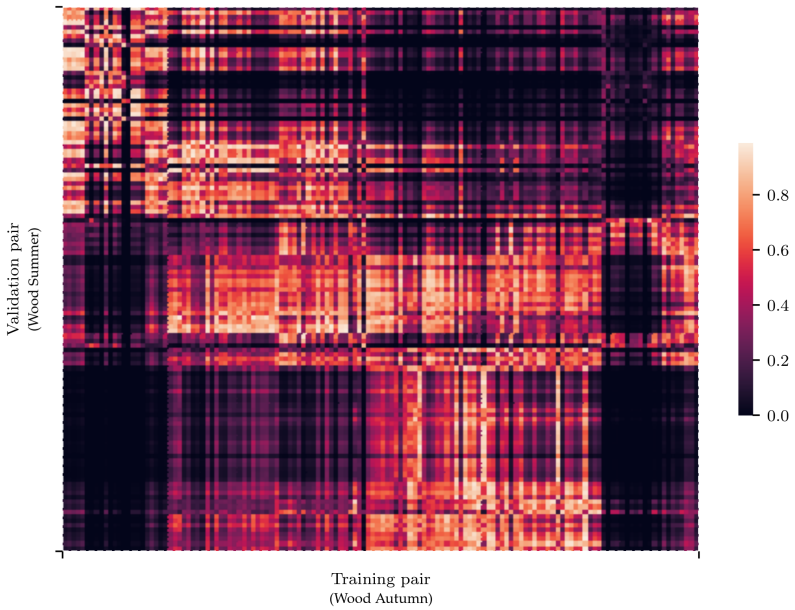


Figure 5.2 – Activation matrix of *Wood Autumn* predicted against *Wood Summer*.

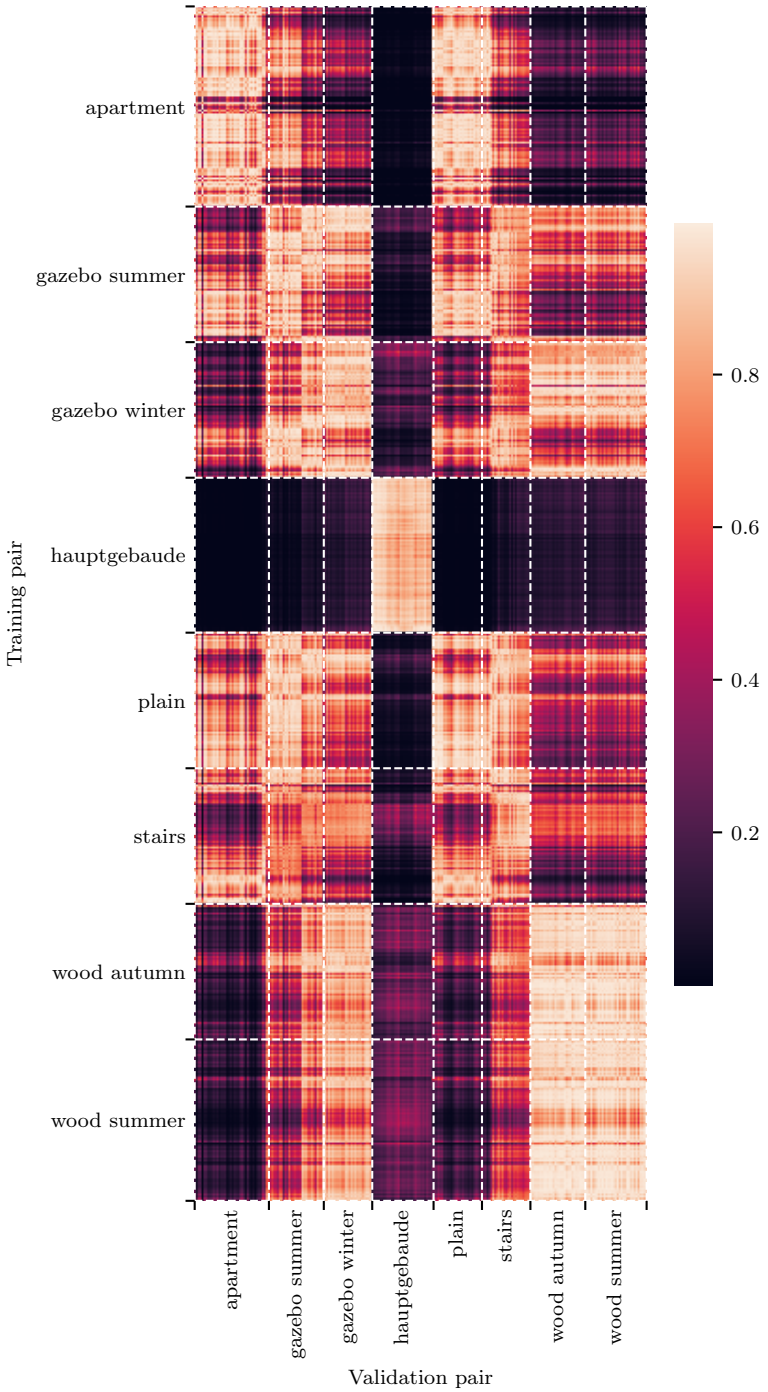


Figure 5.3 – Activation matrix over many datasets. Although the trained model does not benefit from begin trained on a large variety of datasets at the same time, generating this figure can still bring us insight on the performance of the descriptor.

that examples from validation datasets are similar to training examples from that same dataset, which suits our intuition. We also notice an affinity between the environments that come in pair. The examples from *Wood Summer*, for instance, are strongly activated by the examples from *Wood Autumn*, and vice versa. The same is true, to a lesser extent, for the *Gazebo* datasets. However, if we compare Figure 5.3 to Figure 5.2, we notice that we lost the block diagonal aspect *inside* the dataset. When presented with a large variety of environments, the model is not able to capture the same level of detail than when we train it only with data from *Wood*, for instance. This leads us to think that the CELLO model is not rich enough to benefit from being presented with great volumes of data. It does not seem to generalize well from outdoor to indoor datasets, for instance.

This is a hint that the descriptors $\boldsymbol{d}$ from indoor datasets are different to the ones from outdoor datasets. Those descriptors live in two different domains. In machine learning, leveraging datasets that are similar (that come from the same sensor) but different on average (that represent point clouds from indoor or outdoor locations) is called domain adaptation. Domain adaptation is a challenging problem, that could potentially improve the learning results of CELLO-3D. This is left for future work, however. In the meantime, we circumvent the problem of domain adaptation by designing our training set and validation set pairings carefully. This is the rationale that explains the choice of set/validation set pairs in Table 4.2 of the inserted article.

Going back to the activation matrix, we also notice the examples from *Hauptgebaude*, which are not related to examples from any other datasets. We explain this by the exceptionally elongated nature of the covariances of this dataset, because it is a featureless hallway. In other words, the covariances in *Hauptgebaude* are so different from the others that the model made sure the descriptors from *Hauptgebaude* would be very distant from the others.

One thing that is more difficult to explain in Figure 5.3 is the strong activation of the examples from *Apartment* when making predictions for *Plain*. It is difficult to explain because those two environments have very little in common. *Apartment* is indoor and structured, whereas *Plain* is outdoor and unstructured. Furthermore, *Plain* has very little vertical features, which makes it prone to underconstrainedness problems.[1] We speculate two explanations for this phenomenon. Firstly, it might be due to covariances that are similar in both datasets even though their environments

[1] See Section 2.3 for more on underconstrainedness

are very different. Secondly, it may be explained by a side effect of the learning process. It could be that, when optimizing for other factors, the SGD made *Plain* very similar to *Apartment* by accident. In any case, this is a good example of the insight we can draw from the activation matrices. After noticing that the model is confused between *Apartment* and *Plain*, the descriptor designer may want to add features that would help distinguish them.

5.2 Kitti dataset

During the elaboration of the article, we ran experiments on various datasets. This was done in an effort to draw conclusions that were as general as possible. One very attractive dataset for this purpose was the *Kitti Odometry Dataset* [35]. It contains data that is close to what a self-driving car would meet in realistic situations. Consequently, designing an algorithm that performs well on *Kitti* would mean we can be hopeful about it being applicable to very concrete situations. With that in mind, we executed our algorithm on the *Kitti* dataset, with the hope of including those results in the article. Unfortunately, the results were not convincing enough to make a clear case in favour of CELLO-3D. That, combined with the general lack of space, made it preferable to leave the *Kitti* experiments out of the article. This section presents some of the results we obtained on *Kitti*, and then identifies factors that explain the disappointing performance of CELLO-3D on it.

The experimental setup here is essentially the same as in Section 4.5. We use the same protocol, but this time on trajectories 3, 4 and 5 from the *Kitti* dataset. The trajectories are used to train covariance estimation models. The results of the machine learning procedure are shown in Table 5.1, which is analogous to Table 4.2 in the inserted article. For *Kitti*, the results are not as convincing as the ones that were included in the article. Our algorithm, despite being an improvement over baseline, improves the covariance prediction marginally. We must remember, however, that our results still represent a significant improvement over closed form methods, as shown by the very high divergence scores in the rightmost column of the table.

As for the uncertainty estimation over compounded trajectories, the performance of CELLO-3D is listed in Table 5.2. The Mahalanobis Distance D_{M} (defined in Equation 4.12) between the final compounded trajectory and the final estimated distribution is larger than it was for the other datasets. The D_{M} values on the rightmost column are all above 3, which indicates that the simulated trajectories do not belong

Table 5.1 – Loss of the CELLO-3D algorithm for the *Kitti* dataset

Dataset	**Trained on**	**N. Pairs**	**Avg. KL Divergence**		
			Baseline	Ours	Censi
Kitti 03	Kitti 04 & Kitti 05	800	23.81	**21.64**	2.24×10^7
Kitti 04	Kitti 03 & Kitti 05	270	22.83	**21.73**	1.98×10^7
Kitti 05	Kitti 03 & Kitti 04	2760	23.72	**21.35**	1.91×10^7

Table 5.2 – Final odometry error and consistency of CELLO-3D for the *Kitti* dataset

	Length	**Translation**		**Rotation**		D_{M}
Trajectory	**(m)**	$\|\boldsymbol{\xi}_u\|$ **(m)**	D_{M}	$\|\boldsymbol{\xi}_\omega\|$ **(rad)**	D_{M}	
Kitti 03	58.8	1.079	1.193	0.004385	0.4915	3.231
Kitti 04	135.8	48.34	6.375	0.4425	5.536	16.58
Kitti 05	92.98	0.7627	0.9279	0.01416	2.569	5.634

to the distribution created by the covariance estimation algorithm. This is especially true for *Kitti 04*, which has an exceptionally high D_{M} of 17.

These very high values are explained in Figure 5.4. It plots the simulated robot trajectories and the predicted covariances. For *Kitti 04*, the trajectories are more surprising than the predicted covariances. On the one hand, the covariances follow the same general pattern that we observed in the *Challenging* dataset. The trajectories, on the other hand, seem totally divergent. This shows that the odometry itself is to blame for the high Mahalanobis distances. More precisely, our assumption that ICP always converge in the attraction region of the ground truth was broken.

These disappointing results warrant a reflection on the experimental risks associated with the *Kitti* dataset. In hindsight, we can identify several factors that are potentially troublesome when transferring our algorithm to *Kitti*. We believe that further experiments on the *Kitti* dataset should address those issues to improve covariance estimation.

A first challenge that arises is the challenge of scale. The trajectories in the *Kitti* dataset are much longer than the ones of the *Challenging* dataset: they span hundreds of meters. However, we know that we do not have the ability to accurately track uncertainties over large rigid transformations [12, p. 280]. This is a limitation of our experimental protocol that becomes more evident with longer trajectories. It can be overcome by assessing the impact of the compounding error on our results for *Kitti*,

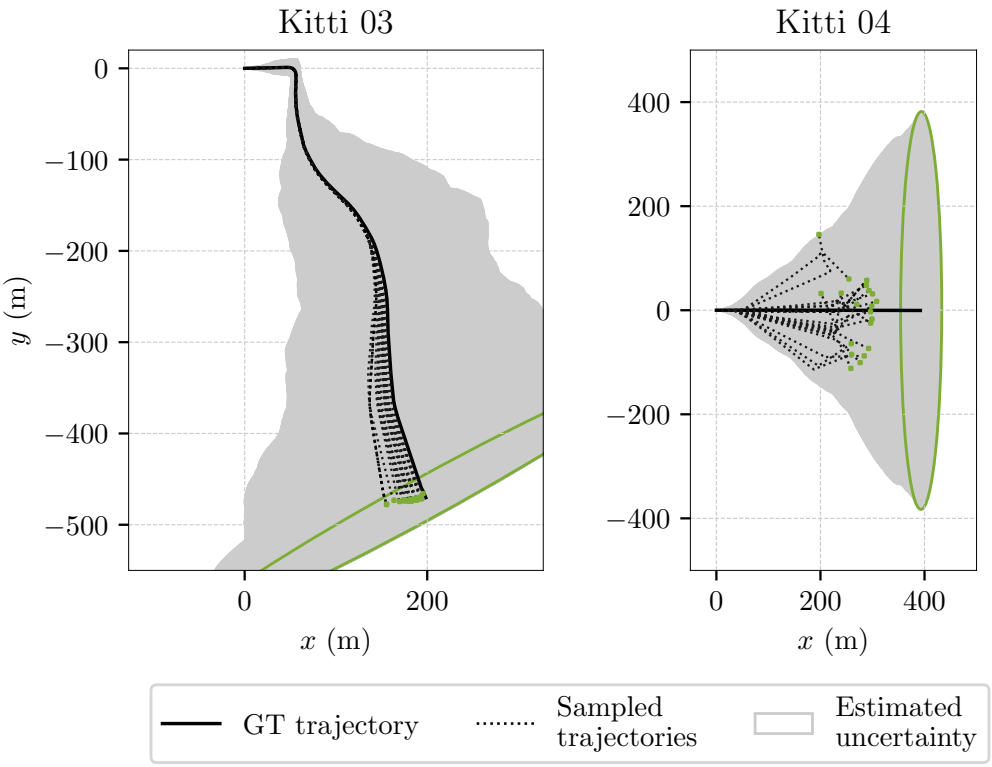


Figure 5.4 – Trajectory over learned covariances for *Kitti 03* and *Kitti 04*. The gray areas are the covariances as estimated by CELLO-3D. The green ellipse and the green dots are respectively the final compounded covariance and the endpoints of the trajectories.

and modifying the protocol accordingly. For instance, the protocol could be adjusted by concentrating our efforts on shorter parts of the trajectories, instead of working on the whole trajectories.

Another hurdle is, quite simply, the sensor. On the one hand, the *Challenging* datasets used a 2D sensor mounted on a panning unit to provide 3D point clouds. Consequently, the assembled scans have a high resolution. The caveat is that the point clouds take longer to produce, because we have to wait for the pan unit to make a complete scan of the area before moving the robot. On the other hand, the sensor of *Kitti* has a much higher time resolution, at the cost of spatial resolution. The point clouds produced by it are more sparse, and cover the environment less uniformly. Trading space resolution for time resolution is very pertinent in an autonomous vehicle scenario, because mobile robots must have quick reaction times. However, it does make point cloud registration more challenging. The effects of this are visible on Figure 5.5. The resolution of points is relatively uniform for the *Challenging*

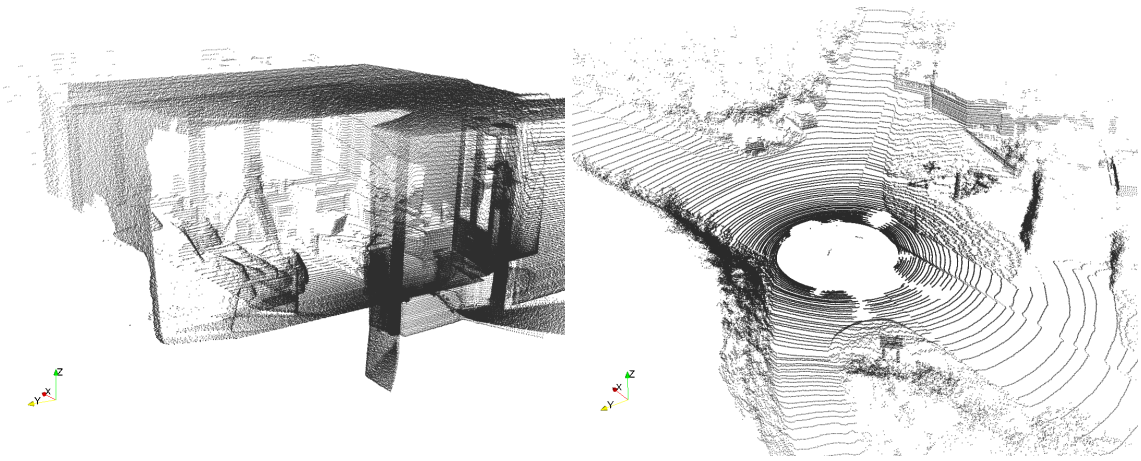


Figure 5.5 – Comparison of point clouds from the *Challenging* and *Kitti* datasets. *Left*: Point cloud from the *Apartment* trajectory of the *Challenging* dataset. *Right*: Point cloud from the *04* trajectory of the *Kitti* dataset. The scan on the left has a better spatial resolution than the one on the right. Furthermore, no rings are visible because the sampling is more uniform on the left.

dataset. On the contrary, the *Kitti* sensor produces rings of points that are increasingly further apart.

This change in sensor is expected to have effects on point cloud registration. Consequently, it can change the distribution of ICP registrations, and the proportion of converging registrations. An adaptation to the ICP pipeline may be necessary to alleviate these changes. The ICP pipeline we set in stone in Section 2.2 might not be appropriate anymore. The change in registration behavior explains the most surprising results from Table 5.2. Most notably, it explains the results for *Kitti 04*, which had an exceptionally high D_{M} of 17.

The change of sensor does not only shows in the ICP registrations themselves, but also into the learning datasets. Since the point clouds from the *Kitti* dataset have different appearances, they may produce different descriptors. The set of $\boldsymbol{d}$ vectors produced by the *Kitti* dataset may not be totally related to the set of $\boldsymbol{d}$ produced by the *Challenging* point clouds. We identify two reasons for this. First, the global descriptor has to be adapted to the new realities of the *Kitti* dataset. Indeed, the descriptor in the paper uses a grid that spans $25 \times 25 \times 10$ meters (see Section 4.5.1). This size is not adapted for self-driving scenarios, because sensor on autonomous vehicles typically have much larger ranges. Second, the local descriptor we had for the *Challenging* dataset uses estimated normals in the point clouds. In Figure 5.5, we already observed that the *Kitti* point clouds have rings, and that their sampling

of the environment is non-uniform. This complicates the estimation of normals for individual points, because normals are estimated using nearest neighbors. If a point is isolated (because it lies on a ring, for instance), it is difficult to estimate the normal of the surface it represents.

Since both the global and local descriptors are affected by the change of sensor, the descriptor performance could degrade when porting CELLO-3D to the *Kitti* dataset. Furthermore, the descriptors $\boldsymbol{d}$ created from *Challenging* point clouds may be, on average, different from the ones generated with *Kitti*. This difference in descriptors, in effect, changes the domain on which we want $\widehat{\boldsymbol{F}}(\cdot)$ to apply. Consequently, we have here another example where CELLO-3D could benefit from domain adaptation techniques. The constant need for domain adaptation, however, underlines a more fundamental problem. It shows that the design of descriptor is a brittle process that hinders the generalization capabilities of CELLO-3D.

Conclusion

The main contribution of this work is CELLO-3D, a covariance estimation algorithm for 3D ICP. This contribution is wholly encompassed in the inserted article. The latter provides a thorough description and experimental evaluation of CELLO-3D. It demonstrates that our algorithm can be used indoor or outdoor, within structured or unstructured environments.

The chapters that surround the article make the contribution clearer. The chapters before it describe concepts and mathematical tools that are useful to express CELLO-3D. They explain some of the intuitions that lead to the design of our algorithm. The supplementary experiments described after the article give us a better portait of the capabilities of CELLO-3D.

With this work behind us, important challenges remain for CELLO-3D. The most important challenge is arguably its limited generalization capability. This limited generalization was underlined by the difficulties of porting the algorithm from one type of environment to another. Also, the point cloud descriptor needs to be manually tuned to different sensors, which requires a great expertise. This challenge needs to be adressed if we are to see a fully satisfactory covariance estimation algorithm for ICP.

We identify two potential directions for future works that could alleviate this problem. These two research avenues would achieve this by removing point cloud descriptors from the equation. The first is to steer back towards closed-form or quasi closed-form approaches. Chapters 2 and 4 show that one issue with current closed-form solutions is their inability to take point reassociations into account. It seems worthwhile to try to alleviate this, while still keeping a solution that operates directly on the pair of point clouds. This would have the advantage of requiring no training data. One way to do it is to sample the objective function of ICP directly to estimate the covariance, instead of relying on the closed form expression of the error.

In other words, estimate the derivative of the error function $J(\cdot)$ numerically instead of using a purely closed form solution. The covariance could then be estimated using this numerical derivative. Although numerical derivatives are not a closed-form solution per say, we expect their computation to be possible in reasonable time intervals. We would have to keep in mind, however, that numerical derivatives are not insensitive to the noise cause by point reassociations. Fortunately, there are many regularization methods that could potentially solve the problem [36]. The same regularization methods are not trivially amenable to closed-form equations, which is an advantage for numerical computations. Using numerical derivatives, point reassociations could explicitly be taken into account, but would not affect the covariance so much that closed form estimates are exceptionally small [10].

The second possible approach is to go further into a data-driven perspective. The experiments in Chapter 5 revealed that the design of descriptors for CELLO-3D is challenging. It is sensitive to a change of sensor and changes in the structure of the environment. Consequently, it seems pertinent to try and avoid the explicit design of a descriptor altogether. Liu *et al.* [28] lays the groundwork for this. Their work is very similar to CELLO, but they make efforts to avoid the explicit design of a input feature descriptor. They prefer an implicit representation through the use of neural networks. As with the original CELLO article, however, this does not mean that scaling this solution to 3D ICP would be trivial. If anything, our work demonstrated that increasing the dimensionality of a problem is not trivial. Notably, the modelling of 3D point clouds using Deep Neural Network (DNN) is the subject of active discussions within the community [37]–[39]. The correct way to model complex 3D scenes using DNN is still unclear, and applying such a model to covariance prediction might be challenging. Still, being able to avoid the descriptor design entirely is very promising, because a lot of the issues with CELLO-3D stem from these descriptors.

In any case, the presence of these two research avenues, one towards closed form approach, and the other towards an even more data-driven perspective, is telling. It shows that there still is room for a vivid discussion about covariance estimation for ICP in the future.

Bibliography

[1] W. Burgard, A. Cremers, and D. Fox, "The interactive museum tour-guide robot", *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial AAAI '98/IAAI '98 intelligence*, pp. 11–18, 1998.

[2] P. Besl and N. McKay, "A Method for Registration of 3-D Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[3] C. Yang and G. Medioni, "Object modelling by registration of multiple range images", in *Image and Vision Computing*, vol. 10, 1991, pp. 145–155.

[4] F. Pomerleau, F. Colas, and R. Siegwart, "A Review of Point Cloud Registration Algorithms for Mobile Robotics", *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[5] P. De Bièvre, "The 2012 International Vocabulary of Metrology: "VIM"", *Accreditation and Quality Assurance*, vol. 17, no. 2, pp. 231–232, 2012.

[6] W. Vega-Brown, A. Bachrach, A. Bry, J. Kelly, and N. Roy, "CELLO: A fast algorithm for Covariance Estimation", in *IEEE International Conference on Robotics and Automation*, May 2013, pp. 3160–3167.

[7] S. Bonnabel, M. Barczyk, and F. Goulette, "On the covariance of ICP-based scan-matching techniques", in *Proceedings of the American Control Conference*, IEEE, Jul. 2016, pp. 5498–5503.

[8] M. Barczyk and S. Bonnabel, "Towards Realistic Covariance Estimation of ICP-based Kinect V1 Scan Matching : the 1D Case", in *Proceedings of the American Control Conference*, 2017, pp. 4833–4838.

[9] T. M. Iversen, A. G. Buch, and D. Kraft, "Prediction of ICP pose uncertainties using Monte Carlo simulation with synthetic depth images", in *IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 4640–4647.

[10] E. Mendes, P. Koch, and S. Lacroix, "ICP-based pose-graph SLAM", in *International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2016, pp. 195–200.

[11] D. Landry, F. Pomerleau, and P. Giguère, "CELLO-3D: Estimating the Covariance of ICP in the Real World", in *IEEE International Conference on Robotics and Automation*, 2019 (forthcoming).

[12] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[13] E. Eade, "Lie groups for 2d and 3d transformations", 2013.

[14] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces", *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

[15] J. L. Bentley, "Multidimensional binary search trees used for associative searching", *Communications of the Association for Computing Machinery (ACM)*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[16] P. Babin, P. Giguère, and F. Pomerleau, "Analysis of robust functions for registration algorithms", in *IEEE International Conference on Robotics and Automation*, 2019 (forthcoming).

[17] A. Censi, "An accurate closed-form estimate of ICP's covariance", in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3167–3172.

[18] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms", *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012.

[19] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited", *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 1–21, 2017.

[20] O. Bengtsson and A.-J. Baerveldt, "Robot localization based on scan-matching — estimating the covariance matrix for the IDC algorithm", *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 29–40, 2003.

[21] P Biber and W Strasser, "The normal distributions transform: a new approach to laser scan matching", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 2743–2748.

[22] J. Nieto, T. Bailey, and E. Nebot, "Scan-SLAM: Combining EKF-SLAM and scan correlation", *Springer Tracts in Advanced Robotics*, vol. 25, pp. 167–178, 2006.

[23] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM", *International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[24] S. M. Prakhya, Liu Bingbing, Yan Rui, and Weisi Lin, "A closed-form estimate of 3D ICP covariance", in *Proceedings of the 14th IAPR International Conference on Machine Vision Applications, MVA 2015*, IEEE, 2015, pp. 526–529.

[25] W. Vega-Brown and N. Roy, "CELLO-EM: Adaptive sensor models without ground truth", in *IEEE International Conference on Intelligent Robots and Systems*, IEEE, Nov. 2013, pp. 1907–1914.

[26] V. Peretroukhin, L. Clement, M. Giamou, and J. Kelly, "PROBE: Predictive robust estimation for visual-inertial navigation", in *IEEE International Conference on Intelligent Robots and Systems*, 2015, pp. 3668–3675. arXiv: 1708.00174.

[27] V. Peretroukhin, W. Vega-Brown, N. Roy, and J. Kelly, "PROBE-GK: Predictive robust estimation using generalized kernels", in *IEEE International Conference on Robotics and Automation*, Jun. 2016, pp. 817–824.

[28] K. Liu, K. Ok, W. Vega-brown, and N. Roy, "Deep Inference for Covariance Estimation : Learning Gaussian Noise Models for State Estimation", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1436–1443.

[29] M. Bosse and R. Zlot, "Place Recognition Using Regional Point Descriptors for 3D Mapping", in *Field and Service Robotics*, Springer Berlin Heidelberg, 2010, pp. 195–204.

[30] J. Demantke, C. Mallet, N. David, and B. Vallet, "Dimensionality based scale selection in 3d lidar point clouds", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. Part 5, W12, 2011.

[31] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, vol. 3, 2009, pp. 23–28.

[32] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets", *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, Feb. 2013.

[33] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems", *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[34] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning", in *International Conference on Machine Learning (ICML)*, 2018.

[35] A Geiger, P Lenz, C Stiller, and R Urtasun, "Vision meets robotics: The KITTI dataset", *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[36] I. Knowles and R. J. Renka, "Methods for numerical differentiation of noisy data", *Electronic Journal of Differential Equations*, no. 2012, pp. 235–246, 2014.

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1578–1584. arXiv: 1612.00593.

[38] R. Klokov and V. Lempitsky, "Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models", in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 863–872. arXiv: 1704.01222.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: deep hierarchical feature learning on point sets in a metric space", in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108.

Appendix A

Training CELLO-3D

The process of training a CELLO-3D model can sometimes be fastidious. The main challenge in this process is the choice of hyperparameters. It is constrained by many concurrent factors, and the implementer has to find compromises experimentally. This appendix describes the two most important hyperparameters in CELLO, the learning rate γ and the regularization parameter α. We explain the rationale that must guide the choice of the implementer for these variables.

A.1 Learning rate

The matrix $\mathbf{\Theta}$ of CELLO is trained using Stochastic Gradient Descent (SGD).[1] Consequently, to train it we must configure a learning rate γ. The latter determines the portion of the gradient that is applied back on $\mathbf{\Theta}$ at each iteration. Thus, it should lie somewhere between 0 and 1.

A learning rate γ that is too small will make the training sensitive to local minimas. If the training algorithm employs a small learning rate, it will tend to stay stuck in these local minimas. To avoid this situation, the experimenter must detect when the training stops prematurely. Symptoms of this are a training that needs very few iterations before converging, or does not seem to improve the loss significantly.

Unfortunately, it is not possible to employ very large values of γ either. If it is too large, the learning procedure will tend to diverge, because at every iteration it makes jumps that are too large to land inside a minimum. In a way, the optimization procedure jumps over its target and lands in another area that is further away. For

[1] See Equation 4.8 for a definition of $\mathbf{\Theta}$.

CELLO-3D, symptoms of this are a loss that increases significantly at every iteration, or a loss that goes wildly up and down without a clear trend towards a minimum.

These two contradictory requirements make it difficult to find a good compromise for the learning rate γ. In the elaboration of the paper, we used values of γ between 1×10^{-3} and 1×10^{-5} depending on the dataset. Those values were mostly found using trial and errors. Since the appropriate value changes according to the dataset, it was computationally prohibitive to run a systematic search every time.

A.2 Regularization

The second hyperparameter of importance is the regularization weight α. This parameter is not stated explicitly in the CELLO-3D paper, but is described in the original CELLO paper [6]. It poses the full training loss $\mathcal{F}$ as a weighted sum:

$$\mathcal{F}(\boldsymbol{\Theta}|\mathcal{D}) = (1-\alpha)\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) + \alpha\mathcal{R}(\boldsymbol{\Theta}|\mathcal{D}). \tag{A.1}$$

Here, $\mathcal{L}$ is the loss on covariance prediction, as defined in Equation 4.9. The regularization term $\mathcal{R}$ is defined as

$$\mathcal{R}(\boldsymbol{\Theta}|\mathcal{D}) = \sum_{i=0}^{n}\sum_{j=0}^{n} \log\Big(s\big(\rho(\boldsymbol{d}_i, \boldsymbol{d}_j)\big)\Big). \tag{A.2}$$

As a reminder, $s(x)$ is a scaling function that we set to $s(x) = e^{-x}$ in Section 4.4.1. Also remember that ρ is our distance metric between descriptors (see Equation 4.8).

The regularization term $\mathcal{R}$ is a measure of the spread of the example descriptors: it detects if the descriptors are very far from one another. The hyperparameter α encodes the relative importance of $\mathcal{L}$ and $\mathcal{R}$. Consequently, it also has to lie between 0 and 1. This relative importance is difficult to establish without experimentation.

A very low regularization is attractive because it lets the SGD focus on optimizing the covariance. However, this has the consequence of making overfitting easy. There is nothing that keeps the algorithm from giving very high weights to $\boldsymbol{\Theta}$. This, in turn, creates very large distances between the training examples (see Equation 4.8). Indeed, if the values in the $\boldsymbol{\Theta}$ matrix are high, multiplying this matrix by a vector will yield large values. In this situation, a handful of training examples which are very similar to the input descriptor $\boldsymbol{d}$ are used to predict $\widehat{\boldsymbol{F}}(\boldsymbol{d})$. This is a case of overfitting, because the model is only able to make predictions for examples that are very similar

to the training data. In the worst case, the weights in $\boldsymbol{\Theta}$ are so large that for some $\boldsymbol{d}$, there are no pertinent examples to make a covariance from. This scenario is a degenerate case that provokes a prediction failure, because no examples are similar enough to the descriptor we want to predict. Experimentally, this degenerate case translated in numerical instability such as the apparition of `NaN`s in the predicted covariances.

Having an α close to 1 is not a solution either. In this case, the loss associated to $\mathcal{R}$ becomes larger than that of $\mathcal{L}$, and the SGD optimizes for the distance between the examples. Consequently, it effectively ignores the quality of the estimated covariances during optimization. A symptom of this situation is when the training loss diminishes, but the validation loss (which ignores the regularization) does not. A $\boldsymbol{\Theta}$ with very small weights also indicates a regularization that is too strong. In our application, we found that a good compromise between these two situations is a value of α between 1×10^{-8} and 1×10^{-4}, depending on the dataset. Once again, those values were mostly found using trial and error, using the experimental intuition of the operator. The appropriate value changed from one learning run to another because of the different datasets. This made it difficult to run systematic searches of appropriate parameter values.

Appendix B

Proof of equivalence of the CELLO-3D learning problem statement

In the inserted paper (Section 4.4.1), we mention that our formulation of the learning problem is different than that of the original CELLO paper [6]. Esentially, the dataset $\mathcal{D}$ was modified so that its learning examples contain covariances matrices $\boldsymbol{\Upsilon} = \frac{1}{n}\sum_j \boldsymbol{\xi}_j \boldsymbol{\xi}_j^\top$ instead of error vectors $\boldsymbol{\xi}$. Where, in the original paper, there was a large amount of example descriptors $\boldsymbol{d}$ with weak reference error $\boldsymbol{\xi}$, our own dataset contains fewer descriptors with stronger examples $\boldsymbol{\Upsilon}$. Our examples are considered stronger because they stem from many ICP registrations given a pair of point clouds. This was justified by the limited availability of 3D point cloud datasets with ground truth.

The formulations are equivalent, but a complete argument was not made in the paper in the interest of space. To prove equivalence, we must reformulate the equations of the original CELLO paper such that they accept covariance matrices. For covariance prediction, we show that Equation 4.7 is equivalent to a degenerate case in the original CELLO formulation, where some of the training pairs $(\boldsymbol{d}_i, \boldsymbol{\xi}_i)$ have the same descriptor. We define a partition over the dataset

$$\mathcal{D} = \mathcal{D}_0 \cup ... \cup \mathcal{D}_m \tag{B.1}$$

$$\mathcal{D}_i \cap \mathcal{D}_j = \varnothing \qquad \forall i, j \tag{B.2}$$

such that

$$\mathcal{D}_k = \{(\boldsymbol{d}_0, \boldsymbol{\xi}_0), ..., (\boldsymbol{d}_l, \boldsymbol{\xi}_l)\} \tag{B.3}$$

$$|\mathcal{D}_k| = l \tag{B.4}$$

$$\boldsymbol{d}_0 = \boldsymbol{d}_1 = ... = \boldsymbol{d}_l \tag{B.5}$$

for every part $\mathcal{D}_k$.

In the original CELLO, the covariance prediction is posed as follows (ignoring notational differences):

$$\widehat{\boldsymbol{F}}(\boldsymbol{d}) = \frac{1}{S} \sum_{i=0}^{n} s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_i)\big) \boldsymbol{\xi}_i \boldsymbol{\xi}_i^\top \tag{B.6}$$

with

$$S = \sum_{i=0}^{n} s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_i)\big). \tag{B.7}$$

By expanding the summation and grouping the terms according to the partition, we get

$$\widehat{\boldsymbol{F}}(\boldsymbol{d}) = \frac{1}{S} \Bigg[\underbrace{\Big(\Big(s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_0)\big) \boldsymbol{\xi}_0 \boldsymbol{\xi}_0^\top \Big) + \Big(s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_0)\big) \boldsymbol{\xi}_1 \boldsymbol{\xi}_1^\top \Big) + \Big(... \Big) \Big)}_{\mathcal{D}_0} + \underbrace{\Big(... \Big)}_{\mathcal{D}_1} + ... + \underbrace{\Big(... \Big)}_{\mathcal{D}_m} \Bigg] \tag{B.8}$$

$$= \frac{1}{S} \sum_{k=0}^{m} \Bigg[s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_k)\big) \cdot \sum_{j=0}^{l} \boldsymbol{\xi}_j \boldsymbol{\xi}_j^\top \Bigg]. \tag{B.9}$$

From Equation 1.19, we have

$$\boldsymbol{\Upsilon}_k = \frac{1}{l} \sum_{j=0}^{l} \boldsymbol{\xi}_j \boldsymbol{\xi}_j^\top, \tag{B.10}$$

where the $\boldsymbol{\xi}_j$ are the ICP errors obtained with the point clouds of descriptor $\boldsymbol{d}_k$. Thus, we can conclude that

$$\widehat{\boldsymbol{F}}(\boldsymbol{d}) = \frac{1}{S} \sum_{k=0}^{m} \Big[s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_k)\big) \cdot l \cdot \boldsymbol{\Upsilon}_k \Big] \tag{B.11}$$

$$= \frac{1}{S'} \sum_{k=0}^{m} s\big(\rho(\boldsymbol{d}, \boldsymbol{d}_k)\big) \boldsymbol{\Upsilon}_k. \tag{B.12}$$

Equation B.12 is exactly how CELLO-3D makes its covariance predictions. There, S' is simply adjusted to compensate for the presence of the l term.

This does not complete the argument of the equivalence, however. We still need to pose the training loss of CELLO in terms of covariance matrices instead of error vectors. Using the same partition, we have

$$\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = -\frac{1}{2}\sum_{i=0}^{n}\Big(\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_i)| + \boldsymbol{\xi}_i^\top \widehat{\boldsymbol{F}}(\boldsymbol{d}_i)^{-1}\boldsymbol{\xi}_i\Big) \tag{B.13}$$

$$= -\frac{1}{2}\underbrace{\Big(\big(\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_0)| + \boldsymbol{\xi}_0^\top \widehat{\boldsymbol{F}}(\boldsymbol{d}_0)^{-1}\boldsymbol{\xi}_0\big) + \big(\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_0)| + \boldsymbol{\xi}_1^\top \widehat{\boldsymbol{F}}(\boldsymbol{d}_0)^{-1}\boldsymbol{\xi}_1\big) + \big(...\big)\Big)}_{\mathcal{D}_0} +$$

$$\underbrace{\Big(...\Big)}_{\mathcal{D}_1} + ... + \underbrace{\Big(...\Big)}_{\mathcal{D}_M} \tag{B.14}$$

$$= -\frac{1}{2}\sum_{k=0}^{m}\Big(l\cdot\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)| + \sum_{j=0}^{l}\boldsymbol{d}_j^\top \widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\boldsymbol{d}_j)\Big). \tag{B.15}$$

Here, we introduce the matrix $\boldsymbol{D}_k$ of error vectors:

$$\boldsymbol{D}_k = \begin{bmatrix}\boldsymbol{\xi}_0 & \boldsymbol{\xi}_1 & ... & \boldsymbol{\xi}_l\end{bmatrix} \tag{B.16}$$

$$\boldsymbol{\Upsilon}_k = \frac{1}{l}\boldsymbol{D}_k\boldsymbol{D}_k^\top. \tag{B.17}$$

This proves useful to re-express the rightmost summation of Equation B.15 as the trace of a matrix product in

$$\sum_{j=0}^{l}\boldsymbol{d}_j^\top \widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\boldsymbol{d}_j = \mathrm{tr}(\boldsymbol{D}_k^\top\cdot\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\cdot\boldsymbol{D}_k) \tag{B.18}$$

$$= \mathrm{tr}(\boldsymbol{D}_k\cdot\boldsymbol{D}_k^\top\cdot\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}) \tag{B.19}$$

$$= \mathrm{tr}(l\cdot\boldsymbol{\Upsilon}_k\cdot\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}) \tag{B.20}$$

$$= l\cdot\mathrm{tr}(\boldsymbol{\Upsilon}_k\cdot\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}) \tag{B.21}$$

$$\tag{B.22}$$

Notice the use of the cyclic permutation on the trace. If we substitute the summation with this new expression in the loss, we get

$$\mathcal{L}(\boldsymbol{\Theta}|\mathcal{D}) = -\frac{1}{2}\sum_{k=0}^{M}\Big(l\cdot\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)| + l\cdot\mathrm{tr}\left(\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\boldsymbol{\Upsilon}_k\right)\Big) \tag{B.23}$$

$$= -\frac{l}{2}\sum_{k=0}^{M}\Big(\log|\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)| + \mathrm{tr}\left(\widehat{\boldsymbol{F}}(\boldsymbol{d}_k)^{-1}\boldsymbol{\Upsilon}_k\right)\Big). \tag{B.24}$$

Equation B.24 successfully expresses the loss of the original CELLO with our formulation of what a learning example is. It was used to implement the loss during training.

Appendix C

Useful identities

Here, we provide the Taylor series for coefficients that are useful to the exponential map in Chapter 1. These series should be used for the sake of numerical stability when the angle θ is close to zero. As a reminder, the symbol $\mathcal{O}(\theta^n)$ is shorthand for the infinite sequence of terms that follows. Since all those terms have a factor θ with an exponent larger or equal to n, their effect on the overall expression is negligible when the angle $\theta \approx 0$.

$$\frac{\sin\theta}{\theta} = 1 - \frac{\theta^2}{6} + \frac{\theta^4}{120} - \frac{\theta^6}{5040} + \mathcal{O}\left(\theta^8\right) \tag{C.1}$$

$$\frac{1-\cos\theta}{\theta^2} = \frac{1}{2} - \frac{\theta^2}{24} + \frac{\theta^4}{720} - \frac{\theta^6}{40320} + \mathcal{O}\left(\theta^8\right) \tag{C.2}$$

$$\frac{\theta-\sin\theta}{\theta^3} = \frac{1}{6} - \frac{\theta^2}{120} + \frac{\theta^4}{5040} - \frac{\theta^6}{362880} + \mathcal{O}\left(\theta^8\right) \tag{C.3}$$