Integrating geometry-based ego-motion estimation processes with deep learning

Andrea de Maio

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *07/06/2021* **par :**

Andrea DE MAIO

Integrating geometry-based ego-motion estimation processes with deep learning

---

JURY

VINCENT LEPETIT	ENPC/TU Graz	Rapporteur
CEDRIC DEMONCEAUX	UBFC	Rapporteur
CHRISTIAN WOLF	INSA Lyon	Membre
DAVID FILLIAT	ENSTA Paris	Membre
MALIK GHALLAB	LAAS/CNRS	Membre
SIMON LACROIX	LAAS/CNRS	Directeur de thèse

École doctorale et spécialité :

EDSYS : Robotique 4200046

Unité de Recherche :

Laboratoire d'analyse et d'architecture des systèmes

Directeur de Thèse :

Simon LACROIX

Rapporteurs :

Vincent LEPETIT et *Cedric DEMONCEAUX*

Résumé

Les processus d'estimation du mouvement sont essentiels dans la plupart des systèmes robotiques et ont maintenant atteint de bonnes performances en précision.

Avec l'avènement de l'apprentissage profond, une part importante des tâches de localisation a été réinventée grâce aux approches basées exclusivement sur les données. Si, en termes de précision, les approches d'apprentissage profond se sont avérées compétitives et aussi performantes que les processus basés sur la géométrie, elles ont également montré des faiblesses. Les méthodes basées sur les données souffrent en effet d'un manque d'explicitation des modèles et ont des limites en termes de généralisation. Ces aspects sont essentielles pour de nombreuses applications de la robotique, par exemple la robotique spatiale, industrielle ou de terrain, dans lesquelles les conditions environnementales peuvent varier considérablement.

Cette thèse développe l'idée que les méthodes d'apprentissage approfondi peuvent être utilisées pour compléter les processus classiques d'estimation des mouvements basés sur la géométrie. Le résultat de l'intégration de ces deux approches vise à surmonter certaines des limitations des méthodes basées sur l'apprentissage. Notre objectif est double : générer des corrections pour les processus basés sur la géométrie dans un cadre probabiliste, et contrôler activement la paramétrisation des processus en exploitant les informations contextuelles (scène, mouvement, etc.).

L'estimation des corrections dans un contexte probabiliste bénéficie de la possibilité de prédire en même temps l'erreur la plus probable et l'incertitude associée. Les informations relatives à cette incertitude sont nécessaires pour qualifier l'estimation naturellement, mais aussi pour développer des processus de fusion de données.

Le contrôle actif des processus de localisation consiste lui à trouver un ensemble optimal de paramètres minimisant une métrique d'erreur prédéfinie. Cet ensemble de paramètres est généralement fixé par des réglages qui exploitent les connaissances du roboticien. Nous démontrons qu'une configuration dynamique des paramètres peut grandement améliorer la précision de l'estimation du mouvement, en proposant une architecture d'apprentissage par renforcement dédiée à ce problème.

Abstract

Ego-motion estimation processes are essential in most robotic systems and have now reached high precision performances. With the advent of deep learning, a conspicuous portion of self-localization tasks has been reinvented using a data-driven approach. While, in terms of accuracy, deep learning approaches turned out to be competitive and in line (or better) than geometry-based processes, they have also shown weaknesses. Data-driven methods suffer lack of model explicability and demonstrated limits in generalization. These issues are key to a large set of robotics application domains, such as space, industrial and field robotics, in which the environmental conditions can vary significantly.

This thesis fosters the idea that deep learning methods can be used to complement classical geometry-based pipelines for ego-motion estimation. The result from the integration of these two approaches is deemed beneficial to overcome some of the limitations presented in learning-based methods. Our objective is twofold: to generate error models for geometry-based processes in a probabilistic framework, and to actively control process parameterization exploiting contextual information (*e.g.* scene, motion).

Estimating error models in a probabilistic context benefits from the possibility to predict at the same time the most likely error and the associated uncertainty. Uncertainty information is required to qualify the precision of the estimation itself, and to further reduce tracking errors in data fusion schemes. We propose two contributions that resort to deep learning schemes to estimate such information: one that is tailored to visual odometry using stereoscopic image pairs, and one that is tailored to the Iterative Closest Point algorithm that exploits LiDAR point clouds. These approaches are evaluated on datasets for autonomous driving using well-established metrics in terms of trajectory accuracy and relevance of the uncertainty with respect to actual errors.

Actively controlling self-localization processes consists in finding an optimal set of parameters minimizing a predefined error metric. This parameter set is generally fixed for long trajectories and tuned using the roboticist's knowledge. We demonstrate that a dynamic configuration can greatly improve motion estimation accuracy and propose a specific reinforcement learning architecture to address this problem. Preliminary results are shown on a simple process as RANSAC and discussed on a more complex case as visual odometry.

To all the people who feel to have played a role in my scientific and personal development during my years at LAAS.

Thank you all for your support.

Contents

Introduction

For autonomous mobile robots, the perception layer is responsible of sensing the surroundings of the robot for various tasks, such as environment modeling, localization, and trajectory servoing. To achieve these tasks, robotics systems are endowed with exteroceptive sensors that acquire data carrying information about the environment and can encode semantically varied features.

Among the perception tasks, robot self-localization plays a key role, as it is required to build environment models, to ensure the proper execution of motion, and to supervise the execution of high level missions. Localization encloses all the processes that integrate proprioceptive and exteroceptive data to estimate the position of the robot with respect to a given frame. To tackle localization in robotics, a wide set of perception processes have been developed, from odometry to place recognition, via simultaneous localization and mapping approaches (SLAM). The wide availability of some sensors (*e.g*. cameras, LiDARs, inertial measurement units) rapidly accelerated the progress of localization processes in terms of accuracy, robustness and efficiency.

Active perception

Robots have been long conceived following the *sense - plan - act* paradigm, in which perception generally serves planning and acting processes. Robotic agents are usually designed to perceive in order to act and not the other way around. On the contrary, the term *active perception* coins approaches in which the robot acts in order to perceive. It identifies the study of strategies and behaviors that aim at maximizing the information gain. Perceiving agents, from humans to animals, act to gain knowledge about their surroundings, their position and the position of other points of interests. In fact, in nature, the link between acting and sensing is significantly tight. In robotics, actions are rarely driven by the need for better environmental models, and even less frequently they are taken to improve the expected quality of the localization process. Instead, it is common to see robotic agents self-localize themselves in order to achieve an assigned task, *e.g*. move to a desired position. It is clear that the achievement of the task is strongly dependent on the efficacy of perception processes. For instance, acquiring more information about the surroundings could elongate the time to reach the goal, but would allow to get closer to it due to a better self-localization.

Active perception entails the definition of where and how to gather data, and also how to process it. Perception processes are most of the times configured once and for all, and they rarely adapt on the basis of the perceived data. Of course, data strongly affects perception: it is well known that localization processes performances vary with respect to the environment, for instance lack of geometrical features in the surroundings or poorly textured surfaces are notable causes of failures for self-localization processes. The dependency of perception performances on the input data and the need to adapt to the input data

bring out the need for a continuous, *data-centered*, evaluation and control of perception process.

Ego-motion estimation

Self-localization processes are sometimes referred to as *ego-motion estimation processes* when the task consists in estimating a pose through consecutive, close, data acquisitions. Such processes are often used to track the evolution of a sensor (*e.g.* a camera) or of a robot over trajectories where data can be acquired consistently and frequently. In this manuscript we discuss about two main threads:

1. The evaluation of internal uncertainty over estimates produced by ego-motion estimation processes.

2. The impact of configuration parameters on ego-motion estimation processes and their active control.

Both problems are tackled from the perspective of the correlation between input data and the process output. The main argument made in this work is that, for the vast majority, performances of a process are a consequence of the input data and of the process configuration. Addressing the first point constitutes a prerequisite for the second. In order to control and change parameters on-the-fly it is necessary to be able to assess the current and expected performances of the process. Internal error models, in the form of confidence intervals over produced estimates, can fulfill this purpose. Moreover, from a broader perspective, to have information on the accuracy of a process is key for several reasons:

- It is intrinsically important to known the confidence of an estimation process over its measures. Expressing how sure we are about an estimate frames the process in a probabilistic manner.

- Uncertainty in the form of a covariance matrix is largely used, from filtering to pose-graph methods, via casting the search for loop closures in SLAM approaches.

- Ideally, it should be possible to use error models to minimize the error of a process as a function of its parameters.

From a formal point of view, in a probabilistic context, which encompasses many perception processes, uncertainty represents faithful error models. Many of these processes are iterative methods which rely on the minimization of a predefined error function. Such techniques naturally yield residuals which can be interpreted as an intrinsic error metric. However, residuals not always coincide with lower errors in practice and it is necessary to resort to alternative error functions.

In this thesis, our ambition is to propose approaches that can fill the gap in this framework. On the one hand, we discuss about the the connection between the input data and the errors of a process. On the other hand, we show the role played by the parameters of a process with respect to its output, and therefore to its accuracy.

Data quality assessment With the goal of coming up with more informative metrics, and outside probabilistic contexts, it is possible to model tailored ones. The advantage in doing so is that they can be specific to the process itself. Such metrics can be defined to detect faults or to identify particular changes in the scene and/or excessive resource usage. Modeling the expected performances of a process is key but to define them in the absence of ground truth is a challenge. Additionally, we wish to capture the sources of errors in estimation processes. It is crucial to know what is the cause of an anomaly and what correlates with it. Error sources can be controllable and uncontrollable. The parameters, the selection of a process components, the devoted resources, the running frequency are controllable. These choices are made with the intent of maximizing the performances/minimizing the errors (even though sometimes operational constraints may require a trade-off between performances and resource usage).

On the other hand, uncontrollable sources of errors are external factors that we can account for. The vast majority of uncontrollable errors come from the environment in which the robot operates. This translates to the data the sensors digest in forms understandable by the estimation process itself. It is the case of how a stereo camera maps the 3D world into a pair of 2D images or how a LiDAR captures environmental information into a point cloud. The data processed by the sensors can carry a large amount of information concerning the surroundings, from geometry to semantic details. Furthermore, data features are processed at several stages in a single process. Every time the data is fused, filtered, or transformed, there exists a chance of introducing noise and errors. When it comes to environmental conditions, it is possible to evaluate the relevance of the available (processed) environmental features in a given stage of a process. Such task can be daunting and challenging to generalize, especially trying to regress input data to error models.

Deep learning for ego-motion Recently, the advances in deep learning significantly facilitated information extraction from large quantities of data. The advent of deep neural networks, and in particular with the use of image convolutions, quickly transferred deep learning to the robotics perception domain. Machine learning proved successful for feature extraction, classification and other tasks. One the main capabilities of these techniques is to learn patterns and extract features in order to generalize a given task. When it comes to modeling errors it is possible to train a system that observing different instances of related data, which are the representation of different environments, has to learn a relation between sensory data and errors. The error models can be learned with respect to a given estimation process. As discussed, the availability of faithful error models is scarce. The majority of deep learning approaches are supervised, *i.e.* make use of ground truth associated to the data to generate models. It can be more challenging to associate uncertainty to a process estimate without knowing the real one. Recent advances allowed to produce various types of uncertainty in a semi-supervised way. Neural network can be trained under a probabilistic scheme, generating error models in form a covariance matrix, leveraging the errors produced by comparing a perception process output and the ground truth. These models can contain semantically different interpretation of uncertainty and are not only linked to the process, but to the input data as well. It is worth to notice that following this approach, it

is possible to maintain a backbone composed by geometrical estimation processes, enhancing it with learning-based methods. The idea of filling the gaps of model-based approaches with machine learning comes in useful when we are not able to describe certain dynamics of a process, such as producing accurate error models.

Adaptive process control Another phenomenon that has been hard to model is the one modeling the behavior of a process when changing its parameters. In fact, most of ego-motion estimation processes, come with a sizeable set of configuration parameters. Varying each of those, given an input data instance, reflects into a different output. A common approach is to pre-configure the process, considering the operational context and constraints that the robot (or the process) will face. Nonetheless, different environments can be faced in the lifespan of a robotic agent. Environmental conditions, resource availability, and other constraints can vary significantly. More importantly, the structure of the perceived world can be wide-ranged. Robotic agents can face both structured and unstructured environments in a small amount of time. It is for instance the case of autonomous vehicles transitioning from urban scenarios to countryside areas. Planetary rovers can in few days acquire images of rocky formations and sandy dunes. Still, very often the parameterization of perception processes does not change according to the input data. One of the goal of this manuscript is to assess whether such need in justified, and to propose possible solutions to address it.

Thesis contributions

The contributions of this thesis are placed in the frame of reference presented above. They address the generation of error models for different estimation processes. The work focuses on two popular processes: Visual Odometry (VO) and Iterative Closest Point (ICP). These two process work with different types of data and in most cases are adopted to perform ego-motion estimation using different sensors. Visual Odometry relies on images to estimate motions, while ICP registers LiDAR scans for the same purpose. The generation of error models for visual odometry is discussed in [De Maio 2020]. Besides error models, we present a way to correct visual odometry estimates suppressing biases originally present in the process itself. A similar approach is adopted for the ICP case. In this work, a deep neural network expressly designed to digest point clouds is used to estimate different uncertainties in association with the ICP process. The work is currently in submission to a computer vision conference for which double-blind review is in act [De Maio 2021].

On the data assessment topic, we propose an open-source implementation of visual odometry, dissecting the process in several atomic functions [De Maio 2018]. The objective is twofold: proposing a modular representation of perception processes, and, with the use of tailored metrics, generate indicators for failure points. The concept is extended to back the idea that through data quality assessment functions, it is possible to identify sub-par scenarios given the current parameterization.

As a natural evolution of the data assessment work, we discuss of the need to dynamically control the parameters in an estimation process. Through the example of visual

odometry, we demonstrate this need and show how the selection of optimal parameter sets can improve the tracking accuracy. The process parameterization is cast as a reinforcement learning problem with the objective of learning a policy associating parameters to input data.

Additional contributions In addition to the aforementioned contributions, the work carried out in the context of this thesis features a few more efforts not included in the manuscript. The work presented in [De Maio 2017] and [Govindaraj 2017] sets the basis for the discussed formalization of perception process, with a particular focus on the concepts of *nodes* and *compounds* which are recalled in [De Maio 2018]. The work of this thesis has been supported by the H2020 EU project InFuse. The project consisted in the development of a data fusion suite for space robotics applications. Additional publication have been produced in the framework of such work. The work in [Lacroix 2019] describes a comprehensive dataset for planetary exploration collected in Erfoud, Morocco. The dataset is composed of a multi-camera image collection with two rovers, LiDAR scans and hyperspectral images. In the same context, [De Maio 2019] illustrates a pose management system developed for InFuse and shows its use fusing visual odometry estimates and global poses computed with absolute localization techniques.

Structure of the manuscript

The manuscript is a collection of three publications that are entirely my own work, plus a chapter that depicts a tentative to actively control VO using a reinforcement learning approach.

Chapter 1 has been published in 2018 in the International Symposium on Artificial Intelligence, Robotics and Automation in Space [De Maio 2018]. It discusses about the formalization of perception processes. The objective is to break a process in atomic functions proposing a structured way to represent them. This approach is instantiated in a proposed visual odometry algorithm. Each of the nodes composing the process are evaluated thanks to dedicated error functions. These functions inspect the quality of the node output as a function of its input and parameters. It is shown how it is possible to detect challenging situations monitoring the evolution of these metrics. Some examples of improving tracking under certain conditions are shown through the hand-tuning of the parameters.

Chapter 2 has been published in the Robotics and Automation Letters in 2020 [De Maio 2020]. It proposes a deep learning architecture to jointly estimate covariance and corrections for visual odometry. The presented network learns to correlate input images with the output of a visual odometry algorithm. Ablation studies are performed to evaluate the best type of data to estimate error models among monocular images, stereo images and depth maps. The evaluation is performed on autonomous driving sequences over long trajectories. Final validation is carried out using the predicted covariances in

pose-graph with simulated loop-closure after having corrected visual odometry estimates with the corrections produced by the neural network.

Chapter 3 is a paper under review submitted to the 2021 edition of a conference on [De Maio 2021]. It tackles the problem of covariance estimation for the iterative closest point algorithm. The subject is a long studied problem and has been recently re-evaluated thanks to learning techniques. This work investigates the different type of uncertainties that can be learned, from the data perspective and from the network perspective. Uncertainty in the form of covariance matrix is estimated from LiDAR scans acquired in urban and rural driving sequences. The evaluation relies on well-known metrics that allow to compare with the state-of-the-art, demonstrating that our method performs consistently well.

Chapter 4 presents preliminary work in the direction of parameters control. It discusses the parameterization of two different visual odometry implementation and selects one for parameter exploration. Based on reinforcement learning works, the parameter search approach is firstly demonstrated on a simple process like RANSAC. Then, the challenges of applying the proposed approach to the visual odometry case are presented. In particular, an overview of the optimal parameters obtained using grid exploration is presented, highlighting the difficulty of successfully finding optima in the parameter space using learning-based approached.

CHAPTER 1

Enabling active perception through data quality assessment: a visual odometry case

Contents

Abstract

The state of the art of perception processes for the autonomy of robots is constantly improving, yet these processes remain mostly pre-configured at the robot design phase. This prevents their adaption to the context at hand, which is all the more needed for long life systems that encounter a large variety of situations. This paper presents work on the modelling of perception processes, exhibiting the need to assess their quality, so as to be able to actively control them. We instantiate the visual odometry case, a crucial functionality for planetary rovers, and define dedicated data quality assessment functions for the elementary processes composing it. These functions are used to monitor the processes, defining control points onto which explore different parameter configurations that better adapt to

the context the robot is facing. Preliminary tests are performed using a planetary analogue data set to show the potential of this approach.

1.1 Introduction

The sensing technologies for robotics are ever improving, along with the state of the art in data processing and data fusion, that now offers a broad choice of solutions for robotics perception. This will allow to endow future generation of space robotics systems with a much richer perception layer, which will maximise the throughput of exploration missions, *e.g.* by yielding the possibility of autonomous long traverses or autonomous science.
However, such broad perception capabilities come with a large amount of configuration parameters. So far, roboticists configured perception processes to find the best expected trade-off between genericity, applicability to the context in which the robots operate, and constraints on the sensing, communication and computing resources. In other words, data fusion and perception processes of space robots are fixed by design, tailored to the task they are needed for, and cannot be optimised online with respect the conditions under which the robots are operating and the objectives they are trying to achieve. This prevents the adaptation of the perception activities to the context and the task at hand, which may yield poor performances in case of unexpected scenarios, and in the worst cases the inability to provide useful information.
The augmentation of the complexity of perception processes raises a need to *actively* control them, by deciding which data to acquire and how to process them. This follows the *active vision* paradigm [Bajcsy 1988], which aims at optimising the throughput of perception, that is the relevance and quality of the information provided to the clients of the perception processes. The clients that exploit the two main perception products, environment models and robot localisation, are numerous and of varied nature, from motion control to decisional and planning processes. To a large extent, the quality of the perception products define the efficiency and adaptivity of the robots, and there is a strong interest in optimising this quality.
In robotics, development of active perception schemes have always targeted to specific given tasks: there is a lack of a system abstracting from the type of sensor or from the nature of the task, meant to define a generic, principled approach to active perception. Our research objective is to propose a formal and operational framework to allow the control of robotics perception activities. This entails a formal modelling of the perception tasks, and the definition of optimisation tools that allow to configure perception activities, as well as to control them in real time. The framework aims at having autonomous systems being able to adapt to a larger variety of contexts and situations, without the need of a human in the loop to manually reconfigure perception processes.

Outline. This paper sketches our work in this direction, and focuses on the case of Visual Odometry (VO), a perception process that has shown its importance for planetary rovers. Section 1.2 first drafts the way we foresee the modelling of the perception processes. The remaining of the paper is dedicated to the specific case of VO: section 1.3 depicts the

models of the underlying processes, section 1.4 defines the associated metrics that allow to qualify, and section 1.5 depicts and discusses preliminary results.

1.2 Modelling Perception Processes

The quality of information produced by the perception processes depends on numerous factors. Some are controllable, like the selection of the algorithms, their configuration and composition, or the resources devoted to their execution. Others are not controllable: the nature of the perceived scenes and the environmental conditions have a strong impact on the output quality of the perception processes.
To maximise the output quality of the perception processes, one must intervene on the controllable factors, while tracking and possibly adapting to the uncontrollable ones. For this purpose, one must assess the influence of the controllable factors on the process output quality: this is done by defining functions that disclose this influence, as well as the influence of the non controllable factors, *i.e.* by defining *models* of the perception processes.

1.2.1 Structuring Perception Processes: Nodes and Compounds

Perception encompasses a variety of processes, ranging from simple data filtering to complex optimisation schemes for state estimation, that must be assembled (composed) in order to fulfill a given functionality, *i.e.* deliver a product such as an environment model or a position estimate. This structure is applicable to all the perception functionalities a given robot must be endowed with. In the context of autonomous navigation, Visual Odometry, SLAM and the generation of a Digital Elevation Map (DEM) are perception functionalities which are composed of several signal processing functions, organised together to return their associated products. As in [Govindaraj 2017], we adopt the notion of *nodes* and *compounds*, where a node represents an atomic perception function performing elementary operations, and a compound is a composition of nodes assembled to deliver a specific data product. A broader view of this concept is presented in [De Maio 2017], along with a sketch of a taxonomy of nodes and compounds defined by their input and output data types.
The benefits of structuring perception activities into nodes and compounds are the ones of any component based software architecture: *e.g.* reconfigurability, ease of development and maintenance, separation of concerns, reusability, openness, and of course composability. In the context of active perception, *controllability* is one of the most important concern. In the remainder of this section, we present a generic formal model of perception nodes that specifies the influence of their control.

1.2.2 A Model for Controlling Perception Nodes

Each perception node is characterised by a set of inputs $\boldsymbol{i}$ and a set of outputs $\boldsymbol{o}$ both represented by specific data types (for simple processes, theses sets are singletons). The combination of these input and output types identify the *type* of the process, which can be achieved using different implementations (for instance numerous approaches extract point features from an image – note this may yield slight variations of the definition of

the associated descriptor, and hence of the output type: such considerations pertain to the definition of taxonomies of processes and data types, with inheritance mechanisms, which will not be developed here).

A set of input parameters $\boldsymbol{u} = \{u_1, \ldots, u_n\}$, generally linked to the considered implementation, is specified and represents the controllable means to intervene on the process, either at configuration stage or during its execution. Finally a set of data quality assessment functions $\boldsymbol{j} = \{j_1, \ldots, j_m\}$ are defined: they assess the quality of the output as a function of the inputs and parameters. These quality variables are of various nature: they can come along the production of the process output (such as variance for a state estimation process), or are more or less explicitly encoded within the process output (such as the proportion of outliers produced by a data association process). The problem of optimising a perception node is simply stated as finding the optimal set of parameters $\boldsymbol{u}^*$:

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} j(\boldsymbol{u}, \boldsymbol{i}) : j(\boldsymbol{u}^*, \boldsymbol{i}) \leq j(\boldsymbol{u}, \boldsymbol{i}) \; \forall \boldsymbol{u} \tag{1.1}$$

Other exogenous concerns condition the behavior of the perception nodes, and hence the quality of their output. These are grouped under the term "context", and include for instance light conditions, terrain and texturing levels. The context gathers a series of non controllable, and sometimes even non observable parameters. Some context information can indeed be directly observable, *e.g.* by specific sensors or given information, some are implicitly encoded in the input parameters $\boldsymbol{i}$, and hence not necessarily observable, and finally some are not known at all. Denoting the $\boldsymbol{z} = (z_1, \ldots, z_p)$ the set of uncontrollable yet observable information, which include the inputs $\boldsymbol{i}$, Eq. 1.1 is rewritten as:

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} j(\boldsymbol{u}, \boldsymbol{z}) : j(\boldsymbol{u}^*, \boldsymbol{z}) \leq j(\boldsymbol{u}, \boldsymbol{z}) \; \forall \boldsymbol{u} \tag{1.2}$$

Lastly, a set of costs $\boldsymbol{c}$ can be associated to the perception node: it encompasses the various resources consumption (time, memory...), and the optimisation problem then comes to maximising quality to cost ratios.

Finally, we can formalise a model for a perception node $\mathcal{N}$ as:

$$\mathcal{N} = (\boldsymbol{i}, \boldsymbol{o}, \boldsymbol{u}, \boldsymbol{z}, \boldsymbol{j}) \tag{1.3}$$

1.2.3 From Nodes to Compounds

The integration and interaction of perception nodes defines a perception compound $\mathcal{C}$, which produces the final data products to be delivered to the client processes. Most of the times, compounds are defined as a pipeline assembly of nodes, but some nodes can be asynchronously invoked, and some feedback sequences can be defined. Here the component based model is very helpful.

Optimising a compound $\mathcal{C} = \bigcup_{i=0}^{k} \mathcal{N}_i$ of k nodes comes to find the set of controls $\boldsymbol{U_{0,k}} = \{\boldsymbol{u_0}, \ldots, \boldsymbol{u_k}\}$ for each of the k implied nodes so as to optimise the final product,

that is the output of the last involved node $\mathcal{N}_k$:

$$\boldsymbol{U}^*_{0,k} = \underset{\boldsymbol{U_{0,k}}}{\arg\min}\, j(\boldsymbol{u_k}, \boldsymbol{z}) \tag{1.4}$$

Finally, more parameters come into play at compound level. For instance, the frequency at which the whole compound runs can be tuned: this affects not only the resource usages but also the accuracy of the data produced by some nodes. Moreover, a compound can be assembled with different compositions of nodes. Some nodes can be optional and some others can be arranged in different orders. For many nodes it is also common to have different implementations performing the same task in different ways. All these cases deal with the *topology* of the compound, offering another layer of control.

In the generic case, given the large parameter space (even if it is reduced by the fact that all parameters are not independent) and the various interleaved semantics between quality measures j and process inputs and outputs, such an optimisation problem is intractable. The next section will illustrate the difficulty of the problem for the case of Visual Odometry, a rather simple pipeline compound.

1.3 Modelling Visual Odometry

Visual odometry is one of the first localization means developed for mobile robotics that exploited vision. Since pioneering work that dates back to the 90's, VO has featured a large number of approaches and methodologies: sparse vs. dense, using monocular vision or stereovision, and many more [Scaramuzza 2011, Aqel 2016]. We focus here on the most classic instance of VO, which derives the motion between two positions at each of which a pair of stereovision images is acquired, by matching point features extracted in the images (Fig. 1.1). This instance of VO is well adapted to the limited resources typically available on board planetary exploration rovers, and has been extensively used on the Mars Exploration Rovers [Maimone 2007].

1.3.1 Involved Processes

This VO scheme is a compound which integrates four nodes: (1) a keypoint extractor, that takes as input an image acquired by a camera and outputs a set of keypoints; (2) a data association algorithm that matches two sets of keypoints extracted from two different images; (3) triangulation, that associates 3D coordinates of keypoints matched between two stereoscopic images, and (4) a motion estimator that computes the relative motion between two stereoscopic image pairs, using the associated keypoints matches and 3D coordinates. Fig. 1.2 summarises how these nodes are pipelined.

The final output of the overall process is a 3D transformation between the two times (or positions) $t-1$ and t at which the stereoscopic images were collected:

$${}^{t-1}\boldsymbol{T}_t = \begin{bmatrix} {}^{t-1}\boldsymbol{R}_t & {}^{t-1}\boldsymbol{p}_{t-1,t} \\ \boldsymbol{0} & 1 \end{bmatrix} \tag{1.5}$$

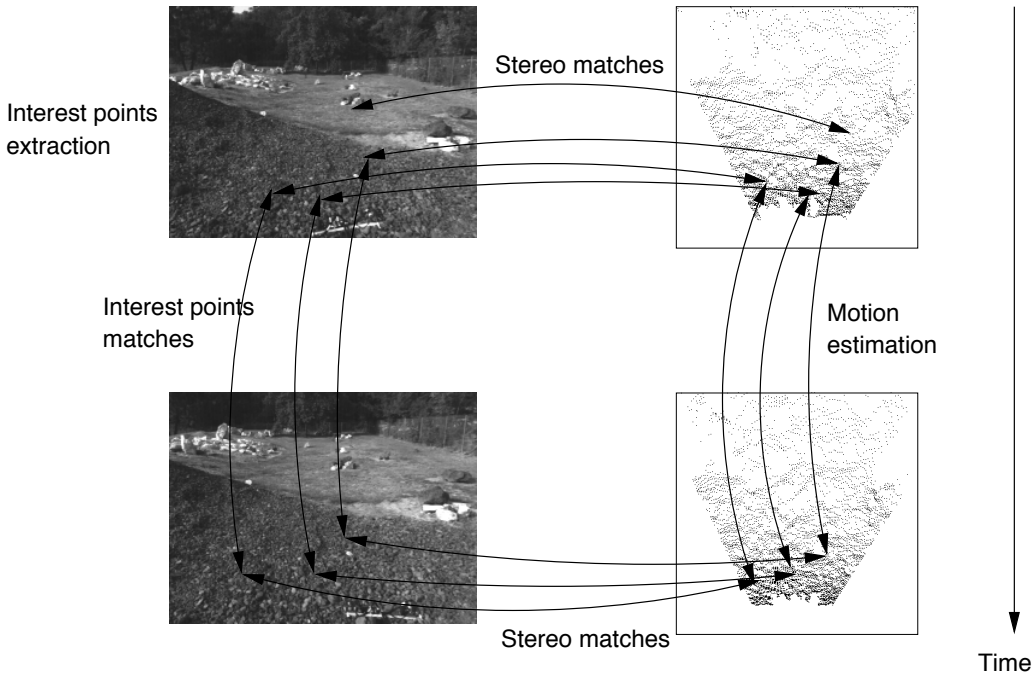


Figure 1.1: Principle of feature-based stereo VO

where ${}^{t-1}\boldsymbol{R}_t$ and ${}^{t-1}\boldsymbol{p}_{t-1,t}$ respectively represents the orientation and translation of the stereo reference frame at time t w.r.t to the same frame at $t-1$. The rover localisation is then estimated by the composition of the motion estimation matrices in between each acquired image pairs:

$$ {}^{O}\boldsymbol{T}_t = {}^{O}\boldsymbol{T}_1 \, {}^{1}\boldsymbol{T}_2 \ldots {}^{t-1}\boldsymbol{T}_t \tag{1.6} $$

where O is the origin reference frame, generally defined at the beginning of a mission/traverse in a specified site.

1.3.2 Details of the Involved Process

We briefly depict here the models of the four perception nodes. The most important part of the models for their controllability being the expression of the quality assessment functions $j(\boldsymbol{u}, \boldsymbol{z})$, a specific section is devoted to their derivation.

Interest point extraction. We have selected the ORB detector for its speed in terms of computation time while maintaining acceptable scale and rotation invariance [Rublee 2011]. A given number of features is extracted from the input image, if more features than requested can be extracted the best n are returned based on a quality score (the "response" of the detector). Following the node model notations (Eq. 1.3), we have:

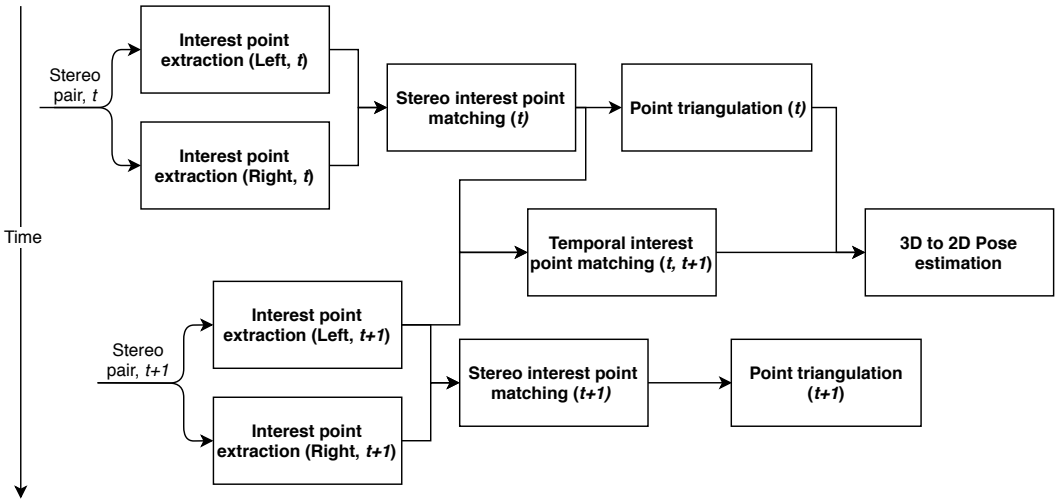


Figure 1.2: Feature-based visual odometry with 3D to 2D motion estimation workflow

$\boldsymbol{i}$ is an image,

$\boldsymbol{o}$ is the set of detected keypoints and associated descriptors,

$\boldsymbol{u}$ encompasses the target number of extracted features n, the pyramid levels, the scaling factor and a detection threshold.

Interest point matching. By comparing keypoints descriptors, the matcher associates keypoints between two images:

$\boldsymbol{i}$ is two sets of keypoints (with their associated descriptors) extracted in two images,

$\boldsymbol{o}$ is a set of matched (paired) keypoints,

$\boldsymbol{u}$ includes the number of k-best matches to return for each keypoint, a cross-check flag request and a criterion to validate matches (e.g. descriptors distance based, best x%).

Note that when applied to rectified stereo images, the matcher search for each feature is narrowed to the same vertical coordinate (*i.e.* the corresponding epipolar line) on the other image $\pm$ 1-2 pixels as an error offset.

3D points triangulation. The data association performed by the matcher on two stereo images allows to estimate their 3D coordinates through a simple triangulation process that does not exhibit any control parameter:

$\boldsymbol{i}$ is a set of keypoint matches established from a pair of stereo images, and the stereoscopic bench calibration matrices,

$\boldsymbol{o}$ is a set of 3D points,

$\boldsymbol{u}$ is an empty set.

Motion estimation. Two approaches for this process are possible: 3D to 3D and 3D to 2D [Scaramuzza 2011]. The first approach finds the roto-translation aligning two sets of corresponding points in $\mathbb{R}^3$, resorting to a least-squares method using singular value decomposition [Sorkine-Hornung 2017]. The second approach minimises the image reprojection error and is now more commonly applied ([Nistér 2004] showed that the 3D to 2D motion estimation performs better than the 3D to 3D due to the large depth error carried by the triangulation process).

We use the latter approach, along with a RANSAC scheme that allows to discard wrong matches possibly produced by the matcher:

$\boldsymbol{i}$ is a set of 3D points observed at time $t-1$, their corresponding image points in one camera frame at time t and the camera intrinsic calibration matrix,

$\boldsymbol{o}$ is a transformation matrix as in Eq. 1.5,

$\boldsymbol{u}$ is a set of parameters for the RANSAC scheme, and an optional motion guess (e.g. based on a motion model or on wheel odometry).

1.3.3 Context

Note that the context $\boldsymbol{z}$ has not been made explicit for any of the four nodes that define our instance of VO. Indeed, the context here mostly pertains to the environment (terrain visual appearance, illumination), which directly (and strongly) influences the first node of the pipeline, that is keypoints extraction. All the following processes are then consequently affected via the output of this first node (number of extracted keypoints and histogram of the associated responses).

1.3.4 Other Parameters

As introduced in Sec. 1.2.3, other parameters pertain to the overall compound. For visual odometry, deciding the process frequency not only accounts for different resource consumption, but also influences the precision of the estimation. Assuming the robot velocity is defined, the problem is dual with controlling the spatial frequency and linked to the *keyframe selection* problem. A high frequency reduces the likelihood of errors in tracking features but is not always achievable due to operational constraints in terms of resources, especially on space hardware. Furthermore, due to the presence of noise in the images and the keypoint extraction process, a too frequent estimation yields a higher drift compared to a slower estimation that is still able to correctly match features.

1.4 Data quality assessment

For control purposes, it is essential that each node is associated with one or more data quality assessment functions $j(\boldsymbol{z}, \boldsymbol{u})$ that express the influence of the control parameters $\boldsymbol{u}$ on the quality of the outputs $\boldsymbol{o}$ as a function of the inputs and context information.

Depending on the considered perception node, this step is not always straightforward. For instance, there is no generic quality metrics applicable to any type of data. While it is typical to resort to uncertainty measures (as covariance of the motion estimates), this is not always possible for many reasons, from the lack of uncertainty in the problem modelling to the impossibility to measure covariance for the input data. Furthermore, the explicit composition of the node data quality assessment functions is hardly feasible, given the variety of data and quality metric types.

Broadly speaking, there are two options to define such data quality assessment functions:

- Model-based approaches, from close-form derivations to rules-of-thumb.
- Data-based approaches, which calls for machine learning approaches where a model representing the characteristics of the data is produced for different contexts and both successful and unsuccessful cases.

Below we present some tentative data quality assessment functions for the nodes that define our VO scheme. They are preliminary, and show the difficulties of finding good measures to assess the quality of a perception process.

Interest point extraction. In order to assess the quality of extracted visual features, we define a figure of merit based on the keypoint response. The response is generally higher in strong features, which are in turn easier to match in successive frames. The average response over N keypoints can be computed

$$j = \sum_{n=0}^{N} \frac{response(n)}{N} \tag{1.7}$$

where N is the number of extracted keypoints and $response(n)$ is the response value for a given keypoint. Keeping in mind that any node which is not the last produces data for the next one, it is possible to reason in terms of utility for the subsequent process. In this case it is possible to learn or build predictive models for data fitness. For instance, the objective of keypoint extraction is to maximise their matchability. A learning based approach trying to predict this factor and to select matchable keypoints is presented in [Hartmann 2014] and has been applied to the Structure-from-Motion (SfM) problem. Alternatively, it is possible to discretise the set of input parameters for the feature extraction process and estimate over a representative dataset what configuration leads to the highest ratio of matched keypoints.

$$j = \frac{|\text{matches}|}{|\text{keypoints}|} \tag{1.8}$$

This may not be sufficient since we want to maximise correct matches rather than just the number of matches. We can then estimate the percentage of correct matches w.r.t. the response of keypoints. However, this does not turn out to be always higher in points with a high response value.

Interest point matching. Having enough matches is crucial for the execution of a visual odometry algorithm. The lack of good matches, or matches at all, easily leads to incorrect pose estimation. It is not only necessary to produce matches but also to ensure their correctness. Evaluating the matching distance can help to assess the quality of the matches, taking into account that a good number of matches can be more robust to outliers. In this case the function j can be the average of the matching distance and $\boldsymbol{u}$ mostly revolves around the percentage of matches to accept.

$$j = \frac{\sum_{m=0}^{M} d(m)}{\sum_{n=0}^{N} \frac{d(n)}{N}} \tag{1.9}$$

Eq. 1.9 computes a utility value based on the ratio between the sum of distances $d(m)$ of the M accepted matches and the average distance of the entire set of N matches. By not dividing the numerator by M (i.e. not using the average), the function favours larger sets of validated matches.

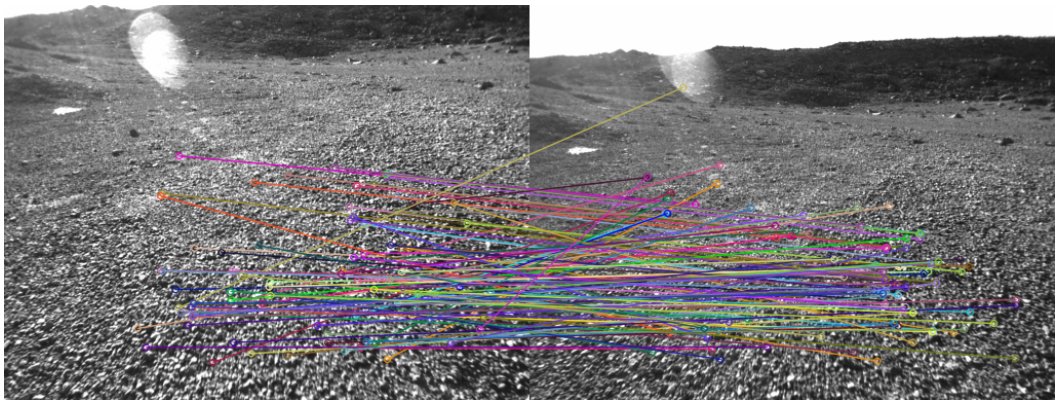
Triangulation. A correct stereo matching process enables to produce 3D points through triangulation. The precision of the keypoint extraction and of the matches, along with their position w.r.t. the camera positions impacts the 3D estimated covariance. As in [Beder 2006], it is possible to compute the covariance matrix C_e of a euclidean 3D point starting from its corresponding coordinates in the stereo images and their respective covariance matrices in 2D. In stereovision, for small angle differences between two points, it is common to have a high covariance over the depth axis. It is desirable to minimise the presence of uncertain points in order to reduce the error at pose estimation stage.

Motion estimation. Finally, as the 3D to 2D estimation is incorporated in a RANSAC scheme it is possible to consider the number of inliers given a desired reprojection error, which can be tuned. It is worth to note that [Ferraz Colomina 2014] and [Urban 2016] introduced methods for including observation uncertainty into the PnP problem. It is part of our planned future works to incorporate the stochastic aspect into the 3D to 2D pose estimation node. It is also worth to remark that in case of a 3D to 3D estimation, an uncertainty measure in form of a covariance matrix is naturally obtained during the computation of the rotation matrix [Sorkine-Hornung 2017].

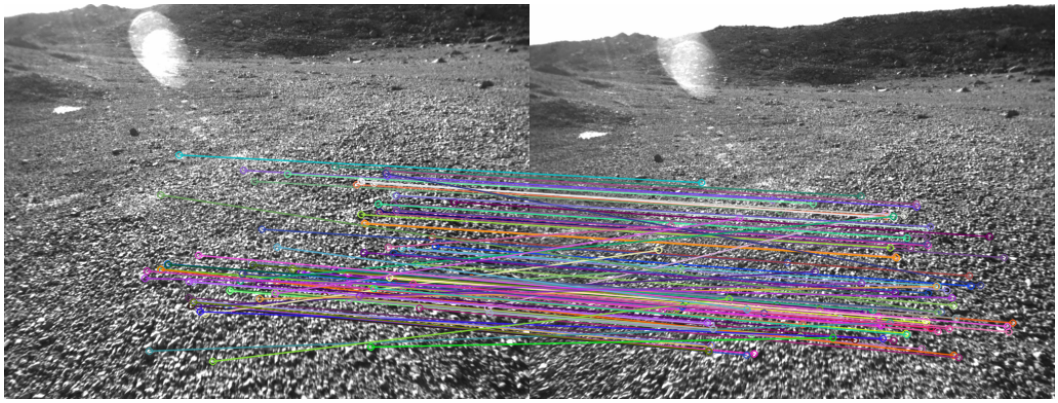
1.5 Parameters Control and Manipulation

We identify two control threads: *passive-active perception*, on which we focus in this paper, bounded in the control of the perception process itself, and *active perception*, aiming to tighten the link between control, perception and planning by acting at different abstraction level to serve the perception layer. In this section we show the execution of our visual odometry algorithm with data evaluation of several nodes in different contexts.

Fixing the data quality assessment functions, we aim to control the set of input parameters $\boldsymbol{u}$ to apply Eq. 1.2 along with other compounds parameters. Despite an autonomous



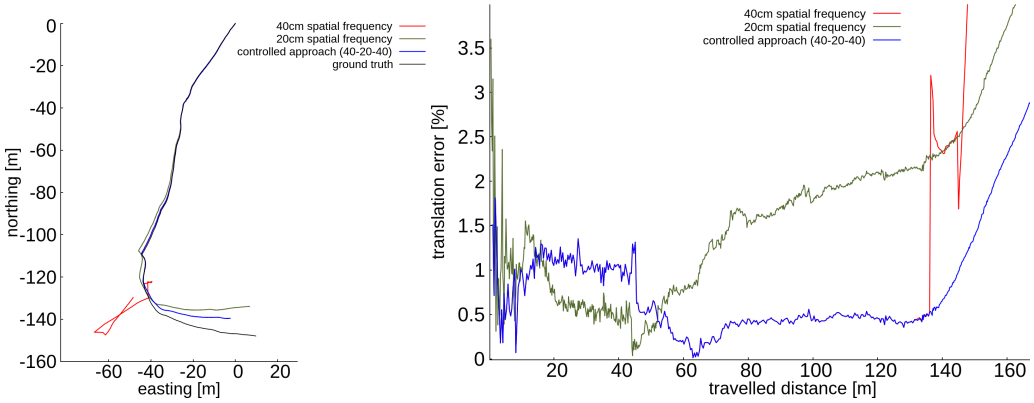
(a) Fine gravel results in a noisy image producing wrong matches between two time instants (left t, left $t + 1$). Far points are filtered by default due to high depth error.



(b) The same image produces many more acceptable matches if compared with another after a smaller motion has been performed.

Figure 1.3: Matching quality difference changing the spatial frequency

reconfiguration of perception nodes is outside the scope of this paper, it is worth to show control means and control points to our use case. The advantage of our approach lies in the easiness of changing the input parameters of a node at runtime, especially if done in the scheme of a predictive model, *i.e.* before faults and with no need to backtrack.



(a) Pose estimation using different strategies. Coordinates are expressed w.r.t. the origin

(b) Errors in percentage of the travelled distance

Figure 1.4: In green, estimation performed at ∼20 cm/frame spatial frequency, the red line, a ∼40 cm/frame, is largely overlapped with the blue line, a controlled execution using both frequencies at different stages. The ground truth is represented with a black line. The controlled approach proves to be more accurate. The 40 cm error suddenly jumps after not finding inliers in the fine gravel area.

1.5.1 Data Sets

The chosen data set is the Devon Island Rover Navigation Dataset collected by the Autonomous Space Robotics Lab of the University of Toronto [Furgale 2012]. The data has been collected at a Mars/Moon analogue site in the Canadian High Arctic region. It represents relevant characteristic distinctive of planetary-like environments. The data set features a set of of rectified stereo pair images collected over a 10 km traverse. Two different resolutions are provided: 1280x960 and 512x384. To simulate a planetary exploration setup we use the smaller resolution, which is less intensive in terms of computation. The results will be shown for the first part of the traverse, approximately 200 m. The result of visual odometry will be evaluated against the ground truth obtained with a differential GPS.

1.5.2 Experiments

It is impossible to manually explore the huge search space of all the parameters involved in visual odometry. Driven by our knowledge, as most roboticians do nowadays, we configured our algorithm based on a priori knowledge of the scene, experience and empirical

results obtained by trial and error. Nonetheless, a robot cannot always predict what situations it is going to face and has to prove adaptive to the operational context.

At some point during the traversal, the terrain changed from well textured rocks producing repeatable keypoints to a fine gravel layer which made the matching process much harder (Fig. 1.3a). By reducing the spatial frequency between two consecutive frames a much better tracking can be achieved (Fig. 1.3b). This could lead to think a higher frequency would yield better results in any case, but as mentioned in Sec. 1.3.4, a too low spatial frequency negatively impacts the estimation, leading to a higher drift. In this case, a relevant the criteria to select the frequency is the percentage of inliers produced by the pose estimation (Fig. 1.6b). The algorithm kept a 40 cm spatial frequency for the first part of the traverse, reducing it to 20 cm in case of fine gravel areas. Once the scene returned better matches, the spatial frequency is reset to the initial value to limit error accumulation (Fig. 1.4).

As indicated by the drop in the number of inliers, it can be beneficial (or even necessary) to modify a subset of the input parameters. For instance, increasing the number of keypoints in the first stage of the algorithm allows to feed a higher number of 3D points to the pose estimation stage, slightly improving its performances (Fig. 1.5). Note that extracting a too high number of keypoints is demanding in terms of computational time and is done only when necessary, as indicated by the the control point. Without action, *i.e.* proceeding with the same spatial frequency, the algorithm produced five gross errors in the pose (jumps), with erroneous translations and rotations (Fig. 1.6a).

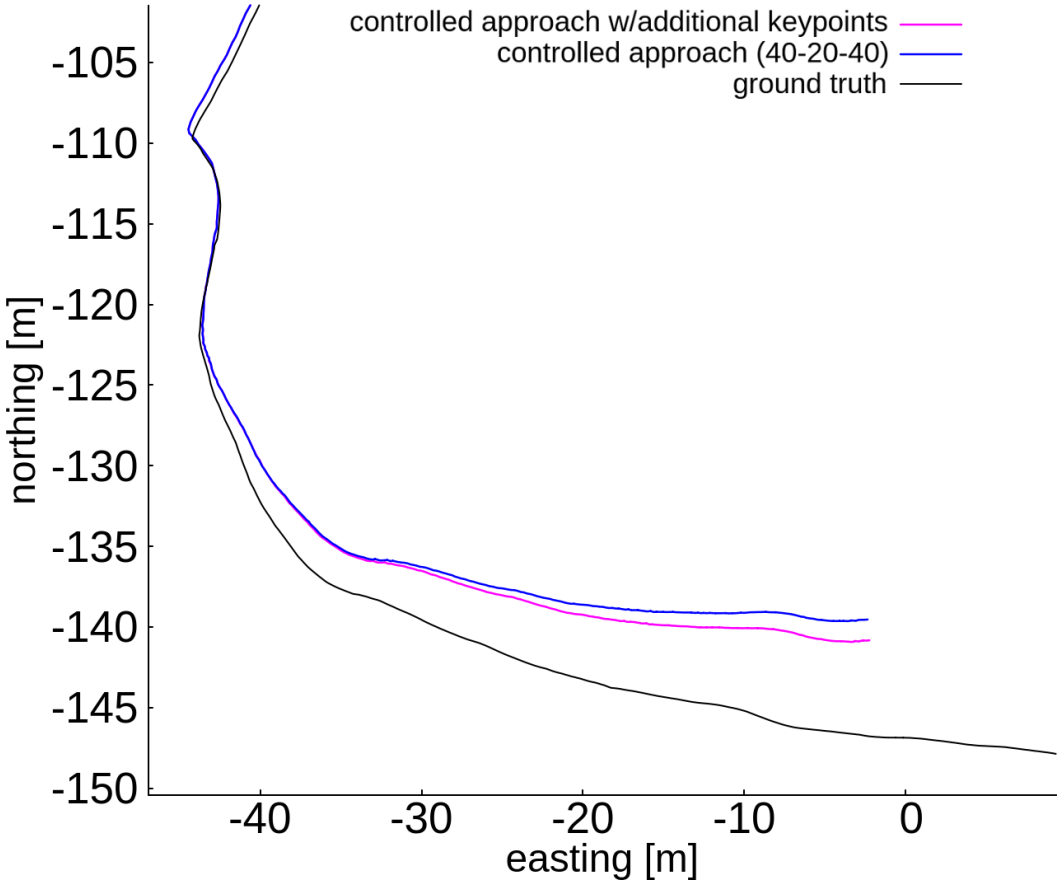
1.6 Conclusions and future works

Perception processes can be formally modelled but it is still a hard task to link their model to the instantiations. We proposed an initial model to help controlling these processes. Despite a scheme to autonomously reconfigure the nodes is yet to be defined, this work shows how it is possible to evaluate either atomic functions or compounds through data quality assessment functions. Monitoring these figures of merit can trigger control of the perception node and compound parameters.

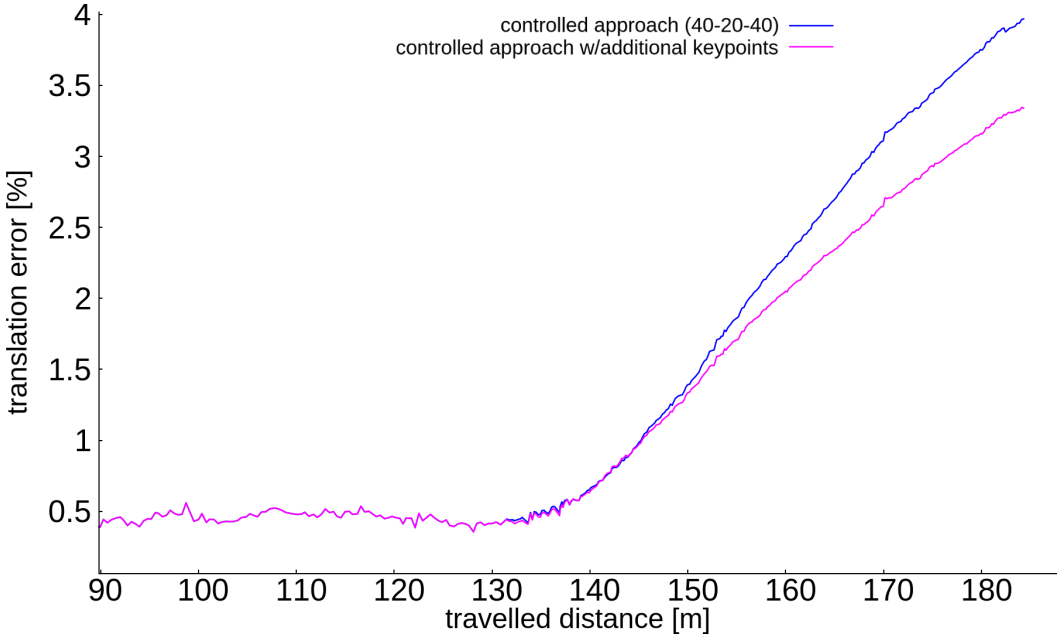
Future extensions of this work directly point towards the definition of a reasoning framework to control perception processes. Adapting to the slightest change in the operational scenario is necessary to produce the best possible results regardless of the working conditions. Nonetheless, the search space for the optimal set of controllable parameters is very large, making a blind search approach unfeasible. Resorting to predictive models helps dealing with this problem by supplying a belief for an optimal configuration. This kind of models can be trained and used to find an initial configuration for the input parameters. The search can then be performed in the neighbourhood of this configuration.

Furthermore, additional elements can be leveraged to optimise the perception outcome. Controlling the robot motion can indirectly achieve the same result as controlling the process execution frequency. Additionally, the viewpoint selection problem directly conditions the output of all the nodes in the compound. Steering the pan-tilt unit of a camera towards more textured areas can make the difference between an efficient estimation and the lack

of convergence. These all represent actions that can be carried out serving the perception layer, driven by the observations carried out by data quality assessment functions.

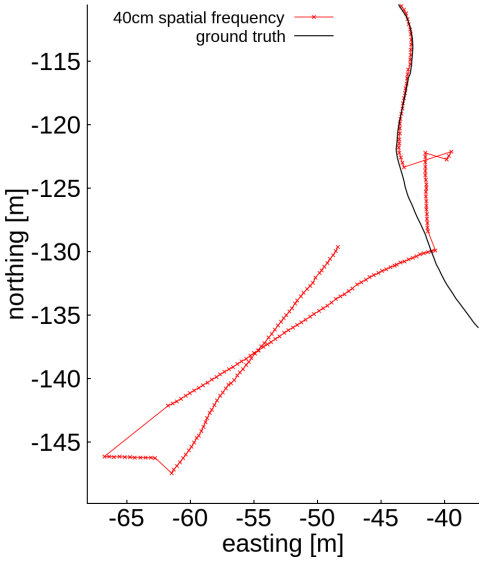


(a) Pose estimation increasing the number of features during a controlled execution

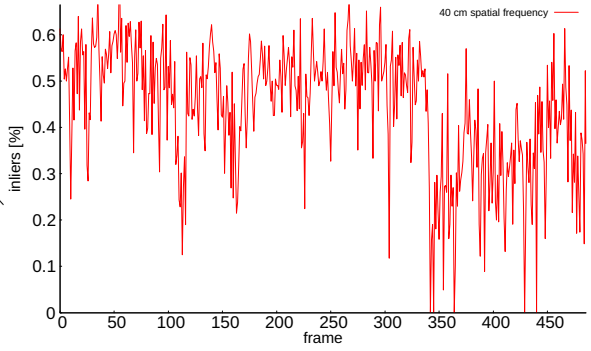


(b) Errors in percentage of the travelled distance

Figure 1.5: Temporarily increasing the number of extracted keypoints in noisy area can help achieving significantly better motion estimation



(a) Close up view of erroneous estimation with low frequency



(b) Inliers percentage at pose estimation stage

Figure 1.6: Maintaining a low spatial frequency fixed, the algorithm produces five completely wrong estimations which are reflected in five drops of inliers to zero.

CHAPTER 2

Simultaneously learning corrections and error models for geometry-based visual odometry methods

Contents

Abstract

This paper fosters the idea that deep learning methods can be used to complement classical visual odometry pipelines to improve their accuracy and to associate uncertainty models to their estimations. We show that the biases inherent to the visual odometry process can be faithfully learned and compensated for, and that a learning architecture associated with a probabilistic loss function can jointly estimate a full covariance matrix of the residual errors, defining an error model capturing the heteroscedasticity of the process. Experiments on autonomous driving image sequences assess the possibility to concurrently improve visual odometry and estimate an error associated with its outputs.

2.1 Introduction

Visual odometry (VO) is a motion estimation process successfully applied in a wide range of contexts such as autonomous cars or planetary exploration rovers [Scaramuzza 2011]. Seminal works largely resorted to stereovision. By tracking point features in images, 3D points correspondences are used to recover the motion between two stereoscopic acquisitions. The integration of elementary motions yields an estimate of the robot pose over its course. Continuous work on VO led to a well established processes pipeline, composed of feature extraction, matching, motion estimation, and finally optimization. This scheme has been extended to single camera setups, in which case motions are estimated up to a scale factor, retrieved *e.g.* by fusing inertial information. Direct methods for VO have also recently been proposed. They bypass the feature extraction process and optimize a photometric error [Engel 2018]. These methods overcome the limits of sparse feature-based methods in poorly textured environments or in presence of low quality images (blurred), and they have proven to be on average more accurate.

The advent of convolutional neural networks (CNN) sprouted alternate solutions to VO. The full estimation process can be achieved by deep-learning architectures in an end-to-end fashion (see *e.g.* [Konda 2015, Li 2017], and especially [Wang 2018] – note these work consider the monocular version of the problem, leaving the scale estimation untackled). In such approaches, the system has to learn the various information necessary to perform vision-based egomotion estimation, which can be a daunting task for a CNN. This paper builds upon existing work that exploits a CNN to predict *corrections* to classic stereo VO methods [Peretroukhin 2017], aiming at improving their precision. We argue that complementing classical localization processes with learning-based methods can return better results than delegating the full pose estimation process to a CNN. On the one hand, classical localization processes do not output totally erroneous poses, as a learning approach could when confronted with features unseen in the training set. On the other hand, classical processes can be monitored by some explicit indicators (e.g. number of tracked points in VO, inliers, etc.), so as to detect erroneous cases. Our developments consider that visual odometry estimation errors do not have zero mean, as assessed in *e.g.* [Dubbelman 2012, Peretroukhin 2014], and provide corrections that improve the precision of VO. Furthermore, at the same time, they produce a full error model for each computed motion estimation (in form of a Gaussian model), akin to [Liu 2018]. This is a significant achievement, as it is generally complex to derive precise error models for geometrical VO methods.

2.2 Problem statement and related work

Consider a robot moving in a three dimensional environment. Let $\boldsymbol{x}_i \in \mathbb{R}^6$ be its pose (3 translations and 3 orientations) at time i in a given reference frame. The actual motion (ground truth) between time instants i and $i+1$ is represented by a homogeneous transformation matrix ${}^i\mathbf{T}_{i+1}$.

A vision-based motion estimator uses raw image data $\boldsymbol{\mathcal{I}}_i \in \mathbb{R}^n$ to obtain an estimate

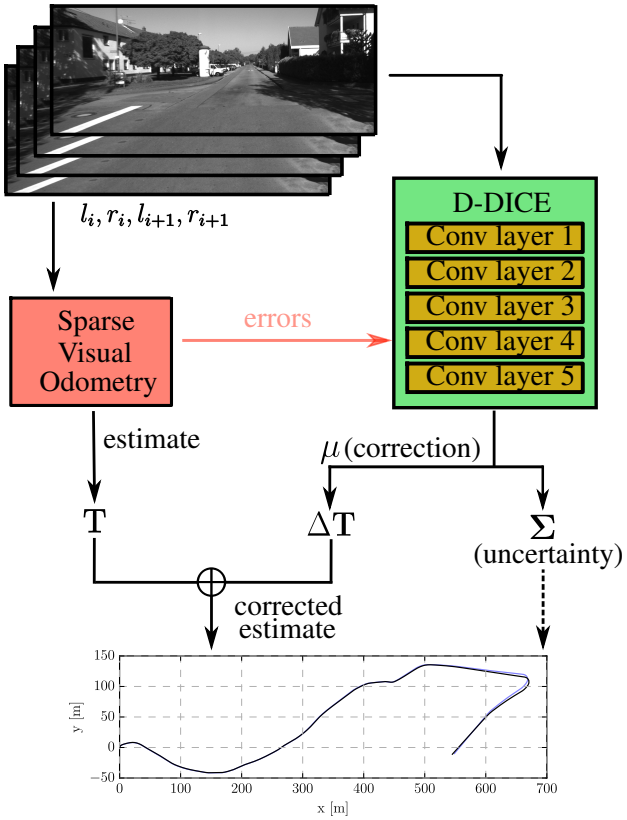


Figure 2.1: D-DICE produces corrections to classic visual odometry methods in a probabilistic framework. The system generates full covariance matrices that can be used to minimize errors in pose-graph optimization.

${}^{i}\hat{\mathbf{T}}_{i+1}$. In the VO case, the raw data $\boldsymbol{\mathcal{I}}_i$ is a pair of monocular or stereoscopic images captured at two different time instants i, $i+1$ (*i.e.* 2 or 4 images). The error $\boldsymbol{e}_i$ of VO is:

$$\boldsymbol{e}_i \;=\; {}^{i}\mathbf{T}_{i+1} \;\cdot\; {}^{i}\hat{\mathbf{T}}_{i+1}^{-1} \tag{2.1}$$

We can create a dataset $\mathcal{D} = \{\boldsymbol{\mathcal{I}}_i, \boldsymbol{e}_i | \forall i \in [1, d]\}$, where d is the size of the dataset. The literature provides two different approaches to leveraging this type of dataset. The two approaches enhance a classic VO process with learning to either estimate *(i)* a *motion correction* to apply to ${}^{i}\hat{\mathbf{T}}_{i+1}^{-1}$, thus improving its accuracy [Peretroukhin 2017], or *(ii)* an *error model* associated with ${}^{i}\hat{\mathbf{T}}_{i+1}^{-1}$ [Liu 2018], thus allowing its fusion with any other motion or pose estimation process. Alternatively, with the same semantic, substituting errors with actual motion transforms, it is possible to directly learn the motion estimate and associated error [Wang 2018].

2.2.1 Directly learning VO and an error model

The work in [Wang 2018] introduces an end-to-end, sequence-to-sequence probabilistic visual odometry (ESP-VO) based on a recurrent CNN. ESP-VO outputs both a motion es-

timate ${}^{i}\hat{\mathbf{T}}_{i+1}^{-1}$ and an associated error. The learned error model is a diagonal covariance matrix, hence not accounting for possible correlations between the different motion dimensions. It is unclear how the probabilistic loss is mixed to the mean squared error of the Euclidean distance between the ground truth and the estimated motions. Finally, the authors make use of a hand-tuned scaling factor to balance rotation and translation. The article presents significant results obtained on a large variety of datasets, with comparisons to state-of-the-art VO schemes. The results show that ESP-VO is a serious alternative to classic schemes, all the more since it also provides variances associated with the estimations. Yet, they are analysed over whole trajectories, which inherit from the random walk effect of motion integration, and as such do not provide thorough statistical insights – *e.g.* on the satisfaction of the gaussianity of the error model or on the evaluation of the mean log-likelihood.

2.2.2 Learning corrections to VO

The work presenting DPC-Net [Peretroukhin 2017] learns an estimate of $\boldsymbol{e}_i$, which is further applied to the VO estimate ${}^{i}\hat{\mathbf{T}}_{i+1}^{-1}$ to improve its precision. The authors introduce an innovative pose regression loss based on the SE(3) geodesic distance modelled with a vector in Lie algebra coordinates. Instead of resorting to a scalar weighting parameter to generate a linear combination of the translation and rotation errors, the proposed distance function naturally balances these two types of errors. The loss takes the following form:

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2} g(\boldsymbol{\xi})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} g(\boldsymbol{\xi}) \tag{2.2}$$

where $\boldsymbol{\xi} \in \mathbb{R}^6$ is a vector of Lie algebra coordinates estimated by the network, $g(\boldsymbol{\xi})$ computes the equivalent of (2.1) in the Lie vector space, and $\boldsymbol{\Sigma}$ is an empirical average covariance of the estimator pre-computed over the training dataset. Such covariance matrix cannot be used as an uncertainty measure but only as a balancing factor between rotation and translation terms. The paper provides statistically significant results showing that DPC-Net improves a classic feature-based approach, up to the precision of a dense VO approach. In particular, it alleviates biases (*e.g.* due to calibration errors) and environmental factors. The system interlace low rate corrections with estimates produced by the underlying VO, which processes all the images, using a pose-graph relaxation approach.

2.2.3 Learning an error model of VO

Inferring an error model for VO comes to learn the parameters of a predefined distribution to couple VO with uncertainty measures. The work in [Liu 2018] introduces DICE (Deep Inference for Covariance Estimation), which learns the covariance matrix of a VO process as a maximum-likelihood for Gaussian distributions. Nevertheless, it considers the distribution over measurement errors as a zero-mean Gaussian $\mathcal{N}(0, \boldsymbol{\Sigma})$. Such model is acceptable for unbiased estimators, which unfortunately it is often not the case of VO. Yet, the authors show that their variance estimates are highly correlated with the VO errors, especially in case of difficult environmental conditions, such as large occlusions.

2.3 Simultaneously learning corrections and uncertainties

To jointly estimate a correction to the VO process and a full error model *after having applied the correction*, we initially enhance the network output of [Liu 2018] adding a vector $\boldsymbol{\mu}_i \in \mathbb{R}^6$ to the output layer to account for biases in the estimator. Such vector is incorporated in the negative log-likelihood loss that is derived as follows. Given a dataset $\mathcal{D}$ of size d, where the observations $\{\boldsymbol{e_1}, \dots, \boldsymbol{e_d}\}^\mathsf{T}$ of VO errors are assumed to be independently drawn from a multivariate Gaussian distribution, we can estimate the parameters of the Gaussian as

$$\underset{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}}{\arg\max} \sum_{i=1}^{d} p(\boldsymbol{e}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2.3}$$

This is equivalent to minimize the negative log-likelihood

$$\underset{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}}{\arg\min} \sum_{i=1}^{d} -\log\left(p(\boldsymbol{e}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right) \tag{2.4}$$

$$= \underset{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}}{\arg\min} \sum_{i=1}^{d} \log|\boldsymbol{\Sigma}_i| + (\boldsymbol{e}_i - \boldsymbol{\mu}_i)^\mathsf{T} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{e}_i - \boldsymbol{\mu}_i) \tag{2.5}$$

$$\approx \underset{f_{\boldsymbol{\mu}_{1:d}}, f_{\boldsymbol{\Sigma}_{1:d}}}{\arg\min} \sum_{i=1}^{d} \log|f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)| + (\boldsymbol{e}_i - f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i))^\mathsf{T} f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)^{-1} (\boldsymbol{e}_i - f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)) \tag{2.6}$$

We split the output of the network in two parts: the mean vector $f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)$ and the covariance matrix $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)$, where $f(\boldsymbol{\mathcal{I}}_i)$ represents the full output given a pair of stereo images.

2.3.1 Retrieving a valid covariance matrix

To enforce a positive definite covariance matrix we tested two different matrix decompositions.

2.3.1.1 LDL

The first one is the LDL matrix decomposition as in [Liu 2018], to which the reader can refer to for a complete description. The predicted covariance matrix $f_{\boldsymbol{\Sigma}_i}(\boldsymbol{\mathcal{I}}_i)$ is generated through a vector $\boldsymbol{\alpha}_i = [\boldsymbol{l}_i, \boldsymbol{d}_i]^\mathsf{T}$, with $\boldsymbol{l}_i \in \mathbb{R}^{(\frac{n^2}{2} - \frac{n}{2})}$ and $\boldsymbol{d}_i \in \mathbb{R}^n$. We have then

$$\boldsymbol{\Sigma}_i \approx f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i) = L(\boldsymbol{l}_i) D(\boldsymbol{d}_i) L(\boldsymbol{l}_i)^\mathsf{T} \tag{2.7}$$

where $\boldsymbol{l}_i$ and $\boldsymbol{d}_i$ are the vectors containing the elements of the respective $\boldsymbol{L}$ and $\boldsymbol{D}$ matrices. The LDL decomposition is unique and exists as long as the diagonal of $\boldsymbol{D}$ is strictly positive. This can be enforced using the exponential function $exp(\boldsymbol{d_i})$ on the main diagonal. By doing so, the computation of its log-determinant, i.e. the first term of (3.7),

can be reduced to $sum(\boldsymbol{d}_i)$, that is the sum of the elements in the vector $\boldsymbol{d}_i$. In the second term $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)^{-1}$ is replaced by the LDL product.

2.3.1.2 LL

Alternatively, it is possible to resort to the classical Cholesky decomposition. In this case $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)$ is replaced by the $\boldsymbol{L}\boldsymbol{L}^*$ product, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{L}^*$ is its conjugate transpose. We consider $\boldsymbol{L}\boldsymbol{L}^\mathsf{T}$ as no complex number is involved in our work. The $\boldsymbol{L}$ matrix can be generated through a vector $\boldsymbol{l}_i \in \mathbb{R}^{(\frac{n^2}{2}+\frac{n}{2})}$. This decomposition also has nice properties around its log-determinant, as it is easy to prove $\log|\boldsymbol{L}\boldsymbol{L}^\mathsf{T}| = 2\sum_i \log(\boldsymbol{L}_{ii})$.

We tested both decompositions and did not experience relevant changes in the network accuracy. Although the presence of the exponential term in the LDL can alter the numerical stability of the loss function, clamping its value mitigates this problem. Therefore, we decided to pursue training using the first method in order to introduce fewer variables that could affect the comparisons in the results (Sec.2.4.3).

2.3.2 Optimization problem

We incorporate the LDL formulation for the covariance matrix in the negative log-likelihood. Replacing $f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)$ with the estimated mean output vector $\hat{\boldsymbol{\mu}}_i$ we finally obtain

$$\begin{aligned}\mathcal{L}(\boldsymbol{\mathcal{I}}_{\mathbf{1:d}}) = \underset{\hat{\boldsymbol{\mu}}_{1:d},\boldsymbol{\alpha}_{1:d}}{\arg\min} \sum_{i=1}^{d} sum(\boldsymbol{d}_i) + \\ (\boldsymbol{e}_i - \hat{\boldsymbol{\mu}}_i)^\mathsf{T}(L(\boldsymbol{l}_i)D(exp(\boldsymbol{d}_i))L(\boldsymbol{l}_i)^\mathsf{T})^{-1}(\boldsymbol{e}_i - \hat{\boldsymbol{\mu}}_i)\end{aligned} \tag{2.8}$$

Formulating the problem as in (3.8), the second term of the loss function loosely recalls the formulation of the Lie algebra loss in (2.2). The covariance matrix in this case is learned in relation to the input, capturing the heteroscedastic uncertainty of each sample. The learned covariance matrix acts as in [Kendall 2017a], weighing position and orientation errors. The main difference resides in the nature of the learned uncertainty, homoscedastic vs heteroscedastic: through back-propagation with respect to the input data, [Kendall 2017b], we aim to learn a heteroscedastic error. Assuming that errors can be drawn from a distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $\boldsymbol{\mu}_i$ matches the expected value for the predicted distribution. This corresponds to the desired correction in our case. At the same time, we estimate a covariance matrix $\boldsymbol{\Sigma}_i$, returning an uncertainty measure relative to each particular input and pose after the predicted correction. Inverting Eq.2.1, corrections are applied as

$${}^{i}\mathbf{T}_{i+1} \approx {}^{i}\mathbf{T}^{VO}_{\hat{i+1}} \cdot {}^{\hat{i+1}}\mathbf{T}^{corr}_{i+1} \tag{2.9}$$

that is the composition of the pose estimate produced by VO and the estimated correction produced by the neural network.

2.4 Experiments

2.4.1 Setup

2.4.1.1 Dataset

We carry out experiments using the KITTI dataset, which provides various sequences of rectified images acquired while driving in urban areas [Geiger 2012]. Depth images are generated using a semi global block matching algorithm to generate and filtered using a weighted least squares filter. We train the network splitting train and validation trajectories in different configurations: for all results shown here we trained using sequences `04` to `10` (which share the same calibration parameters), excluding one for testing and one for validation purposes. Most of our experiments are validated using four sequences (`05`, `06`, `09`, `10`).

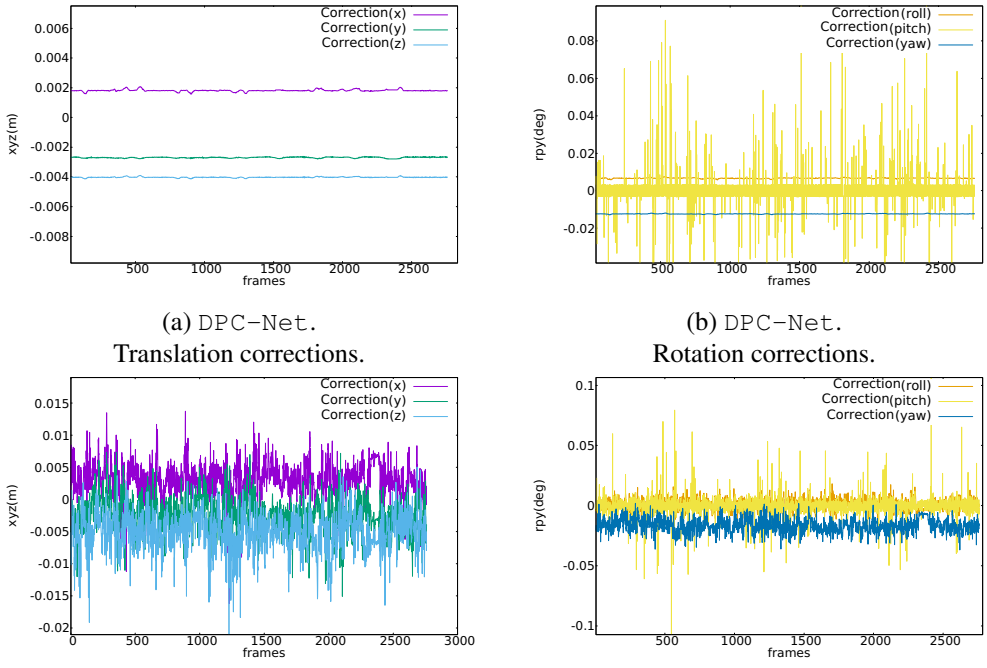
For the initial motion estimates, we use the open-source VO implementation *libviso2* [Kitt 2010]. It is a feature-based approach, that uses a Kalman Filter in combination with a RANSAC scheme to produce SE(3) estimates using rectified stereoscopic pairs. The estimated corrections, and relative uncertainties, are expressed in camera frame (z axis pointing forward), and the Tait-Bryan angles are defined w.r.t. this reference frame (*i.e.* yaw encodes rotations around the optical axis z).

2.4.1.2 Evaluation metrics

To evaluate the precision of the motion estimates, we make use of the absolute trajectory error (ATE) metric [Zhang 2018, Geiger 2012]. It is defined as

$$\begin{aligned} ATE_{rot} &= \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{e}_i^R||_2 \\ ATE_{trans} &= \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{e}_i^t||_2 \end{aligned} \tag{2.10}$$

where $\boldsymbol{e}_i^t$ is the difference between the estimated position and the ground truth, and $\boldsymbol{e}_i^R$ is the rotation angle, in angle-axis representation, of the product $\boldsymbol{R}_i\hat{\boldsymbol{R}}_i^{\mathsf{T}}$. ATE comes with the advantage of returning a single scalar to evaluate rotation and translation errors, making it easy to compare them among multiple estimators. At the same time, its main disadvantage lies in the lack of robustness to isolated poor estimations and their relative position in the trajectory [Zhang 2018, Geiger 2012, Kümmerle 2009]. We use ATE for early architectural choices (section 2.4.2), but in order to provide a more informative analysis, we use relative error statistics. The idea of relative error is to select segments of predefined lengths of the trajectory and compute the error on all the aligned sub-trajectories. This way, it is possible to obtain statistics on the tracking error (mean and standard deviation) and evaluate it for short or long-term accuracy [Peretroukhin 2017, Zhang 2018]. In our evaluations, we select sub-trajectories of 10, 20, 30, 40 and 50 % of the full trajectory length.



(a) DPC-Net. Translation corrections.

(b) DPC-Net. Rotation corrections.

(c) D-DICE. Translation corrections.

(d) D-DICE. Rotation corrections.

Figure 2.2: Estimated translation and rotation corrections using DPC-Net and D-DICE over time.

2.4.1.3 Network structures

We initially compared the results produced using the architectures presented in DPC-Net [Peretroukhin 2017] and DICE [Liu 2018]. The first trial has been to adapt the loss defined in Eq. 3.8 to DPC-Net. We noticed that the mean output vector was still rather constant throughout entire trajectories, regardless of the dataset, and the same behavior was experienced using the loss defined in Eq. 2.2. Similar tests were conducted with DICE. We experienced problems in reducing the average mean error along all the six dimensions at the same time, and an increase in its standard deviation. Alleging these issues as being caused by the shallow architecture of DICE, we modified its network structure, first removing the max pool layers to preserve spatial information [Handa 2016], and achieving dimension reduction by setting the stride to 2 in early layers. We also increased the number of convolutional filters to tackle the estimation of both corrections and error models, adding dropout after each layer to prevent over-fitting. For the rest of the paper, we refer to this network as Deeper-DICE (D-DICE, Table 2.1).

The convolutional layers are followed by two fully connected layer, respectively composed of 256 and 27 output units. In the six-dimensional case, we need 21 values for the LDL decomposition and 6 for the mean vector. We trained the network using both monocular images and stereo images. Additionally, we explored if pairing monocular images to their corresponding disparity maps could be beneficial to the training process, even if the

disparities were produced separately from VO.

We used the Adam optimizer with a learning rate of 1e-04 and halted the learning when test and train loss started diverging. All the experiments have been carried out using an Nvidia GeForce RTX 2080 Ti with a batch size of 32.

Layer	Kernel size	Stride	Number of channels
conv1	5x5	2	64
conv2	5x5	2	128
conv3	3x3	2	256
conv4	3x3	2	512
conv5	3x3	1	1024

Table 2.1: D-DICE convolutional architecture.

2.4.1.4 Influence of input data

We tested the proposed architecture and loss using three different types of input data: monocular images, monocular images with associated disparity images, and stereo image pairs. Table 2.2 shows the ATEs on two sequences for the three input data: the stereo setup outperforms the two others in both sequences. Ideally, one would expect the best results with monocular images associated with disparity images: indeed, with such data the network does not have to infer depth information. However, with respect to monocular images, this input data does not improve the ATE metric as much as stereo pairs. It is likely that efficiently exploiting disparity images would require a specific convolutional architecture, as their nature differs significantly from intensity images.

Throughout the remainder of the paper, we will provide results with D-DICE obtained using stereo images, as for DPC-Net.

ATE	Seq.	D-DICE (mono)	D-DICE (mono+disp)	D-DICE (stereo)
Trans. (m)	05	21.54	19.73	**10.23**
	10	12.15	8.56	**7.20**
Rot. (deg)	05	5.81	5.30	**2.62**
	10	3.01	2.47	**2.28**

Table 2.2: Mean Absolute Trajectory Error (ATE) on two validation sets for three different input data types.

2.4.2 Qualitative Evaluation

To validate the choice of the proposed architecture and loss, we discuss preliminary results obtained exploring different possibilities in this regard.

2.4.2.1 Loss comparison

To show the impact of dynamically estimating a covariance matrix for each transform, we adapt the loss in Eq. 2.2 to our proposed architecture. Table 2.3 compares the ATEs obtained with this loss and the one we proposed in Eq. 3.8. As we are going to showcase in further results, the negative log-likelihood loss generally outperforms the loss associated with DPC-Net. At the same time, we experience a more stable improvement in translation than in rotation. This is especially true in sequences `09` and `10`, where VO presents smaller errors (see figure 2.3). We associate this behavior to the need for tailored large uncertainties to poor estimations, as opposed to sequences where the necessity of corrections is lower due to a better tracking by VO. Besides, our loss yields the prediction of an error model, which can be used to further reduce trajectory errors, as shown in section 2.4.3.2.

ATE	**Estimator**	`05`	`06`	`09`	`10`
Translation (m)	D-DICE + Lie	20.22	5.51	19.51	11.60
	D-DICE + NLL	**10.23**	**4.65**	**16.50**	**7.20**
Rotation (deg)	D-DICE + Lie	4.19	1.85	**2.30**	**1.77**
	D-DICE + NLL	**2.62**	**0.84**	2.79	2.28

Table 2.3: Comparison of ATEs obtained with the Lie loss and the negative log-likelihood.

2.4.2.2 Architecture comparison

Selecting an appropriate architecture was driven by a few factors. A first issue was represented by the significantly higher incidence of numerical instability when pairing it with a NLL loss, particularly in the matrix inversion and in the exponential. We also noticed that the DPC-Net system (architecture + loss) tends to output rather constant corrections throughout a whole sequence, certainly compensating biases. This behavior was less prominent when training D-DICE with the same loss. The two systems behave quite differently, as can be seen in Fig. 2.2, with D-DICE exhibiting more data-dependent corrections. The architectures of DICE and D-DICE are thoroughly compared in Section 2.4.3.2.

2.4.3 Quantitative Evaluation

We evaluate D-DICE performances in two schemes. First, we make use of the estimated mean vector as corrections for each image pair (Sec. 2.4.3.1). This approach does not use the uncertainty model and ensures a fair comparison with DPC-Net systems producing corrections outside a probabilistic context. With respect to uncertainty models (Sec. 2.4.3.2), we test the Gaussian assumption and compare log-likelihood values. Additionally, we use the covariance information to further reduce trajectory errors. In a first step, we correct the trajectory as in Eq. 2.9. Subsequently, we solve a pose-graph optimization problem, weighting errors on the inverse of the covariance matrix and manually adding a ground-truth loop closure.

2.4.3.1 Trajectory correction

We evaluate the accuracy of the correction produced by D-DICE with respect to the chosen baseline VO solution (Fig. 2.3). We also compare to DPC-Net, trained on the same data split as D-DICE. In these comparisons, both systems estimate transforms with consecutive stereo pairs, and the uncertainty estimates provided by D-DICE are not used. D-DICE outperforms DPC-Net in all the selected sequences (Table 2.4). While both systems constantly improve *libviso2*, we noticed a larger improvement in the translation errors especially on the y-axis (upwards) – the vertical drift is one of the most prominent weaknesses of this VO implementation using the KITTI dataset. For more in-depth results, we show mean relative segment errors with associated standard deviations in Table 2.5.

ATE	**Estimator**	05	06	09	10
Translation (m)	Sparse VO	25.90	7.75	52.78	11.79
	DPC-Net	11.04	5.83	23.26	11.67
	D-DICE	**10.23**	**4.65**	**16.49**	**7.20**
Rotation (deg)	Sparse VO	7.73	3.77	6.92	5.19
	DPC-Net	**2.10**	2.25	4.18	4.19
	D-DICE	2.62	**0.84**	**2.79**	**2.28**

Table 2.4: Mean Absolute Trajectory Error (ATE) before and after applying corrections to *libviso2* VO

2.4.3.2 Uncertainty estimation

Since we assume a Gaussian error model $\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we can measure its relevance by checking the fraction of samples that do not respect the following inequality:

$$\mu_i - n\sigma_i \leq e_i \leq \mu_i + n\sigma_i \tag{2.11}$$

where e_i is the error along the dimension i after correction, and μ_i, σ_i are respectively the mean and standard deviation predicted for the input associated with $\boldsymbol{e}$ on the i-th dimension.

Relative segment errors	**Estimator**	05	06	09	10
Translation (%)	Sparse VO	2.51 ± 1.88	1.30 ± 0.52	3.11 ± 1.66	1.16 ± 0.67
	DPC-Net	1.51 ± 0.71	1.76 ± 0.97	1.64 ± 0.84	1.27 ± 0.56
	D-DICE	$\mathbf{1.01 \pm 0.44}$	$\mathbf{0.91 \pm 0.46}$	$\mathbf{1.17 \pm 0.53}$	$\mathbf{0.95 \pm 0.41}$
Rotation (millideg/m)	Sparse VO	10.31 ± 10.40	9.62 ± 4.13	11.19 ± 2.73	12.35 ± 4.60
	DPC-Net	4.50 ± 2.47	7.50 ± 3.20	4.80 ± 1.92	7.75 ± 2.61
	D-DICE	$\mathbf{3.29 \pm 1.47}$	$\mathbf{3.25 \pm 2.02}$	$\mathbf{4.67 \pm 2.36}$	$\mathbf{5.82 \pm 2.58}$

Table 2.5: Relative segment errors (mean error $\pm$ standard deviation, computed on all relative segments for each sequence) before and after applying corrections to *libviso2* VO.

The parameter n is the number of considered standard deviations. We test against the three-sigma interval ($n = 3$) that, in case of samples drawn from a normal distribution, should cover approximately 99.7% of the samples.
Statistics for a pair of KITTI sequences using D-DICE are displayed in Table 2.6. On average, more than 99% of samples falls within the three-sigma interval.

	σ	2σ	3σ		σ	2σ	3σ
x	85.55%	97.45%	98.64%	x	87.24%	98.91%	99.63%
y	83.82%	98.00%	99.36%	y	91.15%	99.42%	99.67%
z	76.27%	95.82%	99.27%	z	79.49%	96.12%	98.94%
$roll$	77.55%	96.00%	99.18%	$roll$	75.72%	94.89%	98.69%
$pitch$	85.64%	97.27%	98.91%	$pitch$	72.97%	95.14%	98.76%
yaw	74.27%	96.64%	99.27%	yaw	70.57%	93.55%	98.69%
mean	80.51%	96.86%	99.10%	**mean**	79.52%	96.34%	99.07%

Table 2.6: Sequence `06` (top) and `05` (bottom). Percentages of samples that lie in the various sigma-intervals around the mean. Mean and standard deviations are produced by D-DICE.

A common way of evaluating uncertainties is to inspect the mean log-likelihood value. Even though it is not possible to use it alone as a measure of fitness, it can be used to compare different distribution parameters. The log-likelihood directly describes how well the estimated distributions capture the errors in the dataset and is easily obtained as it is the function minimized by our neural network. Table 2.7 shows benchmarks for the predicted distributions obtained by DICE and D-DICE with different losses. While it is true that a higher log-likelihood does not systematically translate into a reduced trajectory error, we found this trend generally confirmed in our experiments.

Estimator	`05`	`06`	`09`	`10`	**mean**
DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	41.53	44.42	41.77	42.72	42.61
D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	43.15	44.24	41.77	42.14	42.82
DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	42.04	44.67	42.27	41.86	42.71
D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	44.08	44.67	41.66	41.60	**43.00**

Table 2.7: Mean log-likelihood for different network architectures and losses.

Finally, to study the utility of covariances in diminishing tracking errors, we set up an error minimization problem using a typical 3D pose-graph formulation. The optimizer seeks to minimize the function

$$\sum_{i \in \mathcal{D}} \boldsymbol{e}_{i,i+1}^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \boldsymbol{e}_{i,i+1} + \boldsymbol{e}_{d,1}^{\mathsf{T}} \boldsymbol{\Sigma}_{gt}^{-1} \boldsymbol{e}_{d,1} \tag{2.12}$$

where $\boldsymbol{e}_{i,i+1}$ is the error function between the nodes $\langle i, i+1 \rangle$ and the corresponding con-

ATE	Estimator	05	06	09	10
Trans. (m)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	126.59†	**9.78**	**51.80**†	4.13
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	**7.97**	12.75	52.16†	**4.02**
	DPC-Net	5.73	8.14	4.00	4.22
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	123.21†	3.87	4.96	3.63
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	**5.64**	**2.54**	**3.61**	**3.02**
Rot. (deg)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	106.32†	**3.42**	15.00†	2.70
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	**3.06**	**3.42**	**13.66**†	**2.63**
	DPC-Net	2.12	2.40	**1.13**	1.74
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	106.46†	1.11	1.37	1.31
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	**1.37**	**0.77**	1.52	**0.87**

Table 2.8: ATE for 5 different estimators. Networks paired with $\mathcal{N}(0, \boldsymbol{\Sigma})$ do not use bias estimation to minimize the NLL. Values tagged with † point to cases where the optimizer noticeably got stuck in a local minimum.

straint $\boldsymbol{z}_{i,i+1}$. A loop closure is manually triggered adding the ground truth measurement $\boldsymbol{z}_{d,1}$ and a small covariance $\boldsymbol{\Sigma}_{gt}$ (practically, forcing the last point of the trajectories to fit the ground truth). We use the framework *g2o* to formulate and solve the problem using a Levenberg-Marquardt optimizer [Kümmerle 2011]. The results of the inclusion of the predicted covariances after pose-graph optimization are summarized in Tables 2.8 and 2.9. To provide a complete overview we show D-DICE and DICE errors obtained considering biases or zero-mean Gaussians. Additionally, we use DPC-Net corrections with a fixed covariance. D-DICE consistently shows smaller errors compared to the other methods. It is worth to notice that without the corrections, particularly for long trajectories, the initial problem state given by VO may lock the optimizer in a local minimum. This mainly happens with DICE (`Seq 05, 09`) but also with D-DICE when a zero-mean normal distribution is considered (`Seq 09`).

Despite using only the latest left image available to VO, DICE is able to have good results on all datasets considered. Still, on average, D-DICE outperforms it. While it is hard to assess the exact reason for the more accurate results, we argue that such improvements are due to two major factors. First of all, our network has access to a larger spectrum of information, the stereo images, which are also available to the VO algorithm. Such information can be exploited to retrieve absolute scale for the translation part. This assumption is backed up by experiencing an average larger standard deviation for the translation part when using monocular images instead of stereo pairs. Additionally, even in the monocular case, having both left images gives the information necessary to extract the robot motion. In this case, the derived uncertainty will not be based only on contextual information (e.g. lack of texture, blurred images) but also on the type/magnitude of motion. Secondly, accounting for non-zero mean allows a further minimization of the negative log-likelihood. D-DICE can rely on estimating both the Gaussian parameters to minimize the same loss.

Relative segment errors	Estimator	05	06	09	10
Translation (%)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	$161.74 \pm 23.24^{\dagger}$	$\mathbf{2.16 \pm 1.28}$	$3.89 \pm 4.27^{\dagger}$	1.23 ± 0.59
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	$\mathbf{1.79 \pm 0.97}$	2.40 ± 1.37	$\mathbf{3.86 \pm 4.18}^{\dagger}$	$\mathbf{1.13 \pm 0.54}$
	DPC-Net	1.15 ± 0.71	1.98 ± 1.39	$\mathbf{1.23 \pm 0.53}$	$\mathbf{0.89 \pm 0.33}$
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	21.60 ± 13.85	0.96 ± 0.71	1.37 ± 0.66	1.15 ± 0.46
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{0.98 \pm 0.57}$	$\mathbf{0.67 \pm 0.35}$	1.50 ± 0.70	0.92 ± 0.35
Rotation (millideg/m)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	$21.66 \pm 13.13^{\dagger}$	$\mathbf{12.82 \pm 5.71}$	$15.35 \pm 32.06^{\dagger}$	9.22 ± 4.60
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	$\mathbf{8.20 \pm 5.31}$	$\mathbf{12.88 \pm 5.29}$	$\mathbf{14.09 \pm 28.83}^{\dagger}$	$\mathbf{8.86 \pm 4.41}$
	DPC-Net	4.31 ± 2.50	9.12 ± 4.35	$\mathbf{3.24 \pm 1.56}$	5.50 ± 3.04
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$161.49 \pm 21.35^{\dagger}$	4.31 ± 2.00	3.74 ± 1.70	5.57 ± 2.59
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{3.37 \pm 2.00}$	$\mathbf{2.95 \pm 1.60}$	4.43 ± 2.08	$\mathbf{4.17 \pm 2.33}$

Table 2.9: Relative segment errors for 5 different estimators (mean error $\pm$ standard deviation, computed on all relative segments for each sequence). Network paired with $\mathcal{N}(0, \boldsymbol{\Sigma})$ does not use bias estimation to minimize the NLL. Oppositely, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ results are obtained employing the loss in Eq. 3.8. Values tagged with † denote cases where the optimizer noticeably got stuck in a local minimum.

2.5 Conclusions

We presented an insight into the learning of errors in visual odometry. Relying on existing state-of-the-art techniques, we iterated on analysing what type of error and uncertainty can be learned by deep neural networks. We concentrated our efforts on approaches that complement classical visual odometry pipelines in order to ease the work done by the network and exploiting a robust and well established feature-based processes. We demonstrated that it is possible to assimilate the distribution over visual odometry errors to Gaussians, and proceeded to cast the error prediction to a full maximum likelihood for normal distributions case. Knowing that the errors are biased, we have modelled such Gaussians as non-zero mean distributions, showing the beneficial aspects of this approach compared to works that rely only on the estimation of the covariance matrix. Additionally, we have integrated visual odometry corrections with a more precise error model, inferred thanks to the assumption of biased distributions. To build on this matter, it can result interesting to adopt the same approach with dense estimators which have access to full disparity information. Finally, we plan to explore similar approaches with different perception processes that are yet to be associated with precise error models, *e.g.* iterative closest points algorithm based on LIDAR scans [Pomerleau 2015].

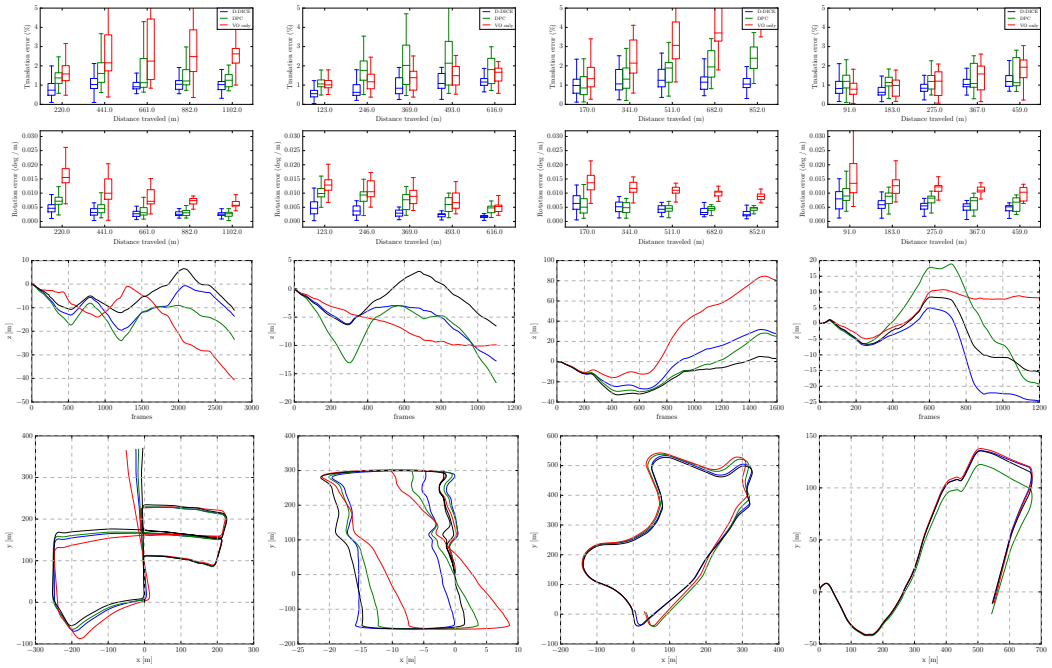


Figure 2.3: Boxplots of the relative segment errors (top two rows), side view and top view of the trajectories (bottom two rows), for KITTI sequences `05`, `06`, `09`, `10` (left to right). D-DICE (blue) and DPC-Net (green) corrections are used to reduce tracking errors using *libviso2* as the baseline estimator (red). The ground truth trajectory is in black.

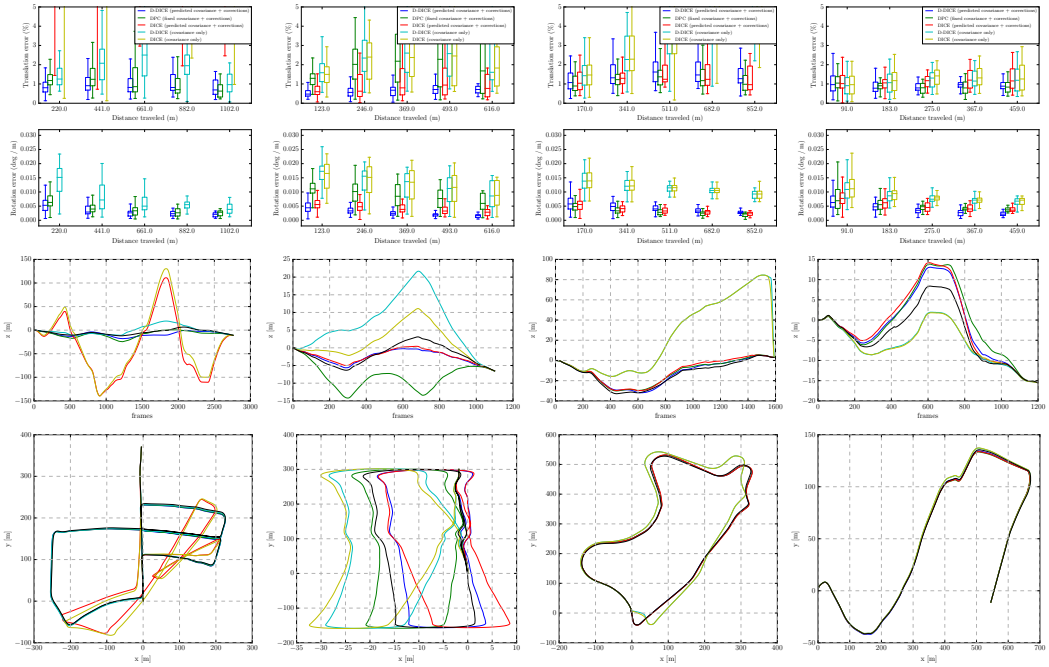


Figure 2.4: Results after pose-graph minimization. Boxplots of the relative segment errors (top two rows), side view and top view of the trajectories (bottom two rows), for KITTI sequences `05, 06, 09, 10` (left to right). D-DICE (blue), DPC-Net (green) and DICE (red) use corrections and uncertainty data (fixed variances for DPC-Net). Cyan and yellow are used for D-DICE and DICE versions that only learned error models. The ground truth trajectory is in black.

CHAPTER 3

Deep Bayesian ICP Covariance Estimation

Contents

Abstract

Covariance estimation for the Iterative Closest Point (ICP) point cloud registration algorithm is essential for state estimation and sensor fusion purposes. We argue that a major source of error for ICP is in the input data itself, from the sensor noise to the scene geometry. Benefiting from recent developments in deep learning for point clouds, we propose a data-driven approach to learn an error model for ICP. We estimate covariances modeling data-dependent heteroscedastic aleatoric uncertainty, and epistemic uncertainty using a variational Bayesian approach. The evaluation is performed on LiDAR odometry on datasets featuring autonomous driving sequences and on challenging scan registration problems. Our method predicts covariances that performs consistently well in comparison to the state of the art.

3.1 Introduction

With the steady increase in 3D sensors availability, point clouds have seen a rapid diffusion and adoption. Their use benefits from the 3D information that they readily express, which allows to easily retrieve geometric properties and yield the possibility to extract complex geometric features to classify object shapes, perform scene segmentation, or assess complex situations. Point clouds are also used to solve motion estimation and self-localization problems in robotics. Scan registration algorithms provide an estimate of the 3D transformation between the positions at which two point clouds are acquired. Such processes are at the heart of LiDAR-based odometry [Zhang 2014] and SLAM frameworks [Mendes 2016]. Originally introduced in [Besl 1992], the Iterative Closest Point (ICP) algorithm, in its many variants, gradually became the standard approach to the point-cloud registration problem [Pomerleau 2015].

In order to qualify the result of ICP, and especially to integrate it within a localization framework, it is essential to estimate its uncertainty. ICP errors stem from various error sources. Among those, the structure of the scene plays an important role: underconstrained situations, such as corridors or mostly flat environments, yield larger errors than geometrically more complex scenes (Fig. 3.1).

Building on the advances in deep learning techniques dealing with point clouds, we present an approach that estimates data-dependent error models for the ICP process in form of a covariance matrix. Our method learns heteroscedastic aleatoric uncertainty from ICP input data and uses bayesian posterior approximation to capture the epistemic uncertainty.

The following section introduces the ICP process, reviews the various sources of errors and works relevant to ours: methods that estimate ICP covariance, data-driven approaches that have proven successful to estimate covariances for other estimation processes, and point-cloud based deep learning techniques. Section 3.3 formalizes the problem and depicts the learning process. Section 3.4 evaluates our results against the state-of-the-art, showing an accurate correlation between the real error of ICP and our predicted covariance.

3.2 Background and related work

3.2.1 Iterative Closest Point process

ICP tackles the problem of aligning a *reading* point cloud $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{3\times n}$ to a *reference* point cloud $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{3\times m}$, finding the true rigid transformation between them. Using an initial estimation $\bar{\boldsymbol{T}} \in$ SE(3) of the transformation (*e.g.* as provided by a prior on the motion), the rigid transformation $\hat{\boldsymbol{T}} \in$ SE(3) that estimates the true transformation is the one that minimizes the error function e:

$$\hat{\boldsymbol{T}} = \arg\min_{\boldsymbol{T}}(e(\boldsymbol{T}(\boldsymbol{\mathcal{P}}), \boldsymbol{\mathcal{Q}})) \tag{3.1}$$

where $\boldsymbol{T}(\boldsymbol{\mathcal{C}})$ is the transformation $\boldsymbol{T}$ applied to the point cloud $\boldsymbol{\mathcal{C}}$. The error function is the result of a weighted average distance between appropriately matched point pairs or point to plane associations. The optimization is an iterative process, and there is no

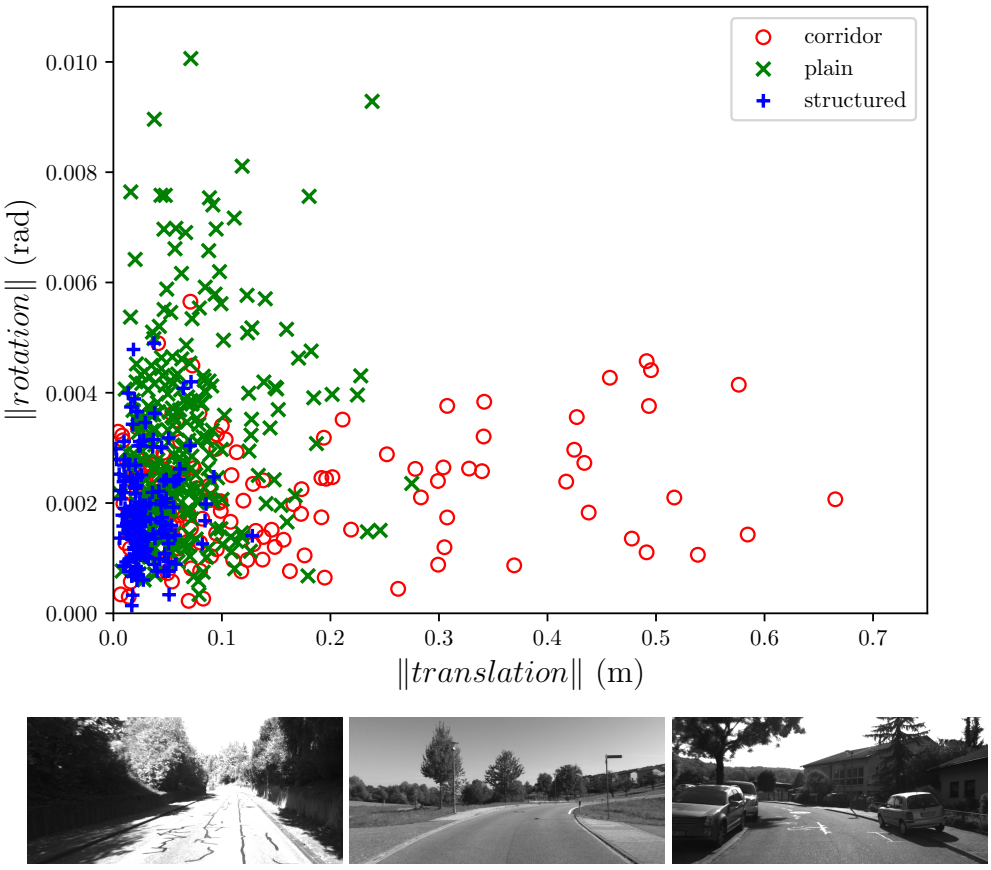


Figure 3.1: Distribution of errors on a series of ICP estimates computed in three different kind of environments: corridor-like (left), wide plain (center), structured (right).

guarantee to reach the optimal solution.

3.2.2 Sources of error for ICP

There are five main sources of error for ICP described in the literature: wrong convergence, wrong initialization, sensor noise and bias, underconstrained situations, and intrinsic randomness of the process [Censi 2007, Brossard 2020].

Convergence and initialization. Wrong convergence comes from the iterative nature of ICP, which does not guarantee global convergence and can get stuck in local minima. This is closely related to wrong initialization, which can be a predominant source of error when the initial guess is far from the true solution [Brossard 2020].

Sensor noise and biases. Each sensor naturally suffers changes in its accuracy that depend on several environmental conditions (*e.g.* temperature) [Pomerleau 2012a]. Also, sensor calibration can introduce distortion [Deschaud 2018]. While these phenomena introduce a bias in the estimate [Laconte 2019], there is an additional source of sensor noise added to each point independently, which generally is a function of the distance, orientation and physical nature or texture of the perceived material.

Underconstrained situations. These occur when the environment does not offer enough information to fully estimate the transformation between the point clouds, mainly by precluding the establishment of good matches between points. While in the planar 2D case they can be exhaustively identified, namely the corridor and the circle, in the 3D world this becomes much more challenging to do. Periodic geometric patterns also yields series of local minima in which ICP can converge.

Intrinsic ICP randomness. Depending on the configuration and implementation, a certain degree of randomness is introduced by subprocesses of ICP. For example, filtering clouds with outlier rejection and sub-sampling can yield different transformation estimates, even with the same input data.

3.2.3 Estimating ICP covariance

Various methods for estimating ICP covariance have been proposed. Monte-Carlo algorithms [Buch 2017] generate the covariance from a large number of samples of scan registration results. Different clouds and initializations are randomly subsampled to obtain a dispersion of the transformation that is used to compute the covariance. Closed-form methods [Censi 2007, Brossard 2020, Prakhya 2015] provide algebraic solutions to the ICP covariance problem by linearizing the error functions defined in the ICP algorithm. These methods are potentially ill-founded if point reassociations happen at a scale smaller than the actual error [Bonnabel 2016]. Experiments on synthetic data have shown that this is often the case [Landry 2019]. Early closed-form works, as the one in [Censi 2007], have been demonstrated over optimistic [Mendes 2016]. Recently, [Landry 2019] proposed a

learning approach to the ICP covariance estimation problem. The work estimates a 3D covariance based on preprocessed feature descriptors that are used in the learning scheme. While such descriptors are generic, it is desirable to have a more flexible representation of features impacting on the error of ICP.

It is worth to notice that most of these work tackle different sources of error. The method presented in [Brossard 2020] shows the need for a proper initialization of ICP and proposes a novel approach to 3D uncertainty of ICP accounting for initialization errors. Censi's pioneering work [Censi 2007] studies the error due to sensor noise and computes the covariance of this error as a function of the error metric minimized by any ICP implementation. The work in [Landry 2019], closer to our approach, explores a data-driven approach to estimate the covariance focusing on sensor noise and aiming at detecting underconstrained situations.

3.2.4 Learning motion estimation error models

The literature in deep learning features a large amount of work in the estimation of uncertainty. The approaches are often tied to the process for which they predict error models. Interesting works in domains similar to ours have been proposed, mainly tackling the Visual Odometry (VO) case. As ICP, VO is a well-known family of techniques which estimate a rigid transformation between two consecutive frames for the monocular or stereovision cases. End-to-end methods perform frame to frame motion estimation [Wang 2017] and covariance prediction [Wang 2018]. Both estimates are in this case delegated to neural networks, and do not rely on any geometrical pipeline. The works in [Liu 2018, De Maio 2020] respectively show covariance and joint bias-covariance estimation to the geometric implementation of VO.

3.2.5 Deep learning for point clouds

Recently, deep learning methods applied on point clouds have proven successful for several tasks [Qi 2016, Qi 2017, Thomas 2019, Qi 2019]. In particular, PointNet [Qi 2016] paved the way for neural networks that directly use 3D point clouds for object classification and segmentation purposes, and has been extended towards learning point cloud features in metric space [Qi 2017]. This extension increased the effectiveness of PointNet to learn features on different scales, in the manner of traditional convolutional networks. Other approaches rely on the spherical projection of point clouds, usually coined as range (or depth) image – which is, in most cases, the original structure of the sensor data. This is used in the LiDAR odometry techniques [Cho 2019, Wang 2020, Li 2019] which makes use of the entire scan to estimate the motion between two frames in an end-to-end manner.

3.3 Data driven learning of ICP uncertainty

Excluding the process initialization, all error sources of ICP are inherited from the input data. Sensor noise is directly reflected in the data, as well as ICP subprocesses modifying its size, density and structure (i.e. randomness due to filtering). Additionally, the scene

structure, with its inherent geometry and features, largely impacts the error of ICP (as illustrated in Fig. 3.1), as it defines the observability of the motion along the different dimensions.

Addressing all these sources in a unified framework is a challenging task. Leveraging recent progresses in deep learning for point clouds, we address the problem of estimating the covariance of ICP that stems from the last three error sources presented in Sec. 3.2.2. Our hypothesis is that a learning scheme that processes the raw input data is able to capture these error sources and properly estimate the covariances of the ICP estimate.

This section depicts our approach to estimate ICP covariance and discuss the nature of the learned uncertainties. The optimization problem is contextualized in a probabilistic framework and a description of the learning architecture incorporating different types of uncertainty is provided.

3.3.1 Minimization problem

Modeling errors for an estimation process can be tackled via a supervised approach. Consider an estimator, ICP in our case, using sensory data $\boldsymbol{\mathcal{C}}_i = \{\boldsymbol{\mathcal{P}}_i, \boldsymbol{\mathcal{Q}}_{i+1}\}$ with $\boldsymbol{\mathcal{P}}_i \in \mathbb{R}^{n_i}$ and $\boldsymbol{\mathcal{Q}}_{i+1} \in \mathbb{R}^{n_{i+1}}$. Its output ${}^i\hat{\boldsymbol{T}}_{i+1}$ is an estimate of the ground truth transformation ${}^i\boldsymbol{T}_{i+1}$. In practice, the point clouds used in the algorithm are the result of several filters [Pomerleau 2015]. This has an impact on ICP estimates, but is needed from a computational point of view and also to remove outliers and stabilize the algorithm. The actual input data is a pair of decimated point clouds $\bar{\boldsymbol{\mathcal{C}}}_i = \{\bar{\boldsymbol{\mathcal{P}}}_i, \bar{\boldsymbol{\mathcal{Q}}}_{i+1}\}$ with $\bar{\boldsymbol{\mathcal{P}}}_i \in \mathbb{R}^{m_i}, m_i \ll n_i$ and $\bar{\boldsymbol{\mathcal{Q}}}_{i+1} \in \mathbb{R}^{m_{i+1}}, m_{i+1} \ll n_{i+1}$. The error of ICP is computed as

$$\boldsymbol{e}_i \;=\; {}^i\hat{\boldsymbol{T}}_{i+1}^{-1} \;\cdot\; {}^i\boldsymbol{T}_{i+1} \tag{3.2}$$

We represent an error $\boldsymbol{e} \in \mathrm{SE}(3)$ as a vector $\boldsymbol{\xi} \in \mathfrak{se}(3)$ member of the Lie algebra [Sola 2018, Barfoot 2014]

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} \tag{3.3}$$

where the *hat* operator $\wedge$ turns the vector $\boldsymbol{\xi} \in \mathbb{R}^6$ into a matrix $\boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times4}$. The opposite linear map is often represented as the *vee* operator $\vee$. The exponential map $\exp(\boldsymbol{\xi}^\wedge)$ allows to retract an element of the Lie algebra back to the group. Its inverse map is $\log(\boldsymbol{T})$. Once the error is defined as $\boldsymbol{\xi} = \log(\boldsymbol{e})^\vee$, the associated uncertainty $\boldsymbol{\Sigma} \in \mathbb{R}^{6\times6}$ can be expressed as

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\rho\rho} & \boldsymbol{\Sigma}_{\rho\phi} \\ \boldsymbol{\Sigma}_{\phi\rho} & \boldsymbol{\Sigma}_{\phi\phi} \end{bmatrix} \tag{3.4}$$

This uncertainty representation is generally valid for small perturbations that are added to a given pose $\hat{\boldsymbol{T}} = \boldsymbol{T}\exp(\boldsymbol{\xi})$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$[Barfoot 2014]. The equation above matches the error definition in Eq. 3.2, that is $\boldsymbol{\xi} \in \mathbb{R}^6$ can be viewed as the error between the real transformation and its estimate.

We define a dataset $\mathcal{D} = \{\bar{\boldsymbol{\mathcal{C}}}_i, \boldsymbol{\xi}_i | \forall i \in [1, d]\}$, where d is the size of the dataset. The goal is to predict the uncertainty of the error vector $\boldsymbol{\xi}_i$ along its dimensions. In the data, we

noticed that the unbiased error assumption is valid for the vast majority of ICP estimates and the bias is very small otherwise. To keep the uncertainty assumption valid, we encode a bias term for the error vector in the distribution that ensures the error remains close to a zero-mean Gaussian. Therefore, it is possible to estimate the parameters of the Gaussian as

$$\underset{\boldsymbol{\mu}_{1:d},\boldsymbol{\Sigma}_{1:d}}{\arg\max} \sum_{i=1}^{d} p(\boldsymbol{\xi}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3.5}$$

To maximize the probability of estimating the correct error given the estimated Gaussian parameters is equivalent to minimize the negative log-likelihood

$$\underset{\boldsymbol{\mu}_{1:d},\boldsymbol{\Sigma}_{1:d}}{\arg\min} \sum_{i=1}^{d} -\log\left(p(\boldsymbol{\xi}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right) \tag{3.6}$$

$$= \underset{\boldsymbol{\mu}_{1:d},\boldsymbol{\Sigma}_{1:d}}{\arg\min} \sum_{i=1}^{d} \log|\boldsymbol{\Sigma}_i| + (\boldsymbol{\xi}_i - \boldsymbol{\mu}_i)^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\xi}_i - \boldsymbol{\mu}_i) \tag{3.7}$$

To estimate a positive definite covariance matrix we use the LDL composition as in [Liu 2018]. The elements used to reconstruct the covariance are defined in a vector $\boldsymbol{\alpha}_i = [\boldsymbol{l}_i, \boldsymbol{d}_i]^{\mathsf{T}}$, with $\boldsymbol{l}_i \in \mathbb{R}^{\frac{(n^2-n)}{2}}$ and $\boldsymbol{d}_i \in \mathbb{R}^n$. Substituting the decomposition into Eq.3.7 and denoting estimated quantities with $\widehat{(\cdot)}$, the final loss function becomes

$$\begin{aligned} \mathcal{L}(\bar{\boldsymbol{C}}_{1:d}) = &\underset{\hat{\boldsymbol{\mu}}_{1:d},\boldsymbol{\alpha}_{1:d}}{\arg\min} \sum_{i=1}^{d} sum(\boldsymbol{d}_i) + \\ &(\boldsymbol{\xi}_i - \hat{\boldsymbol{\mu}}_i)^{\mathsf{T}} (L(\boldsymbol{l}_i) D(exp(\boldsymbol{d}_i)) L(\boldsymbol{l}_i)^{\mathsf{T}})^{-1} (\boldsymbol{\xi}_i - \hat{\boldsymbol{\mu}}_i) \end{aligned} \tag{3.8}$$

where the covariance matrix is reconstructed as $\hat{\boldsymbol{\Sigma}}_i = L(\boldsymbol{l}_i) D(exp(\boldsymbol{d}_i)) L(\boldsymbol{l}_i)^{\mathsf{T}}$. For the details of the loss composition we refer the reader to [De Maio 2020, Liu 2018].

3.3.2 Uncertainty estimation

The estimated overall ICP covariance can be decomposed into two categories: epistemic and aleatoric [Kendall 2017b]. Both uncertainties are formulated as probability distributions over different entities.

Aleatoric uncertainty models the uncertainty over the input data to the process. It assumes an observation noise that can be either constant or vary with the input. As discussed in Sec.3.2.2, we observed that noise levels change with the scene (Fig. 3.1), hence we are interested in modeling the variation of the noise in correlation with the input data. This is referred to as heteroscedastic aleatoric uncertainty (as opposed to homoscedastic uncertainty for the constant noise case).

Additionally, it is possible to model the variance intrinsic to the learned model, that is how certain of a given prediction the network is. Epistemic uncertainty can, in principle, be reduced if more data is used for training. In our case, it accounts for the network's poor predictions on out-of-data samples such as scene features that were rarely encountered during training. Epistemic uncertainty can be modeled by placing a distribution over the

network's weights. Unfortunately, given a prior distribution $\mathcal{N}_{\boldsymbol{W}}(0, \sigma)$ over the weights, it is computationally difficult to analytically infer the posterior $p(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{Y})$, that is the probability of drawing a set of weights given the network input and output. There is a number of proposed methods to approximate this posterior [Stephan 2017, Postels 2019]. The work in [Gal 2016] produces an approximation using dropout variational inference. This has proven to be an effective yet easily obtainable Bayesian approximation when dealing with deep complex models.

From Eq.3.7 we can consider the mean as a noisy error estimate, and the covariance matrix as the aleatoric uncertainty due to the observation noise. To approximate the posterior for a Bayesian neural network estimating the ICP errors we resort to Monte-Carlo dropout. The epistemic uncertainty can be captured by keeping dropout activated at test time. Sampling N times the predicted ICP error vector and ICP covariance yields a distribution $[\hat{\boldsymbol{\mu}}_{1:N}, \hat{\boldsymbol{\Sigma}}_{1:N}] = f_{\boldsymbol{W}}(\bar{\boldsymbol{\mathcal{C}}})$ where $f_{\boldsymbol{W}}(\bar{\boldsymbol{\mathcal{C}}})$ is the output of a Bayesian neural network, function of the input point clouds.

The total covariance matrix, integrating aleatoric and epistemic uncertainty is computed as

$$\frac{1}{N}\sum_{n=1}^{N}(\hat{\boldsymbol{\mu}}_n - \bar{\boldsymbol{\mu}})(\hat{\boldsymbol{\mu}}_n - \bar{\boldsymbol{\mu}})^{\mathsf{T}} + \frac{1}{N}\sum_{n=1}^{N}\hat{\boldsymbol{\Sigma}}_n \tag{3.9}$$

where $\bar{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N}\hat{\boldsymbol{\mu}}_n$ is the mean error vector over the sampled outputs. Intuitively, the first term in Eq. 3.9 computes the uncertainty over the predicted error vector, which presents a variance proportional to the network confidence over the estimate. The mean vector is not used to correct ICP estimates as the original error is generally noisy. It is rather evaluated in terms of dispersion after having approximated the model posterior.

3.3.3 Learning architecture

The improvements in point clouds based network involved a number of estimation processes. Recently, FlowNet3D [Liu 2019] proposed an approach to tackle 3D scene flow expanding on the building blocks introduced in [Qi 2017]. It provides the 3D flow for each point in a reference point cloud deducing its motion from comparison with a second, successive point cloud. Estimating the scene flow is a problem closely related to the scan registration. Intuitively, a network able to approximate such task would result suitable for ICP related inferences. In fact, the main difference between the two tasks lies in the generalization needed to shift from a high dimensional flow to the estimate of a rigid transformation in the ICP case. We leverage the main structure of FlowNet3D to tackle the estimation of ICP errors, modifying few key aspects to adapt it to our problem.

FlowNet3D has three core building blocks: *set conv* layers, a *flow embedding* layer and *set upconv* layers. All modules deal with the irregular and orderless nature of a point cloud, which makes convolutions inappropriate and calls for alternative formulations to extract features.

1) Set conv. This layer has already been proposed in [Qi 2017] showing the capability to learn hierarchical features used for classification and segmentation problems. It also proves useful for the motion estimation case.

Set conv first samples n' points with farthest point sampling from a point cloud with n points, where each point is $p_i = \{x_i, f_i\}$, $x \in \mathbb{R}^3$ represents its 3D coordinates and $f_i \in \mathbb{R}^c$ an associated feature. Then, it groups them by a region defined in a 3D sphere defined by a radius r. Finally, it extracts local features with a classical multi-layer perceptron (MLP)

$$f'_j = \underset{(i|\ ||x_i - \hat{x}_j|| \leq r)}{\mathrm{MAX}} h(f_i, x_i - \hat{x}_j) \tag{3.10}$$

where $h : \mathbb{R}^{c+3} \rightarrow \mathbb{R}^{c'}$ is the MLP (non-linear) which gets features and points in input, MAX is the max-pooling operation and the output is a set of features f' part of the sub-sampled cloud with n' points $p'_j = \{x'_j, f'_j\}$, $f'_j \in \mathbb{R}^{c'}$.

2) Flow embedding. In order to generate information about point motion the outputs of the *set conv* layers are processed by a dedicated flow embedding module. It takes a pair of point clouds $\{p_i\}_{i=1}^n$ and $\{q_j\}_{j=1}^m$ in the same form as in *set conv* $p_i = \{x_i, f_i\}$, $q_j = \{y_j, g_j\}$. The layer computes an embedding e_i for each point in the first cloud, which even if computed similarly to the previous layer has a different meaning. In fact the distance is now computed between neighboring points belonging to different sampled clouds, recalling the flow idea. It is then used as input, along with respective features, to the MLP

$$e_i = \underset{(j|\ ||y_j - x_i|| \leq r)}{\mathrm{MAX}} h(f_i, g_i, y_j - x_i) \tag{3.11}$$

Once the flow embeddings are computed, extra features are extracted with few additional set conv passes.

3) Set upconv. With this layer the embeddings computed in the previous step are up-sampled to be matched with the original point set. Similar to 2D upconv in images, *set upconv* is performed in the same way as *set conv* but using a different sampling strategy. The target points to upsample are retrieved from samplings in the previous layers until the original point cloud size is retrieved. Feature propagation is performed using the same technique showed in Eq.3.10.

To complete the architecture we add a final MLP layer reducing the dimensionality to the number of parameters needed to predict the Gaussian parameters. First we follow the original regression layer from FlowNet3D reducing the feature space to $\mathbb{R}^3 \times n$. The MLP is extended to take as input the feature produced by the regression layer and to return the parameters generating the covariance matrix and predicted error vector. All the MLP (originally Linear-BatchNorm-ReLU) in FlowNet3D are stacked with Dropout modules to allow the estimation of the epistemic covariance matrix. This also proved to reduce overfitting as expected.

We initially assumed that including the *set upconv* layer would have not been needed. Intuitively, as in a classical convolutional scheme, the regression task we try to solve can be solved with direct dimensionality reduction, from sensory data to parameters describing uncertainty [De Maio 2020, Liu 2018]. Our ablation studies showed that retaining upsampling layer produced better results in terms of predicted uncertainty and mean. We associate this behavior with the architecture using skip connections to use the same features extracted from the same point sets originally used ICP. This creates a 1:1 matching between the data

Layer type	Radius	Sampling	MLP size
set conv	2.0	1.0×	[32, 32, 64]
set conv	4.0	0.25×	[64, 64, 128]
flow embedding	knn	1.0×	[128, 128, 128]
set conv	8.0	0.25×	[128, 128, 256]
set conv	16.0	0.125×	[256, 256, 512]

Table 3.1: Network architecture parameterization. Set upconv layers are the same as in the updated version of [Liu 2019].

used by the geometric estimator and the network, leaving less room for outliers. Even if the features output from the feature propagation layer are not the same generated by the original FlowNet3D, the use of the original point cloud set is beneficial to learn the importance of single points in ICP error estimation. The new parameter table with sampling rates and radius can be found in Table 3.1.

3.4 Evaluation

3.4.1 ICP process and dataset

To learn errors produced by a geometric ICP algorithm we selected *libpointmatcher* [Pomerleau 2011]. It is a widely used open-source library offering a chain that filters point clouds, establishes points associations, removes outliers and estimates rigid transformations using an error minimizer. For our setup we parameterized the process similarly to [Landry 2019]. Cloud random sampling is fixed at 2048 random points. We generate normals, used for the point-to-plane minimizer, using the 10 nearest points before subsampling.

To select the initial transform for ICP, we add an error to the ground truth $\boldsymbol{T}$. In order not to draw errors from an overoptimistic distribution we sample guesses for the initial transforms from a non-zero mean normal distribution $\mathcal{N}(\mathcal{N}(\boldsymbol{T}, a \cdot \boldsymbol{T}), b \cdot \boldsymbol{T})$, with $a = 0.25, b = 0.2$. These figures yield more pessimistic estimates than commonly used aiding sensor (e.g. odometry or inertial measurements) posing a challenge to our system. The result is clipped to be at most an error of 35% w.r.t. the ground truth along each dimension.

We select the KITTI odometry dataset [Geiger 2012], using LiDAR scans, which is made of urban and rural road scenes. We train and test using all ground truth tagged sequences but sequence `01`, on which ICP fails to register most of the scans to scenes lacking features. To compare against the state of the art in covariance estimation for ICP, we also present results on the challenging data sets of [Pomerleau 2012b]. Since for the LiDAR odometry case this dataset is rather small, we refine our KITTI trained model with a few epochs of training, approximately 30 or less for early convergence, on all but one outdoor sequences and validate on the one left out. As expected, features learned on KITTI are ineffective to predict covariances on the *apartment* and *stairs* sequences, which are highly structured small indoor environments.

3.4.2 Single-pair validation

We first assess the quality of the predicted covariances in a single-pair fashion, and then check their consistence on a full trajectory. The metrics we use are based on the relation between the observed errors and the predicted covariance. This is due to the absence of the true distribution which would allow for more classic figures of merit, such as the Kullback-Leibler divergence. While it would be possible to obtain a pseudo-true distribution by random sampling over the point cloud, this would affect the estimated covariance as well, because the network prediction is tied to the ICP configuration, and most importantly to the input data used by ICP.

As in [Censi 2007, Brossard 2020], we compute the *Normalized Norm Error (NNE)* to evaluate the scale of the predicted covariance w.r.t. the actual error:

$$\mathrm{NNE} = \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{||\boldsymbol{\xi}_n||_2^2}{tr(\hat{\boldsymbol{\Sigma}}_n)}} \tag{3.12}$$

where $tr(\hat{\boldsymbol{\Sigma}}_n)$ is the trace of the predicted covariance matrix. The optimal value for NNE is one. Values below and above one respectively represent pessimistic and optimistic uncertainty estimates. Table 3.2 shows NNE values computed over test KITTI sequences. The scale of the covariance is well predicted, and is only slightly pessimistic for two of the five validation sequences (`05` and `07`).

NNE (KITTI)	`05`	`06`	`07`	`09`	`10`
Trans (A)	**0.70**	**0.61**	**0.58**	**0.94**	**0.90**
Trans (E+A)	0.66	**0.61**	0.55	0.93	0.88
Rot (A)	**0.65**	**0.81**	**0.55**	**0.90**	**1.04**
Rot (E+A)	0.53	0.72	0.45	0.81	0.93

Table 3.2: NNE translation and rotation values for the KITTI dataset using aleatoric only (A) and combined epistemic-aleatoric (E+A) uncertainty.

The NNE is an intuitive measure of the adequacy of the predicted variances, but it does not account for off-diagonal covariances. For this purpose, the Mahalanobis distance between ICP estimates and ground truth $\boldsymbol{\xi}_n = \log({}^n\hat{\boldsymbol{T}}_{n+1}^{-1}\ {}^n\boldsymbol{T}_{n+1})^\vee$, weighted by the predicted covariance $\boldsymbol{\Sigma}_n$ averaged over each consecutive pair in a sequence, is a better metric:

$$\mathrm{D_M} = \frac{1}{N}\sum_{n=1}^{N}\sqrt{\frac{\boldsymbol{\xi}_n^\mathsf{T}\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\xi}_n}{dim(\boldsymbol{\xi})}} \tag{3.13}$$

As in the NNE case, the optimal value for a normal distribution is one, and a distance below and above one respectively denotes pessimistic and optimistic estimates.

Table 3.3 shows the average Mahalanobis distance for the validation sequences. As for the results of Table 3.2, it is worth to see that the spread between the aleatoric and combined aleatoric-epistemic is rather small: this is because the network encounters known

Single-Pair **Mah. Dist (KITTI)**	05	06	07	09	10
Trans (A)	**0.80**	**0.83**	**0.74**	**0.88**	**1.03**
Trans (E+A)	0.73	0.73	0.66	0.79	0.92
Rot (A)	**0.77**	**0.63**	**0.70**	**0.94**	**0.96**
Rot (E+A)	0.75	0.62	0.67	0.93	0.93

Table 3.3: Single-pair Mahalanobis distance in terms of translation and rotation for the KITTI dataset using aleatoric only (A) and combined epistemic-aleatoric (E+A) uncertainty.

patterns in the validation set. This behavior is expected as the dataset is rather homogeneous in terms of motions and environments. The fact that the aleatoric-only formulation outperforms the combined aleatoric-epistemic is due to the nature of the chosen metric along with the predicted covariance. For any pessimistic prediction, given the positive definite nature of a covariance matrix, it is likely that both the Mahalanobis distance and the NNE would become smaller when adding multiple sources of uncertainty. This normally translates in better results for over-pessimistic predictions and worse results in the opposite case. Inverse trends, i.e. metric values growing when adding the epistemic uncertainty, are possible in cases of large off-diagonal values (*e.g.* rotational Mahalanobis distance for sequence 05 and 09). It is interesting that, due to the data homogeneity, the epistemic uncertainty is rather constant throughout the whole KITTI dataset. Still, slight variations are encountered for seldom cases: an example is shown in Fig. 3.2 displaying the epistemic covariance before, during, and after an area corresponding to *wide plain* scenes shown in Fig. 3.1.

3.4.3 Trajectory validation

In order to assess the quality of our predicted covariances on trajectories we take a multi-step approach. First, we compose ICP estimates to generate the final pose as ${}^0\hat{\boldsymbol{T}}_f = {}^0\hat{\boldsymbol{T}}_1 \, {}^1\hat{\boldsymbol{T}}_2 \cdots {}^{f-1}\hat{\boldsymbol{T}}_f$ for a sequence with f poses. The ground truth final pose ${}^0\boldsymbol{T}_f$ is computed similarly. We propagate the covariance associated to each estimate using the 4-th order approximation as in [Barfoot 2014], to obtain an uncertainty estimate for the final ICP pose. This way, it is possible to evaluate the Mahalanobis distance (Eq. 3.13) in a spatially consistent manner. Table 3.4 shows the results. They are only slightly optimistic, which is a remarkable outcome given the length of KITTI trajectories (up to more than 2km). Sequence 10 presents the farthest values from the optimum, even though it returned excellent estimates in the single pair case. As already mentioned, even if not numerically significant, in the case of an optimistic prediction the aleatoric-epistemic formulation is preferable to the aleatoric only.

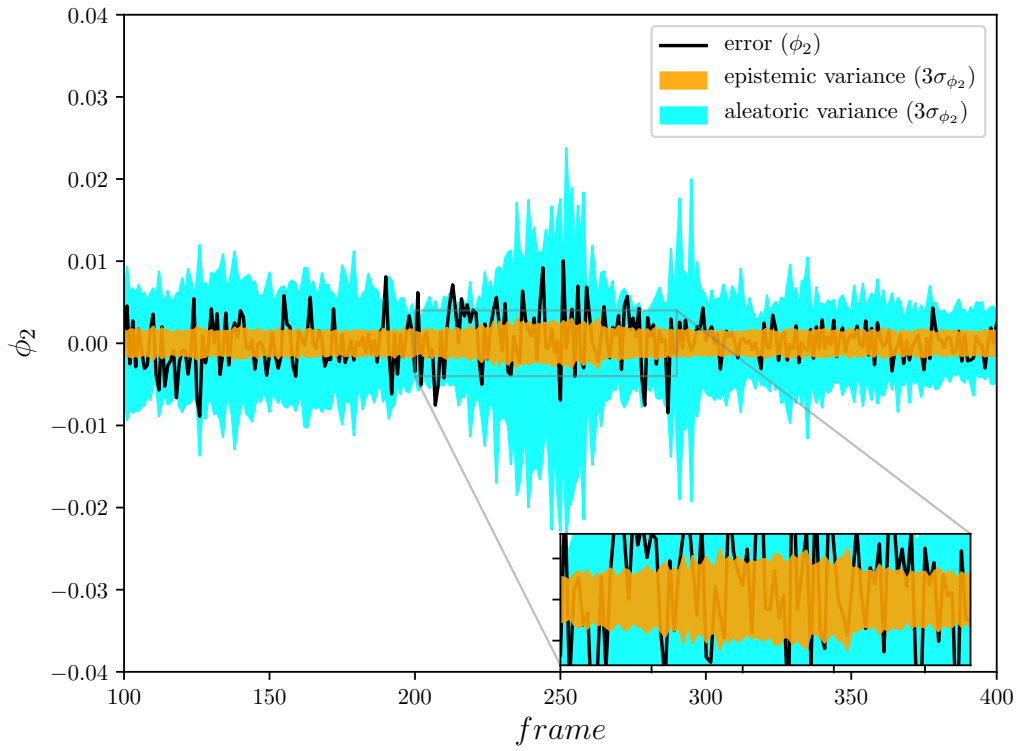


Figure 3.2: Epistemic and aleatoric 3σ intervals in a rarely seen area lacking sharp geometrical features. The real error computed from the ground truth is represented in black. The zoomed-in area shows epistemic uncertainty increase (up to 2 times), which corresponds to a *wide plain* environment.

3.4.4 Fine tuning on other datasets

We now evaluate the results on the smaller dataset of [Pomerleau 2012b] and compare to the state of the art. As our model requires a significant amount of data we initially learn on a large dataset. KITTI makes available a large number of LiDAR scans with different scenarios, but all correspond to outdoor road environments. To apply our approach to smaller datasets acquired in very different environments, a fine-tuning process is applied using the pre-trained model on the KITTI dataset. This allows to obtain much better results compared to re-training on such a small dataset.

Table 3.5 and 3.6 show the result of our approach respectively on single-pair estimates and on aggregated transforms with the dataset of [Pomerleau 2012b]. While the environments are very different, the model performs remarkably well in outdoor scenes. Even architectural features, such as in *hauptgebaude* can be leveraged to produce faithful covariance predictions as, intuitively, the scene presents a structure (flat ground plane, lack of indoor man-made elements, lack of relevant features) that is geometrically similar to the some scenes encountered in KITTI.

The tables also compare our results against CELLO-3D [Landry 2019] and *Brossard et*

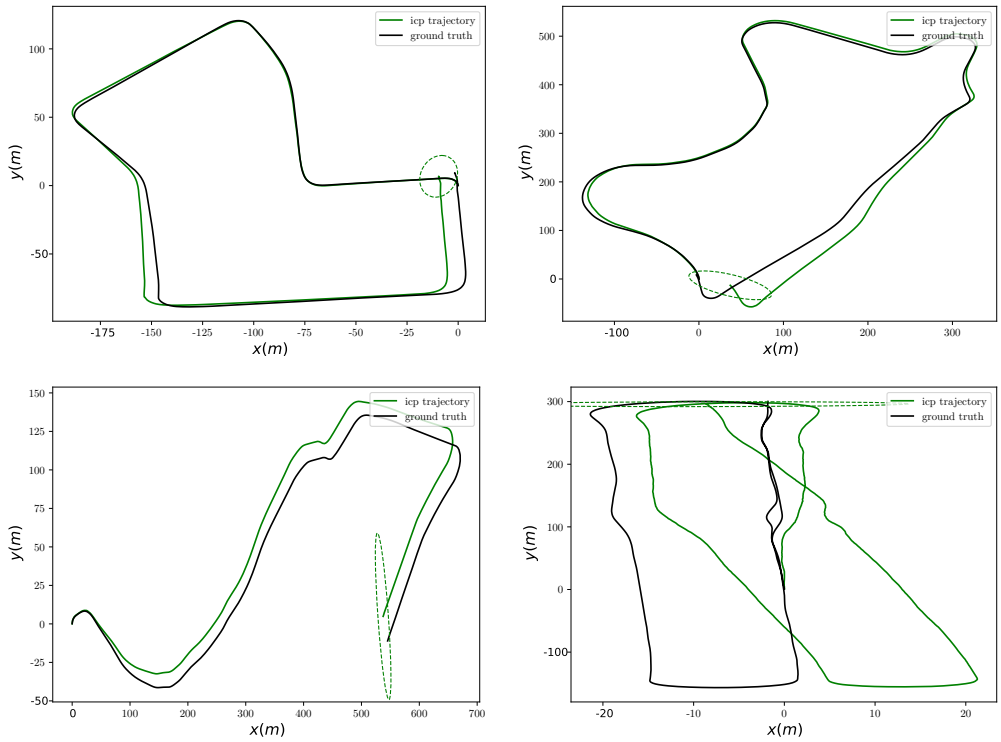


Figure 3.3: Four propagated covariances in the 2σ interval (95%) on considered ICP trajectories compared to ground truth.

Trajectory **Mah. Dist (KITTI)**	05	06	07	09	10
Trans (A)	**2.31**	1.51	1.26	**1.36**	4.86
Trans (E+A)	**2.31**	**1.48**	**1.23**	**1.36**	**4.77**
Rot (A)	**1.49**	1.95	2.19	**1.89**	2.95
Rot (E+A)	1.51	**1.85**	**2.17**	1.92	**2.88**

Table 3.4: Trajectory Mahalanobis distance in terms of translation and rotation for the KITTI dataset using aleatoric only (A) and combined epistemic-aleatoric (E+A) uncertainty.

Single-Pair **Mahalanobis Distance (ETH)**		*Gazebo winter*	*Gazebo summer*	*Mountain*	*Haupt--gebaude*	*Wood autumn*	*Wood summer*
Ours (Direct)	Trans (A)	2.05	2.32	4.15	2.68	4.01	4.03
	Trans (E+A)	2.04	2.12	3.88	2.49	4.19	3.78
Ours (Finetuned)	Trans (A)	**0.64**	**1.12**	**1.27**	**1.04**	**1.86**	**1.03**
	Trans (E+A)	0.59	1.06	1.21	0.98	1.70	0.90
Ours (Direct)	Rot (A)	1.33	1.93	3.06	2.61	2.10	2.33
	Rot (E+A)	**1.30**	1.89	3.03	2.47	2.07	2.19
Ours (Finetuned)	Rot (A)	0.48	**0.53**	**0.65**	**1.02**	**0.54**	**0.82**
	Rot (E+A)	0.44	0.49	**0.65**	0.96	0.51	0.71

Table 3.5: Single-pair Mahalanobis distance in terms of translation and rotation for the ETH dataset using aleatoric only (A) and combined epistemic-aleatoric (E+A) uncertainty. A comparison between the pre-trained model on KITTI and after finetuning is presented.

al. [Brossard 2020]. It is worth noticing that given the strong focus on ICP initialization from *Brossard et al.*, it is more relevant to compare our approach to CELLO-3D, as our work solely focus on the errors introduced directly and indirectly by the data on a purely data-driven approach.

CELLO-3D is often over pessimistic while *Brossard et al.* is generally optimistic. On rotational uncertainty, our method outperforms the state of the art by a significant margin. This shows the challenge of estimating accurate covariances for ICP. We highlight the impact of the finetuning, not only in the expected improved results, but also in the magnitude of the epistemic uncertainty. The improvement brought by the considering model uncertainty is in fact evident when testing the model trained on KITTI directly on an unseen dataset. Numerically speaking, the variance difference in play are significantly larger in the *direct* case compared the *finetuned*. As expected, this shows that the network has little confidence in its prediction for new scenarios. While some outdoor features may recur in the ETH dataset, motion type and scene size are remarkably different when compared to the KITTI dataset. After a finetuning, not only the weight of the epistemic covariance is greatly reduced (see Fig. 3.4) but the overall results are closer to the optimal value.

3.5 Conclusion

We presented an approach to estimate faithful covariances for the ICP scan registration process. Using a data-driven paradigm we have shown that it is possible to learn uncertainty for the selected algorithm, and to generalize to a certain degree on multiple datasets. Leveraging a neural network architecture conceived to directly process point clouds, we estimate covariances with a close connection to the data used in the ICP algorithm. Standard metrics show the reliability of our approach, and that it generally outperforms the state-of-the-art. Future improvements will concern the network architecture, possibly tailoring it to the scan registration problem. A possible approach is to have the flow embedding layer (and the successive layers) doubled to account for both point clouds motions, whereas in

Trajectory **Mahalanobis Distance (ETH)**		*Gazebo winter*	*Gazebo summer*	*Mountain*	*Haupt--gebaude*	*Wood autumn*	*Wood summer*
CELLO-3D [Landry 2019]	Trans	0.1	0.2	-	0.3	0.1	0.1
Brossard et al. [Brossard 2020]	Trans	1.8	1.0	1.2	**1.8**	1.2	1.5
Ours (Direct)	Trans (A)	2.50	4.95	3.94	8.12	4.59	3.24
	Trans (E+A)	2.45	4.87	3.90	7.90	4.58	3.09
Ours (Finetuned)	Trans (A)	0.83	**1.29**	**0.87**	**1.83**	**0.93**	**0.69**
	Trans (E+A)	**0.86**	1.32	**0.87**	1.92	0.91	0.66
CELLO-3D [Landry 2019]	Rot	0.2	0.2	-	0.2	0.3	0.3
Brossard et al. [Brossard 2020]	Rot	3.7	2.3	1.2	2.9	4.2	4.7
Ours (Direct)	Rot (A)	2.63	3.46	3.60	5.76	7.34	7.81
	Rot (E+A)	2.52	3.39	3.52	5.44	7.33	7.42
Ours (Finetuned)	Rot (A)	**0.71**	**1.03**	0.81	**1.48**	1.69	**1.65**
	Rot (E+A)	**0.71**	**1.03**	**0.84**	1.56	**1.63**	1.69

Table 3.6: Trajectory Mahalanobis distance in terms of translation and rotation for the ETH dataset using aleatoric only (A) and combined epistemic-aleatoric (E+A) uncertainty. A comparison with the state-of-the-art is provided.

the scene flow the problem is cast on the reference cloud only. Additionally, the role of initialization and the use of the initial guess in the learning process can also be considered. Late stages of the network could encode the initial roto-translation as an additional input to relate the estimated error to the initial transform.

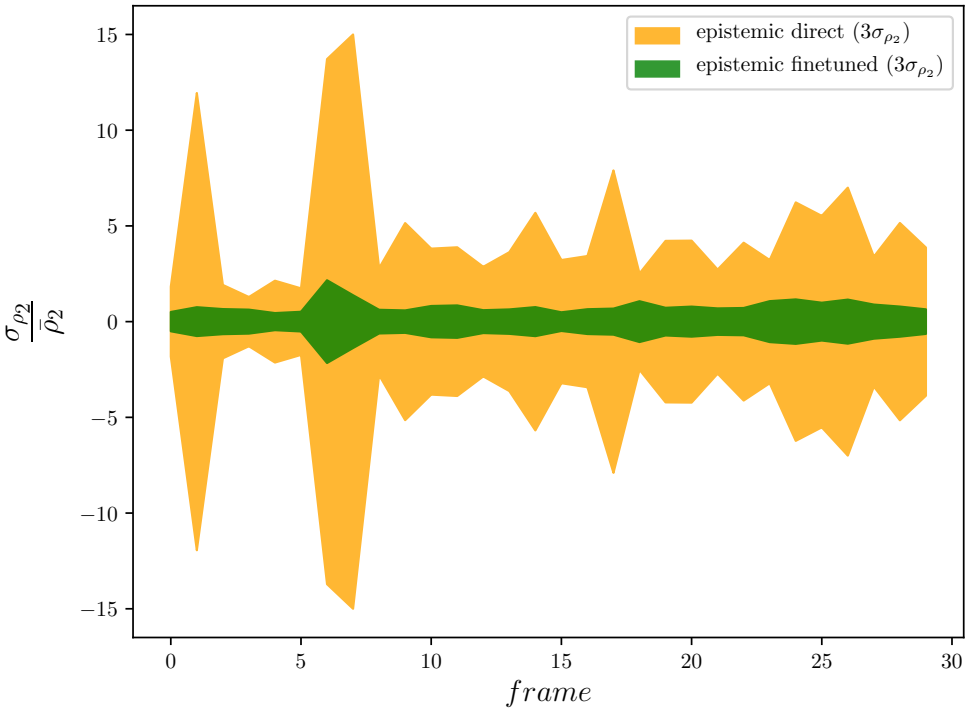


Figure 3.4: Epistemic uncertainties normalized by the mean in the direct and finetuned case.

CHAPTER 4

Process parameterization using (deep) reinforcement learning

Contents

4.1 Introduction

Visual odometry (VO) is a well established pose estimation process in robotics. Its applications vary within a wide range of domains, from autonomous cars to space exploration rovers. As many other perception processes, building VO entails dealing with a large number of configuration parameters. Tuning them ensures not only a proper execution of the process but strongly impacts the performances of the process itself. With the goal of showing a direct connection between parameters and the adaptability of VO to different contexts, we discuss an active parameterization based on deep learning techniques. One way to classify the different VO flavors is to distinguish its input: monocular VO, for single camera setups, and stereo VO, for stereoscopic cameras. While not being completely apart, in this chapter we tackle an autonomous parameterization of the latter version. Relying on stereo VO has several benefits, the most renowned being the capability to retrieve full scale information about robot motion.

VO featured significant advancements and implementations in the years. Since its earliest versions, VO has been built upon geometry-based (indirect) methods. Sparse feature-based techniques represented for many years the standard approach to the VO problem. A continuous work in this direction led to a well established pipeline composed by feature extraction, matching, motion estimation, outlier rejection and an eventual optimization [Scaramuzza 2011]. Oppositely to indirect methods, which aim at minimizing a geometric error, direct methods optimize a photometric error and have become more adopted in the recent years [Engel 2018]. They have been proposed to overcome the limits of sparse feature-based methods in absence of textured environments and in presence of low quality images such as motion blur. While dense/direct methods have proven to be on average more accurate than sparse methods, their accuracy degrade in presence of large stereo baselines [Engel 2013, Okutomi 1993].

Recently, the huge popularity gained by convolutional neural networks and deep learning in general, sprouted several works tackling pose estimation, *e.g.* [Konda 2015, Li 2017, Wang 2018]. These methods focus on delegating the full estimation process to the deep-learning (DL) architecture in an end-to-end fashion. Additionally, they often consider the monocular version of the problem, leaving the depth perspective either untackled or used as ground truth. Despite innovative methods, the classical sparse VO pipeline remains broadly used in the industry, especially in domains requiring robustness and proven efficiency. A notorious application field of sparse VO is the space sector, with planetary exploration rovers [Maimone 2007, Johnson 2008]. The high risks deriving from a system failure in these domains generates a natural reluctance to the adoption of more recent methods, preferring sturdiness to learning-based techniques. While geometry-based methods have found numerous applications, they lack the capability to adapt to different contexts, hence to different inputs. In fact, the process is always pre-configured with the goal to find the best average performance under the expected working environment. On the other hand, deep learning methods can be more adaptive to their input and learn to face multiple situations in their training phase but have not proven robust enough to be widely used in robotics.

We propose an intertwined approach between geometry-based and deep-learning methods. Our idea is to keep the classical structure of stereo sparse based VO systems putting in the loop a reinforcement learning scheme based on convolutional neural networks to actively drive the parameterization of the complete VO pipeline. Our approach aims at increasing the adaptivity of the systems to a series of different contexts whilst retaining the same robustness of sparse feature-based methods.

4.2 Visual Odometry Parameterization

As many other perception processes, VO is constituted of several atomic *nodes* which pipelined generate a desired data product [Govindaraj 2017]. Each of these nodes comes with a number of configurable parameters that when multiplied by the number of nodes in a process quickly grow to a large amount. Parameters can impact both the accuracy of the process and the performances in terms of used resources (cpu, memory, etc.). Naturally

different VO implementations come with different parameters. We discuss two implementations and their respective parameterization.

4.2.1 First instance: OpenCV-based 3D-2D Visual Odometry

In our VO implementation, presented in Sec. 1.3.2, we identified approximately twenty controllable parameters of different nature (integers, floats, enumerations). The parameters are responsible for the configuration of all the nodes composing VO, from the feature extraction to the motion estimation. Below, we discuss our VO implementation node by node (Fig. 4.1), and proceed to list their parameters along with their respective description. The first node is interest point extraction. An ORB implementation [Rublee 2011] can be parameterized by (at least) the parameters in Table 4.1. Feature extraction is normally fol-

parameters	**description**
features	maximum number of features to extract
scale factor	pyramid decimation ratio
pyramid levels	number of pyramid levels
edge threshold	border size where no features are extracted
score	point ranking system (Harris or FAST)
patch size	size of the patch used by BRIEF
fast threshold	FAST detection threshold

Table 4.1: ORB configuration parameters

lowed by feature matching (or tracking). Errors in matching are quite common, especially when matching points extracted at different time instants (stereo matching is easier, especially using rectified images). Part of the matches can be rejected based on the distance in the description space. The percentage of accepted matches is identified as another parameter to optimize at process level. Since stereo matches are transformed into 3D points, it is important to exclude wrong matches to ease the motion estimation process. As shown in [Beder 2006], the confidence ellipsoid of a 3D scene point varies accordingly to the angle between the intersecting rays projected from the two corresponding stereo image points. As the angle grows above a given threshold, the scene point covariance quickly increase on the depth axis. Furthermore, it is well known that the depth error grows quadratically with the depth itself [Gallup 2008]. Hence, it is desirable to control the maximum distance at which 3D computed points are accepted to be used in the motion estimation process. However, this threshold should be kept dynamic as in some cases, even imprecise points can prove useful in an estimation where no good matches can be retrieved closer to the cameras. There are several case scenarios for this situation, from unfavorable light conditions to noisy keypoints in the near plane. Finally, having an outlier rejection algorithm as RANSAC in the pipeline, *e.g.* at matching stage or at estimation stage, normally means dealing with three more parameters: samples, inlier threshold and probability of an outlier-free sample (confidence). The proposed description listed twelve controllable parameters for this VO implementation.

4.2.2 Second instance: LIBVISO2

Let us a consider a different VO implementation. A well-known alternative is *libviso2*, an open-source sparse VO algorithm [Kitt 2010]. Some of its traits are shared with the implementation discussed in previous section. It is a feature-based approach, with a RANSAC scheme that produces motion estimates using rectified stereoscopic pairs. Nonetheless, even for a similar architecture there are key differences intrinsic to the nodes that directly reflect onto the available parameters.

The feature matching process in *libviso2* relies on low-level blob and corner detection. Features are described concatenating the Sobel filter responses using a predefined layout. Non-minimum and non-maximum suppression (NMS)[Neubeck 2006] is applied on the filtered images to obtain the final feature candidate set. This process allows to select only maxima and minima in a neighborhood, suppressing similar features. Instead of detecting features using rotation and scale invariant features as ORB, assuming consecutive images coming from a smooth trajectory in the VO case, *libviso2* simply compares the sum of absolute differences (SAD) for the Sobel responses. This way the matching process is significantly sped-up. The features are matched in a *circular* way: features in the left image at time $t + 1$ are matched against the ones extracted in the left image at time t, then the process moves to right at time t, right at time $t + 1$, and finally the circle is closed returning at left $t + 1$. Ideally the starting feature will coincide with the final on, which would make the match accepted. Outlier rejection on matched features is performed using a 2D Delauney triangulation [Shewchuk 1996]. Matches are retained only if supported by at least tow neighboring matches falling withing disparity threshold τ_{disp} and flow threshold τ_{flow}.

To further speed up the matching process, the library takes advantage of a multi-stage approach. In the first stage, the considered candidates are filtered using a large NMS window, greatly reducing the number of evaluated features. The resulting matches are subdivided in bins with a predefined size (50x50 pixel). All the sparse matches are used to compute statistics and to narrow the final search space based on feature displacement. To finally estimate the motion, a bucketing process is performed with a twofold objective: to further reduce the number of features used in the egomotion estimation, and to spread features uniformly over the image space. The number of features per bucket is also predefined. The list of most important *libviso2* parameters in shown in Table 4.2.

4.2.3 Discussion

As seen in Sec. 4.2.1 and 4.2.2, even two similar approaches to the VO problem can have very different, large set of parameters. It is evident that an exhaustive search can become quickly unfeasible for more than two parameters, especially over the assumption that there is a dependence between the input images and the optimal parameter set. It would be necessary to perform a search that grows exponentially with the number of considered parameters for every estimation step. The solution would be a predictive approach modeling the connection between the input images and the error of the VO estimate. The model should dynamically return a set of optimal parameters given input stereo pairs and possi-

parameters	description
NMS n	Non-maximum suppression neighborhood size
NMS τ	Feature threshold for minima/maxima
matching binsize	Bin size for speeding up search
matching radius	Maximum radius to accept a match
matching disparity tolerance	Epipolar constraint tolerance
outlier disparity tolerance	Max distance to accept left-right matches
outlier flow tolerance	Max distance to accept flow matches
multi stage	Boolean to use multi-stage matching or single pass
refinement	Boolean to use sub-pixel refinement via parabolic fitting
bucket height	Height of each bucket grid window
bucket width	Width of each bucket grid window
bucket max features	Max number of features per bucket

Table 4.2: *libviso2* parameter list

bly additional information (*e.g.* initial motion guess from other sensors, velocity, process frequency). In the next section we will discuss a generic approach to parameterize process using (deep) reinforcement learning.

4.3 Adaptive process parameterization

We look for a method to predict the optimal set of VO parameters minimizing a given error function. In absence of a model-based approach to generate the parameter set, it comes natural to think about model-free learning approaches. However, the considered problem is not suitable for canonical approaches.

One of the main challenges is that the ground truth over the parameters is unknown. This prevents the possibility to employ an end-to-end supervised approach that directly learns to estimate parameters from images. Another issue comes from the knowledge we have as roboticists over the "quality" of a parameter set. In fact, a possible evaluation for a VO process is based on its accuracy with respect to the true transform between two frames. However, in practice, the most common way to assess the performances of VO is to compute informative metrics over a trajectory, taking into account translational drift in form of a percentage error over the traveled distance. One possibility would be to define a quadratic loss function as

$$\mathcal{L}(\mathcal{I}) = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{VO}(\mathcal{I}_i, \boldsymbol{x}_i) - \boldsymbol{y}_i)^2 \tag{4.1}$$

where $\boldsymbol{x}$ represents the parameters predicted for an image batch $\mathcal{I}$, the true transform is $\boldsymbol{y}_i$ and $\mathrm{VO}(\mathcal{I}_i, \boldsymbol{x}_i)$ is the output of VO using the corresponding parameters and stereo pair. In order to backpropagate the error of visual odometry through the network estimating the parameters, VO should be a differentiable function with respect to the $\boldsymbol{x}$ vector. However, VO cannot be described by any elementary function or as a composition of elementary functions, therefore preventing the use of automatic differentiation tools and learning frameworks in general.

4.3.1 Reinforcement Learning Background

Reinforcement learning has seen a vast adoption for control policies learned from high-dimensional sensory inputs since the pioneering work in [Mnih 2013]. The results obtained in teaching to an agent to successfully play Atari games ignited the interest towards deep reinforcement learning in several fields, including robotics. Initially, agents were trained using a variant of the Q-learning algorithm [Watkins 1992], where the inputs was a set of raw images and the output was a value function for each action in a given state. The problem is generally formalized as the maximization of future *rewards* obtained by an agent operating in an environment $\mathcal{E}$. A reward function is defined as the sum of rewards discounted over time by a factor γ as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ at time t. The output is defined as an action-value function $Q^*(s, a)$ expressing the maximum reward obtainable starting from state s and taking an action a. Following a policy π, the optimal Q-value can be expressed as $Q^*(s, a) = \max_\pi \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$. The optimal action-value is generally expressed through the Bellman equation, which makes the problem tractable iteratively. It is possible to rewrite the previous equation as

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{a'} Q^*(s', a') | s, a] \tag{4.2}$$

where r is the immediate reward for taking the action a in the state s and the second term returns the discounted future rewards from following the optimal policy, *i.e.* selecting the action in the next state with the maximum return. The iterative approach historically used in Q-learning implementation was only feasible for problems with a small set of states and actions. To tackle more complex problem it is common to estimate the Q-value using a function approximator, *e.g.* a neural network, having $Q(s, a; \theta) \approx Q^*(s, a)$, where θ represents the weight of the neural network. This approach takes the name of Deep Q-Learning and the networks trained with approach are referred to as Deep Q-Networks (DQN). As the goal of the network is to approximate the Q function, it is possible to train it using stochastic gradient descent with the following loss

$$\mathcal{L}_i(\theta_i) = (y_i - Q(s, a; \theta_i))^2 \tag{4.3}$$

where $y_i = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$ is the target value for iteration i recalling Eq. 4.2. This is different from a supervised learning approach where the target values are known beforehand. In this case y_i is dependent on the network weights from the previous iteration. This makes the algorithm model-free, *i.e.* it solves a problem without any knowledge about the environment, only sampling states and actions from the environment $\mathcal{E}$. While it would be possible to entirely train an agent with the following method, one of the issues highlighted in the literature relates to the inefficiency in training using consecutive, correlated samples. A solution is to use an *experience replay buffer* into which store the agent's experiences in the form of a tuple $e_t = (s_t, a_t, r_t, s_t + 1)$ for each time-step t. Once filled with an initial set of experiences, at training time, random samples are extracted to compose a minibatch that is normally used to train the network according to Eq. 4.3.

This approach has proven valid to address a set of problems. Successive improved version proved and addressed the issue of action value overestimation [Van Hasselt 2016].

One of the main limits of Deep Q-Learning is that the actions space is represented by the final DQN layer, with each output representing the predicted Q-value $Q(s, a)$ for an action given an input state. The work presented in [Lillicrap 2016] extends Deep Q-Learning ideas to continuous action spaces. While in the domain of the Atari games the action space was discrete, *i.e.* the number of commands input to the game is finite and enumerable, robotics problems, noticeably in the control theory field, often have continuous inputs. This is the case of the cartpole swing-up, pendulum, dexterous manipulation, locomotion and many others. Discretizing this action space comes with some problems, noticeably loss of generality and exploding complexity in high dimensional cases. For a process with n continuous inputs, even a very coarse representation of the type $a_i \in \{-k, 0, k\}$ quickly grows according to the input dimensionality with 3^n total possible action in each state. A finer representation would make the action space grow faster. To tackle this problem, [Lillicrap 2016] formalized the Deep Deterministic Policy Gradient (DDPG) algorithm. DDPG is based on the deterministic policy gradient (DPG) [Silver 2014], a model-free, off-policy actor-critic, adding to its ideas the use of a large, non-linear function approximator as in DQN. The actor-critic ideas is rather intuitive. The actor has to decide what action to perform in a given state and the critic evaluates the action taken by actor. DPG uses a parameterized actor function $\mu(s|\theta^\mu)$ which returns the current policy in a deterministic way. The critic function estimates $Q(s, a) \approx Q(s, a|\theta^Q)$ and is trained using the Bellman equation as in Eq. 4.3. The actor is updated applying the chain rule to the expected return J

$$\mathbb{E}_{s_t \sim \rho^\beta} = [\nabla_a Q(s, a|\theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s=s_t}] \tag{4.4}$$

where the state visitation distribution ρ^β of a stochastic behavior policy β is typically ignored. To explore continuous action spaces is more challenging than the discrete case (where usually an ε-greedy strategy is used). DDPG architecture has the benefit of separating the policy from the value function, thus making the exploration problem independent from the learning. It is possible to perturb the action in order to generate an exploration policy μ' as

$$\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N} \tag{4.5}$$

with $\mathcal{N}$ being a stochastic noise process. DDPG proposes the use of an Ornstein-Uhlenbeck (OU) process [Uhlenbeck 1930]. In our tests, similarly to [Fujimoto 2018], we found no particular advantages making use of OU exploration and rather perturb the actions with a Gaussian noise $\mathcal{N}(0, \sigma)$.

4.3.2 Parameterization of perception processes with reinforcement learning

The use of reinforcement learning for perception processes is not as widespread as in control tasks. This comes mainly from the nature of reinforcement approaches which are modeled for problems intrinsically presenting a state-action dependence. In fact, a sort of duality can be seen between a control and a perception problem. Input to robotic controllers reflect into changes in the *real* 3D state of the robot. Subsequent actions will be decided accounting for the actual robot configuration that is the next state. Oppositely, input to perception processes modify the *belief state* of the robot, that is the internal representation

of its own position and orientation. Actions will, however, not impact the next state.

Controlling a robot, *e.g.* for trajectory tracking purposes, lies in deciding the controller inputs that maximize some reward function evaluating how well the the trajectory is being tracked. A similar example is the collision avoidance task. A robot can learn to navigate an unknown environment solely relying on depth images. The learned agent will have to map optimal controller inputs to sensory data maximizing a return proportional to the time for each episode without a collision [Xie 2017]. It is clear that the taken action, *i.e.* deciding a linear and angular velocity, directly affects the next state of the robot, *i.e.* the perceived images.

In the perception case, a similar problem is the already presented visual odometry case. We map the agent state to the process input, in our case stereo pairs. The action space is represented by the natural boundaries for each parameter, *e.g.* distances in pixels cannot be negative. The output of the process is a motion estimate that will not affect the successive state. Independently from the estimation accuracy, the next stereo pairs will only be determined by the trajectory that the robot is following, which is out of our control.

This type of problem is commonly referred to as a *contextual bandit*. It naturally represents the link between the classic multi-armed bandit and the more complex full reinforcement learning world. Contextual bandits have seen a vast adoption in web applications for contextual recommendation systems. Their goal is indeed to map actions to single states, used as *context*. Then, the objective is to maximize the cumulative reward over a set of episodes

$$\max \sum_{t=1}^{T} r_t(s_t, a_t) \tag{4.6}$$

where the reward r_t is a function of the state-action pair (s_t, a_t). This is equivalent to have a discount factor $\gamma = 0$ in the Bellman equation. Adapting Eq. 4.2 to the problem we measure the performance of a policy as the expected reward a state distribution

$$Q^{\pi}(s) = (\mathbb{E}_s[r(s, a)|a \sim \pi(s)] \tag{4.7}$$

where the action a is chosen following the policy π.

We set-up an off-policy actor-critic approach, such as the one in [Fujimoto 2018], aiming at finding the optimal set of parameters for each time step maximizing the cumulative sum of rewards. Let us cast this approach on a simple process at first.

4.3.3 RANSAC adaptive inlier threshold

We demonstrate that this approach can be successfully applied to simple processes related to the perception layer. Let us consider the RANSAC process, an iterative method that estimates the parameters of a model from observations. It implicitly considers that the observations are noisy and that the data contains outliers. Being robust to outliers makes it broadly used to computer vision applications (*e.g.* matching, motion estimation) and can be more generally used as an outlier rejection method.

A simple RANSAC use case features linear line fitting in presence of outliers which corresponds to estimating a 1D affine transformation $y = ax + b$. Least-squares method

would largely suffer from outliers whereas RANSAC is more robust. The problem is twofold in reality: estimating the model parameters, and classify each data point as an inlier or an outlier. To solve the second part of the problem, an *inlier threshold* has to be chosen. This is generally decided depending on the noise information about the data. For instance on a normally distributed dataset with zero mean and variance σ, the threshold could be decided as $t = 3\sigma$. To ensure that a selected point is an inlier with a probability α, the threshold is dependent from several factors, and in particular from the probability distribution for the distance of an inlier from the model [Hartley 2004]. In practice, in absence of a lot of information about the data, the threshold is rather empirically chosen. Most algorithms adopt the Median Absolute Value (MAD):

$$M(|x_i - M(\boldsymbol{x})|) \tag{4.8}$$

where $M(\boldsymbol{x})$ is the median of a vector $\boldsymbol{x}$, *i.e.* the value separating the lower and upper half of the sorted version of the vector.

We propose to use the proposed approach to actively parameterize the *inlier threshold* for a 2D line fitting dataset. The algorithm is a variant of the DDPG. The main difference is that given $\gamma = 0$, there is no need for some components, noticeably no target networks are present. The actor selects a suitable inlier threshold based on the feedback it receives from the critic. The state is represented by the 2D coordinates of the entire point set. The

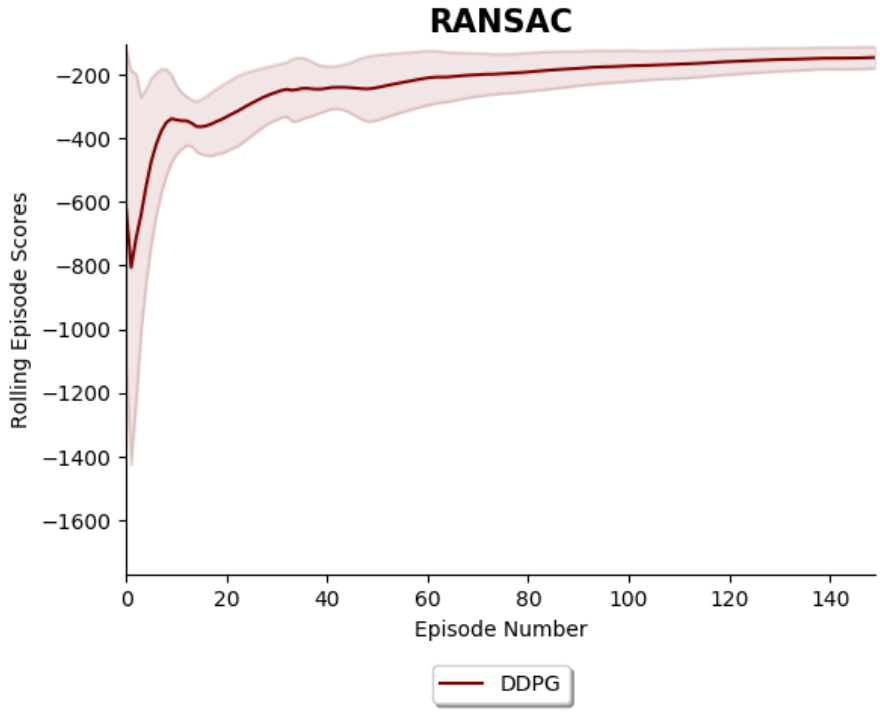


Figure 4.1: Rolling score over time for the RANSAC problem.

state space is $2n$, for a dataset containing n points and the action space is one-dimensional. The reward function is modeled with respect to the available ground truth, that is the true

	score	score^2	σ_{score}
adaptive	**-0.46**	**-0.62**	**0.54**
MAD	-0.61	-0.81	0.64

Table 4.3: Adaptive inlier threshold evaluation. Higher is better for *score* results. Lower is better for standard deviation.

line coefficient

$$r_t = -(a - \hat{a})^2 \tag{4.9}$$

where the function measures the distance between the coefficient estimated by RANSAC $\hat{a}$ and the true coefficient. Fig. 4.1 shows the cumulative reward on a moving average. The approach learns to predict the correct threshold, even outperforming the score obtained using the Mean Absolute Deviation. We evaluate over a dataset with dimension d the mean and squared distance from the true coefficient estimated using different thresholds. To consider the stability of this approach we also inspect the standard deviation of the reward. Given RANSAC randomness we evaluate the trained actor over 100 runs. Table 4.3 shows the results for a validation dataset.

In the following section, we will discuss about a possible formulation of the same problem in the VO context. The challenges and limits of the proposed method will be analysed focusing on the evolution of the VO error with respect to the chosen parameters.

4.4 Towards VO parameterization

As discussed, not having an available metric directly accounting for parameters, we have to rely on the ground truth. We model the reward function as an error minimization problem. The SE(3) distance is measured splitting the rotation and translation error. Given two transformation matrices respectively representing the true transform and its VO estimate, the errors are computed as

$$e_i^{rot} = ||\boldsymbol{k}_i\theta_i - \hat{\boldsymbol{k}}_i\hat{\theta}_i||_2 \quad \text{and} \quad e_i^{trans} = ||\boldsymbol{t}_i - \hat{\boldsymbol{t}}_i||_2 \tag{4.10}$$

where $\boldsymbol{k}$ and θ are the components of the axis-angle representation after the conversion from rotation matrix, and t is translation component. In order to have a single scalar for the reward function, the errors are composed as $r_t = -(e_i^{trans} + \beta e_i^{rot})$, with β working as a scaling factor to balance the two terms. The actor is a neural network returning a set of parameters from a stereo pair. The critic network has to first extract relevant features from the images and then return a value for the state-action pair. Similarly to [Fujimoto 2018], we propose an architecture that in early layers reduces the feature dimensionality from raw pixels to high-level features, and in fully connected layers reduces the state-action pair to the value function. The action is concatenated to the output of the previous layer for each fully connected layer. To model these networks, we start from the architectures proposed in Chap. 2 (Table 2.1). Ideally, as the two tasks are tightly coupled, convolutional filters trained for uncertainty estimation would be helpful in the parameter search for the same process.

4.4.1 Parameter space

To search in 10+ dimensional space is a daunting problem, to learn to predict actions in such a vast space is no exception. However, optimizing a subset of parameters can yield significant improvements in terms of accuracy. Fig. 4.2 shows the results over a KITTI trajectory optimizing a subset of parameters with different size. The three parameters control mainly the amount of features and matching that are used to estimate motion. It is worth to notice how vastly the accuracy of the estimation is improved by selecting the optimal parameters compared to the standard configuration. The optimal set was found using grid search over the parameter space defining lower and upper bounds for each parameter. This approach can be computationally expensive and took more than 2 days for an entire sequence.

4.4.2 Challenges and limitations

While the proposed approach worked well on a simple problem like RANSAC, in our tests did not consistently outperform the standard VO configuration. The reward value was not converging and the action distribution featured oscillation as well. We raise a question about the smoothness of true value function, *i.e*. the evolution of VO error wrt its parameters. Fig 4.3 shows the optimal parameters, amongst the ones considered in Fig. 4.2, for each stereo pair in Sequence 10. Consecutive samples on the x-axis correspond to images acquired in close instants while driving. In turn, they tend to largely overlap, presenting similar visual features and geometry. For this reason, one would expect some smoothness in the curve, however the plot shows a noisy signal with the optimal radius being uncorrelated with the image features. To motivate this behavior is not an easy task. On the one hand, we can assume that given the intrinsic randomness of the process (RANSAC, outliers in matching, etc.), even slight changes in the data can result in large variation on the optimal set. On the other hand, it is hard to predict how a change in the input data will propagate on the estimation, and, most importantly in our case, on the optimal parameter set. Fig 4.4 highlights the same problem in the two-dimensional case, exploring match radius and NMS τ. For consecutive images, with few visible changes to the human eye, the optimal duo of parameters changes both for rotation and translation error minimization purposes.

4.5 Conclusions

For the reasons discussed in the previous sections, the proposed approach did not result suitable for the VO parameterization problem. While the main challenge of this problem may, at first, seem the lack of ground truth on the optimal parameters, which calls for a contextual bandit modeling, this is not the case. Even casting this as a supervised problem, producing actual ground truth for one or two parameters, remains intractable to us. After the failure of the proposed approach, using grid search we generated the best parameters set for small spaces but were unable to achieve good results with this approach either. The intrinsic noisy signal presented by the ground truth has little correlation with the input data

as we previously discussed. Additionally, the presence of an unknown geometric process, with its own implementation, to the network does not facilitate its task. However, this study makes it possible to look closer at what happens *inside* Visual Odometry algorithms. In particular, as also discussed in Chap. 1, it highlights the importance of an adaptive parameterization of perception processes. The computer vision and robotics community rarely tend to dynamically modify these inputs, instead, the "configure and forget" is a common approach which should be put in discussion.

In this chapter, we presented a possible approach to search for optimal parameters for perception processes. It is discussed why this is needed with a special focus on the VO case. A possible solution based on *contextual bandits* and reinforcement learning is proposed. This demonstrated viable for simple processes like RANSAC but still not mature for more complex cases. In future, the transformation of this approach into a full reinforcement learning problem could be studied, by injecting the estimated belief state into the state-action-next state logic. In parallel, continuing to reason in perception *nodes* and *compounds*, the parameterization of smaller processes could be tackled. In principle, this approach should be an easier task compared to the tackled one.

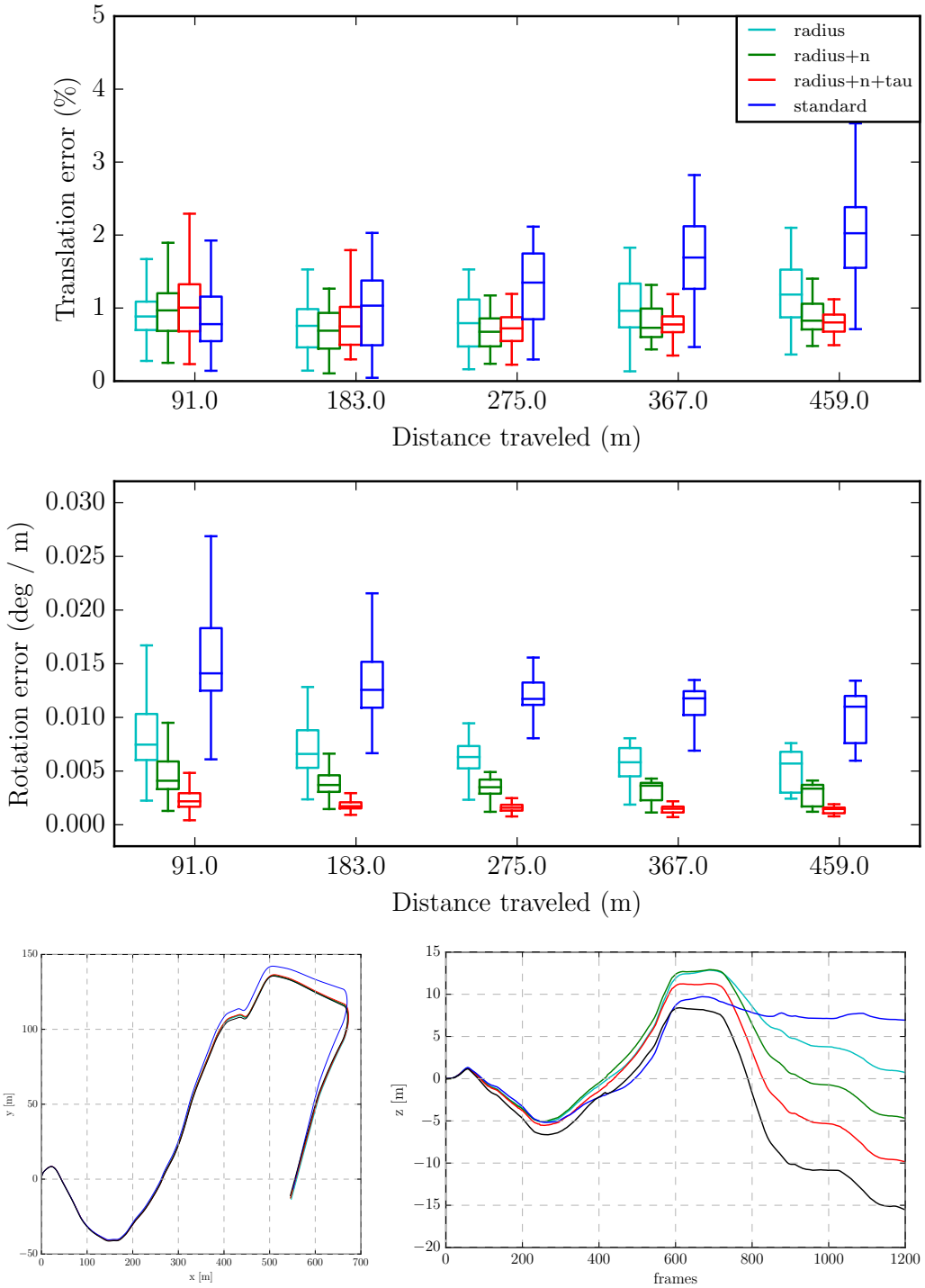


Figure 4.2: Error subplots (top) and trajectory evolution (bottom) of *libviso2* parameterized choosing the optimal parameters (up to 3 - cyan/green/red). Results of the standard configuration are shown in blue for comparison.

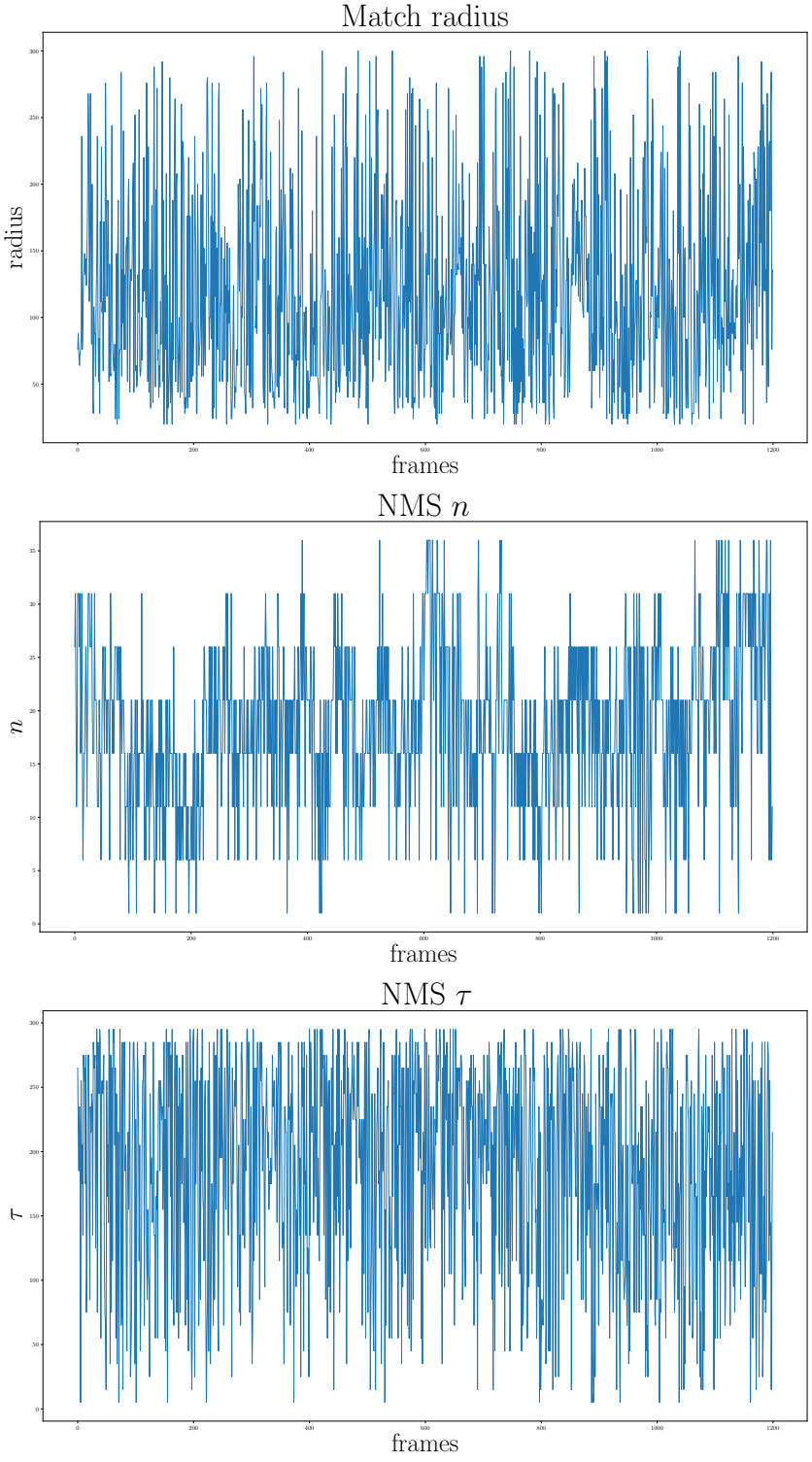


Figure 4.3: Optimal match radius (top), NMS n (middle), and NMS τ (bottom) for Sequence 10.

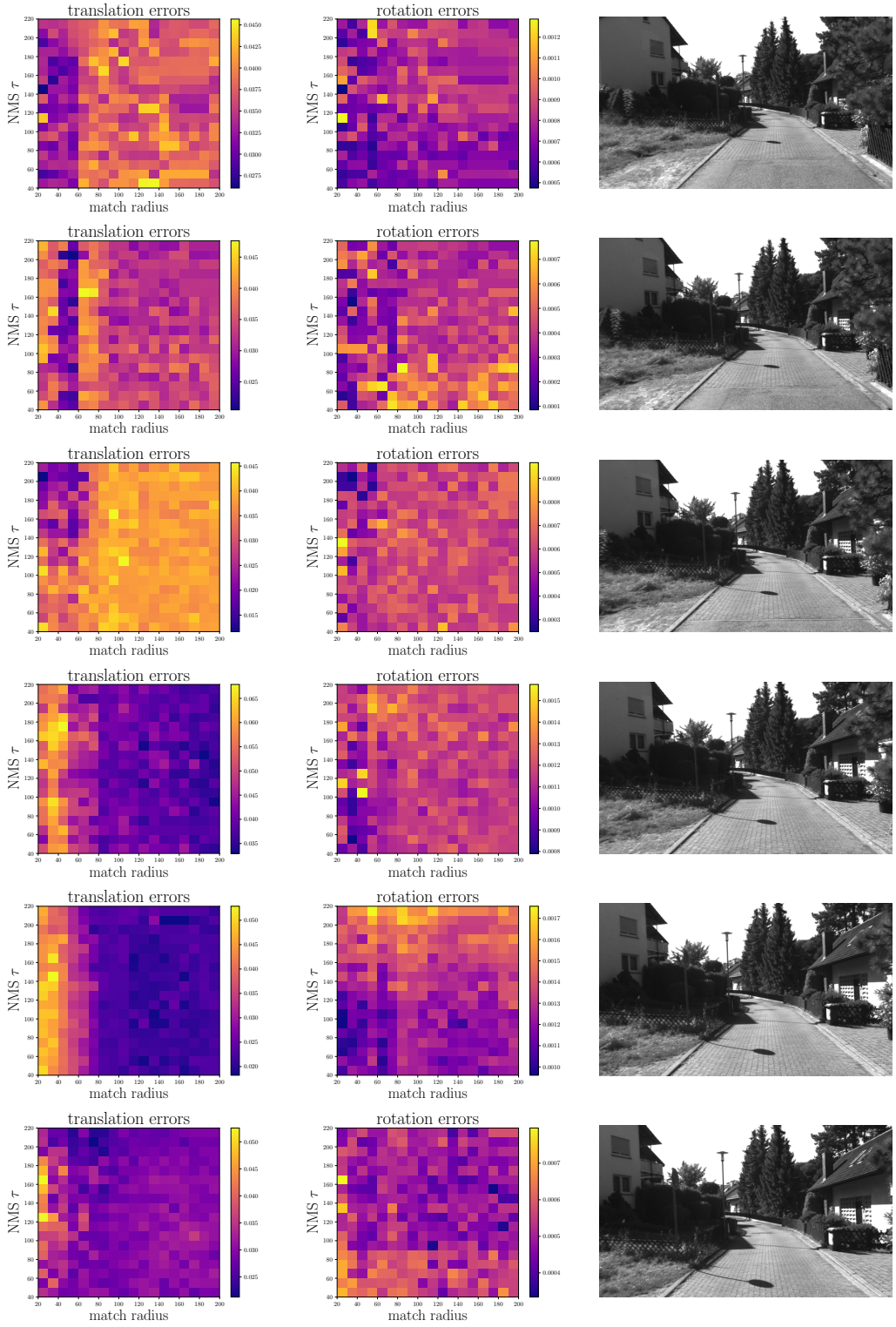


Figure 4.4: Errors in two-dimensional parameter (match radius vs NMS τ) exploration. Note the changes in optima for six consecutive images. Blue cells represent lower errors. Yellow cells represent higher errors.

Discussion

Summary

This manuscript presented a collection of works in the domain of active perception. Deep learning techniques are used to tackle the problems of estimating error models with respect to the input data of ego-motion estimation processes. Additionally, the modeling of perception process, along with their parameterization, is discussed in detail. The need for active control of these processes is showed with experiments leveraging visual odometry algorithms as a use case.

The first chapter shows the utility of having error functions. Moreover, it empirically verifies that an adaptive parameterization can yield improved results compared to preconfigured processes. The second and third chapters propose learning-based techniques to estimate error models for the visual odometry and iterative closest point case respectively. These approaches rely (almost) solely on the input data to these geometrical processes, showing that the most predominant error sources lie in the data itself. The final chapter transitions to the parameterization of perception processes. While error functions are key to estimate the performances of a process, it is important to adapt its configuration accounting for the operational context. A method based on actor-critic approaches tackles the problem of optimizing continuous parameter spaces. While the proposed solution is not successful in the visual odometry case, it allows to gain an insight into the dynamics of the parameters with respect to the ground truth errors. It is shown that the visual odometry errors are extremely dependent on the parameter choice but it is hard to find the optimal process configuration.

Concluding remarks and future works

This work addressed mixed concerns and tackled problems on different abstraction levels. Nonetheless, all the raised concerns are worthy of attention as we believe they are central to advance towards a more "intelligent" perception layer. On the one hand, we discussed about concrete cases of error models, corrections, and parameters, being at a relatively low-level of abstraction. On the other hand, in a more conceptual way, we introduced concepts dealing with the organization of the perception layer and of the formalization of its components. Each chapter represents a self-standing contribution which can be further enhanced.

The formalization proposed in Chap. 1 is not as important as the points it raises. The idea of formalizing the perception layer paves the way not only to control and evaluate the processes as already discussed, but also to recompose them using common building blocks. Feature extractors, filtering processes, and many others, are often interchangeable among their different implementations. It could be possible to demonstrate that a certain implementation would serve better in a given scenario while another one would perform better in a different one. Naturally, this would require a high-level supervision that in the

ideal case would enclose not only the perception layer but would orchestrate all the actors in the robotic architecture.

The architecture shown in Chap. 2 could be integrated with different type of uncertainties, notably accounting for epistemic uncertainty. Further validation could be carried out using alternative implementations of visual odometry and/or trying to associate the error model not only with the input data but also with the input parameters. In a more general perspective, the covariance prediction using learning-based techniques is an interesting field featuring a rich and active community. On the subject of ego-motion estimation, it can be interesting to test the approach on many different processes (and instantiation) since each own has its characteristics. This is already evident from the smaller biases present in ICP.

Chap. 3 architecture features an innovative network architecture based on point clouds. While this was originally designed for point flow, a related task to the one of ICP, a more specific one could be conceived. In particular, accounting for two flow embedding layers, one for each point cloud, in their respective frames, is a possibility. A dynamic range sampling based on point cloud boundaries could also be advisable. Moreover, the validation of the covariances in a pose-graph, similarly to what we showed in Chap. 2, would return more information on the efficacy of this method in real use cases.

The preliminary work in Chap. 4 leaves a large margin for improvements. While shallow function approximators work for simple processes, the presence of deep network normally complicates the task of reinforcement learning algorithms. Not only for visual odometry, but also for any other process, a possible way would be to pre-compute relevant features to feed to the critic and actor networks. The proposed use case leaves little hope that such approach would work given the chaotic nature of the optimal parameters with respect to the data, but this could be different for other process. Trying the proposed approach on ICP (or on other estimation process) would be worth, after a study on the impact of the parameters on the process performance (akin to what is shown in Fig. 4.3 and Fig. 4.4). Finally, it could be possible to transform the problem into a classical reinforcement learning approach. This would require to modify the state definition and create a state-actions-next state dependence. The implications of this choice should be carefully considered nonetheless. The association of a *belief state*, *i.e.* the 6D estimated pose via pose composition, would be a solution, but whether this information should affect the actions taken is debatable.

Bibliography

[Aqel 2016] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan and Napsiah Bt Ismail. *Review of visual odometry: types, approaches, challenges, and applications*. SpringerPlus, vol. 5, no. 1, page 1897, 2016.

[Bajcsy 1988] Ruzena Bajcsy. *Active perception*. Proceedings of the IEEE, vol. 76, no. 8, pages 966–1005, 1988.

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

[Beder 2006] Christian Beder and Richard Steffen. *Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence*. In Joint Pattern Recognition Symposium, pages 657–666. Springer, 2006.

[Besl 1992] P.J. Besl and N.D. McKay. *A Method for Registration of 3-D Shapes*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pages 129–256, 1992.

[Bonnabel 2016] Silvere Bonnabel, Martin Barczyk and François Goulette. *On the covariance of ICP-based scan-matching techniques*. In 2016 American Control Conference (ACC), pages 5498–5503. IEEE, 2016.

[Brossard 2020] Martin Brossard, Silvere Bonnabel and Axel Barrau. *A new approach to 3D ICP covariance estimation*. IEEE Robotics and Automation Letters, vol. 5, no. 2, pages 744–751, 2020.

[Buch 2017] Anders Glent Buch, Dirk Kraft*et al.* *Prediction of ICP pose uncertainties using Monte Carlo simulation with synthetic depth images*. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4640–4647. IEEE, 2017.

[Censi 2007] Andrea Censi. *An accurate closed-form estimate of ICP's covariance*. In Proceedings 2007 IEEE international conference on robotics and automation, pages 3167–3172. IEEE, 2007.

[Cho 2019] Younggun Cho, Giseop Kim and Ayoung Kim. *Deeplo: Geometry-aware deep lidar odometry*. arXiv preprint arXiv:1902.10562, 2019.

[De Maio 2017] Andrea De Maio and Simon Lacroix. *Towards a versatile framework to integrate and control perception processes for autonomous robots*. In 12th national conference on Software & Hardware Architectures for Robots Control & Autonomous CPS, 2017.

[De Maio 2018] Andrea De Maio and Simon Lacroix. *Enabling active perception through data quality assessment: a visual odometry case*. In 14th International Symposium

on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2018), Madrid, Spain, June 2018.

[De Maio 2019] Andrea De Maio, Quentin Labourey, Pierre Narvor, Ellon Paiva Mendes and Simon Lacroix. *Pose Management for Planetary Rovers*. In 15th Symposium on Advanced Space Technologies in Robotics and Automation, 2019.

[De Maio 2020] Andrea De Maio and Simon Lacroix. *Simultaneously Learning Corrections and Error Models for Geometry-based Visual Odometry Methods*. IEEE Robotics and Automation Letters, vol. 5, no. 4, pages 6536–6543, 2020.

[De Maio 2021] Andrea De Maio and Simon Lacroix. *Deep Bayesian ICP Covariance Estimation*. In Submitted to a computer vision conference, 2021.

[Deschaud 2018] Jean-Emmanuel Deschaud. *IMLS-SLAM: scan-to-model matching based on 3D data*. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 2480–2485. IEEE, 2018.

[Dubbelman 2012] Gijs Dubbelman, Peter Hansen and Brett Browning. *Bias compensation in visual odometry*. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2828–2835. IEEE, 2012.

[Engel 2013] Jakob Engel, Jurgen Sturm and Daniel Cremers. *Semi-dense visual odometry for a monocular camera*. In Proceedings of the IEEE international conference on computer vision, pages 1449–1456, 2013.

[Engel 2018] Jakob Engel, Vladlen Koltun and Daniel Cremers. *Direct sparse odometry*. IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 3, pages 611–625, 2018.

[Ferraz Colomina 2014] Luis Ferraz Colomina, Xavier Binefa and Francesc Moreno-Noguer. *Leveraging feature uncertainty in the PnP problem*. In Proceedings of the BMVC 2014 British Machine Vision Conference, pages 1–13, 2014.

[Fujimoto 2018] Scott Fujimoto, Herke Hoof and David Meger. *Addressing function approximation error in actor-critic methods*. In International Conference on Machine Learning, pages 1587–1596. PMLR, 2018.

[Furgale 2012] Paul Furgale, Pat Carle, John Enright and Timothy D Barfoot. *The Devon Island rover navigation dataset*. The International Journal of Robotics Research, vol. 31, no. 6, pages 707–713, 2012.

[Gal 2016] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. In Proceedings of the 33rd International Conference on Machine Learning (ICML-16), 2016.

[Gallup 2008] David Gallup, Jan-Michael Frahm, Philippos Mordohai and Marc Pollefeys. *Variable baseline/resolution stereo*. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[Geiger 2012] Andreas Geiger, Philip Lenz and Raquel Urtasun. *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. In Conference on Computer Vision and Pattern Recognition (CVPR), pages 3354–3361. IEEE, 2012.

[Govindaraj 2017] Shashank Govindaraj, Jérémi Gancet, Mark Post, Raúl Dominguez, Fabrice Souvannavong, Simon Lacroix, Michal Smisek, Javier Hildalgo-Carrio, Bilal Wehbe, Alexander Fabisch*et al. InFuse: A Comprehensive Framework for Data Fusion in Space Robotics*. In 14th Symposium on Advanced Space Technologies in Robotics and Automation, page 8p, 2017.

[Handa 2016] Ankur Handa, Michael Bloesch, Viorica Pătrăucean, Simon Stent, John McCormac and Andrew Davison. *gvnn: Neural network library for geometric computer vision*. In European Conference on Computer Vision, pages 67–82. Springer, 2016.

[Hartley 2004] R. I. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[Hartmann 2014] Wilfried Hartmann, Michal Havlena and Konrad Schindler. *Predicting matchability*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9–16, 2014.

[Johnson 2008] Andrew E Johnson, Steven B Goldberg, Yang Cheng and Larry H Matthies. *Robust and efficient stereo feature tracking for visual odometry*. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 39–46. IEEE, 2008.

[Kendall 2017a] Alex Kendall and Roberto Cipolla. *Geometric loss functions for camera pose regression with deep learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5974–5983, 2017.

[Kendall 2017b] Alex Kendall and Yarin Gal. *What uncertainties do we need in bayesian deep learning for computer vision?* In Advances in neural information processing systems, pages 5574–5584, 2017.

[Kitt 2010] Bernd Kitt, Andreas Geiger and Henning Lategahn. *Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme*. In 2010 ieee intelligent vehicles symposium, pages 486–492. IEEE, 2010.

[Konda 2015] Kishore Reddy Konda and Roland Memisevic. *Learning Visual Odometry with a Convolutional Network.* In VISAPP (1), pages 486–490, 2015.

[Kümmerle 2009] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss and Alexander Kleiner. *On measuring the accuracy of SLAM algorithms*. Autonomous Robots, vol. 27, no. 4, page 387, 2009.

[Kümmerle 2011] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige and Wolfram Burgard. *g2o: A general framework for graph optimization*. In 2011 IEEE International Conference on Robotics and Automation, pages 3607–3613. IEEE, 2011.

[Laconte 2019] Johann Laconte, Simon-Pierre Deschênes, Mathieu Labussière and François Pomerleau. *Lidar measurement bias estimation via return waveform modelling in a context of 3D mapping*. In 2019 International Conference on Robotics and Automation (ICRA), pages 8100–8106. IEEE, 2019.

[Lacroix 2019] Simon Lacroix, Andrea De Maio, Quentin Labourey, Ellon Mendes, Pierre Narvor, Vincent Bissonette, Clément Bazerque, Fabrice Souvannavong, Raphaël Viards and Martin Azkarate. *The Erfoud dataset: a comprehensive multi-camera and Lidar data collection for planetary exploration*. In 15th Symposium on Advanced Space Technologies in Robotics and Automation, 2019.

[Landry 2019] David Landry, François Pomerleau and Philippe Giguere. *CELLO-3D: Estimating the Covariance of ICP in the Real World*. In 2019 International Conference on Robotics and Automation (ICRA), pages 8190–8196. IEEE, 2019.

[Li 2017] Ruihao Li, Sen Wang, Zhiqiang Long and Dongbing Gu. *UnDeepVO: Monocular visual odometry through unsupervised deep learning*. arXiv preprint arXiv:1709.06841, 2017.

[Li 2019] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng and Jonathan Li. *Lo-net: Deep real-time lidar odometry*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8473–8482, 2019.

[Lillicrap 2016] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver and Daan Wierstra. *Continuous control with deep reinforcement learning.* In International Conference on Learning Representation (ICLR), 2016.

[Liu 2018] Katherine Liu, Kyel Ok, William Vega-Brown and Nicholas Roy. *Deep inference for covariance estimation: Learning Gaussian noise models for state estimation*. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1436–1443. IEEE, 2018.

[Liu 2019] Xingyu Liu, Charles R Qi and Leonidas J Guibas. *FlowNet3D: Learning scene flow in 3D point clouds*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 529–537, 2019.

[Maimone 2007] Mark Maimone, Yang Cheng and Larry Matthies. *Two years of visual odometry on the mars exploration rovers*. Journal of Field Robotics, vol. 24, no. 3, pages 169–186, 2007.

[Mendes 2016] Ellon Mendes, Pierrick Koch and Simon Lacroix. *ICP-based pose-graph SLAM*. In 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 195–200. IEEE, 2016.

[Mnih 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller. *Playing Atari With Deep Reinforcement Learning*. In NIPS Deep Learning Workshop. 2013.

[Neubeck 2006] Alexander Neubeck and Luc Van Gool. *Efficient non-maximum suppression*. In 18th International Conference on Pattern Recognition (ICPR'06), volume 3, pages 850–855. IEEE, 2006.

[Nistér 2004] David Nistér, Oleg Naroditsky and James Bergen. *Visual odometry*. In Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, volume 1, pages I–I. Ieee, 2004.

[Okutomi 1993] Masatoshi Okutomi and Takeo Kanade. *A multiple-baseline stereo*. IEEE Transactions on pattern analysis and machine intelligence, vol. 15, no. 4, pages 353–363, 1993.

[Peretroukhin 2014] Valentin Peretroukhin, Jonathan Kelly and Timothy D Barfoot. *Optimizing camera perspective for stereo visual odometry*. In 2014 Canadian Conference on Computer and Robot Vision, pages 1–7. IEEE, 2014.

[Peretroukhin 2017] Valentin Peretroukhin and Jonathan Kelly. *DPC-net: Deep pose correction for visual localization*. IEEE Robotics and Automation Letters, vol. 3, no. 3, pages 2424–2431, 2017.

[Pomerleau 2011] François Pomerleau, Stéphane Magnenat, Francis Colas, Ming Liu and Roland Siegwart. *Tracking a Depth Camera: Parameter Exploration for Fast ICP*. In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3824–3829. IEEE Press, 2011.

[Pomerleau 2012a] François Pomerleau, Andreas Breitenmoser, Ming Liu, Francis Colas and Roland Siegwart. *Noise characterization of depth sensors for surface inspections*. In 2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI), pages 16–21. IEEE, 2012.

[Pomerleau 2012b] François Pomerleau, Ming Liu, Francis Colas and Roland Siegwart. *Challenging data sets for point cloud registration algorithms*. The International Journal of Robotics Research, vol. 31, no. 14, pages 1705–1711, 2012.

[Pomerleau 2015] François Pomerleau, Francis Colas and Roland Siegwart. *A review of point cloud registration algorithms for mobile robotics*. Foundations and Trends® in Robotics, vol. 4, no. 1, pages 1–104, 2015.

[Postels 2019] Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab and Federico Tombari. *Sampling-free epistemic uncertainty estimation using approximated*

variance propagation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2931–2940, 2019.

[Prakhya 2015] Sai Manoj Prakhya, Liu Bingbing, Yan Rui and Weisi Lin. *A closed-form estimate of 3D ICP covariance*. In 2015 14th IAPR International Conference on Machine Vision Applications (MVA), pages 526–529. IEEE, 2015.

[Qi 2016] Charles R Qi, Hao Su, Kaichun Mo and Leonidas J Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. arXiv preprint arXiv:1612.00593, 2016.

[Qi 2017] Charles R Qi, Li Yi, Hao Su and Leonidas J Guibas. *PointNet++: Deep hierarchical feature learning on point sets in a metric space*. arXiv preprint arXiv:1706.02413, 2017.

[Qi 2019] Charles R Qi, Or Litany, Kaiming He and Leonidas J Guibas. *Deep Hough voting for 3D object detection in point clouds*. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9277–9286, 2019.

[Rublee 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. *ORB: An efficient alternative to SIFT or SURF*. In Computer Vision (ICCV), 2011 IEEE international conference on, pages 2564–2571. IEEE, 2011.

[Scaramuzza 2011] Davide Scaramuzza and Friedrich Fraundorfer. *Visual odometry [tutorial]*. IEEE robotics & automation magazine, vol. 18, no. 4, pages 80–92, 2011.

[Shewchuk 1996] Jonathan Richard Shewchuk. *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*. In Workshop on Applied Computational Geometry, pages 203–222. Springer, 1996.

[Silver 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra and Martin Riedmiller. *Deterministic policy gradient algorithms*. In International conference on machine learning, pages 387–395. PMLR, 2014.

[Sola 2018] Joan Sola, Jeremie Deray and Dinesh Atchuthan. *A micro Lie theory for state estimation in robotics*. arXiv preprint arXiv:1812.01537, 2018.

[Sorkine-Hornung 2017] Olga Sorkine-Hornung and Michael Rabinovich. *Least-Squares Rigid Motion Using SVD*. Computing, vol. 1, page 1, 2017.

[Stephan 2017] M Stephan, Matthew D Hoffman, David M Blei*et al.* *Stochastic Gradient Descent as Approximate Bayesian Inference*. Journal of Machine Learning Research, vol. 18, no. 134, pages 1–35, 2017.

[Thomas 2019] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette and Leonidas J Guibas. *Kpconv: Flexible and deformable convolution for point clouds*. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6411–6420, 2019.

[Uhlenbeck 1930] George E Uhlenbeck and Leonard S Ornstein. *On the theory of the Brownian motion*. Physical review, vol. 36, no. 5, page 823, 1930.

[Urban 2016] Steffen Urban, Jens Leitloff and Stefan Hinz. *MLPnP-A Real-Time Maximum Likelihood Solution to the Perspective-n-Point Problem*. arXiv preprint arXiv:1607.08112, 2016.

[Van Hasselt 2016] Hado Van Hasselt, Arthur Guez and David Silver. *Deep reinforcement learning with double q-learning*. In Proceedings of the AAAI Conference on Artificial Intelligence, 2016.

[Wang 2017] Sen Wang, Ronald Clark, Hongkai Wen and Niki Trigoni. *DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks*. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 2043–2050. IEEE, 2017.

[Wang 2018] Sen Wang, Ronald Clark, Hongkai Wen and Niki Trigoni. *End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks*. The International Journal of Robotics Research, vol. 37, no. 4-5, pages 513–542, 2018.

[Wang 2020] Naiyan Wang and Zhichao Li. *Dmlo: Deep matching lidar odometry*. arXiv preprint arXiv:2004.03796, 2020.

[Watkins 1992] Christopher JCH Watkins and Peter Dayan. *Q-learning*. Machine learning, vol. 8, no. 3-4, pages 279–292, 1992.

[Xie 2017] Linhai Xie, Sen Wang, Andrew Markham and Niki Trigoni. *Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning*. In RSS 2017 workshop on New Frontiers for Deep Learning in Robotics, 2017.

[Zhang 2014] Ji Zhang and Sanjiv Singh. *LOAM: Lidar Odometry and Mapping in Real-time.* In Robotics: Science and Systems, 2014.

[Zhang 2018] Zichao Zhang and Davide Scaramuzza. *A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry*. In IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS), 2018.