University of Zurich[UZH]

Department of Informatics

Decentralized Multi-Agent Visual SLAM

Dissertation
submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by

Titus Cieslewski
from Visp, VS, Switzerland

approved in February 2021

at the request of
Prof. Dr. Davide Scaramuzza, advisor
Prof. Dr. Marc Pollefeys, examiner
Prof. Dr. Torsten Sattler, examiner

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, 17.02.2021

The Chairman of the Doctoral Board: Prof. Dr. Thomas Fritz

Perfection is achieved,
not when there is nothing more to add,
but when there is nothing left to take away.
— Antoine de Saint-Exupéry

To my friends and family.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Davide Scaramuzza for accepting me as a PhD student in his lab. Davide provided me with many exciting opportunities, while also giving me the freedom and resources to pursue my own ideas.
This thesis would not have been possible without the help, fruitful discussions, and fun distractions from my colleagues. I therefore wish to express my gratitude to all the current and past members, visitors, and students. I would particularly like to thank Christian Forster, Elias Müggler, Matthias Fässler, Henri Rebecq, Davide Falanga, Zichao Zhang, Antonio Loquercio, Philipp Föhn, Elia Kaufmann, Mathias Gehrig, Daniel Gehrig, Manasi Muglikar, Yunlong Song, Giovanni Cioffi, Nico Messikommer, Manuel Werlberger, Jeff Delmerico, Suseong Kim, Guillermo Gallego, Dario Brescianini, Dimche Kostadinov, Javier Hidalgo Carrio, Christian Pfeiffer, Michael Gassner, Alessandro Simovic, Julien Kohler, Felix Böwing, Thomas Längle, Manuel Sutter, Barza Nisar, Kevin Kleber, Kunal Shrivastava, Stefano Ghidoni, Rubén Gómez Ojeda, Naveen Kuppuswamy, Kosta Derpanis, Timo Stoffregen, Francisco Javier Perez Grau, Bianca Sangiovanni, Yuto Suebe, Roberto Tazzari, Sihao Sun and Tamar Tolcachier. I also had the pleasure to work with great students, namely Mathis Kappeler, Filippo Martinoni, Kaju Bubanja, Yawei Ye, Andreas Ziegler, Yves Kompis, Thomas Eppenberger, Amadeus Oertel, Roy Rutishauser, and Maxime Choulika. Furthermore, I was fortunate enough to work with great international collaborators, namely Siddharth Choudhary, Michael Bloesch, and Kosta Derpanis. I would like to thank the agencies funding my research, namely Armasuisse, the National Centre of Competence in Research (NCCR) Robotics, the Swiss National Science Foundation, the DARPA FLA Program, and the UZH Forschungskredit Candoc. I would like to thank Prof. Marc Pollefeys and Prof. Torsten Sattler for accepting to review my thesis.
Last but not least, I am very grateful to my parents, and my friends who supported me at all times.

Zurich, October 2020 T. C.

Abstract

Simultaneous Localization and Mapping – SLAM – is an algorithm which confers to agents a sense of the environment and how they move through it. Examples of such agents are autonomous robots, but also augmented- or virtual reality devices worn by humans. It allows them to build a map of the environment, to localize themselves within that map, and to plan routes between points of interest inside that map. Multi-agent SLAM extends this sense of the environment to a group of agents, allowing them to additionally profit from each others' knowledge of the environment. In this thesis, this knowledge is acquired using vision. Cameras provide rich information for various tasks and are compact, low-cost, and ubiquitous at the same time.
The drawback of this rich information is that it would per default require a lot of bandwidth to transmit between agents. While new infrastructure provides more and more high-bandwidth wireless communication, it is far from covering all multi-agent SLAM application environments. It has trouble penetrating through rock, walls and water, and across large distances. Besides, saving bandwidth enables scalability to larger groups of agents, and leaves the bandwidth available to other uses. This thesis thus focuses particularly on multi-agent mapping using minimal data exchange. For similar reasons – scalability and bandwidth savings – the emphasis is put on developing a decentralized, as opposed to a centralized SLAM system.
In the pursuit of such a system, three components of visual SLAM, where data is exchanged between the participating agents, emerge: place recognition, relative pose estimation, and map optimization. In this thesis, I mainly work on ways of achieving the former two with as little data exchange as possible. As for map optimization, I make the point that it is most likely not needed in most decentralized visual SLAM applications. Map optimization mitigates the global inconsistency that inevitably happens due to drift in visual odometry. In the past, however, it has been shown that global consistency is not necessary for navigation in the map. In my thesis, I show that it is not needed for exploration, either. The following is a list of contributions of this thesis, in chronological order:

- A method for decentralized bag-of-words-based visual place recognition for a group of n agents which requires n times less data exchange than conventional decentralized place recognition.
- A similar method for decentralized learned-embedding-based visual place recognition which requires even less data exchange.
- A first truly data-efficient, full decentralized visual SLAM system based on state-of-the-art components, including the previous contribution, and additional

methods for reducing the data exchange, without sacrificing the accuracy of the shared map. Ten robots operating for one minute only exchange 2MB of data **in total** for full multi-agent visual SLAM functionality.

- A detector for visual features which is capable of extracting a minimal set of feature points that enables localization with as little as 50 visual features.
- A completely new approach to feature detection and matching, where features that are implicitly matched between images are detected, thus rendering feature descriptors obsolete in the considered application case.
- A data representation for exploration which enables exploration using a globally inconsistent state estimate.

List of Contributions

Journal Publications

- Amadeus Oertel, **Titus Cieslewski**, and Davide Scaramuzza. "Augmenting Visual Place Recognition with Structural Cues". In: *IEEE Robotics and Automation Letters (RA-L)* (2020). DOI: 10.1109/LRA.2020.3009077
  Links: PDF, Video

- Hyungpil Moon, Jose Martinez-Carranza, **Titus Cieslewski**, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, Guido de Croon, Sunyou Hwang, Sunggoo Jung, Hyunchul Shim, Haeryang Kim, Minhyuk Park, Tsz-Chiu Au, and Si Jung Kim. "Challenges and implemented technologies used in autonomous drone racing". In: *Springer: Intelligent Service Robotics* (2019). DOI: 10.1007/s11370-018-00271-6
  Links: PDF

- **Titus Cieslewski**, and Davide Scaramuzza. "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index". In: *IEEE Robotics and Automation Letters (RA-L)* (2017). DOI: 10.1109/LRA.2017.2650153
  Links: Appendix A, PDF, Presentation

Peer-Reviewed Conference Papers

- Philipp Foehn, Dario Brescianini, Elia Kaufmann, **Titus Cieslewski**, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. "AlphaPilot: Autonomous Drone Racing". In: *Robotics: Science and Systems (RSS)* (2020). DOI: 10.15607/RSS.2020.XVI.081
  Links: PDF, Video

- **Titus Cieslewski**, Andreas Ziegler, and Davide Scaramuzza. "Exploration Without Global Consistency Using Local Volume Consolidation". In: *IFRR International Symposium on Robotics Research (ISRR)* (2019).
  Links: Appendix F, PDF, Video

- **Titus Cieslewski**, Konstantinos G. Derpanis, and Davide Scaramuzza. "SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning". In: *IEEE International Conference on 3D Vision (3DV)* (2019). DOI: 10.1109/3DV.2019.00072
  Links: Appendix D, PDF, Poster, Video, Code

- **Titus Cieslewski**, Michael Bloesch, and Davide Scaramuzza. "Matching Features without Descriptors: Implicitly Matched Interest Points". In: *British Machine Vision Conference (BMVC)* (2019).
  Links: Appendix E, PDF, Poster, Code

- Jeff Delmerico, **Titus Cieslewski**, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2019). DOI: 10.1109/ICRA.2019.8793887
  Links: PDF, Video, Dataset Website, Code

- **Titus Cieslewski**, Siddharth Choudhary, and Davide Scaramuzza. "Data-Efficient Decentralized Visual SLAM". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2018). DOI: 10.1109/ICRA.2018.8461155
  Links: Appendix C, PDF, Video, Presentation, Code

- **Titus Cieslewski**, and Davide Scaramuzza. "Efficient Decentralized Visual Place Recognition From Full-Image Descriptors". In: *IEEE International Symposium on Multi-Robot and Multi-Agent Systems (MRS)* (2017). DOI: 10.1109/MRS.2017.8250934
  Links: Appendix B, PDF, Presentation

- **Titus Cieslewski**, Elia Kaufmann, and Davide Scaramuzza. "Rapid Exploration with Multi-Rotors: A Frontier Selection Method for High Speed Flight". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017). DOI: 10.1109/IROS.2017.8206030
  Links: PDF, Video, Presentation

- Yawei Ye, **Titus Cieslewski**, Antonio Loquercio, and Davide Scaramuzza. "Place Recognition in Semi-Dense Maps: Geometric and Learning-Based Approaches". In: *British Machine Vision Conference (BMVC)* (2017). DOI: 10.5244/C.31.74
  Links: PDF, Poster

- Michael Gassner, **Titus Cieslewski**, and Davide Scaramuzza. "Dynamic Collaboration without Communication: Vision-Based Cable-Suspended Load Transport with Two Quadrotors". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2017). DOI: 10.1109/ICRA.2017.7989609
  Links: PDF, Video, Presentation, Poster

Open-source Software

- SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning
- Matching Features without Descriptors: Implicitly Matched Interest Points
- Code to accompany the UZH-FPV Drone Racing Dataset
- TensorFlow implementation of NetVLAD
- Data-Efficient Decentralized Visual SLAM

Awards

- **RSS Best Systems Paper Award, 2020.**
- **IROS Best Search and Rescue Robotics Paper Award Finalist, 2017.**

Contents

1 Introduction

Any agent acting in the real world needs to have a notion of its environment to efficiently operate in it. More concretely, typical skills that should be enabled by this "sense of the environment" are the ability to recognize whether the place the agent is in right now is a previously visited place or a new place, or the ability to navigate from the current location to a desired location. The knowledge required for this is typically represented in a map. Thus, an agent that wants to acquire the aforementioned skills will both continuously build its map, while also always trying to localize itself within the existing map. In other words, it will perform Simultaneous Localization and Mapping – SLAM.

Due to its importance for autonomous robots, SLAM has been studied in the robotics community for a long time, and single-agent SLAM has recently reached maturity to the point where it is deployed in industry, such as for augmented and virtual reality (AR/VR) [1] or for drones [2]. Still, it remains an interesting research topic, as different aspects of it remain open problems.

In this thesis, I go one step beyond single-agent SLAM and consider multi-agent SLAM. Multi-agent SLAM gives groups of agents a shared sense of the environment, and a sense of each other, thus being crucial for collaborative tasks. Examples of collaborative tasks are multi-robot exploration (faster than single-robot exploration), multiplayer AR/VR games, group coordination through AR devices, or heterogeneous tasks, like construction, where different robots with different skill are required. Figure 1.1 illustrates the utility of both collaborative AR and multi-robot exploration with an example from science fiction. In multi-agent SLAM, the aforementioned desired skills are extended to a group of agents: rather than one agent recognizing places it has been to before, any agent should be able to recognize places any other agent in the group has been to before. Rather than one agent being able to navigate to a place of interest it has been to before, any agent should be able to navigate to any place of interest previously seen by any other agent. This requires communication between the agents, and one of

[1] https://www.microsoft.com/en-us/hololens, https://www.oculus.com/, https://www.magicleap.com/en-us

[2] https://www.skydio.com/

Figure 1.1 – Multi-agent augmented reality (AR) and multi-robot exploration as envisioned by modern science fiction. In the movie "Prometheus", a group of scientists equipped with AR devices encounters an unknown cave system. Rather than incurring risk and wasting time by exploring the system themselves, they deploy a group of robots that scans the caves. The map is sent back to headquarters, including the positions of robots (red) and scientists (yellow) within the map (© Scott Free Productions, Brandywine Productions, Dune Entertainment, 2012).

the key challenges I am tackling in this thesis is to make this communication as minimal as possible. Keeping the communication minimal allows to develop multi-agent SLAM system with a larger group of agents, or it allows for deployment in environments where the available bandwidth is limited, such as underwater, in cave systems, indoors, or across large distances.

In Section 1.1, I discuss further decisions in setting up the problem statement, such as why I focus on Visual SLAM in particular, and why I focus on decentralized, rather than centralized multi-agent SLAM. In Section 1.2, I set up the research objectives that make up this thesis and which have driven the individual contributions that this cumulative thesis consists of. Section 1.3 reviews and discusses the related literature that has been relevant to my thesis. Chapter 2 then summarizes the papers in the appendix and their connections to each other. Finally, Chapter 3 provides future research directions.

1.1 Problem Statement and Design Decisions

1.1.1 Visual State Estimation and Mapping

While Visual SLAM and its components are described in due detail in Section 1.3.1, this section is dedicated to motivating why vision is used as the modality for state estimation and mapping. The fact that evolution has led us to use vision to find our whereabouts might be an indicator enough that it is a good idea to use vision also for artificial agents. However, there are several alternative technological solutions for state estimation, such as GPS, optical tracking systems, or laser range finders.

GPS and optical tracking systems indeed have the advantage over vision that they can give an accurate and drift-free pose, or at least position estimate. Furthermore, given existing products, they are programmatically rather simple to implement. However, they are both restricted in where they can be applied: On the one hand, GPS only works reliably outdoors, even there succumbing to failures such as reflections in urban canyons [44]. Optical tracking, on the other hand, relies on external infrastructure and can thus only be used in rooms where such infrastructure has been deployed. In contrast, visual state estimation works in any environment with sufficient texture – a light source is typically present where humans are present, but can also be brought along otherwise.

A popular alternative method for state estimation in robotics are laser range finders. They typically provide one or several planar scans that determine the free space between the sensor and the nearest obstacle. For now, however, laser range finders remain significantly more expensive than cameras, and they are typically heavier, making them less useful for applications where state estimation needs to be provided in a lightweight package, such as on quadrotors. Finally, they usually have a limited range and capture

much less information about the environment than cameras. Cameras can be used for many other tasks beyond state estimation, and the fact that they capture more information means that they are also able to describe places more uniquely, which is relevant for recognizing previously visited places and navigation.

1.1.2 Why a Decentralized System?

A conscious choice in this thesis has been to work specifically on **decentralized** multi-agent SLAM, rather than centralized multi-agent SLAM. Centralized algorithms are generally simpler, as all data can be sent to a central entity. Centralized algorithms thus have access to all data. Decentralized algorithms, in contrast, have to deal with the fact that none of the involved entities has access to all data unless all data is exchanged – which would be inefficient in terms of bandwidth use. Hence, decentralized algorithms are conceptually more challenging, but there are several advantages to using them.

There are two ways to implement a centralized algorithm: either there is a dedicated central machine, or one of the agents participating in the multi-agent activity is designated as a central entity. The former case is restricting in applications, as all agents need access to the central entity – the group cannot move too far away from the central entity or from communication infrastructure that allows access to the central entity. In the latter case, the dedicated agent represents a bottleneck in several ways: firstly, in terms of computation and memory. This can be fine if known a priori: one of the agents can be equipped with a more powerful computer than the others. However, this might preclude ad-hoc assemblies of existing lightweight devices, such as multi-rotors or augmented reality devices. Secondly, the dedicated central entity poses a bottleneck in terms of bandwidth. This can be particularly problematic in scenarios with limited bandwidth, such as environments with high electromagnetic absorption like under water [158], in environments with many walls, or across large distances, absent communication infrastructure. In a centralized system, the central entity has to receive all of the data, whereas, in well-designed decentralized algorithms, data tend to travel less, and can be well-distributed across the participating agents. Thirdly, the central entity poses a bottleneck in terms of robustness to failure. In a central system, if the central entity fails, everything fails. Decentralized systems, in contrast, can be designed for robustness to the loss of some of the participating agents. However, this typically needs to be explicitly designed for, typically at the cost of more communication [161]. A similar case can be made for privacy, especially in adversarial scenarios: in a centralized system, capturing or hijacking the central entity gives access to the full information [32], or, worse, could lead to control over the full system.

Finally, in a decentralized system, bandwidth can be saved since it might not be necessary to send out certain data at all. For example, decentralized map optimization can be achieved with only exchanging a fraction of the overall map data (see Section 1.3.4), as

each agent mostly processes its own data, whereas a centralized optimization algorithm would need to gather all of the data from the agents.

1.1.3 The Components of Decentralized Visual SLAM

Single-agent visual SLAM has three components: firstly, **Visual Odometry (VO)** processes the incoming sensor data and tries to establish, ideally in real-time, an initial estimate of both the trajectory of the agent, as well as of the environment as observed by the camera, represented as a sparse or semi-dense set of 3D landmarks [176]. The trajectory estimate is typically given as a series of six-degree-of-freedom (DoF) poses. A pose $\boldsymbol{T}_A^B$ expresses the spatial transformation between two Cartesian 3D reference frames A and B. It consists of a 3D rotation described by a rotation matrix $\boldsymbol{R}_A^B$ and a position vector $\boldsymbol{t}_A^B$, both expressed in frame A, unless otherwise stated. The axes of frame B can be found by first translating the axes of A by $\boldsymbol{t}_A^B$ and then by rotating them by $\boldsymbol{R}_A^B$. Using linear algebra, the concatenation of such transformations can be expressed with 4×4 matrix multiplication [3]. Given three frames A, B and C:

$$\boldsymbol{T}_A^B := \begin{bmatrix} \boldsymbol{R}_A^B & \boldsymbol{t}_A^B \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \tag{1.1}$$

$$\boldsymbol{T}_A^C = \boldsymbol{T}_A^B \cdot \boldsymbol{T}_B^C. \tag{1.2}$$

Formally, then, visual odometry provides a series of estimates of the agent frame A with respect to the odometry frame O over time, $\bar{\boldsymbol{T}}_O^A(t)$, O conventionally coinciding with the initial pose of the agent $\bar{\boldsymbol{T}}_O^A(t=0) = \boldsymbol{I}$. Visual odometry is subject to drift, meaning that the error between the estimated pose $\bar{\boldsymbol{T}}_O^A$ and the actual pose $\boldsymbol{T}_O^A$ tends to grow over time.

The second component of single-agent SLAM, **Loop Closure Detection**, compares new sensor data against the already established map to see whether the currently visited location has already been seen in the past. The output of loop closure detection is a relative pose estimate $\bar{\boldsymbol{T}}_Q^M$ between the current (query) agent frame Q and a previous, matched frame M.

Loop closures can then be used in the third component, **Optimization**, to mitigate drift: since VO is subject to drift, there will be an inconsistency between the poses estimated by the VO and the relative pose provided by loop closure detection:

$$\bar{\boldsymbol{T}}_O^Q \cdot \bar{\boldsymbol{T}}_Q^M \neq \bar{\boldsymbol{T}}_O^M \tag{1.3}$$

The fact that, given (1.2), equality should hold in (1.3), is exploited in an optimization

[3] For efficiency reasons, pose manipulation is usually implemented using quaternions. Conceptually, however, pose manipulation using rotation matrices is much more efficient to communicate.

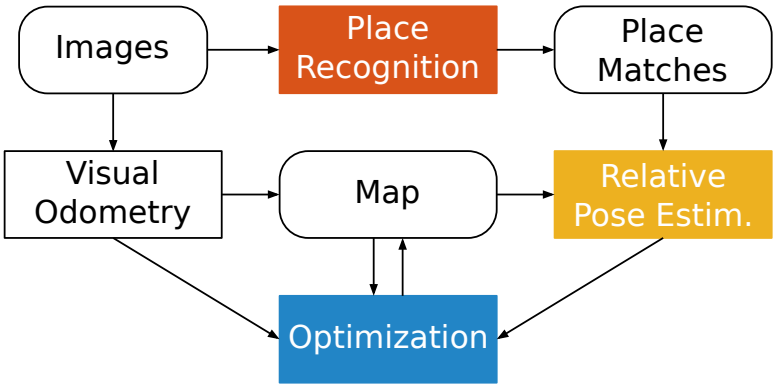


Figure 1.2 – The components of decentralized SLAM as I proposed in [35]. The colored blocks represent components that involve data exchange between agents.

process that corrects the drift within the set of poses affected by this inconsistency [100], along with the 3D structure observed from those poses. This results in a globally consistent map, which typically exhibits less error than it did before incorporating the loop closure in the optimization.

These components transfer to decentralized multi-agent SLAM, with some modifications: **Visual Odometry (VO)** works exactly like in single-agent SLAM, it can run independently on each individual agent. As for **Loop Closure Detection**, however, I propose to explicitly decompose it into **Place Recognition** and **Relative Pose Estimation**. The reason for this is that it only makes sense to establish a precise relative pose once it has been established that the query place has indeed been previously visited by another agent: firstly, a place recognition query requires less data than a query for a full loop closure detection. Secondly, a place recognition query needs to be addressed to every other agent, while the pose estimation query only needs to be addressed to the single agent with the place match. So it only makes sense to decompose loop closure detection into these two blocks in order to reduce the amount of exchanged data. Finally, the **Optimization** component remains the same, but now needs to be performed in a coordinated manner between several, if not all agents, since loop closure constraints and the accordingly affected pose estimates can now span across the trajectories of several agents. These components are visualized in Figure 1.2. Of these components, place recognition, relative pose estimation, and optimization are involved in data exchange, and are hence the focus of this thesis.

1.2 Research Objectives

The research objective of this thesis, in one sentence, is to achieve decentralized SLAM with as little data exchange as possible. The extent to which SLAM is "achieved" depends on evaluation metrics. Traditionally, SLAM systems are evaluated by comparing

the error between the estimated trajectory to the true trajectory of the agent. I will, however, make the argument that this metric does not best capture the utility of SLAM for many applications where a globally consistent or accurate map is not needed. This is important because if a globally accurate map is not needed, the optimization component of SLAM can be dropped, resulting in far less data exchange. In that case, the considered performance metric is reduced to the accuracy of loop closure detection. VO quality can still matter for accuracy in a local scope. The contributions of this thesis, however, are VO-agnostic (apart from Paper C, which uses optimization), meaning that they can be combined with any VO meeting any desired standard for local accuracy.

While a part of this thesis is about implementing a full decentralized visual SLAM system, the overall objective can be tackled by considering the individual components of SLAM as outlined in Section 1.1.3. Hence, in the following sections, I give a quick overview of the challenges faced when attempting to implement them in a data-efficient and decentralized manner.

1.2.1 Place Recognition

A property that makes place recognition unique among the other components is that a naive decentralized implementation requires more data exchange than a centralized implementation: In order to find out whether an agent sees a place that has previously been seen by another agent, every agent would constantly send a place recognition query to every other agent. Consequently, the complexity of overall data exchange with respect to the number of agents n is $O(n^2)$. In contrast, in a centralized system, the agents would send these queries only to the central entity, resulting in a complexity of $O(n)$. Thus, an important challenge in making decentralized SLAM viable is in solving this problem of scalability.

When it comes to the data requirements of individual queries, place recognition does not have as much potential for further size reduction. As I will elaborate in Section 1.3.1, modern place recognition methods rely on nearest-neighbour retrieval in high-dimensional vector spaces. The fewer dimensions such a space has, the more efficient such retrieval can be, and so a lot of effort has already been undertaken in the existing literature to reduce the size of the vectors, which are the main constituent of the query data.

1.2.2 Relative Pose Estimation

To save data exchange, the agents involved in decentralized multi-agent SLAM will each store their own map data. Consequently, to establish relative poses between agents, queries will be sent to the agent which contains the portion of the map indicated by the response to a place recognition query. Since the pose estimation query will be

sent only to one agent, the amount of data exchanged is the same in a centralized and decentralized system.

So the focus in this thesis will be on the composition of the query. As will be elaborated in Section 2.3, the typical way to establish relative pose visually is through a set of n visual features, each with their position and their descriptor, resulting in a query size of

$$d_{\text{RPE}} = n \cdot (d_{\text{Coord.}} + d_{\text{Desc.}}), \tag{1.4}$$

where $d_{Coord.}$ is the byte size of the coordinates of the feature in the image and $d_{\text{Desc.}}$ the size of the descriptor. Choosing a minimal $d_{\text{Coord.}}$ is trivial, so the goal is to find methods for relative pose estimation that reduce n and/or $d_{\text{Desc.}}$ – or use a completely different approach to relative pose estimation.

1.2.3 Map Optimization

Unlike for the previous two components of decentralized visual SLAM, there is already a rich literature that has attempted to achieve decentralized map optimization with minimal data exchange. Consequently, this has not been the main focus of this thesis. However, as elaborated in Section 1.3.5, a technique called visual teach-and-repeat has shown that it is not necessary to optimize maps in order to be able to navigate within them. This raises the question: is it necessary to optimize at all? Rather than to consider visual SLAM as an end in itself, the goal of this part of the thesis is to investigate the applications of SLAM and to see whether there might be applications other than navigation that do not require optimized maps.

1.3 Related Work

1.3.1 Single-Agent Visual SLAM and its Components

In this section, I review the most relevant work in single-agent visual SLAM, which serves as a basis for this thesis, as well as recent developments that have affected my research during the thesis. I start by reviewing the individual components of SLAM as outlined in Section 1.1.3 and finally discuss full SLAM systems. An overview of the geometry and the interactions between the components when using a standard approach is provided in Figure 1.3. I omit literature related to single-agent map optimization, as it is not relevant to my thesis. Multi-agent map optimization is reviewed in Section 1.3.4.

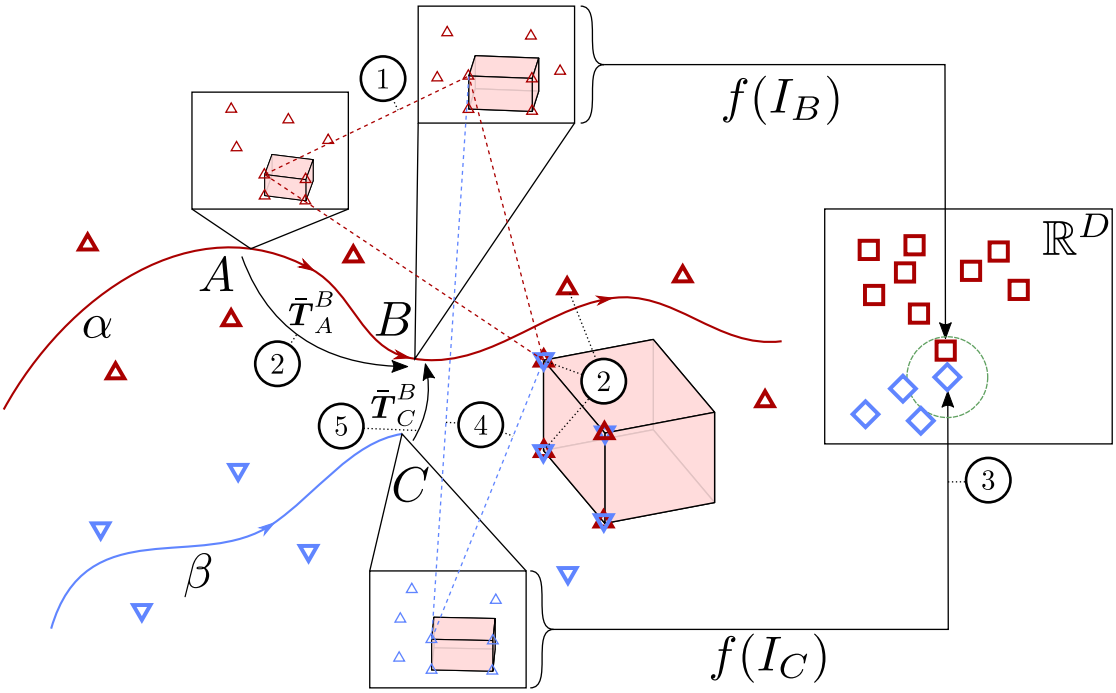
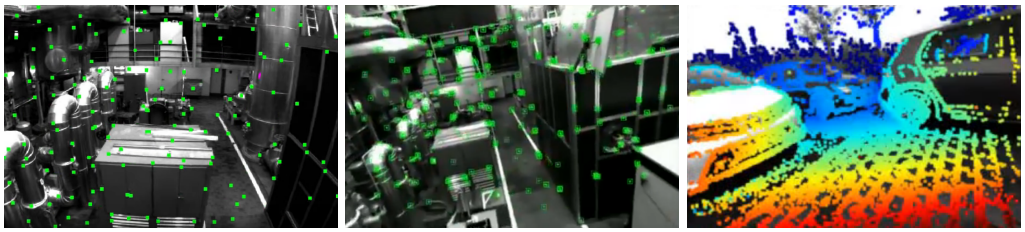


Figure 1.3 – The approach to Visual SLAM used in this thesis, based on the literature reviewed in this chapter. The curved lines indicate the trajectory of two agents, α and β. Along these trajectories, the agents record images I; here the images at poses A, B and C are shown. ① Visual features (△, Section 1.3.2) are extracted in each image, and associated between subsequent images. ② Based on these associations, **Visual Odometry** estimates both the 6-DoF transformation $\bar{\boldsymbol{T}}_A^B$ between poses, and the 3D positions of the points corresponding to each feature, allowing an estimate of the overall trajectory. ③ To localize in the map of agent α, agent β first establishes which image of agent α – if any – best matches the currently observed image, using **Place Recognition**. In learned embedding-based place recognition, a Neural Network f transforms all images into D-dimenional vectors (*embeddings*) $f(I)$. Due to how this network is trained, because frames B and C observe the same scene, their embeddings are within a detection threshold (circle). Thus, β can efficiently determine that its image I_C observes the same scene as α's I_B using nearest neighbor retrieval. ④ For **Relative Pose Estimation**, β matches the features of I_C with those of I_B based on this place recognition result. ⑤ This matching implies an association to the 3D structure observed by α, from which β can derive a relative transformation $\bar{\boldsymbol{T}}_C^B$ to the map of α.



(a) SVO [67] **(b)** ORB-SLAM [149] **(c)** DSO [60]

Figure 1.4 – Different approaches to Visual Odometry with different point densities. SVO only extracts and tracks 121 points that can be directly tracked inexpensively. These points, however, are not detected repeatably enough for loop closure detection. ORB-SLAM extracts 2000 points, out of which it usually manages to track upwards of 500. The same points are used for visual odometry and loop closure detection, both times relying on descriptor matching. DSO and its predecessor LSD-SLAM attempt to track every point with a sufficient gradient.

Visual Odometry Visual Odometry (VO) takes as input a series of images taken by a camera and provides as output an estimate of its 6Dof trajectory $\bar{\boldsymbol{T}}_O^A(t)$ [176]. Depending on its implementation, a VO can return additional information about the environment. VOs typically work image-by-image, associating the points observed in the current image with points observed in the previous image. Over a sequence of images, a VO then knows how points have moved through the image plane and can calculate from that both the 6dof trajectory of the camera and the 3D position of each point. VO approaches, then, differ in the choice of the points to be tracked, the method by which points are associated between images, and the math that is used to transform point tracks into camera trajectory and point location.

When it comes to the choice of points to be tracked, VO approaches can foremost vary in the density of the extracted points, see Figure 1.4. Some VO algorithms select a sparse set of points, and can thus be either very computationally efficient [67] or specifically select points that are easy to track also under challenging conditions such as motion blur [19]. Other VO algorithms select a large number of points to track, or simply attempt to track all points, resulting in a much more detailed representation of the environment, and robustness to individual point tracking failures [59, 60]. When selecting a sparse set of points, VO algorithms can rely on a wide body of work on feature selection methods, which are discussed in further detail in Section 1.3.2.

To track points across images, VO approaches can rely either on direct tracking methods or on feature matching. Direct tracking methods can either be applied to points individually [12, 19, 125], or the known geometry can be used to more rapidly track all points simultaneously [67]. Instead of direct tracking, sparse sets of feature points can also be associated by matching their descriptors (see Section 1.3.2). This can be more robust to strong changes in viewpoint and is thus more commonly used for relative pose estimation in loop closure detection, but it is also used in VO, such as in the VO

implementation of [149].

Once point tracks have been established, the most common method to derive the camera trajectory and point location is to use perspective-n-point algorithms. 2D-2D algorithms such as the five-point [109, 152] or eight-point [122] algorithm are used to provide an initial estimate of two camera frame poses and the 3D location of the points matched between them. The problem of simultaneously estimating camera poses and 3D point positions is naturally scale-agnostic: The same image observations can be explained by either a small trajectory in a small environment or a large trajectory in a large environment. While this scale-invariance transfers to visual odometry, repeatedly applying 2D-2D algorithms to subsequent frame pairs would not even guarantee a consistent scale between subsequent camera pose estimates. Hence, VO algorithms only use 2D-2D algorithms to provide an initial set of 3D point locations. To keep a scale that is consistent with this initial estimate, the subsequent camera frame poses are then localized with respect to the initial set of 3D points using a 2D-3D algorithm like P3P [83, 105]. Simultaneously, the set of 3D points is expanded by triangulating new point tracks from the poses of newly added camera frames. Since point tracking is subject to errors, VO implementations combine 2D-2D and 2D-3D algorithms with robust estimation methods, most commonly random sample consensus (RANSAC) [64].

To estimate the correct scale, VO algorithms typically either use a stereo camera setup with known geometry or fuse visual state estimation with the output of an inertial measurement unit (IMU) [19, 118, 160]. A recent review of VO algorithms additionally using inertial data (Visual- Inertial Odometry, VIO) is provided in [50].

Very recently, VO methods have emerged which rely on machine learning, implicitly performing many of the above subtasks inside of a neural network architecture [119, 214]. To obtain scale, these approaches often implicitly guess the scale [79, 209], like in monocular depth estimation [77].

Place Recognition While for SLAM, place recognition is only one of two steps that are required for loop closure detection, it is often the sole subject of publications, as it is a very complex problem in its own right. A naive problem statement for visual place recognition is "Given two images, determine whether they were taken at the same location". However, in SLAM a single query image typically needs to be matched to a correct image among a very large set of images. Doing this by individually solving the aforementioned problem statement for all possible matches would scale very poorly, and so solving place recognition in practice also implies using an efficient retrieval method. This problem of image retrieval is very similar to document retrieval that is used by text search engines. One of the earliest and still popular approaches, *bag of visual words* (BoW), has thus solved image retrieval by treating images like text documents [186]. This is achieved by extracting visual features (see Section 1.3.2) and

associating them with *visual words* from within a dictionary. This "visual dictionary" is constructed a priori by clustering the feature descriptor space. Each cluster is considered a word and features are associated with the word within whose cluster their descriptor lies. Image retrieval is then performed by doing document retrieval with visual words. This approach has subsequently been improved upon by, among others, using hierarchical clustering in a vocabulary tree [151, 159], by applying it to features with binary descriptors [73], and by properly handling repetitive structures in the scene [199].

With the recent rise of the popularity of machine learning and convolutional neural networks, a paradigm shift has occurred in the place recognition literature. Neural networks can be constructed in such a way that their input is an image I and their output is a high-dimensional vector $f(I)$. Such an architecture can then be trained in a way that if two images I_1 and I_2 are from the same place, a distance (euclidean or other) between $f(I_1)$ and $f(I_2)$ is small, while if I_1 and I_2 are from different places, that distance is larger. This still lends itself to efficient retrieval, as efficient nearest-neighbor algorithms, such as k-d trees [15], can be used to retrieve the closest match from among a large set of potential matches. During this thesis, the most popular such approach has been NetVLAD [7], the distinctive feature of which is that its architecture is derived from VLAD [8, 97], an advanced alternative to BoW.

So far, this review has discussed place recognition methods that are based on single-image retrieval, but it needs to be noted that place recognition can also be achieved using other methods, such as using more than just a single image [126, 139]. Two of my contributions that are not related to the main thesis have considered this aspect of place recognition and are discussed in Section 2.5. Furthermore, a well-known problem in place recognition is perceptual aliasing, where there is an ambiguous environment in which there are multiple places that look very similar. Several approaches to deal with this problem exist [111, 131, 190].

Relative Pose Estimation Relative pose estimation is the part of localization/loop closure detection that is applied to obtain a 6DoF relative pose to previous parts of the trajectory once a candidate map region has been determined using place recognition. Similarly to visual odometry, this is most often achieved using feature detection and matching and a Perspective-n-Point algorithm coupled with robust estimation. Like in visual odometry, it can also be achieved using direct image alignment [59].

Localization Localization, which is usually composed of place recognition and pose estimation, serves for loop closure detection in SLAM but is an area of research of its own [171, 172, 173]. It has many other applications, most importantly providing the pose of an agent in an already-established map of an environment. A recent problem

that is considered in the localization community is robustness to changes in appearance, such as due to weather, light, or seasonal changes [128].

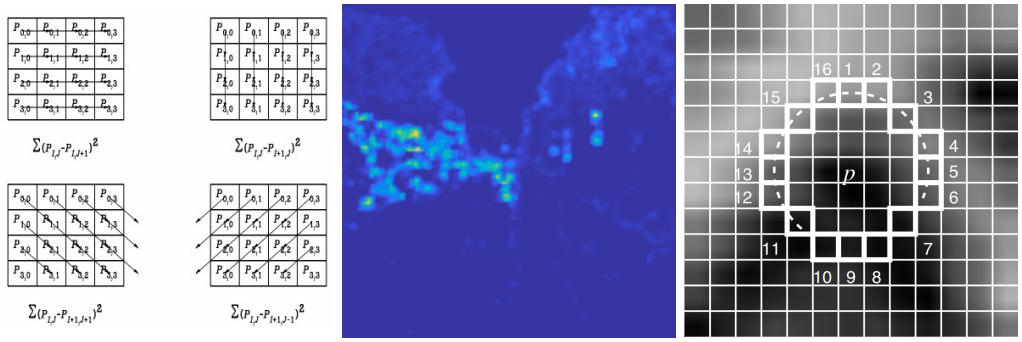
Full Single-Agent Visual SLAM systems The de-facto standard visual SLAM system for monocular and stereo cameras encompassing all of the above components at the time of this thesis has been ORB-SLAM [148, 149]. In visual-inertial SLAM, VINS-Mono [160] also has the ability to detect and incorporate loop closures, relying on [73], just like ORB-SLAM. A peculiarity of ORB-SLAM is that it uses the same features (ORB features [170]) and feature matching method for odometry and loop closure detection, thus recycling some of the computation between these tasks. While this can save computation overall, using different features for odometry and loop closure detection is just as valid an approach [180], allowing for a more modular SLAM system.

1.3.2 Feature detection and description

Feature selection and description is a crucial component of many visual odometry and relative pose estimation approaches. It is of particular importance for this thesis, as it defines the data that is exchanged to establish relative poses between agents: a large part of this thesis has dealt with designing features that can be used to achieve SLAM while being as compact as possible, see Section 2.3.

Features are usually established by first *detecting* them in an image, that is, determining the pixel location of features in an image, and then *describing* them. The goal of feature description is to provide a feature *descriptor* (typically a high-dimensional vector), which is invariant to geometric and photometric transformations, such that a feature can be correctly associated between two images, even if the camera viewpoint or the scene lighting has changed. Similarly to place recognition, the rise in popularity of machine learning and convolutional neural networks has inspired a paradigm shift in the feature community.

Traditional Feature Detection Classically, features have been selected from distinctive image locations, that is, image locations that significantly differ from neighboring image locations. This is a prerequisite for a reliable association of points between different images: points that are not distinctive get confused with their neighbors. Whether an image location is distinctive can be determined explicitly [144] or using a first-order approximation, such as the Harris [84] or Shi-Tomasi [183] detector (Figure 1.5). Alternatively, distinctive features can be detected by convolving the image with a suitable kernel, such as the Laplacian of Gaussian (LoG), or its faster approximation, the Difference of Gaussians (DoG) kernel [124]. Other filter-based detectors have been proposed by [14, 130]. Another method for detecting features is to identify



(a) Moravec [144] **(b)** Shi-Tomasi [183] **(c)** FAST [167]

Figure 1.5 – Traditional concepts for feature detection. In his 1980 thesis "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover", Moravec proposed an "interest operator" which checks whether an image region looks different when shifted sideways and diagonally. The more it does, the more interesting the center point is considered. Harris and Shi-Tomasi proposed an efficient approximation to this check, the figure shows an example image response. FAST proposes to instead only consider the pixels in a circle around the point and to use arc lengths of consistent intensity to explicitly distinguish points on corners, edges, or homogeneous surfaces.

image regions that explicitly resemble a known type of distinctive points, such as corners. Efficient ways to do this have been proposed by [81, 187, 200] and have found widespread popularity with [167] due to its highly efficient implementation. Technically, convolution-based methods also respond to specific patterns in the image. However, the explicit methods attempt to achieve detection in a computationally less expensive way than with convolution and non-maxima-suppression, see Figure 1.5c. Similarly to feature description, an important problem in feature detection is ensuring that the detection is independent of geometric transformations. Scale invariance can be achieved using multi-scale detection [135], while rotational invariance is part of the design in several of the aforementioned detectors.

Learning-Based Feature Detection With the recent popularity of convolutional neural networks (CNNs), CNNs have also been considered for feature detection. They are particularly well-suited for this, as a per-pixel interest score as shown on the top of Figure 1.6 can be directly calculated with a series of convolutional layers. LIFT [216] was the first to exploit CNNs for several components of feature detection and description. It uses a separate network for detection, orientation estimation, and description of features. This work is trained on patches provided by an SfM algorithm executed a priori. Lenc and Vedaldi [116] train a feature detector using a "covariance constraint", which trains the response of a network to be invariant to viewpoint changes. Savinov et al. [175] generalize this constraint to enforce a consistent ranking of responses of corresponding points between viewpoints. Superpoint [52] proposes a sophisticated training method that involves first fitting a network to detect labeled corners in a

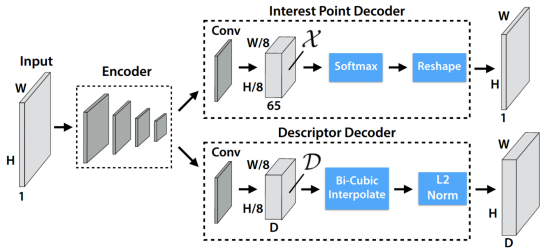


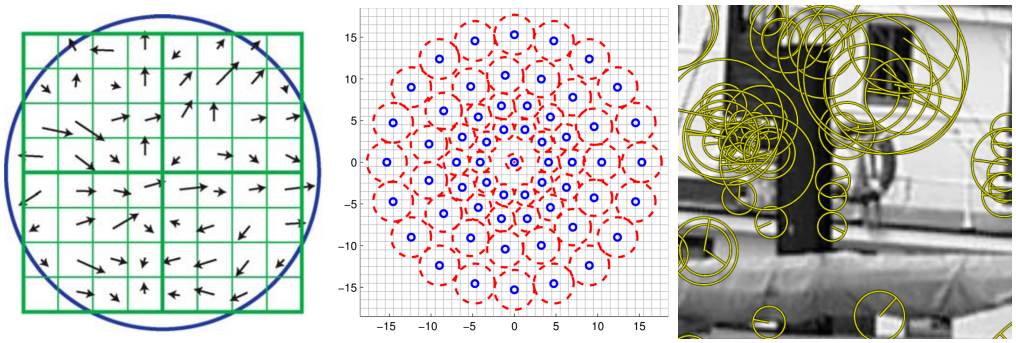
Figure 1.6 – Superpoint [52] (image source), and several other CNN-based integrated feature extractors share parts of the initial CNN layers for both detection and description.

synthetic dataset, and later to train invariance on real images by warping real images. In contrast to previous work, the detector and descriptor are trained jointly. Similarly, LF-Net [155] is trained on the full point correspondence algorithm. The loss acting on the LF-Net detector is similar to the covariance constraint of [116], augmented with a heuristic to favor responses that resemble narrow Gaussian kernels. See [115] for a recent comparison of several feature detection methods.

Traditional Feature Description One could naively use the image patch surrounding a feature detection as a descriptor, but this is fragile to slight changes in illumination and viewpoint. To work for visual odometry and localization, descriptors need to be invariant to some amount of illumination and viewpoint change, yet still distinctive enough to differtiate between the different features extracted in one image. A popular class of traditional descriptors is histograms of gradients (HoG) [124] as shown in Figure 1.7a. Another example is binary descriptors, which are particularly efficient to calculate [27, 117, 170], and thus particularly popular in the robotics community, see Figure 1.7b.

Most descriptors, however, are still sensitive to large affine transformations such as changes in scale and orientation. Consequently, modern feature matchers use multi-scale detection [117, 135], and orientation estimation [124, 170] as shown in Figure 1.7c. Using these, the image patches used for descriptor calculation can be normalized with respect to these geometrical transformations. These normalizations are often also explicitly modeled in learning-based feature descriptors [155, 216].

Learning-Based Feature Description Similar to feature detection, the recent success of convolutional neural networks (CNNs), has led the community to revisit traditional feature description methods. D output channels of a CNN can simply be interpreted as the coefficients of a D-dimensional descriptor, as shown on the bottom of Figure 1.6. Alternatively, recent work has shown that these channel outputs can also be interpreted as a visual word response [184, 197]. Like in place recognition, the CNN can then be



(a) SIFT [124] **(b)** BRISK pattern [117] **(c)** BRISK normalization

Figure 1.7 – Traditional concepts for feature description. HoG-based descriptors like SIFT accumulate a histogram of gradient orientations in regions surrounding the interest point. Binary pattern-based descriptors like BRISK make a binary comparison of intensity between average values within image regions in a pattern around the interest point. One can imagine that an appropriately constructed CNN should be able to generalize both of these approaches. To be invariant to geometric changes, most traditional and many learned descriptors rely on an explicit scale and rotation (and sometimes affine) normalization. (c) shows scale and orientation detections of BRISK.

trained using metric learning, ensuring that image patches showing the same point seen from different viewpoints result in similar descriptors [82, 101, 123, 141, 185, 210, 219].

Integrated detection and description Most features that are known in the community by their acronyms (such as SIFT [124], ORB [170] or BRISK [117]) usually imply both a detection and description method, and these methods are usually evaluated jointly (even though one could often easily combine the detector from one method with the descriptor from another). Furthermore, when using learned components, it is common to train both detection and description jointly, often using parts of the overall CNN architecture for both tasks [52, 56, 155, 216], see Figure 1.6. A recent comparison of traditional and learned features was provided in [181]. When comparing CNN-based methods to traditional methods, their results do not yet suggest an absolute superiority of CNN-based methods.

Feature detection and description is a very active area of research, with many new publications having appeared since this evaluation, and since the publication of the results presented in this thesis. Recently, D2-Net [56] and D2D [196] more closely consider the relation between detector and descriptor. They exploit the fact that with modern machine learning hardware, densely extracting descriptors for every pixel is not as prohibitive as it used to be when the detect-then-describe paradigm emerged. Thus, they densely extract descriptors and derive feature locations using saliency in the descriptor map. Most recently, [17, 205] have proposed integrated feature detectors

and descriptors, which, similarly to our contributions in Papers D and E, and [202], use supervision schemes based on reinforcing the true objective of matching as many points as possible. Unlike Paper D and [202], they also learn the descriptor.

Unlike the contributions presented in this paper, few works explicitly care about minimizing the representation size needed for localization. On the contrary, the community moves towards exploiting as much data as possible to get as detailed scene reconstructions as possible. The number of features extracted for evaluation by recent approaches such as [56, 196, 205] is typically between $2K$ and $20K$. Furthermore, direct and dense methods, which attempt to match as many pixels as possible are recently gaining in popularity [189, 203], exploiting the high computational power provided by recent machine learning hardware.

1.3.3 Multi-Agent SLAM

Centralized Multi-Agent SLAM The conceptually simplest way to implement centralized multi-agent SLAM would be by streaming the raw sensor data from each participating agent to a central entity that would execute an extended version of a single-agent SLAM algorithm. Doing this, however, would require an unnecessary amount of bandwidth, as for SLAM, most of the initial sensor data is discarded. Visual SLAM algorithms extract local image features that can be used for both visual odometry and loop closure detection, and discard the rest of the initial sensor data. Accordingly, [66] proposed a centralized multi-robot SLAM system where features are already extracted on the individual robots, as well as an initial guess of the individual robots' trajectories. The central entity would then only execute loop closure detection and map optimization. While this system builds a full map of the environment seen by all robots, the authors do not consider how the individual robots can make use of that map. This has more recently been considered by [177, 178], where the authors proposed to stream back parts of the map to individual robots if map data is already available at the current locations of the robots. The authors have shown that by doing this, the individual robots can immediately fuse their individual state estimates with the pre-existing map to immediately get better state estimates.

Decentralized Multi-Agent SLAM Prior to my thesis, few works have considered true decentralized visual SLAM. There has been a lot of work on decentralizing what would be considered the optimization component of SLAM. I will discuss these works in more detail in Section 1.3.4. Many of them establish loop closure constraints through direct observations of other agents. Doing this has several inconveniences: firstly, achieving relative pose estimation through direct observation traditionally requires an explicit visual design, such as the addition of markers to agents [75, 103]. In the age of machine learning, this could probably be solved without explicit markers, but

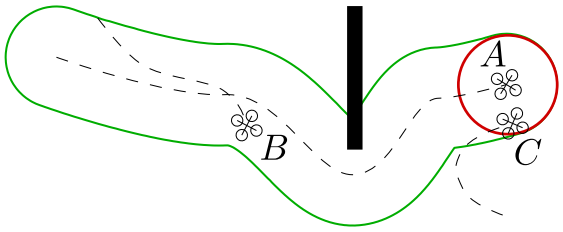


Figure 1.8 – When relying only on direct relative pose estimation, agent A can only establish a relative pose with agent C. In indirect relative pose estimation through the map, a relative pose can be established with any agent that observes the same scene as A has seen in the past, such as B, and with any agent that has observed the scene in which A is currently.

would still require careful training for every new agent with which relative poses are desired to be established. Secondly, as shown in Figure 1.8, establishing relative poses only through direct observations severely reduces the likelihood of establishing a relative pose, as two agents must be in the same place, at the same time. This would for example be extremely inconvenient in multi-robot exploration, as, without costly coordination mechanisms, it would cause the robots to do a lot of redundant work. In any case, using direct relative pose estimation, cannot really be considered full multi-agent simultaneous *localization* and mapping, as localization implies the ability to localize inside the map shared by the agents.

Some works exist which do decentralized loop closure detection using other modalities than vision. DDF-SAM [46] has been extended to a decentralized SLAM system in [47] by adding a data association system that matches planar landmark points. This system has been validated in the real world, but the landmarks consisted of manually placed poles that were detected by a laser range finder. [114] establishes loop closures by exchanging and aligning 2D laser scan edges. [32] has been extended to a decentralized SLAM system in [31] where data association is provided by identification and pose estimation of commonly observed objects. This approach relies on the existence of unique, pre-trained objects in the environment.

A first system that could be considered full decentralized visual SLAM has been proposed in [22], but the authors did not undertake efforts to minimize data exchange. In fact, that system simply relies on exchanging the full maps between robots. Furthermore, it is only evaluated on small trajectories and with only two robots, with images obtained from a simulated environment.

A decentralized visual SLAM system that came out after the system presented in this thesis is DOOR-SLAM [112]. The authors focus on resilience to outliers such as due to perceptual aliasing, incorporate an outlier rejection scheme [131] into decentralized visual SLAM, and validate their system in the real world. They do not focus as much on minimizing data exchange, and while a direct comparison to our work is not available,

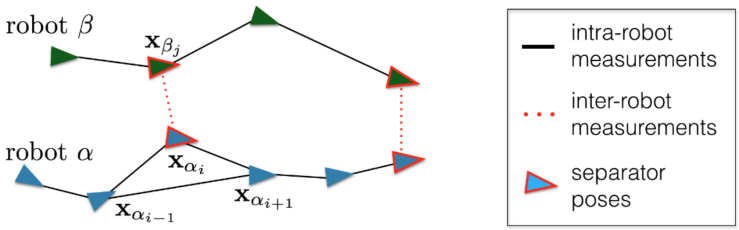


Figure 1.9 – Separators, in red, as visualized in [32]. They establish relative pose links between the maps of robot α (in blue) and robot β (green). In decentralized map optimization, only information pertaining to these separators needs to be exchanged.

it seems clear that their approach requires more data exchange than ours.

1.3.4 Decentralized Map Optimization

While few works have implemented full decentralized SLAM systems, many works have considered the specific problem on what would be considered the optimization component of SLAM, that is, exploiting information from other robots to improve each robots' state estimate. This work can broadly be divided into work which is only concerned about the current state estimate of the agents and work which tries to estimate the full trajectory of all robots, which is closer to SLAM.

Most of the works which only consider the current state estimate are based on bayesian filters, such as Kalman filters [168], information filters [194] or particle filters [29, 89]. These are however only of limited interest to decentralized SLAM, as SLAM systems not only need to establish the current pose of all agents, but also a map based on the full trajectory.

Works that aim to estimate the full trajectories are more often based on optimization algorithms [5, 46, 68, 106, 150]. The core principle behind data-efficient decentralized trajectory estimation is that agents do not exchange their full trajectory estimates with other robots. Instead, each robot optimizes its own trajectory with additional constraints that represent an interface to other robots' trajectories. In literature, these links to the other agents' maps are called *separators*, see Figure 1.9. It is then only information about these separators that are exchanged between the agents. In DDF-SAM [45, 46], an approach has been developed where robots use Gaussian elimination and exchange Gaussian marginals over the separators. While Gaussian elimination has become a popular approach, it has two major shortcomings. First, the marginals to be exchanged among the robots are dense, hence the communication cost is quadratic in the number of separators. This motivated the use of sparsification techniques [158]. Second, Gaussian elimination is performed on a linearized version of the problem, hence these approaches require good linearization points and complex bookkeeping to

ensure consistent linearization across the robots [45]. The need for a linearization point also characterizes gradient-based techniques [106]. An alternative approach to Gaussian elimination is the Gauss-Seidel approach of Choudhary et al. [32], which requires communication linear in the number of separators. The Distributed Gauss-Seidel approach only requires the latest estimates for poses involved in inter-robot measurement to be communicated and therefore does not require consistent linearization and complex bookkeeping. This approach also avoids information double counting since it does not communicate marginals. All aforementioned decentralized map optimization methods are synchronous, meaning that each optimization iteration involves all participating agents, and the next iteration can only be executed once all agents have exchanged data in the current iteration. Recently, an asynchronous decentralized optimization method has been proposed in [195]. This allows natural handling of communication delays, while performing similarly to synchronous methods.

1.3.5 Navigation in Unoptimized Maps

One of the contributions in my thesis is the questioning whether optimization is something that is really needed in SLAM. This is driven by the applications for which SLAM is used: Do they need a consistent map? The idea that this might just not be the case has been inspired by work that has shown that consistent maps are not needed for navigation: visual teach and repeat [23, 61, 71].

The naive way of imagining navigation in maps might be that there is a globally consistent 3D map where a goal is identified. The navigating agent would then know its initial pose and could then derive a collision-free 3D trajectory to the target from the high-fidelity 3D map. It would then track that trajectory in 3D space. This naive way has several disadvantages: most importantly, it is subject to visual odometry drift, both during map creation as well as during trajectory tracking. Another disadvantage is that it requires a very accurate volumetric map to plan a collision-free 3D trajectory.

To combat drift, the navigating agent in this naive scenario could constantly re-localize in the 3D map and update its state estimate, as well as the planned trajectory. The idea in teach-and-repeat, in contrast, is to suspend the need for a global state estimate in the navigating agent and to instead rely on constant re-localization. Rather than planning a collision-free trajectory in a detailed volumetric map, the trajectory is instead planned in the topological space of previous agent trajectories. Here, the planned trajectory is only partially constrained by the previous trajectories: robustness to viewpoint changes of the localization method naturally allows for deviation from old trajectories, and loop closures within or between the old trajectories allow for taking "shortcuts" [61], see Figure 1.10.

The most important insight for this thesis, however, is that the map(s) used for teach

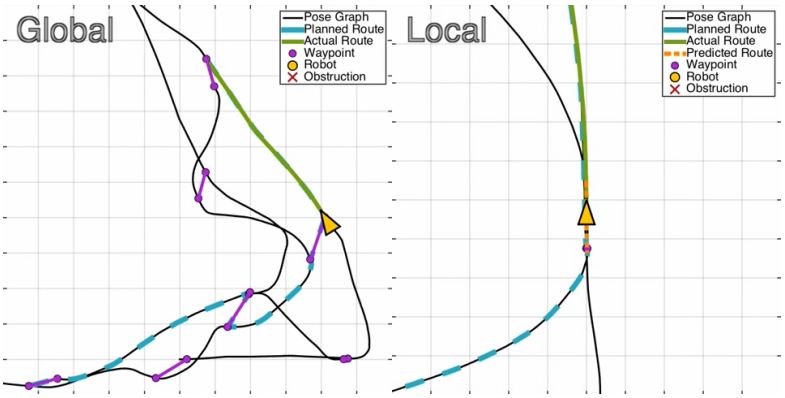


Figure 1.10 – Navigation in an unoptimized map using visual teach and repeat, as proposed in [61]. *Global*: given a goal, the robot can plan a route (blue, dashed) along previously traversed routes (black), using relative pose estimates as shortcuts (violet). Teach and repeat can be applied along previous routes, skipping across discontinuities as they are encountered (violet). *Local*: If a dynamic trajectory needs to be planned ahead, a local, consistent map can be assembled by using relative poses, including the loop closure relative pose. See https://www.youtube.com/watch?v=SDemQzTOtZM for the corresponding video.

and repeat do not need to be globally optimized. If a pose inconsistency is encountered within the planned path, the agent can simply first localize within the map up to that inconsistency, and then switch to localizing within the other side of the inconsistency, once the inconsistent edge of the trajectory has been traversed. This holds even when dynamics need to be considered in path planning, such as for quadrotors. In that case, a locally consistent trajectory can simply be assembled from relative poses along the teach-and-repeat path.

The question of how much computation needs to be done by a SLAM system can be highly application-dependent. Recently, [132] proposed a multi-robot exploration deployment with a group of tiny flying robots, which does not rely on a metric state estimate at all, but rather just optical flow for velocity control. Using reactive behaviors based on four punctual laser rangers for obstacle avoidance and radio frequency measurements for avoiding other robots, they demonstrate a simplistic search-and-rescue scenario where a group of six robots is tasked with discovering survivors in an unknown environment. Up to a group of ten robots, they show that the more robots they use, the more area is covered. However, their system is not able to fully cover the environment, as it does not have any notion of the explored volume. Furthermore, their exploration does not result in a map that can be subsequently navigated by other agents. In this thesis, I instead focus on approaches that result in navigable maps, and the exploration approach I propose in Paper F is capable of determining whether it has fully covered an unknown environment.

2 Contributions

This chapter summarizes the key contributions of the papers that are reprinted in the appendix. It further highlights the connections between the individual results and refers to related work and video contributions.

In total, this research has been published in 5 peer-reviewed conference publications and 1 journal publication (in the *Robotics Automation Letters (RA-L)*).

Additionally, these works led to several open-source software repositories:

- SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning
- Matching Features without Descriptors: Implicitly Matched Interest Points
- TensorFlow implementation of NetVLAD
- Data-Efficient Decentralized Visual SLAM

2.1 Scalable Decentralized Visual Place Recognition

As mentioned in Section 1.2.1, by virtue of every agent having to query every other agent for place recognition, the naive bandwidth complexity required for this in a group of n agents would be $O(n^2)$ overall, or $O(n)$ per query. In this part of my thesis, I address this problem of scalability. The solution was inspired by previous, unrelated work [33, 161], where I have used Distributed Hash Tables (DHTs). DHTs efficiently implement key-value lookup in a decentralized network [162, 169, 188, 221]. The key feature is that they implicitly distribute data and query load by using a deterministic hashing function that assigns keys to network peers, which become responsible for hosting the corresponding value. Thus, at query time, the relevant peer can be determined without first having to contact a central coordinator, while at the same time, the load of serving requests is balanced among peers based on the randomness of the hashing function.

In the following two publications, I have applied this principle to image retrieval-based place recognition by approximating image retrieval using key-value-like queries. In the first publication, this has been done for the traditional bag-of-visual-words

(BoW) place recognition approach, while the second publication does the same for machine-learning-based place descriptors, which have gained popularity during this thesis.

2.1.1 Paper A: Decentralized Bag-of-Words-Based Visual Place Recognition

(P1) T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 640–647. DOI: 10.1109/LRA.2017.2650153

To achieve efficient retrieval, Bag-of-Visual-Words (BoW) place recognition relies on an inverted index: for every word in the visual vocabulary (see Section 1.3.1), this index maintains a list of previously observed images that contain a feature associated with that word. During a query, every word that occurs in the query image is looked up in that index, and the resulting information can be used to assess which previously observed image is most similar to the query image in terms of the word distribution.

Our contribution is to leverage the efficiency of distributed hash tables for image retrieval by implementing the inverted index in a distributed manner, deterministically assigning words in the visual vocabulary to the participating agents, see Figure 2.1a. In our proposed system, the inverted index is queried by sending to each agent a partial query consisting only of the words that are assigned to that agent. To save bandwidth, the agents do not respond with the full inverted index data. Instead, they each provide a best guess based only on the occurrence of the visual words they are responsible for and send back that best guess along with a confidence score. The querying robot then derives the overall result from these guesses. While it can occasionally happen that the accumulation of best guesses based on partial data results in a wrong match (see Figure 2.1b), we experimentally show that in the majority of cases, it does not. Compared to the performance that can be achieved with a centralized BoW system, our method typically achieves 90% of the performance, at worst 80% of the performance while balancing the bandwidth- and computational load among the peers. These results hold for group sizes of up to 20 agents, and were evaluated on the KITTI [76] and Málaga [18] datasets, see Figure 2.2a. At the same time, the amount of data that is sent between agents is only slightly larger than in a centralized system. More importantly, the complexity of "data sent per query" with respect to agent count n is almost $O(1)$, while it would be $O(n)$ for a naively implemented decentralized system, see Figure 2.2b. This means that the proposed decentralized place recognition approach is far more scalable to large groups of agents. Unfortunately, due to communication overhead and the fact that every other agent returns a partial response, the complexity is technically still $O(n)$.

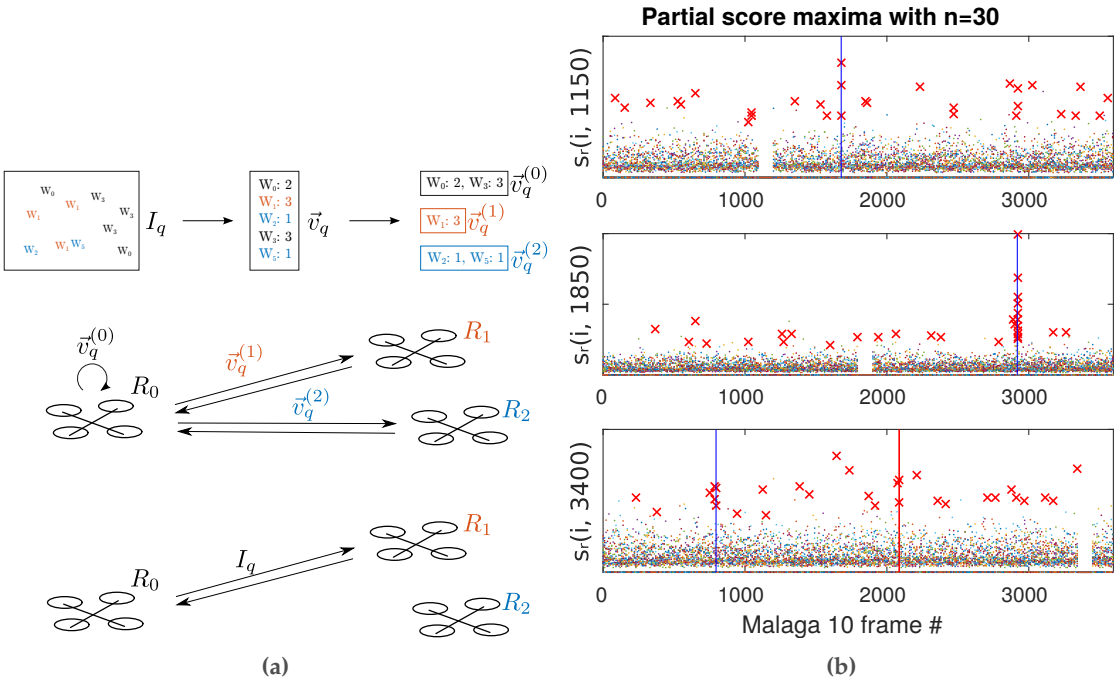
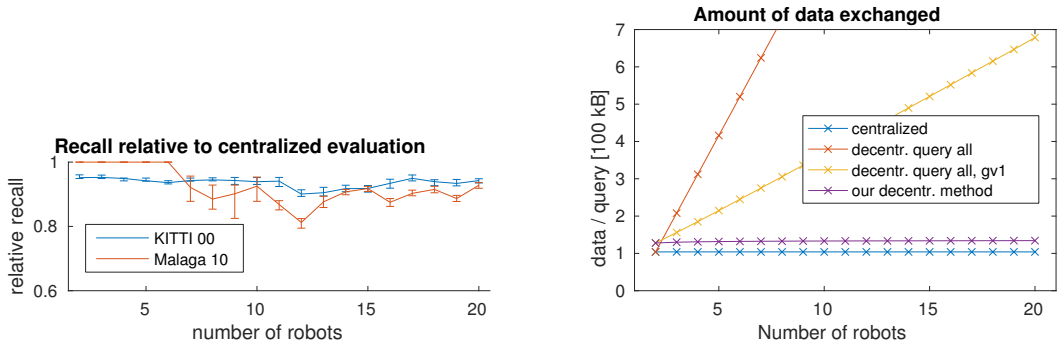


Figure 2.1 – (a) Simplified illustration of a query in our decentralized BoW algorithm. Top: Five visual words $\in \{W_0..W_5\}$ are encountered at different frequencies in the query image I_q, resulting in the BoW query vector $\vec{v}_q$. Each word is deterministically assigned to one of the three participating agents, $\vec{v}_q$ can accordingly be subdivided in three partial vectors $\vec{v}_q^{(0)}, \vec{v}_q^{(1)}, \vec{v}_q^{(2)}$. Middle: These vectors are sent as partial queries to the participating agents, and each of them responds with the best guess of the overall response according to the portion of the inverted index that it sees. Bottom: Based on the partial responses, the querying agent determines the most likely true inverted index match, and sends the relative pose estimation query to the corresponding agent. **(b)** Partial responses (red crosses – frame with highest partial BoW score s_r) and ground truth response (blue line) for three different queries in a system with 20 agents. The colored dots represent the partial BoW scores for every potential match, for the given query, the color indicating the agent at which that partial score is generated. The querying agent picks the frame for which the sum of partial BoW scores is highest. In most cases, this corresponds to the equivalent single-agent BoW solution, but in the bottom case shown here, the partial responses around the ground truth do not "agree" on a frame, resulting in a false positive around frame 2100.



(a) (b)

Figure 2.2 – (a) Recall relative to a centralized BoW on the evaluated datasets, for an agent/robot count up to 20. **(b)** Bandwidth used per query as a function of group size for centralized BoW and different versions of decentralized methods. The naive decentralized variants are visibly $O(n)$, our method is "almost" $O(1)$.

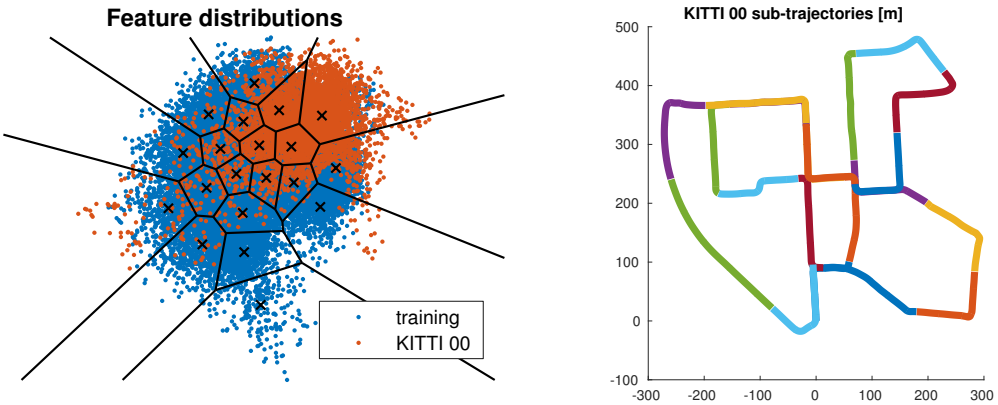
2.1.2 Paper B: Decentralized Embedding-Based Visual Place Recognition

(P2) T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition From Full-Image Descriptors". In: *IEEE Int. Symp. on Multi-Robot and Multi-Agent Sys.* (Dec. 2017), pp. 78–82. DOI: 10.1109/MRS.2017.8250934

In this work, we respond to the paradigm shift to place recognition based on neural-network produced embeddings [7], and use it as a basis for decentralized place recognition. With this approach, a place recognition query is composed of only a single D-dimensional vector (128 dimensions or similar). The most important improvement over the previous paper is that this now allows us to truly achieve a query-size-per-agent-count complexity of $O(1)$, as the system only needs to send a single query to a single agent, rather than multiple partial queries to all other agents.

To leverage the efficiency of distributed hash tables for nearest-neighbor search in high-dimensional space, we propose to transform it into a key-value-like lookup. This is achieved by clustering the output space of the neural network as illustrated in Figure 2.3a, and assigning each cluster to one robot. The proposed system can thus use any neural network to produce embeddings/descriptors of the observed images. At query time, each agent can then consult the pre-established clustering to determine which agent is assigned to the query embedding. The query is then only sent to that agent, which responds with the nearest neighbor among all previously observed images within the cluster.

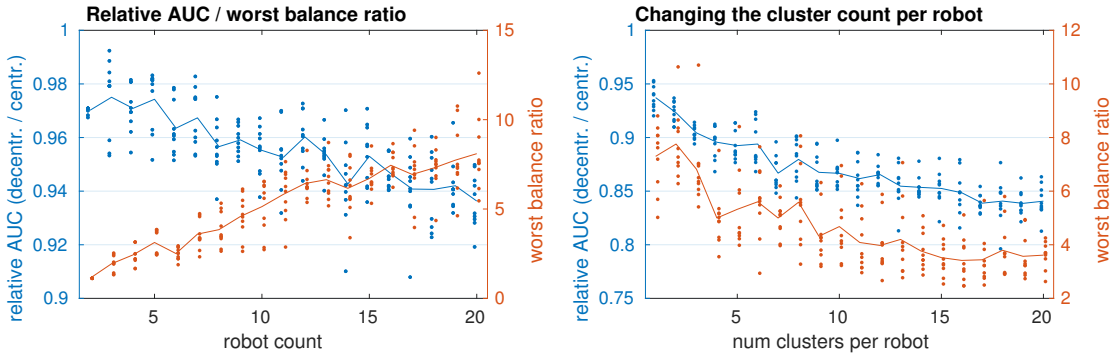
We investigate two shortcomings of our approach: firstly, a difference between the embedding distribution used for clustering and the distribution encountered during deployment can result in uneven load balancing as illustrated in Figure 2.3a. We analyze to which extent this can be addressed by training multiple clusters per agent,



(a) (b)

Figure 2.3 – (a) The clustering of the image descriptor space, illustrated on a projection of the space onto two dimensions. Note the difference between the training distribution and the distribution when evaluating on `KITTI 00`. This results in an uneven load balancing. **(b)** We simulate n agents by subdividing a single-agent dataset, here `KITTI 00`, into n parts. Simulated place recognition between agents occurs on portions of the trajectory that are observed multiple times in the single-agent recording.

and randomly assigning them. Secondly, this system does not return the correct nearest neighbor to a query if that nearest neighbor is in another cluster than the query. We evaluate our method by simulating n agents on KITTI [76] (see Figure 2.3b) and show that, as the agent count increases, the performance compared to a centralized system decreases slightly, remaining mostly above 93% up to 20 agents, see Figure 2.4a. Load balancing depreciates more strongly, but, as shown in Figure 2.4b can be traded off by sacrificing performance, using several clusters per robot. For example, using 20 clusters per robot instead of one reduces average performance from 94% to 84%, but reduces the load of the busiest agent by half.



(a) (b)

Figure 2.4 – (a) As the number of participating agents increases, the place recognition performance as compared to a single-agent or centralized system decreases and the load balancing becomes worse. The "worst balance ratio" indicates the factor by which the busiest agent is busier than it would be with perfectly even load distribution. Points indicate different trials with different (random) cluster distributions. **(b)** The load can be better distributed by assigning $k > 1$ clusters per agent (x-axis). However, this comes at the cost of further reduction in recall.

2.2 A Full Decentralized Visual SLAM System

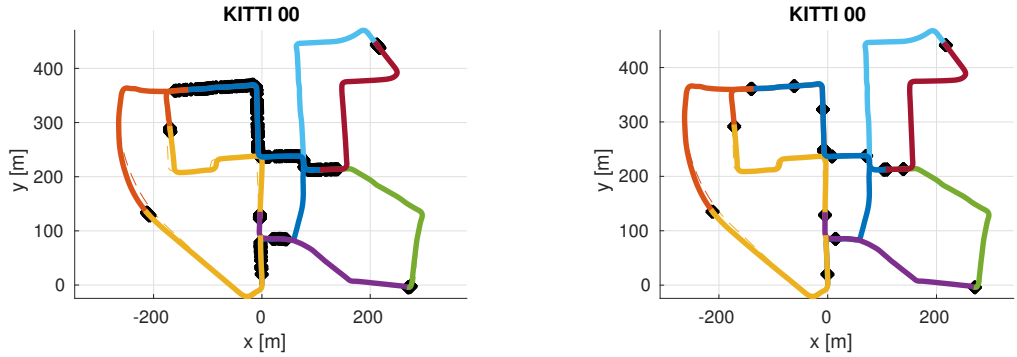
Having proposed scalable methods to do place recognition in a decentralized group of agents, the next step of this thesis has been to assemble a full decentralized visual SLAM system. By doing this, two goals were pursued: firstly, to establish a state-of-the-art baseline for data-efficient decentralized visual SLAM. And secondly, to identify the main bottlenecks in the full system, to guide the focus for the rest of the thesis.

2.2.1 Paper C: Data-Efficient Decentralized Visual SLAM

(P3) T. Cieslewski, S. Choudhary, and D. Scaramuzza. "Data-Efficient Decentralized Visual SLAM". in: *IEEE Int. Conf. Robot. Autom. (ICRA)* (2018). DOI: 10.1109/ICRA.2018.8461155

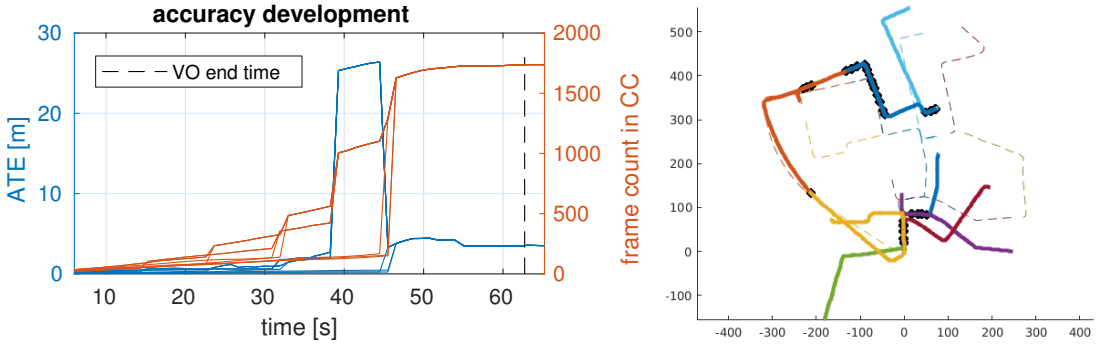
This publication was a collaborative effort with Siddharth Choudhary, the author of the state-of-the-art decentralized trajectory optimization method at the time [32]. It is in this paper that I first formalized the decentralized SLAM system composition described in Section 1.1.3 and shown in Figure 1.2. We used the visual odometry and map data structure from ORB-SLAM [148]. To transform ORB-SLAM into a decentralized SLAM system, we had to transform the following three components into decentrally executed components: firstly, we replaced the binary BoW implementation [73] used in ORB-SLAM with the CNN-based NetVLAD [7]. This allowed us to use the highly scalable embedding-based decentralized visual place recognition system from Paper B. Secondly, for relative pose estimation between maps of two agents, we mostly use the localization implementation of ORB-SLAM unaltered. The data that needs to be sent from one agent to the other here is the set of features extracted in the query image. To save bandwidth, we have evaluated using the method of [192], wherein feature descriptors are replaced with their associated visual word identifier. Finally, we have replaced ORB-SLAM's map optimization method with Siddharth's decentralized trajectory optimization [32]. Additionally, we have proposed a way to reduce data exchange in the overall system: assuming that the VO is locally consistent, we have proposed to skip relative pose estimations between the maps of two different robots unless a certain distance lies between them: If agent A has found a relative pose constraint to agent B's map, it will not attempt to establish another relative pose constraint with B until after it has traveled a certain "skip-distance". Figure 2.5 shows the difference between skipping relative pose estimations between two agents for $64m$ versus not skipping them at all.

We have thoroughly evaluated this system with a simulated group of 10 agents on the KITTI dataset. This has allowed us to characterize the accuracy of the resulting map (Figure 2.6), as well as the trade-off between data exchange and map accuracy, depending on parameter choices (see Appendix C for more details). The most important result, however, is the overall data exchange between the agents, shown in Figure 2.7. Without skipping relative pose estimations, the total amount of data exchanged in the simulated scenario is 10MB, dominated by decentralized optimization. In contrast, by



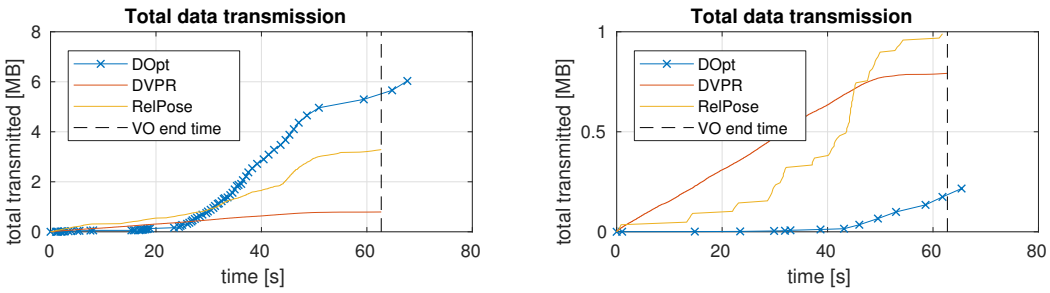
(a) Skipping $0m$. **(b)** Skipping $64m$

Figure 2.5 – When relying on a good VO, relative pose esimations between two agents can be skipped for a certain distance without significantly affecting the overall map accuracy. Instances of relative pose estimation are shown as black diamonds, the ground truth trajectory is dashed. The different colors indicate the simulated trajectories of the agents simulated from the single-agent dataset `KITTI 00`. Note that skipping is only done for pose estimations between the same two agents: The distance between pose estimations seen from a single agent can be shorter if each of them links to a different agent's map.



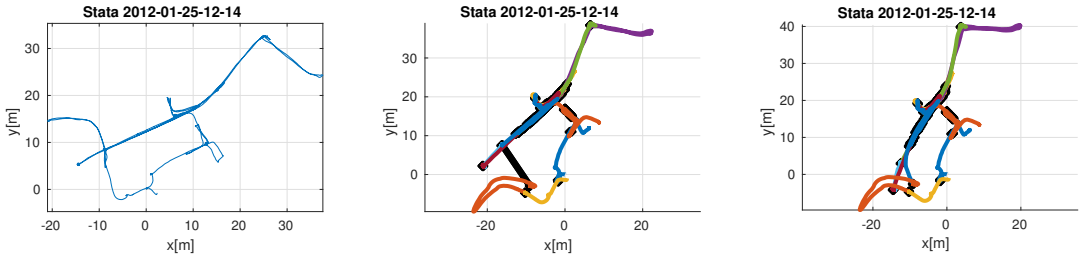
(a) **(b)** Estimate at time $40s$

Figure 2.6 – (a) Average Trajectory Error (ATE) and size of the connected components (CC), that is, the different disconnected parts of the map, over time. There is one line for each CC, by the end, all map data is merged into a single CC. A small error in a relative pose estimate between agent trajectories can result in a large ATE, as two components get connected at the wrong "angle", see **(b)** (same visualization as Figure 2.5 – note that some subtrajectories, like the green one, are not yet linked and are thus still plotted at the origin). This error can only be corrected by optimization once additional constraints between the individual agents' maps are added.



(a) Skipping $0m$ **(b)** Skipping $64m$

Figure 2.7 – Cumulative data exchanged by the data exchanging components of decentralized visual SLAM over time, when not skipping relative pose estimations versus skipping relative pose estimations for $64m$. The corresponding final estimates are shown in Figure 2.5. Note that in the latter case, only 2MB is sent in total!



(a) Ground truth **(b)** Before **(c)** After

Figure 2.8 – Decentralized optimization failure case on the MIT Stata dataset. Ground truth and estimates shown before and after performing the optimization that incorporates the final loop closure (black line). Instead of evenly deforming the affected subtrajectories, the brunt of the deformation is absorbed by the blue subtrajectory from which the loop closure to the red trajectory has been observed.

using a skip-distance of as much as $64m$, the overall bandwidth use is reduced to 2MB, now dominated by relative pose estimation, without loss in overall map accuracy.

Finally, we have found a failure case of decentralized trajectory optimization when evaluating on the MIT Stata dataset [62], see Figure 2.8.

2.3 Compact Representations for Relative Pose Estimation

The results from implementing a full decentralized visual SLAM system led me to the following conclusions when it comes to deciding the subsequent research path: first of all, the main consumers of bandwidth are relative pose estimation, and, depending on the choice of skip-distance, decentralized optimization. While with a large skip-distance, place recognition becomes a large contributor to data exchange, note that the data exchange of place recognition is not dependent upon the robot's trajectories (it needs

to be executed anyway), while the data exchange of both relative pose estimation and optimization are – the more overlap between robot trajectories, the more data will be exchanged. Considering that in the situation simulated in Paper C, most agents spend most of their time in unique locations not seen by any other robot, one could easily conceive that in many realistic multi-agent scenarios, such as multi-player augmented reality games or multi-robot exploration, there would be more overlap between the agents' trajectories. Besides, in that paper, I have already reduced the place recognition bandwidth tenfold with the contributions of Paper B. This leaves map optimization and relative pose estimation as the main bottlenecks to achieving more data-efficient decentralized visual SLAM.

Among those two, I decided to focus on relative pose estimation, as, based on my literature review, optimization seemed to be an already quite well-explored area of research. More importantly, however, based on some familiarity with teach-and-repeat, I began developing the notion that using map optimization to build a globally consistent 3D map might just not be that necessary in many applications where multi-agent visual SLAM would be used (see Section 1.3.5). This notion is further developed in Section 2.4.

For relative pose estimation then, according to the standard approach, the minimal data to be exchanged consists of a set of visual features, as those are needed for the 2D-3D matching used in Perspective-n-Point algorithms. One might envision a model-free machine-learning architecture where one network converts images and/or 3D scenes into some compact embedding that is later used by another network that combines two embeddings to return the relative pose between the original images. Intuitively, however, it is not clear to me that such an approach would be able to generalize to arbitrary environments, nor that it would be able to converge in the first place. Maybe as an indication, previous work where machine learning is used for pose estimation in a model-free manner is (by design) severely limited in that it cannot generalize its pose estimation ability beyond the dataset on which it is trained [102, 174]. Maybe this will one day prove to be better, but for now, I have decided to rely on the domain- and geometrical knowledge implied in using feature points.

The minimal amount of data that needs to be exchanged for relative pose estimation using features, using a standard approach, is:

$$d_{\mathrm{RPE}} = n \cdot (3B + d_{\mathrm{Desc.}}), \tag{2.1}$$

where n is the number of features used for relative pose estimation and d_{RPE} the number of bytes needed to represent a descriptor of the chosen feature. The feature pixel coordinates are also needed, and I assume that one would do that with 3 bytes, which allows expressing them with a resolution of 4096×4096 – presumably, using

two bytes for 256×256 would also work. In the system developed in Paper C, the amount n of ORB features was around 500. The size of the ORB descriptor is 32 bytes, resulting in $d_{\text{RPE}} = 17.5kB$.

When replacing descriptors with word indices as proposed by [192], this can be reduced to $2.5kB$: Per default, the used vocabulary has one million words, but it uses hierarchical clustering of 6 layers with 10 clusters each. In Paper C, we found the third-lowest layer of that hierarchical clustering is sufficient for good feature matching, meaning that two bytes suffice to identify among the $10'000$ visual word clusters at that level.

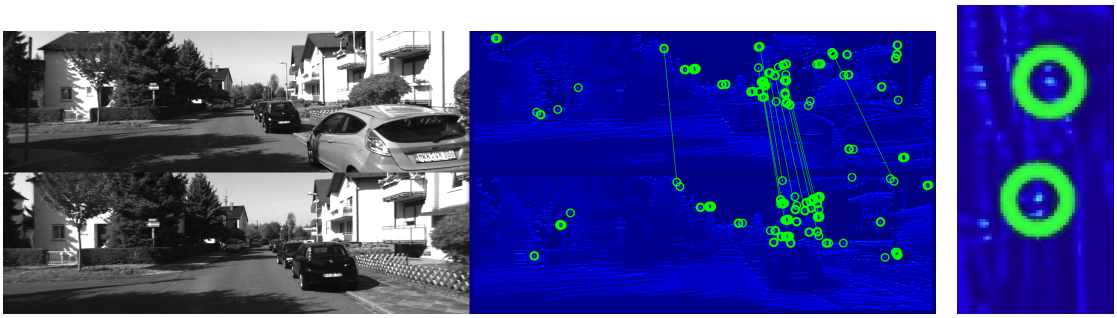
So, to further decrease the amount of data for relative pose estimation formulated in (2.1), one can reduce either n or $d_{\text{Desc.}}$. In Section 2.3.1 and Paper D, I have worked on reducing n. In Section 2.3.2 and Paper E, I have proposed a method by which sets of implicitly matched features are extracted, thus reducing $d_{\text{Desc.}}$ to zero.

2.3.1 Paper D: Detecting a Succinct Set of Features

(P4) T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. "SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning". In: *3D Vision (3DV)* (2019). DOI: 10.1109/3DV.2019.00072

Between being extracted and being a set of proper inliers that can be used for relative pose estimation, feature sets usually undergo a series of reductions in their amount. When building a map, only a subset of the initially extracted features can be properly tracked across several frames, and only a subset of those can be confidently triangulated. When attempting to localize, only a subset of the initially extracted features can be matched to features in the map, and only a subset of those will be correct matches, and thus for example result in RANSAC inliers. In relative pose estimation algorithms, one can typically choose, as a parameter, the number of inliers k below which a relative pose estimate is rejected as wrong, or too uncertain. The theoretical minimum using P3P is four (three to obtain two P3P hypotheses and one to determine the right one), but the presence of repetitive features and uncertainty in the feature locations suggest to base relative pose estimates on more inliers in practice. Without having considered representation size constraints, ORB-SLAM requires a minimum of $k = 20$. In our work, and with our datasets, we have empirically found that $k = 10$ represents a good trade-off between pose estimate quality and representation size requirements.

In order to send the minimum amount of features n, then, it would be ideal to only send inliers. This, of course, is not realistic – a very salient feature could be close to the edge on one image, but not be present on the other image due to an ever so slight change in viewpoint. Still, considering that $k = 10$ is enough inliers, it is easy to imagine that the $n \in [500, 2000]$ or more that one typically encounters in literature is far more than necessary.



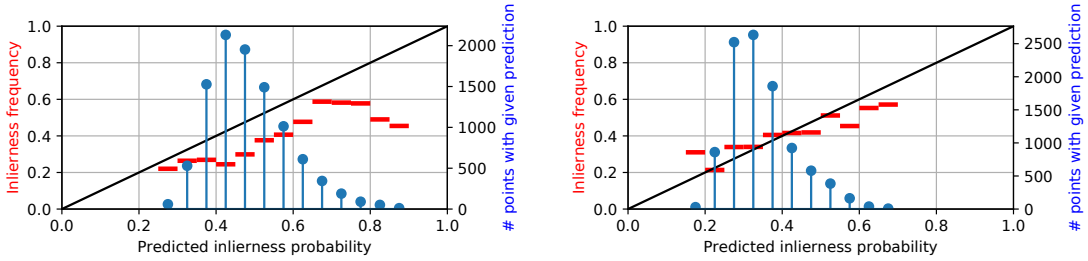
(a) (b)

Figure 2.9 – (a) Input and output of our proposed feature detection network for two images observing the same scene. Green circles indicate the selected interest points, their thickness indicates their score. **(b)** A close-up shows that our method results in very well-localized features – an explicit loss for this, as proposed in some of the related work, is not required.

If one would treat a map with triangulated visual features such as SIFT or ORB as given, one can use heuristics based on feature tracking and triangulation characteristics to remove feature points that are likely not going to result in inliers [28, 120, 157, 204] (in my prior affiliation, I have co-authored one such work, [57]). The problem with this, however, is that it cannot be used for localization with a given image instantly, but first needs to wait for information on these characteristics to accumulate. It has thus previously been proposed to learn classifiers to distinguish good from bad pre-extracted features [58, 86, 217]. In this work, we propose to simplify the system even further and to directly train a feature detector that only extracts features that will likely be inliers.

While one can come up with theoretical properties that determine whether a 2D feature will be "good" (see the literature review on traditional detectors in Section 1.3.2), there is always the danger that an image location looks good, but is not really, for example due to occlusions or reflections [51, 58]. A good avenue, then, is to additionally rely on learned heuristics.

Like most related works, we propose to train a pure convolutional neural network (CNN) which implicitly selects features by predicting a per-pixel score, see Figure 2.9. Feature points are selected from that score map using non-maxima suppression. One of our contributions is that we propose to train that network to predict, from the image patch implicit in a pixel's receptive field, the probability that that pixel will result in an inlier ("inlierness probability"). This is done by applying the feature selection derived from the network (bootstrapped with random weights) to relative pose estimation, and then directly using the fact whether features result in inliers or outliers for training.Note that this training task is strictly speaking ill-posed and has a recursive component: It is ill-posed because whether a point will be an inlier will depend on much more than just the image patch (such as the number of points extracted, in training we use $n = 500$). And it is recursive because if a "good" point does not have a high score, the



(a) KITTI **(b)** EuRoC

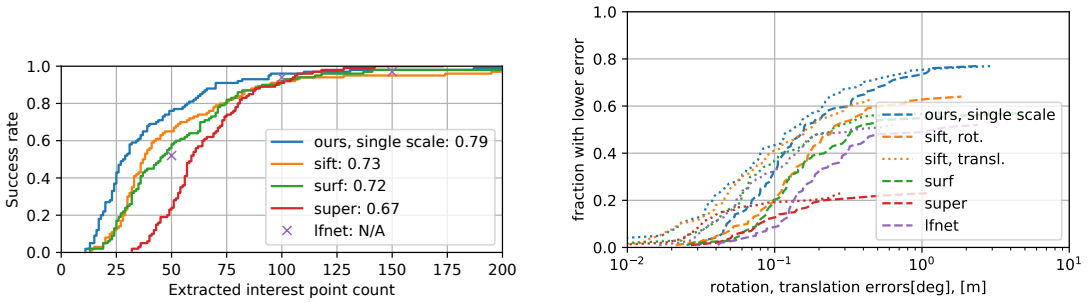
Figure 2.10 – On the two testing sets, we have binned the 50 interest points according to their interest point score, in our method the same as the predicted "inlierness probability". If this prediction were perfect, the fraction of interest points in the bin that are inliers would, at large numbers, be the same as the predicted score (black diagonal). The actual fraction of inliers per bin is shown with red steps – the prediction is quite accurate on EuRoC, less so on KITTI. The cardinality of the bins is shown with the blue stem plot.

training procedure will never pick up on its goodness, as it might never be selected to be a feature point. However, we have found the proposed method to work well, even making decent predictions of inlierness probability on the testing set, see Figure 2.10.

To measure progress towards our goal, we have proposed a new metric for feature detectors which we call k-succinctness. It measures the distribution of how many points need to be extracted to achieve k inliers. Previous metrics for feature detection have been repeatability (detection independent of matching) and matching score (detection + matching), but neither of them directly captures what we are looking for. Both of them are typically evaluated at a fixed number of feature points, and they would return a distribution of k as a result – we instead want to invert this to see how many points are needed (distribution) to get k inliers (fixed). Since trying to plot a distribution from samples is ill-posed, we instead look at the cumulative distribution, see Figure 2.11a. On the KITTI [76] and Euroc [24] datasets, we show that the proposed network performs particularly well in terms of k-succinctness when compared to other state-of-the-art feature detectors. Figure 2.11 shows the results for the `KITTI` dataset. On the evaluated image pairs taken from KITTI and EuRoC, a relative pose estimation success rate of 80% can be achieved with as little as $n = 50$, and $> 90\%$ with $n = 100$. Compared with a more standard $n = 500$, this results in a $10\times$ bandwidth savings – this comes at a reduced recall, but in Paper C, we have seen with the skip-distance that in SLAM, a high density of relative pose constraints is not crucial.

2.3.2 Paper E: Descriptorless, Implicitly Matched Features

(P5) T. Cieslewski, M. Bloesch, and D. Scaramuzza. "Matching Features without Descriptors: Implicitly Matched Interest Points". In: *British Mach. Vis. Conf. (BMVC)*. 2019



(a) **(b)**

Figure 2.11 – (a) 10-succinctness plot for the KITTI testing dataset: how many features need to be extracted to achieve 10 inliers. For any (low) amount of extracted interest points, our method has the highest success rate in having 10 inliers or more. **(b)** shows the cumulative distribution of rotation and translation errors in relative pose estimation when extracting 50 features. The distribution is only shown for image pairs with 10 or more inliers. Consequently, the highest fraction of pairs with lower error is the same as the success rate at $n = 50$ in (a).

Between two images, features are usually matched through their descriptors. In this work, we have proposed an alternative approach, thanks to which it is not necessary to extract and exchange descriptors anymore. This is achieved with a CNN with n (for example128) output channels, where each channel picks a single, "favourite" point from within the input image, see Figure 2.12. This CNN is trained in such a way that, ideally, for two different images of the same scene, each channel will pick corresponding points in both images. This corresponds to the idea that CNN channels can be interpreted as visual word responses [184, 197]. The authors of [197] have exploited this idea only for image retrieval, consider the maximum response in the image or a subregion, whereas my work is interested in the particular location of this maximum response. Concurrently to my work, [184] use this idea for deep spatial matching, but they extract and match several regions for each channel. This gives an implicit matching of features between both images – the points selected by the same channel in each image. As a consequence, for relative pose estimation queries, it is not necessary anymore to transmit feature descriptors, only the well-ordered set of point coordinates selected by the output channels.

Similarly to Paper D, we have trained this network using a loss which reinforces the score of pixels that result in inliers and suppresses scores of pixels resulting in outliers. Additionally, we have added a loss which ensures that different channels do not pick the same "favourite" points. Unlike in Paper D, these two losses were not sufficient to result in convergence. To address that, we added a third loss which reinforces the true point correspondences of the points selected by a channel, if it does not pick the same point in both images of a training pair.

We have verified that also using this feature matching approach, 10 is a sufficient inlier

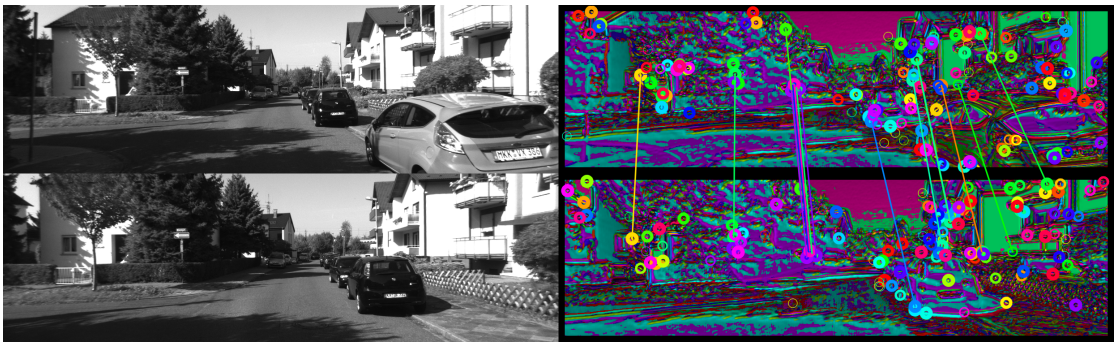


Figure 2.12 – Input and output of our implicitly matched feature detection network for the same image pair as in Figure 2.9. This is the output with 128 output channels, circles indicate points selected by the channels, hue stands for channel identity. Additionally, the hue of each pixel indicates which channel has the highest response at that pixel. Brightness indicates the strength of the response. Lines connect inlier matches that can be used for relative pose estimation between the two images.

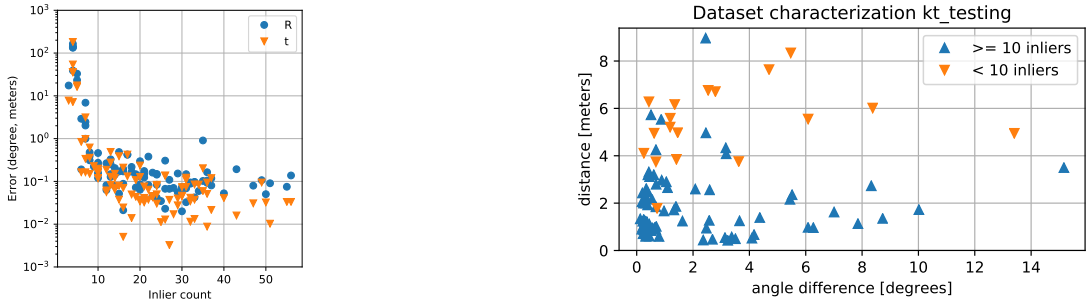
count to get good relative poses, see Figure 2.13a. On image pairs from the Euroc and KITTI datasets (see Figure 2.13b), relative pose estimation using our approach yields results comparable to conventional features, while requiring several orders of magnitude less representation size, as shown in Figure 2.14.

To summarize, starting from $17.5kB$ of representation size with ORB-SLAM, I have gotten down to 384 bytes with this work, or even 256 bytes, if a resolution of 256×256 is sufficient for relative pose estimation.

2.4 SLAM Without Optimization

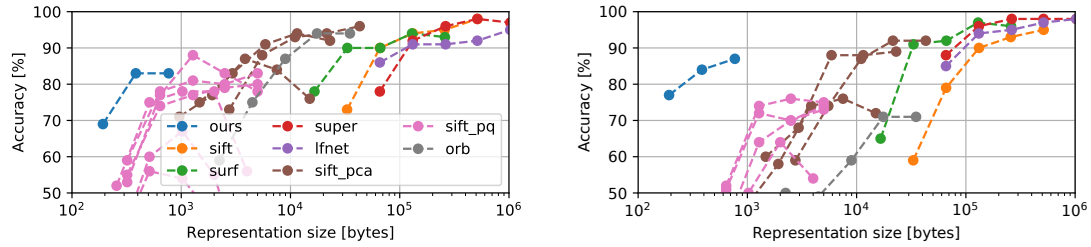
In the previous sections, I have established three data-exchanging components of decentralized visual SLAM and developed methods to minimize data exchange for two of them, place recognition and relative pose estimation. In this section, I consider the third component, decentralized map optimization. However, rather than attempting to beat the state-of-the-art in terms of data-efficient decentralized map optimization, I decided to take a step back and question whether map optimization is even a problem worth solving. With teach-and-repeat [61, 71], it has already been shown that consistent maps are not necessary for navigation, see the discussion in Section 1.3.5.

The goal of this part of the thesis has been to show that more applications of SLAM than just navigation do not need a globally consistent map. Certainly, we might not get around the need for as good a local accuracy as possible for some applications. For example, any augmented reality application where objects need to be rendered or accurately localized at a large distance that exceeds the immediately observable environment (for example, a shooter game) will require as good a long-range relative



(a) **(b)**

Figure 2.13 – (a) Empirical relationship between inlier count and pose estimation accuracy when using our method on our KITTI evaluation set. **(b)** Distance and angle differences of the poses for each image pair in our KITTI evaluation set, the same pairs were used in Paper D. This figure also indicates which pairs resulted in more than 10 inliers with our method.



(a) KITTI **(b)** EuRoC

Figure 2.14 – Accuracy versus representation size for different configurations of our method (64, 128, 256 output channels) and baselines. Note the logarithmic x-axis.

pose estimate as possible. While global consistency as achieved through optimization is no guarantee for a perfect long-range pose estimate, it certainly helps in reducing the error.

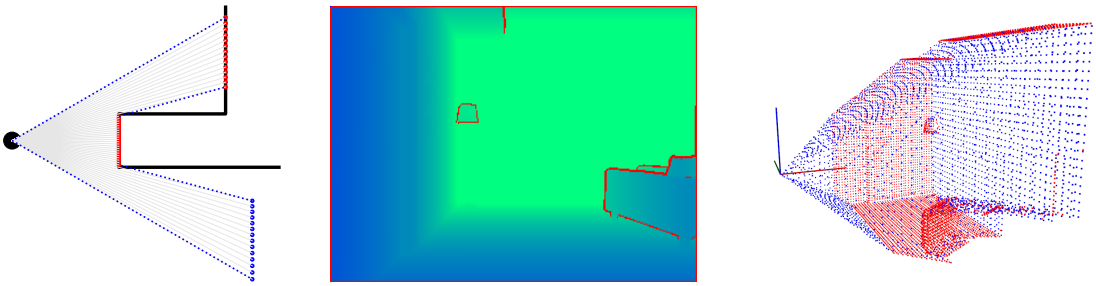
However, I doubt that decentralized visual SLAM would be used for such applications anyway. I believe that for most applications that require localization, we will have curated, GPS- or otherwise anchored maps that will be optimized in large data centers that are not subject to significant communication constraints. The main application for decentralized visual SLAM will be rapid-response scenarios in unknown environments, such as military or search-and-rescue missions, or the initial map data collection for the subsequent creation of curated maps. I envision an in-situ robot group deployment with limited bandwidth after which the robots return "home" and their data is collected to build a curated map offline. Decentralized SLAM, then, would form the backbone for spatial coordination between the robots in such deployments. For many of the aforementioned in-situ applications, exploration is essential. In particular, the key ability provided by exploration is to determine whether an unknown environment has been fully covered, and if not, how to move in the environment to find previously unseen places. State-of-the-art approaches to exploration assume a globally consistent and accurate map. In the final part of this thesis, I contribute a novel representation for exploration, which, in contrast, can deal with state estimate drift and does not require map optimization to work.

2.4.1 Paper F: Exploration Using an Unoptimized State Estimate

(P6) T. Cieslewski, A. Ziegler, and D. Scaramuzza. "Exploration Without Global Consistency Using Local Volume Consolidation". In: *Proc. Int. Symp. Robot. Research (ISRR)* (2019)

In exploration, the goal is to build a map of an unknown environment. To achieve this, an exploration algorithm needs a representation that tells it whether it has fully covered an environment, and if not, where in the environment there is potential for further exploration. Typically, this representation consists of a volume representing the known free space, which is bounded either by obstacles or by the interface to unknown space – so-called *frontiers*, see Figure 2.15. Then, the algorithm knows that exploration is not yet complete if there are still frontiers left in the map, and the location of them indicates where to move to do further exploration.

Most state-of-the-art exploration algorithms use map representations that either require a drift-free state estimate to function properly [154, 211, 212], or they rely on map optimization [140], as they are not designed to deal with a globally inconsistent map. In this work, we have presented a 2D representation for exploration that is robust to drift. An extension to 3D is still in preparation at the time of the writing of this thesis, having already been verified in simulation, see Figure 2.19. Conceptually, both the 2D and the 3D method work in the same way, so the description that follows holds for



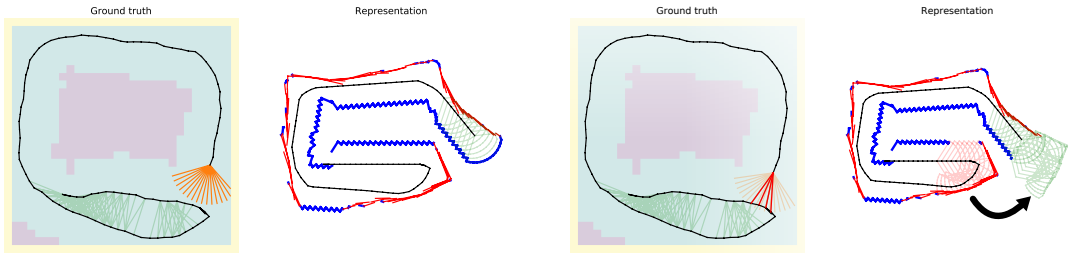
(a) (b) (c)

Figure 2.15 – Free volume, obstacles and frontiers, illustrated on a single depth measurement for both 2D and 3D. **(a)** 2D case: the black circle is a robot with a depth sensor. The rays of that depth sensor are shown in gray. The volume spanned by them is considered to be free. The rays either hit an obstacle, resulting in an obstacle surface bounding the free volume (red), or reach the maximum range, resulting in a frontier surface (blue). Additional frontier surfaces can be found on the sides of the field of view and at occlusions. **(b)** 3D case: a depth image with occlusions highlighted in red, the corresponding obstacle- and frontier-bounded volume is shown in **(c)**. The shown 3D representation does not represent the state of the art, it is rather already part of my planned contribution. The state of the art relies mostly on volumetric grids with voxels labeled as free space, obstacles, and frontiers.

both.

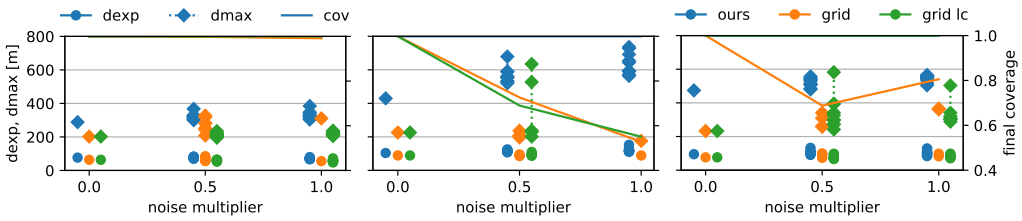
Rather than a global volumetric map, our representation for exploration uses local metric volumes which are derived from individual depth measurements as shown in Figure 2.15. These volumes are referenced to poses on the robot's trajectory, and this trajectory estimate does not need to be globally consistent.

In traditional, volumetric approaches, all depth measurements would be fused in a global, metric map (some previous approaches exist which use submaps [140, 179], but the individual submaps are still plagued by the same problems as a global map). The fusion of two depth measurements can be thought of as the overlapping of two volumes as shown in Figure 2.15: all frontiers of one volume that fall within the inside of the other volume are removed, as they no longer represent the interface to unknown space. The problem of this global fusion is that it results in errors if two depth measurements are incorrectly aligned, which can easily be caused by drift. In our approach, instead, each local volume individually tracks its frontiers. Then, rather than fusing volumetric information of all volumes to remove frontiers, any local volume only fuses with topologically nearby volumes, see Figure 2.16. As a consequence, if volume B is falsely estimated to be close to volume A, even though in reality it is far away, both A and B will not wrongly eliminate each other's frontiers. The approach relies on loop closure detection to topologically connect volumes that are actually nearby (even if the global estimate of them is far away). Rather than using the global estimate's position for frontier removal, this is done using a locally consistent assembly derived from local relative transformations.



(a) **(b)**

Figure 2.16 – Local volume fusion/consolidation in our approach: **(a)** Shortly before loop closure detection, the volume resulting from the current depth measurement (orange rays) is only consolidated with other recent volumes (green polygons on the right). **(b)** Once the loop closure is detected, volumes from the beginning of the trajectory (red polygons) become topologically close to the current depth measurement. For fusion, they are transformed into the local frame of the current measurement, according to the relative pose transformations on the shortest path from the current measurement.

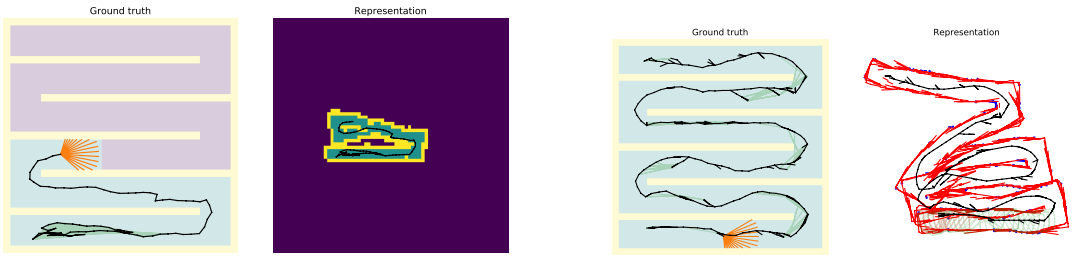


(a) Open **(b)** Maze **(c)** Forest House

Figure 2.17 – Expected distance until discovery d_{exp} (average distance traveled before a given voxel has been discovered) and full distance until the last voxel has been explored d_{max}, with different drift/noise intensity, on different maps (see Appendix F). Additionally, final coverage for the evaluated approaches is shown. While our approach takes longer to complete, it manages to fully cover the environment, unlike the traditional approaches(see Figure 2.18).

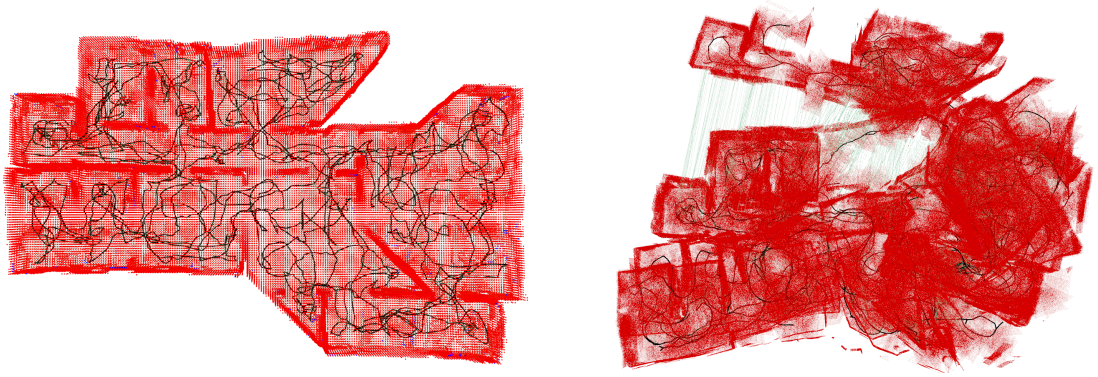
We demonstrate our representation with a frontier-based exploration approach, evaluate it under different conditions (including a real-world experiment) and compare it with a commonly-used grid-based representation. As can be seen in Figure 2.17 using the proposed representation allows full coverage of space even for very large drift in the state estimate, only at the cost of somewhat longer exploration time. More importantly, in 2D exploration, we found a situation where a traditional approach fails to completely explore the environment, while our approach succeeds, see Figure 2.18.

I am currently working on the publication of an extension of this approach to 3D, see Figure 2.19.



(a) **(b)**

Figure 2.18 – A failure case of grid-based exploration which does not affect our proposed representation. **(a)** Due to drift in the state estimate, the estimated trajectory folds upon itself and prevents the grid-based representation from exploring the top half of the map, as it thinks there are no frontiers left. **(b)** Since our approach does not fuse volumes without explicit topological proximity (due to odometry or place recognition), it is not affected by this failure case.



(a) **(b)**

Figure 2.19 – Visualizations showing the working planned extension to 3D. **(a)** The environment, fully explored with a traditional, grid-based approach. The black line shows the trajectory of the agent. A constant and increasingly complex map optimization effort was required to combat the drift of the state estimate when building this map. **(b)** The same environment, fully explored with our representation. It might not look user-friendly to humans, but, using visual teach-and-repeat (which is anyways the superior approach to navigation, see Section 1.3.5), it is just as easy to navigate for robots as the grid-based map. More importantly, no map optimization was needed to fully explore and map the environment using our representation.

2.5 Unrelated Contributions

In this section, I briefly describe additional publications I contributed to during my PhD, even if they did not directly contribute to this thesis.

[75] M. Gassner, T. Cieslewski, and D. Scaramuzza. "Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 5196–5202. DOI: 10.1109/ICRA.2017.7989609

[215] Y. Ye, T. Cieslewski, A. Loquercio, and D. Scaramuzza. "Place Recognition in Semi-Dense Maps: Geometric and Learning-Based Approaches". In: *British Mach. Vis. Conf. (BMVC)*. 2017, pp. 74.1–74.13. DOI: 10.5244/C.31.74

[37] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. "Rapid exploration with multi-rotors: A frontier selection method for high speed flight". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017, pp. 2135–2142. DOI: 10.1109/IROS.2017.8206030

[49] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2019. DOI: 10.1109/ICRA.2019.8793887

[143] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T. Au, and S. J. Kim. "Challenges and implemented technologies used in autonomous drone racing". In: *Springer: Intelligent Service Robotics* 12 (2019), pp. 137–148. DOI: 10.1007/s11370-018-00271-6. URL: https://link.springer.com/article/10.1007/s11370-018-00271-6

[65] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. "AlphaPilot: Autonomous Drone Racing". In: *Robotics: Science and Systems (RSS)*. July 2020. DOI: 10.15607/RSS.2020.XVI.081

[153] A. Oertel, T. Cieslewski, and D. Scaramuzza. "Augmenting Visual Place Recognition With Structural Cues". In: *IEEE Robot. Autom. Lett.* 5.4 (2020), pp. 5534–5541. DOI: 10.1109/LRA.2020.3009077

Team efforts [49, 65, 75, 143] can best be described as "team efforts", where I contributed to efforts mainly managed by other people. In [75], we implemented a pair of quadrotors that collaboratively carried an elongated object. We called it "collaboration without communication", because the robots collaborate without communicating through data channels. Instead, there is a master and a slave, and the slave "follows" the master in the plane using force inputs, and in altitude and yaw by tracking a QR code attached to the back of the master. This somewhat relates to the thesis in that it shows an example of collaboration without the need for decentralized SLAM. However, it is a very restricted and niche application – the robots would not be able to coordinate, neither before nor after grabbing the object. In [49], I helped with the creation of an aggressive flight dataset for the evaluation of Visual- Inertial- Odometry (VIO) algorithms. After the publication of the dataset, I hosted several competitions and supervised the development of an automatic evaluation website which is now accessible at http://fpv.ifi.uzh.ch. Finally, [65, 143] are publications that resulted from

our lab's participation in autonomous drone racing competitions, where I also took part in the efforts. We ranked second both times.

Structure-based place recognition [153, 215] are works developed in a side research project of mine, structure-based place recognition. This project started with my work at ETH Zurich before my PhD [40]. It was inspired by the observation that the 3D structure output by visual odometry, even if sparse, still looks recognizable – so why not use it for place recognition, rather than (just) images? In [215], we extended [40] by switching from a handcrafted to a learning-based approach, using 3D convolutional neural networks, outperforming [40], but receiving mixed results when it comes to the comparison between image-based and structure-based place recognition. Thus, in [153], we proposed a network that combines both visual and structural information, outperforming both of them and several state-of-the-art place recognition methods. Both papers were developed as master theses that I directly supervised.

Exploration Finally, [37] was another master thesis I supervised, by Elia, who later joined our group as a PhD and became a dear friend. We developed an exploration policy that considers the fact that changing direction at high-speed quadrotor flight is costly. We then showed that the policy that we proposed is significantly faster than previous state-of-the-art exploration policies. This project had some influence on my thesis, as it exposed me to the exploration literature, as well as shortcomings of traditional exploration representations. The work contained the exploration of several real-world scenarios, also using a state estimate that is subject to drift. While sufficient for the experiments in that work, this resulted in visible errors in the volumetric map and occasional failure cases.

3 Future Directions

In this thesis, I have developed the first truly data-efficient decentralized visual SLAM system. I have furthermore developed new methods for the loop closure- related, data exchanging part of decentralized visual SLAM, reducing data exchange by orders of magnitude. These new methods are not restricted in their viability to decentralized visual SLAM, but could also help make distributed localization systems more scalable. Finally, I have made the argument that map optimization is often not necessary for decentralized SLAM, and have added exploration to the list of SLAM applications that do not require global consistency in the map.

During this work, I have encountered several ideas for future work, both in extensions or alternative approaches to the proposed methods, as well as in further development of applications that make use of decentralized visual SLAM.

Better Implicitly Matched Features Since the Implicitly Matched Features that we have proposed in Paper E are a fundamentally new approach to feature extraction and matching, I believe that we have barely scratched the surface of what is possible with this approach. Besides warranting further exploration in training methodology, it would be interesting to better investigate invariances, such as robustness to viewpoint and lighting change. Furthermore, our approach has been to train a monolithic network from scratch. As a consequence, our network needs to learn not just the ability to pick good "favourite" points, but also everything else that is needed from a feature detector and descriptor, such as distinctiveness, repeatability, and several invariances. It could hence be interesting to instead compose the network of two parts, one for feature extraction and description, and one for point-picking. Then, the feature extraction and description part could be initialized with, or even fixed to, existing methods, such that the desired properties of this module do not become another training objective for the network, potentially distracting it from better solutions for the point picking task.

Revisiting and Benchmarking Decentralized Map Optimization Like several before me, I have made the argument that map optimization is not actually necessary for

many applications of SLAM. This is important to remember, as often, SLAM, when evaluated as a module, is assessed by its ability to achieve globally accurate maps – which does not necessarily best reflect its true utility in many applications. However, there still are applications where map optimization is useful. I have developed some intuitive ideas for how to do decentralized map optimization myself, and the decision to not delve into decentralized map optimization has not just been because I saw the limits in its utility for many applications, but it has also been in part strategic. Many approaches have already been proposed. One of the problems when reviewing them is that it is hard to assess which of them is the best, as each approach evaluates on its own dataset, and there is often even no common metric between the evaluations. So I believe that the map optimization literature would benefit from a proper benchmark of decentralized map optimization approaches. I believe that this would significantly improve the finding of and confidence in the right approach for a given application, and it could greatly facilitate further development.

Data-efficient Visual Teach and Repeat The contributions of this thesis have been focused on decentralized visual SLAM. At the same time, I have considered applications of it and identified navigation as an important application of a SLAM system. As elaborated in Section 1.3.5 and elsewhere, visual teach and repeat is a very powerful solution to navigation in visual maps. This is particularly important when combined with a decentralized visual SLAM system, to avoid additional data exchange due to optimization. The obvious next question is: what about data exchange needed for teach and repeat? In order to navigate from point A to point B – points potentially seen by two different, far-away agents – the agent group would first need to collaboratively establish a path between these points – ideally the shortest one. Then, the agent that plans to undertake that journey would need to gather the necessary map data from the other agents. To the best of my knowledge, a decentralized implementation of this is yet to be proposed, but it is also easy to envision methods to additionally reduce data exchange, such as methods to reduce the exchanged maps to a necessary minimum.

Data-efficient Multi-robot Exploration In this thesis, I have proposed a representation for exploration that allows it to be performed with unoptimized maps. However, this representation remains to be used in a multi-robot setting. It seems conceptually obvious that this method would work just as well with multiple exploring agents: much like SLAM itself, the only conceptual change is that loop closure detections can also happen between, not just within agent trajectories. The fusion of volume information, then, simply needs to happen across agent boundaries. The question, however, is how much data exchange this requires, and how it might compare to data exchange that would be used in alternative methods, potentially also taking optimization data exchange into account. This could involve further refining the proposed representation to make it more compressible. Besides consolidating volumes with other agents, agents

also need to coordinate where to move next: while each robot could potentially only navigate towards its own frontiers, it might be more efficient to instead move to frontiers discovered by other robots. Some work on task allocation in multi-robot exploration, based on global metric representations, already exists. It would be interesting to see how these methods would transfer to our topological representation.

A Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index

Reprinted, with permission, from:

T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 640–647. DOI: 10.1109/LRA.2017.2650153

Note: Figure A.2b has been modified by adding a red line that indicates the false positive in the third case.

Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index

Titus Cieslewski and Davide Scaramuzza

Abstract — State-of-the-art systems that do place recognition in a group of n robots either rely on a centralized solution, where each robot's map is sent to a central server, or a decentralized solution, where the map is either sent to all other robots, or robots within a communication range. Both approaches have their drawbacks: centralized systems rely on a central entity which handles all the computational load and cannot be deployed in large, remote areas; decentralized systems either exchange n times more data or preclude matches between robots that visit the same place at different times while never being close enough to communicate directly. We propose a novel decentralized approach which requires a similar amount of data exchange as a centralized system, without precluding any matches. The core idea is that the candidate selection in visual bag-of-words can be distributed by pre-assigning words of the vocabulary to different robots. The result of this candidate selection is then used to choose a single robot to which the full query is sent. We validate our approach on real data and discuss its merit in different network models. To the best of our knowledge, this is the first work to use a distributed inverted index in multi-robot place recognition.

A.1 Introduction

Many robotic applications can benefit from parallel deployment of multiple robots. In a search and rescue mission for example, the search area can be subdivided, so that each robot has less space to cover, resulting in quicker task completion. In order for the robots to efficiently collaborate, they need to know where they are with respect to each other and to the search area. In unstructured, GPS-denied environments, a robust way to achieve this is to have each robot create a map of the environment and then let it localize itself in the maps created by other robots. In such a scenario, if two robots A and B want to find out their relative poses, robot A can send its map data M_A to robot B (or vice versa). Robot B can then perform place recognition between M_A and M_B. When extending this problem to $n > 2$ robots, roboticists typically use one of the following approaches:

Centralized: Each robot sends its map to a central server, which performs all place recognition computations[48, 66, 72, 146, 164].

Decentralized, query all: Each robot sends its map to every other robot. Then, each robot can run place recognition between its own map and the maps of all other robots [93].

Decentralized, query in range: Same, but maps are only shared with robots in a certain communication range [45, 69, 142].

In this paper we show that, in terms of the amount of data exchanged as a function of robot count, current decentralized place recognition schemes are less scalable than their centralized counterparts. We then propose a novel decentralized scheme specifically designed for feature-based visual approaches, which scales similarly to a centralized method (see Figure A.1). This comes at the cost of only mildly affected recall. The proposed method is inspired by work on distributing visual queries in server clusters [99, 121]. The gist is that we distribute the task of candidate frame selection into n queries of size $\frac{C}{n}$, where C is the amount of data that would otherwise be sent to every other robot. We then send the full query for geometric verification only to the one robot that has observed the best candidate. The motivation for using a decentralized system instead of a centralized one is that the latter has a computational bottleneck at the central station, which should always stay operational. Furthermore, centralized systems cannot be deployed in remote, large areas because of limited communication range.

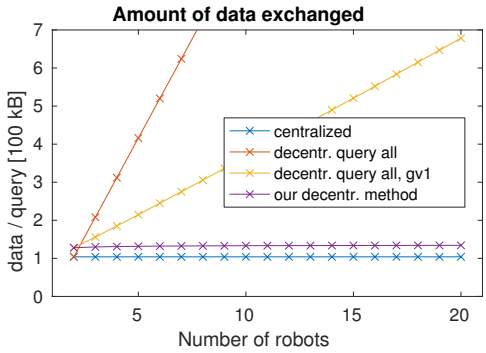


Figure A.1 – Our method significantly reduces the amount of data exchanged for place recognition in a decentralized multi-robot system. This is achieved by distributing the candidate selection of bag of words place recognition. *decentr. query all, gv1* stands for a query-all approach where geometric verification is performed only at one robot.

A.2 Related Work

The presented decentralized place recognition algorithm is based on bag-of-words visual place recognition. This method has been pioneered by [186], and we make use of further improvements as presented by [73, 151]. An essential part of our system is the use of an inverted index, a technique borrowed from document retrieval [9]. There, the inverted index is a key-value map where the keys are words and the value for each word is a list of documents containing it.

The fact that inverted indices can be distributed has already been discovered in document retrieval literature in the early 90s [98, 198]. Back then, the main focus was on distributing document retrieval to multiple processors on the same machine. With the advent of distributed hash tables [162, 188] and faster Internet connections in the early 2000s, however, researchers started to propose distributed document retrieval over peer-to-peer networks [133, 163], all pivoting around distributed inverted indices.

Only a small number of these can be effectively applied to visual place recognition, as there are significant differences between document and place retrieval. In visual place recognition, the query size is typically equal to the document size. Furthermore, each query is simultaneously inserted as a new document into the database. Lastly, while document retrieval is typically about obtaining all documents relevant to a query, we will contend with at most a single match per query in our scenario.

Some work exists in distributed image retrieval: Yan et al. [213] have developed distributed image search in camera sensor networks. Their query approach, however, is decentralized, query all. Aly et al. [3] proposed to distribute kd-tree-based image retrieval. However, the hierarchical nature of kd-trees requires that approach to pass through a central server. Ji et al. [99, 121] made the step of going from distributing images among servers to distributing the vocabulary, which is essential to effectively

reduce the amount of data exchange. Doing this work in the context of image queries in a local network, however, their focus is more on overall query speed than it is on the amount of transmitted data. Thus, they do not consider that it is not scalable, in terms of bandwidth, to return partial scores for all images — which we address in section A.4.3. Furthermore, we exploit geometric verification, which allows us to simplify the distributed query without affecting precision. Finally, while their approach distributes the workload, they use a central server that coordinates each query. In contrast, our approach is perfectly decentralized as soon as every robot knows the address of every other robot.

To the best of our knowledge, we are the first ones to apply a distributed inverted index in multi-robot place recognition.

A.3 Amount of data exchanged: scalability

Regarding place recognition systems, we will from here on use the term *add-querying* a place, meaning *to simultaneously add a place to the database and query it*. We assume that in a multi-robot system, places (in our case visual key-frames) observed by each robot are add-queried regularly. The expected total amount of data exchanged in the system due to place recognition within a unit of time is then:

$$d_{tot} = n \cdot \mathbf{E}[f] \cdot d_q, \tag{A.1}$$

where n is the number of robots in the system, $\mathbf{E}[f]$ the expected frequency at which each robot add-queries a place, and d_q the data that needs to be transmitted for each add-query. For now, if the same data is transmitted to two different robots, we count it twice, and we consider direct connections between robots. See Section A.7 for a discussion of multi-cast and multi-hop networks. We assume that both n and $\mathbf{E}[f]$ are factors of d_{tot} independent of the approaches considered hereafter, so we characterize their scalability solely by d_q.

In centralized place recognition, d_q is independent of n, or $\mathcal{O}(1)$, since each add-query is sent only to the central server. In decentralized, query-all place recognition, each add-query is sent to $n-1$ robots, and d_q assumes a complexity of $\mathcal{O}(n)$. In the *query in range* variant, the amount of robots to which an add-query is sent varies, but the complexity is $\mathcal{O}(n)$ in the worst case. It could be limited to $\mathcal{O}(1)$ by only sending queries to the closest k robots, but that would preclude localization with respect to robots outside of that clique. Limited-communication approaches that accumulate add-queries or propagate them through the network (such as [69]) reduce to a d_q complexity of $\mathcal{O}(n)$, since in the end each place gets shared with every robot. Thus, previous decentralized schemes, which have a d_q complexity of $\mathcal{O}(n)$ are inherently less scalable than the centralized scheme with its complexity of $\mathcal{O}(1)$.

In contrast, we propose a decentralized scheme whose d_q is close to the centralized case. This is achieved by splitting the add-query into two queries: first, a distributed query in which messages of $\mathcal{O}(\frac{1}{n})$ size are sent to every other robot [1] second, a targeted query where the full place information is sent to one robot only. The recipient of the targeted query is determined by the outcome of the distributed query.

A.4 Methodology

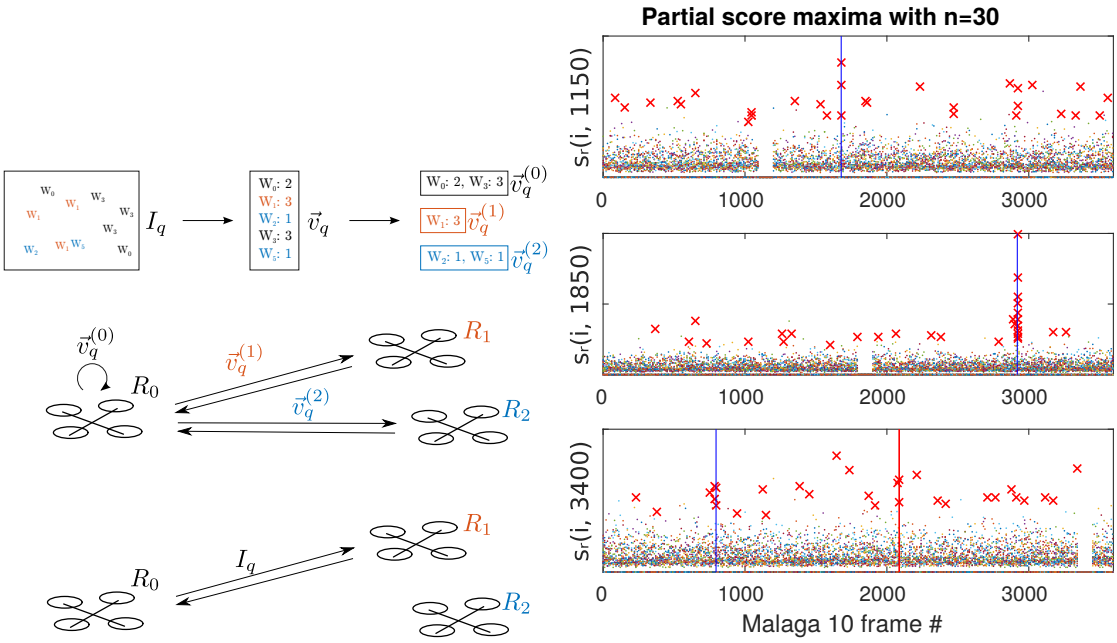
For simplicity, we consider place recognition algorithms which identify places with images. This reduces visual place recognition to matching of images. State-of-the-art visual place recognition is typically performed in two steps: 1) The full set of previously seen images $\{I_i\}$ is reduced to a set of candidates using visual appearance-based matching or other criteria. 2) These candidates are verified for geometrical consistency with the query image I_q. Only places passing this step are considered to be a match. A popular choice for the first step is comparing visually salient features, typically image patches around keypoints. Since image patches are high-dimensional and matching them is expensive, they are reduced to lower-dimensional features. This reduction can be achieved using PCA or variations thereof [21, 126], quantization of the high-dimensional space, e.g. using k-means [186], or more recently, convolutional neural networks [191].

We focus on approaches which quantize the feature space, since they are most amenable to distribution, as will become clear soon. When quantizing the feature space, each cell is identified by what is called a *word* W_w in a *vocabulary* $\{W\}$. A frame I can now be represented as a *bag-of-words* vector $\vec{v} \in \mathbb{R}^{|\{W\}|}$, where the w-th coefficient of $\vec{v}$ is the TF-IDF value of W_w in I. The TF-IDF (term frequency - inverse document frequency) value is the frequency of this word in I normalized by its overall frequency. It reflects that overall less frequent words carry more information when matching frames containing them [9]. Now, a given frame I_i can be matched to the query frame I_q using the score

$$s(i,q) = 1 - \frac{1}{2}\left|\frac{\vec{v}_i}{|\vec{v}_i|} - \frac{\vec{v}_q}{|\vec{v}_q|}\right|_1, \tag{A.2}$$

where $|\cdot|_1$ stands for the ℓ_1 norm. If $s(i,q)$ is close to 1, $\vec{v}_i$ and $\vec{v}_q$ are similar, in which case I_i is passed to the geometric verification step. If the vectors $\vec{v}$ are pre-normalized,

[1] Strictly speaking, the complexity is still $\mathcal{O}(n)$ due to overhead and responses from sending data to n other robots. However, since the amount of the bulk of the data is reduced by a factor of n, the resulting total amount of data sent per query is much closer to the centralized case, see Figure A.1.;



(a) (b)

Figure A.2 – (a) The essential steps of the proposed method. Illustrated for $n = 3$ and where the querying robot R_{r_q} is R_0. Top: The query frame I_q contains several features which are mapped to visual words W_i. I_q can thus be represented by a bag-of-words vector $\vec{v}_q$. Responsibility for visual words is distributed among the robots and $\vec{v}_q$ is subdivided accordingly. Middle: R_{r_q} sends to each robot R_r the portion of $\vec{v}_q$ which it is responsible for. Each robot R_r then returns a *partial response*. Bottom: From the partial responses, R_{r_q} derives which robot is the most likely to have seen a place matching I_q. Only to that robot it sends the full I_q to perform standard place recognition with geometric verification. **(b)** A close look at three instances of partial maxima-based candidate selection (A.7). For three query frames with a unique ground truth match (blue line), dots represent the partial scores $s_r(i, q)$ for all robots r (color) and all other frames i (x-axis) except the ones inside a radius of 50 key-frames (gaps). Red crosses represent the best partial candidate $(i, s_r(i, q))$ at each robot. In the first two cases, the true positive accumulates enough partial scores to outperform the outliers, which do not manage to outperform the true positive consistently across robots. In the third case, however, an outlier at around $i \sim 2000$ successfully accumulates two maximum partial scores, which is enough to outperform the weak true positive. Note how the true positive suffers from having neighboring frames which also have maximum scores at some robot. This suggests that clustering partial maxima that are close in time, or using a lower frame rate for place recognition could improve recall.

s can be rewritten as a sum for each w:

$$2 \cdot s(i,q) = 2 - |\vec{v}_i - \vec{v}_q|_1 = |\vec{v}_i|_1 + |\vec{v}_q|_1 - |\vec{v}_i - \vec{v}_q|_1 \\ = \sum_w v_{i,w} + v_{q,w} - |v_{i,w} - v_{q,w}| \doteq \sum_w s_w(i,q) \quad \text{(A.3)}$$

An important property of this is that a given summand s_w is zero if either $v_{i,w}$ or $v_{q,w}$ is zero. This allows fast candidate selection using inverted indices. We exploit the sum form of the score to distribute the candidate selection.

A.4.1 Distributing candidate selection

Regarding key-value lookup in a peer-to-peer network, distributed hash tables show that significantly less data can be exchanged by deterministically assigning keys to peers, and having these peers store the corresponding values. If peers that want to perform a query can locally compute which peer is responsible for storing the sought value, then there is no need to send the query to other peers. This cannot be directly applied to place recognition since, to the best of our knowledge, there is no deterministic mapping of observations of the same place to a unique string. In contrast, words of a pre-calculated vocabulary can be assigned to specific peers. We therefore distribute the evaluation of scores in (A.3) by assigning each word W_w to a robot $R_{h(w)}$ with $h(w) = w \mod n$. Each bag-of-words vector $\vec{v}_i$ can then be subdivided into n partial vectors $\vec{v}_i^{(r)}$, which only contain the coefficients $w : h(w) = r$ (see Figure A.2a). Note that this scheme does not support dynamic sets of robots natively. Adding robots can be handled by not distributing queries to them. For orderly departure of robots, however, we cannot conceive a method that would not transmit a lot of data. The special case of node failure is analyzed in a similar system by [99].

Now, each robot can calculate a partial score

$$s_r(i,q) = \sum_{w:h(w)=r} s_w(i,q) \quad \text{(A.4)}$$

for every potential match using only $\vec{v}_q^{(r)}$ and $\vec{v}_i^{(r)}$. Since in multi-robot place recognition the potential matches for the q-th query is the set of all previous queries, robot R_r only needs to receive $\vec{v}_q^{(r)}$ at the q-th query to calculate the partial scores $s_r(i,q)$ for all candidates. It can then add $\vec{v}_q^{(r)}$ to $\{\vec{v}_i^{(r)}\}$ for future queries. Thus, the first part of our place recognition query consists of sending $\vec{v}_q^{(r)}$ to each robot R_r (see Figure A.2a). For now, let's assume that robot R_r responds with a map $\{i \to (r_i, s_r(i,q))\}$. We address the scalability of these responses in section A.4.3. Having received $\{i \to (r_i, s_r(i,q))\}$

from each robot R_r, the querying robot can calculate the score for all candidate frames:

$$\{i \to (r_i, s(i,q) = \sum_r s_r(i,q))\} \tag{A.5}$$

A.4.2 Asking the right peer for geometric verification

State-of-the-art methods such as DBoW2 [73] or its implementation in ORB-SLAM [148] pass multiple candidates to geometric verification. To reduce data exchange, we send the geometric verification query to only one robot R_{i^*} (Figure A.2a), potentially sacrificing recall. Note that this one robot may still evaluate multiple candidate frames. We let it perform local candidate selection between I_q and all the frames that it has previously observed itself. R_{i^*} is chosen from (A.5) with:

$$i^* = \arg\max_i \sum_r s_r(i,q) \tag{A.6}$$

A.4.3 Scalability with the map size - minimal partial responses

As hinted at the end of Section A.4.1, having each peer respond with $\{i \to (r_i, s_r(i,q)) \forall i\}$ is not scalable, as the response size would grow linearly with the amount of frames observed by all robots. We have found a simple way to make the response size scalable: we make each robot return only the one frame with the highest partial score $s_r(i,q)$. While the underlying assumption

$$i^* = \arg\max_i \sum_r \begin{cases} s_r(i,q) & \text{if } i = \arg\max_{i'} s_r(i',q) \\ 0 & \text{otherwise} \end{cases} \tag{A.7}$$

might not hold for general vectors, we have found that it works well enough for visual bag-of-words vectors and the range of the robot count n that we have performed our experiments with. We identify some mechanisms that help with this: as has been previously shown [204], a small part of observed words contributes a big part towards the score. In the extreme case where only one word is responsible for the entire score, it is trivial show that (A.7) holds. In practice however, there are up to a couple dozen words that contribute to most of the score. For higher relevant word counts, another mechanism is in place: in order for an outlier to be considered the correct match by our evaluation (A.7), it needs to consistently have better partial scores $s_r(i,q)$ than the true positive. This is unlikely unless it observes a very similar scene. In contrast, if the outliers do not have consistently better partial scores, it should be sufficient for the true positive to have the best partial score at only two robots — if all of the top contributing words contribute similarly, the true positive will be the maximum in (A.7). Finally, for matches with a larger amount of contributing words, it is more likely that

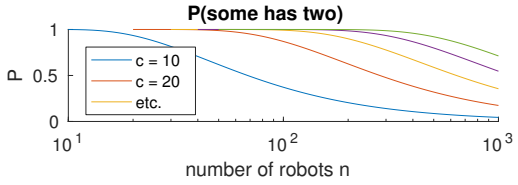
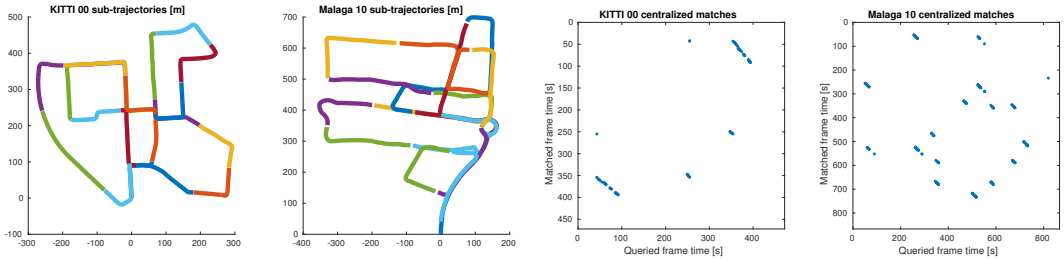


Figure A.3 – Probability that out of n robots, at least one is assigned at least two out of c words. Being responsible for multiple words increases the odds of contributing to a successful distributed candidate selection even if only the single best partial score is returned.



(a) KITTI 00 **(b)** Málaga 10 **(c)** KITTI 00 **(d)** Málaga 10

Figure A.4 – (a), (b): Sub-trajectories assigned to different robot processes. **(c), (d):** Confusion matrices for centralized evaluation on the used datasets.

some robots will be responsible for several of them, which decreases the odds that an outlier outperforms the true positive at that robot while at the same time increasing the weight of the partial score by that robot. For instance, the probability that there is at least one of n robots responsible for two of the c contributing words is 1 minus the probability that no robot is assigned two or more words, or $1 - \prod_{j=1}^{c} \frac{n+1-j}{n}$ (see Figure A.3). Evidently for as little as twenty contributing words, groups of up to 100 robots will almost certainly have at least one robot that is responsible for two words. A detailed study of mathematically derived guarantees for these mechanisms is outside of the scope of this paper. Instead, we provide a detailed case analysis for $n = 30$ in Figure A.2b.

A.5 Experiments

In order to evaluate our system, we use the data from KITTI 00 [76] and Málaga 10 [18] as if it were recorded by a group of robots: we first split each dataset into 20 non-overlapping sequential parts (Figure A.4), run the visual odometry (VO) thread of ORB-SLAM [148] on each part, and save each resulting map together with all timestamps that capture at what time what portion of the map has been created. We then load all maps and run the ORB-SLAM version of DBoW2 [73] between all possible pairs of maps, after having performed bundle adjustment. We save the resulting

matches as ground-truth reference, as we evaluate precision and recall relative to this centralized place recognition. See Figure A.4 for the resulting confusion matrices. We run 10 trials of our system for each of $n \in [2, 20]$, where the maps are chosen such that there is always at least one ground truth match. This avoids meaningless recall values for small n.

All experiments are performed on a single i7 machine and run in real time. This is enabled by pre-calculating the VO, bundle adjustment (required for good results in geometric verification) and all descriptor-to-word mappings. Each robot is implemented as an independent process and networking is done using ZeroMQ and Google Protocol Buffers. We use the pre-trained visual vocabulary that is provided with ORB-SLAM.

A.5.1 Data exchange evaluation

For every place recognition query we log both the amount of data exchanged for the distributed candidate selection query and responses d_c, and the amount of data exchanged for the geometrical verification query and response d_g. For each evaluated robot count n we average $(d_c + d_g)$ and report them as data exchanged per query by our method. We compare this to d_g, our model traffic for centralized place recognition and $(n-1) \cdot d_g$, our model traffic for classic decentralized place recognition. Since we observe that $d_g > d_c$, we also propose an alternative query-all approach where first only $\vec{v}_q$ is sent to all robots for candidate selection based on their individual maps. Similarly to our approach, the querying robot then only selects the one robot with the best candidate and sends to it the full information required for performing geometric verification. We model the data exchange for this with $(n-1) \cdot d_c + d_g$.

A.5.2 Accuracy evaluation

In order to evaluate the impact of our approach adjusted for the accuracy of the underlying (centralized) place recognition algorithm, we evaluate precision and recall relative to the matches of that centralized algorithm. Absolute precision and recall for the centralized algorithm are reported in [73]. Relative precision and recall are then calculated using the binary classification values described in Table A.1. Since the reference place recognition can match backwards in time, we expand the matches returned by our method by all their inverse matches; that is, if our method matches I_A to I_B, a match from I_B to I_A is added to the set of matches for evaluation. Furthermore, matches where the query or match frame is less than $2s$ off a corresponding ground truth match are also considered true positives.

Since both the centralized and the decentralized method use the same geometric verification method, one can expect that relative precision is 1. We have verified that this is indeed the case, therefore we only report relative recall. Very occasionally there

	tp	fp	fn
If $q(I_q) \neq \emptyset$ and $q(I_q) \subset q'(I_q)$	1	0	0
If $q(I_q) \cap q'(I_q) \neq \emptyset$ yet $q(I_q) \not\subset q'(I_q)$	1	1	0
If $q(I_q) \neq \emptyset$ and $q(I_q) \cap q'(I_q) = \emptyset$	0	1	0
If $q'(I_q) \neq q(I_q) = \emptyset$	0	0	1
If $q'(I_q) = q(I_q) = \emptyset$	0	0	0

Table A.1 – Relative true positive, false positive and false negative evaluation for each query frame I_q. Here, $q(I_q)$ is the set of matched frames returned by our algorithm, while $q'(I_q)$ is the set of matched frames returned by the centralized version.

are single false positives among ~ 100 true positives.

A.5.3 Computation time and memory evaluation

Finally, we evaluate the computational cost of our approach by measuring the CPU times for five subroutines: t_{lp}, the local processing time without processing related to the distributed candidate selection. This contains mostly addition of the query frame to the local database and reference score calculation, a DBoW trick to adaptively select a threshold for candidate rejection [73]. We use this reference score only for the full query that is sent to the robot selected by the distributed candidate selection query. The local processing time dedicated to the distributed candidate selection is measured separately as t_{ld} and contains mostly splitting up of the bag-of-words vector $\vec{v}_q$ and the evaluation of the partial scores (A.4) of self-assigned words. The handling of distributed candidate selection queries from other robots is measured as t_{rd}. Finally, full query handling is measured with candidate selection time t_{rcs} and geometric verification time t_{rgv}.

We are able to reliably measure these times with up to 16 robot processes, beyond which we encounter anomalies which we attribute to context switching, file system waits and other low-level causes. Such anomalies also occur spuriously with fewer robot processes, which is why we use the median and quartiles to accumulate these measurements.

With these measurements, we can estimate the total computation cost per query with $t_{lp} + t_{ld} + (n-1)t_{rd} + t_{rcs} + t_{rgv}$. We compare this with a decentralized, query-all approach whose time is modeled with $t_{lp} + (n-1)(t_{rcs} + t_{rgv})$, as well as a decentralized, query-all approach with only one geometric verification modeled with $t_{lp} + (n-1)t_{rcs} + t_{rgv}$. We do not compare with a centralized approach, since there, all the computational and memory load is focused on a single machine.

In terms of memory requirements, our approach should be somewhat more expensive than the query-all approach. Since for the geometrical verification query we maintain the same data structures as the query-all approach, our memory requirements are

Distributed query payload size = 2000·	
Word index and score	$4 + 4$
Geometric verification query payload size = 2000·	
Short integer keypoint coordinates	$2 \cdot 2$
Single precisiton keypoint uncertainty	4
ORB descriptor	32
Single precistion landmark 3d positions	$3 \cdot 4$

Table A.2 – Payloads, in bytes, of the distributed candidate selection queries (corresponds to d_c) and the targeted geometric verification query (corresponds to d_g). The factor 2000 stems from ORB-SLAM, which tracks 2000 keypoints at every frame.

strictly larger by the data structure necessary for distributed candidate selection. The accumulation of this data should be exactly the same as the inverted index data that is used for local candidate selection at each robot, except that it is re-arranged between the robots, and each frame reference also contains a reference to the robot that has seen it. In particular, if all robots query the same amount of frames, the size of the data structure should not depend on the robot count n. We verify this using a profiling tool.

A.6 Results

The amount of data exchanged per query is reported in Figure A.1. Unsurprisingly, our approach requires less bandwidth than both decentralized, query all approaches starting with as little as 3 robots. Furthermore we see that $d_c < d_g$. A detailed listing of the data transmitted at each query, minus overhead and responses, is provided in Table A.2. We are using the geometric verification method from ORB-SLAM [148], which requires both the original ORB descriptors of keypoints for accurate keypoint matching in presence of ambiguities as well as the 3D positions of corresponding landmarks for scale drift-aware 3D-3D RANSAC. Thus, d_g could be reduced by omitting the landmark 3D positions and using 2D-3D RANSAC. We speculate that d_g could be further reduced by not sending ORB descriptors, but only node IDs at a certain level of the vocabulary tree, and using a RANSAC that can take 1-to-n matches into account. Finally, as annunciated at the end of section A.3, both d_c and d_g can be reduced by only sending a subset of most relevant keypoints. Neither of these reductions would change the fact that our approach is more scalable than classical decentralized place recognition. Moreover, since $d_c < d_g$ should be the case even if d_g is strongly reduced, our approach is always worthwhile with more than three robots — at least considering the amount of data that needs to be exchanged.

From an accuracy perspective, Figure A.5 shows that for groups of up to 20 robots, recall is only mildly affected, being typically above 0.9, and at worst 0.8 times as high as with centralized place recognition. Interestingly, the biggest amount of robots does

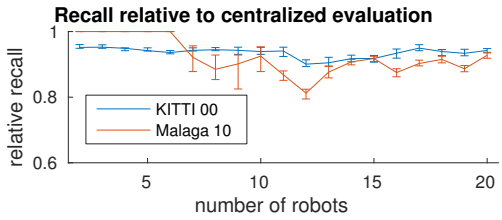
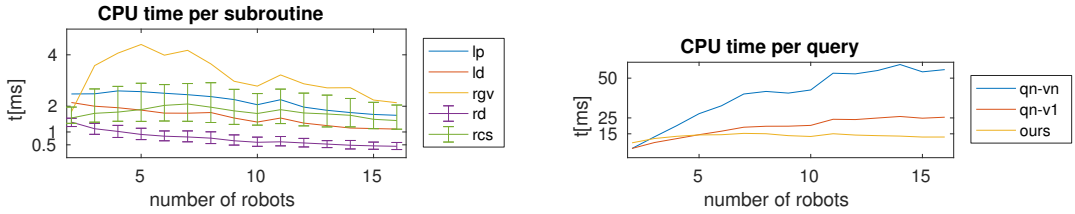


Figure A.5 – Recall relative to centralized place recognition. Mean, minimum and maximum are reported for 10 trials each.



(a) **(b)**

Figure A.6 – (a) Subroutine durations as a function of robot count. *lp*: local processing, *ld*: local DII query, *rgv*: remote geometric verification, *rd*: remote DII query, *rcs*: remote candidate selection. **(b)** Estimation of per-query total duration based on subroutine durations. *qn-vn*: query all with geometric verification at each robot, *qn-v1*: query all with geometric verification at only best robot.

not result in the worst recall. We have found that the groups of robots that exhibit lower recall happen to traverse portions of the map (Figure A.4) that are hard to match — note that adding a robot to the system also adds a new sub-trajectory, which has new potential matches with respect to the rest of the map. As we add more robots with trajectories that are easy to match to a group with low recall, the overall true positives increase faster than the false negatives, and recall increases.

The computation times per subroutine, as evaluated on KITTI 00, are reported in Figure A.6a. The local processing time t_{lp} is relatively large, and profiling with `valgrind` reveals that most of it can be attributed to the calculation of the reference score, which in our case involves intricate data structure traversal. Both the local and remote distributed candidate selection decrease with the amount of robots. This is on one hand well explained by the fact that the size of the partial vectors $\vec{v}_q^{(r)}$ which are processed per robot decreases as the robot count increases. On the other hand, it might seem surprising, since at the same time, the total amount of frames grows with the amount of robots. This, however, just indicates the efficiency of the inverted index approach. The candidate selection from a full query at a single robot's map t_{rcs} fluctuates, which we attribute to heuristics applied in DBoW. In fact, profiling in one instance reveals that ℓ_1 score evaluation takes around 40% of the time. However, heuristics are applied for candidate filtering based on common word count before score

evaluation. Similar heuristics are applied for geometric verification time t_{rgv}, which exhibits even stronger fluctuations. The estimated overall execution time is reported in Figure A.6b. The sampled memory footprint of the distributed inverted index per robot in the KITTI 00 dataset fluctuates between roughly 7 and 10 MB out of 100 MB overall memory use and is not correlated with the robot count. We explain these fluctuations with uneven word distribution and pre-allocation of `std::vector`, which we use as mapee in the inverted index.

A.7 Discussion: Scalability in real networks

So far, we have seen that our approach requires one order less data to be exchanged than previous decentralized visual place recognition approaches that also do not preclude matches. In this section, we re-consider all the data exchanged per unit of time (A.1) and discuss how the scalability with respect to the amount of exchanged data translates to scalability in real networks. Seeing the main application of our approach in mobile robotics, we only consider wireless data transmission. The following discussion is based on a comprehensive survey of wireless sensor networks [218]. Considering the large amount of available protocols and considerations listed in that survey, a complete discussion is well outside of the scope of this paper. Instead, we discuss two points that we consider most relevant to a prospective deployment of our system. To compare our approach to a decentralized query-all approach we only analyze the communication necessary for candidate selection. This is the only communication in our approach which is not strictly a subset of the communication in the former. Note that the communication needed for centralized place recognition is strictly a subset of all considered decentralized approaches, and so we will not consider it in this comparison. In the considered scenario, all robots make a query simultaneously. We define $d^{t\to r}$ the data transmitted from robot t to robot r, $b(d)$ the byte size of d, $t(d)$ the time required to transmit d over a direct connection and p_l the probability of packet loss in a direct connection. Subscript o denotes the use of our approach while subscript a denotes the use of a query-all approach. We use the following simplified model:

$$b(d^{t\to i}) = b(d^{t\to j}) \quad \forall t, i, j \in \{r\} \tag{A.8}$$

$$b(d_a^{t\to i}) = n \cdot b(d_o^{t\to i}) \doteq b^\star \tag{A.9}$$

$$t(d^{i\to j}) = B \cdot \frac{1}{1-p_l} b(d^{i\to j}), \tag{A.10}$$

where B is the bandwidth that is assumed equal for all connections and i and j in (A.10) are within communication range. (A.10) reflects that a packet might need to be re-transmitted once with a probability of p_l, twice with a probability of p_l^2, etc.

Channels, multi-casting and packet loss A given wireless channel can only have one active transmitter at a given time, while it may have multiple concurrent receivers.

Hardware designed for a given frequency typically only has access to a limited amount of channels. If the amount of robots in a group is smaller than the amount of available channels, each robot could receive on a separate channel and switch the transmitter channel depending on which other robot it would like to talk to. Assuming the robots are all in communication range and assuming that they could coordinate with negligible cost that only one robot talks to a given robot at the same time, each robot r would simply need to sequentially transmit $d^{r\to i}\ \forall i \neq r$ in parallel with the other robots and so the total time for candidate selection $t = (n-1)\cdot t(d^{t\to i})$. Applying (A.8)-(A.10), we obtain $t_a = n \cdot t_o$. Note that as long as enough channels can be provided, the duration of candidate selection in our approach is independent of the size of the robot group!

If only $c < n$ channels can be provided, several robots need to listen to the same channel. With that, the data thus additionally needs to be addressed, and not all robots can receive data at the same time. In the worst case, if $c = 1$, we can show that with the above transmission approach $t = (n-1)\cdot n \cdot t(d^{t\to i})$. $t_a = n \cdot t_o$ still holds, but the scalability with respect to the number of robots is now limited. Note that with the query-all approach we could exploit that the queries originating from one robot are all the same. Since all robots are listening to the same channel, the querying robot could thus address its data to all robots (multi-cast), and if $p_l = 0$ we can show that $t_a = t_o$, and our approach has no advantage over the query-all approach besides the use of less CPU time.

However, our approach still has merit in the presence of packet loss. Assuming that the packet loss dice are rolled for each receiver independently, (A.10) needs to be adapted for multi-cast. Now, a packet needs to be retransmitted with probability $(1-(1-p_l)^n)$, and thus

$$t(d^{i\to\{r\neq i\}}) = B \cdot \frac{1}{(1-p_l)^n} b(d^{i\to\{r\neq i\}}) \tag{A.11}$$

and our approach remains faster with $t_o = (1-p_l)^{n-1} t_a$. Reliability of multi-cast is a well-known issue in wireless networks [108].

Multi-hop networks We have previously assumed that all robots are within communication range of each other. However, n-to-n communication can be achieved with more relaxed constrains using multi-hop networks. To that end, data exchanged between two robots that are not directly connected needs to be routed according to the available topology [53] and the robots need to be controlled in a way that maintains overall connectivity [134]. Due to the sheer amount of possible topologies in a multi-hop network, we cannot derive and compare t_o and t_a within the scope of this paper. We have already shown that $t_o \leq t_a$ for the case where all robots can directly communicate. It is simple to show that for the remaining cases the amount of data that needs to be routed is smaller with our approach than with a query-all approach. If not all robots can communicate directly, there is at least one robot a such that from the remaining robots

one can form two non-empty robot sets B and C, such that all communication from a to C needs to go through at least one robot in B. This can be shown by contradiction: if no such robot a exists, this means that for every robot, no sets B and C fulfilling the above condition can be constructed. In other words, for every robot a' there exists no other robot c' such that the communication between a' and c' would need to be routed through at least one other robot b'. In that case, however, all robots can communicate directly, which contradicts the initial premise.

In our application, the robots in B are cumulatively responsible for forwarding $d^{a\to C} := d^{a\to j}\ \forall j \in C$. With the query-all approach, $d_a^{a\to C}$ is the full query originating in a, which needs to be sent to all robots, and thus $b_a(d^{a\to C}) = b^\star$. With our approach, $d_o^{a\to C}$ is only the portion of the $\frac{b^\star}{n}$-sized queries originating in a that are addressed to C, and thus $b_o(d^{a\to C}) = \frac{|C|}{n} b^\star$, which is strictly smaller than $b_a(d^{a\to C})$. We have thus shown that the amount of data that needs to be forwarded with our approach is less than with a query-all approach. We assume that in a multi-hop network, data forwarding can only increase the overall time needed to transmit the same data as in a fully connected work, and that this increase should correlate positively with a combination of $b(d^{a\to C})$ for different a, C. Since $b_o(d^{a\to C}) < b_a(d^{a\to C})\ \forall\ a, C$ we conclude that our approach should also be worthwhile in a multi-hop network.

A.8 Conclusion

In this paper, we have proposed a method that makes decentralized place recognition competitive compared to centralized place recognition in terms of the amount of data transmitted per robot count n, while only mildly affecting recall. This is opposed to previous decentralized place recognition approaches, which, while generally maintaining perfect relative recall, typically use $n-1$ times more bandwidth than centralized place recognition. Unlike centralized place recognition, our decentralized method evenly distributes the workload among the robots involved. At the same time, it is deployable anywhere as long as the robots can communicate with each other. We have validated our approach for groups of up to 20 robots on the public datasets KITTI and Málaga and discussed what it would mean to deploy it in wireless networks.

B Efficient Decentralized Visual Place Recognition From Full-Image Descriptors

Reprinted, with permission, from:

T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition From Full-Image Descriptors". In: *IEEE Int. Symp. on Multi-Robot and Multi-Agent Sys.* (Dec. 2017), pp. 78–82. DOI: 10.1109/MRS.2017.8250934

Efficient Decentralized Visual Place Recognition From Full-Image Descriptors

Titus Cieslewski and Davide Scaramuzza

Abstract — Visual multi-robot simultaneous localization and mapping (SLAM) is an effective way to provide state estimation to a group of robots that operate in an unstructured and GPS-denied environment. This is a problem that can be solved in a centralized way, but in some instances it can be desirable to solve it in a decentralized way. Decentralized visual place recognition, then, becomes a key component of a decentralized visual SLAM system. Achieving it by having all robots send queries to all other robots would use vast amounts of bandwidth, and diverse approaches have been explored by the robotics community to reduce that bandwidth. In previous work, we have proposed a decentralized version of bag-of-words place recognition, which, by using a distributed inverted index, is able to reduce bandwidth requirements by a factor of n, the robot count. In this short paper, we instead propose a decentralized visual place recognition method that is based on full-image descriptors. The method consists in clustering the full-image descriptor space into several clusters and assigning each cluster to one robot. As a result, place recognition can be achieved by sending each place query to only one robot. We evaluate the performance of our new method versus a centralized implementation using the Oxford Robotcar and KITTI datasets and explore an inherent trade-off between performance and load balancing.

B.1 Introduction

Many robotic applications can benefit from parallel deployment of multiple robots. In a search and rescue mission for example, the search area can be subdivided, so that each robot has less space to cover, resulting in quicker task completion. In order for the robots to efficiently collaborate, they need to know where they are with respect to each other and to the environment. In unstructured, GPS-denied environments a popular method for this, which generalizes well to all kinds of different robots, is visual simultaneous localization and mapping (SLAM). Visual SLAM takes as input camera images and produces as output an estimate of the robot's trajectory as well as typically some crude representation of the environment (map).

Visual SLAM as deployed on a single robot has recently reached maturity [26]. We identify three components of a state-of-the-art visual SLAM system:

1. *Visual Odometry* is a real-time component that converts sensor measurements into a pose estimate considering only data from the most recent past. It is in the nature of visual odometry algorithms to exhibit drift [70]: metric accuracy decreases over time and if the robot returns to a place it has visited before, the current pose estimate will most likely be inconsistent with the previous pose estimate at that place.
2. To mitigate such drift, SLAM systems have a *Place Recognition* module which uses visual cues to recognize previously visited places in spite of the inconsistent pose estimate.
3. Once previously visited places have been recognised, an *Optimization* module incorporates them to make a consistent map. The optimization module can also be used without place recognition, to reduce linearization errors of the visual odometry.

How can these components be extended to multiple robots? It is important to note that in multi-robot SLAM both place recognition and optimization need to consider data from all robots to build a meaningful and consistent global map. Hence, a centralized system, where all the data is sent to a central instance that runs place recognition and optimization is a popular choice for such a system [48, 66, 72, 146, 164, 178].

In certain situations, however, it can be interesting to opt for a decentralized system. A centralized system, for example, has a computational bottleneck at the central station and thus limited scalability. A well-designed decentralized system could defy this bottleneck. Centralized systems furthermore typically require permanent or regular connection to a dedicated central machine and thus preclude for example applications where a group of lightweight robots goes deep into the field. Finally, there exist militaristic and privacy arguments for using a decentralized system [31, 32].

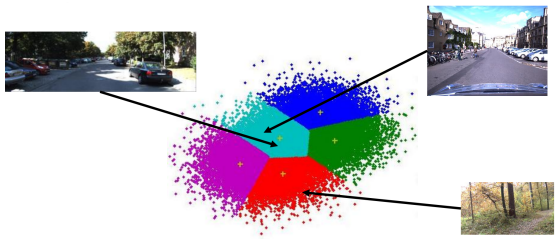


Figure B.1 – We propose an efficient, full-image descriptor-based method for decentralized visual place recognition. The method consists in clustering the descriptor space of NetVLAD [7] into several clusters and assigning each cluster to one robot. As a result, place recognition can be achieved by sending each place query to only one robot. Image credit: Yi Cao / Mathworks, [76, 128]

Decentralized systems, then, have the challenge of implementing place recognition and optimization in a decentralized way that does not require too much communication: technically, it would be possible for every robot to share its data with every other robot, but that would typically require a lot of data to be transmitted. In our work, we focus on decentralized place recognition. For recent work in decentralized optimization instead, we refer the reader to [32, 45, 158].

In this short paper we propose an efficient, full-image descriptor-based method for decentralized visual place recognition. The core of the method consists in clustering the descriptor space of NetVLAD [7] into several clusters and assigning each cluster to one robot. As a result, place recognition can be achieved by sending each place query to only one robot.

B.2 Related work

Decentralized visual place recognition by sending queries to every other robot scales poorly in terms of the robot count n. The complexity of every query is $\mathcal{O}(n)$. Of course, the overall bandwidth can me mitigated by adopting one of many existing approaches to map compression: among others, visual maps can be compressed by pruning unnecessary map features [57, 207], reducing the dimensionality of feature descriptors [127], or using an overall non-canonical visual place representation, such as a frequency-domain place representation [95] or one that relies on object extraction [31]. These approaches, however, ultimately do not reduce the complexity in robot count.

In previous work [39], we have proposed a Bag-Of-Words based [73, 151, 186] decentralized place recognition algorithm that reduces the query complexity by one order. The algorithm has been inspired by distributed hash tables [162, 188], and some previous work in image retrieval [99, 121]. The method we proposed there, however, suffers from two drawbacks: firstly, the method is somewhat complex and uses an assumption

whose full implications we do not yet fully understand. Secondly, the method requires for every query to have a message sent to every other robot (of size $\mathcal{O}(\frac{1}{n})$), thus still causing a lot of traffic.

In contrast, the method proposed here is much less complex, and, for every query, a message only needs to be sent to a single robot (plus to another robot for geometric verification, if place recognition succeeds). The key lies in substituting the bag-of-words approach with a full-image descriptor approach. This allows us to cluster the image descriptor space with k-means, and assign each cluster to a robot: any query from a cluster will be sent only to that robot. We show that this results in a competitive place recognition algorithm with minimal bandwidth requirements. We furthermore show how a problem of poor load balancing arises in practical deployment, and how it can be mitigated by sacrificing some recall.

B.3 Methodology

B.3.1 Bag-of-Words method

In [39], we have shown how the data exchange incurred in decentralized visual place recognition can be reduced by a factor of up to n, the robot count. This can be achieved by casting the place recognition problem to a key-value lookup problem, which can be efficiently distributed using deterministic key-to-peer assignment, as is for example common in distributed hash tables [169, 188]. In [39], we have thus cast the bag-of-words (BoW) place recognition method [151, 186] used in [73, 148]. In broad strokes, this is how the resulting method works:

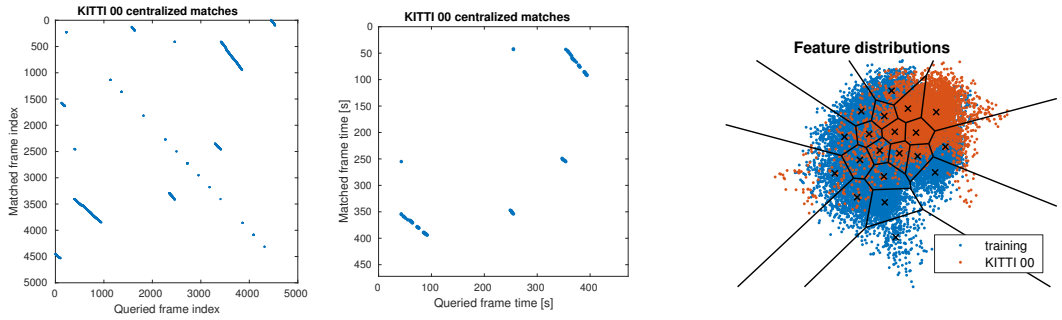
1. Before deployment, deterministically assign words of the visual vocabulary to the different robots.

2. When querying place recognition of an image frame, calculate the BoW vector and split it up into partial BoW vectors such that one partial BoW vector can be sent to each robot r, containing the coefficients of the words assigend to r.

3. The robots receive and process each their own partial query, returning the identity of the single frame which best matches the query frame according to the partial BoW vector. They also store the query, making it available as a result for subsequent queries.

4. Gather all partial results and determine which frame is most consistently returned as result.

5. Send a full query to the robot that has observed that frame for geometric verification.



(a) NetVLAD **(b)** Bag-of-Words **(c)**

Figure B.2 – (a) Confusion matrix for a centralized evaluation of the KITTI 00 dataset with 20 subtrajectories, using NetVLAD [7]. The threshold is manually selected and no geometric verification is performed. NetVLAD exhibits a visibly larger recall than the bag-of-words method we used in [39], shown in **(b)**. The dots on the diagonal indicate place matches on the boundaries between sub-trajectories. Matches within the same trajectory are excluded. **(c)** The bad load balancing during evaluation can be explained by the difference in distribution of the image features. This is the distribution of the first two dimensions for the training and testing data. Training data should be more general / cover more environments than the deployment data, which could come from only a very specific kind of environment. The superimposed k-means clustering is not an actual clustering used in our method, but a 2D k-means clustering that serves as illustration. As we can see, clustering the training data can result in an uneven distribution of features among clusters at deployment time. This leads to poor load balancing.

The last step involving geometric verification serves the purpose of rejecting false positives of the method and can at the same time be used to establish relative pose between the query and matched image frames. We have shown that this methods results in a bandwidth reduction of up to n (depending on the network infrastructure), while reducing recall by $10 - 20\%$ depending on the robot count. A lot of the recall reduction is due to steps 3) and 4) of the method, which are based on a simplifying assumption that we do not yet fully understand. See Sec. IV C. and Fig. 4 of [39] for a detailed discussion.

B.3.2 Full image descriptor method

Here, we instead propose to use a full-image descriptor place recognition method as a basis. In particular, we use the recent, deep-learning based NetVLAD method [7] which has been shown to perform excellently even under severe appearance and viewpoint changes. Indeed, as can be seen in the centralized evaluation of this method (Fig. B.2), its recall qualitatively looks better than the one of the BoW method we used in [39].

NetVLAD uses a deep neural network to calculate a low-dimensional feature vector $\vec{v} \in \mathbb{R}^d$ from an input image. Place matches can then be found by looking for the nearest vectors of other images according to the ℓ_2 distance.

This method can now efficiently be decentralized in the following way:

1. Before deployment, cluster the feature vector space and assign each cluster center to a robot.
2. When querying place recognition of an image frame, calculate the feature vector and send it as query to only the robot assigned to the corresponding cluster.
3. That robot processes the query, stores it for future reference, and replies with the best matching frame identifier.
4. Send a full query to the robot that has observed that frame for geometric verification.

Similarly to our previous method, the last step rejects false positives and provides a method to find the relative pose between the query and matched image frames. Evidently, this method is less complex than the one proposed in [39]: before geometric verification, data is sent to only one robot, and no assumptions on the fidelity of partial responses are made. As for the geometric verification query, it is with the new method possible to skip it if the NetVLAD distance of step 3) exceeds a certain threshold, resulting in further bandwidth reduction at the cost of potentially reduced recall. We use k-means clustering as clustering method for step 1).

It is essential to note that every query that is sent to a robot is stored to the place recognition database of that robot for retrieval in future relevant queries. In [39], we have referred to this as *add-querying*. Since we assume that the robots constantly send place recognition queries, at any given time all places seen up to that time will be stored in the place recognition database of the appropriate robot. This is even robust to message delay: two queries of the same place sent by different robots will both arrive at some point at the robot responsible for that place, independently of the message delay, so at least one of the two robots will be notified of the place match.

B.3.3 Mitigating poor load balancing

Clustering the image feature space should ideally be done on a very general dataset, in order to account for deployment in many different environments. Deployment, on the other hand, can often occur in very specific environments, which only constitute a subset of the general, trained descriptor space, see Figure B.2c.

This poses an interesting problem: because the features at deployment time only come from a subspace of the trained feature space, there will be some clusters that will contain only very little features at deployment time, while other clusters will contain disproportionally many features from deployment time. In practice, this should

translate in poor load balancing, since the robots assigned to the clusters with many features will have to handle much more queries than the other robots, which reduces scalability of the approach.

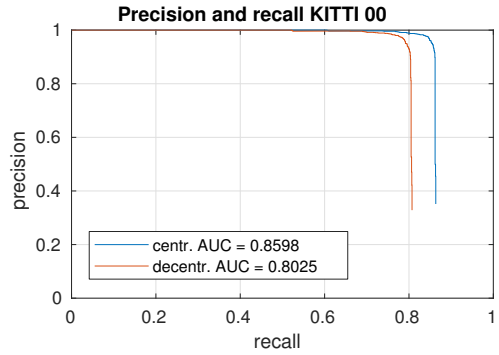
To mitigate this poor load balancing we propose to instead train several clusters per robot, and randomly assign the clusters to robots. This should distribute the robot's responsibilities in the feature space more evenly, and ensure that all robots are assigned features, even in narrow subspaces. However, this comes at the cost of increasing the amount of cluster boundaries, and, with that, the possibility that a matching pair of features is not in the same cluster. This would result in those features not being possible to match, and thus reduce recall.

B.4 Experiments

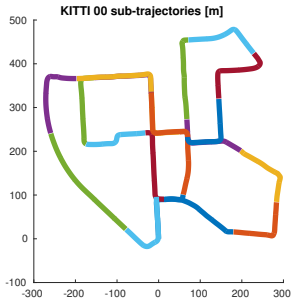
Unlike in [39] we do not actully implement the method on multiple processes. It is evident form the method that it needs n times less data exchange than if all queries were sent to all robots. To evaluate the place recognition performance the method would have if deployed on a group of robots, we simply exclude all images that are not in the same cluster as the query from the pool of possible responses to a query. Furthermore, we evaluate our method without geometric verification, again unlike in [39]. Evaluation with geometric verification is on one hand closer to practical deployment, but on the other hand contaminates the results with the performance of the used geometric verification implementation. To provide a fair evaluation, we evaluate precision and recall for all possible feature vector distance thresholds and consider the area under that curve (AUC) as metric for place recognition performance, see Figure B.3a. We use then NetVLAD feature vector dimension $d = 128$ (tunable thanks to a final layer that does principal component analysis). The clustering is trained on image data from the Oxford RobotCar dataset [128] and the method is evaluated on KITTI 00 [76] by splitting the sequence into n sub-sequences, one per robot, see Figure B.3b. We first evaluate the method in general, and show how it performs as we increase the number of robots. At the same time, we show how load balancing behaves as we increase the amount of robots. Then, we evaluate how using several clusters per robot can improve load balancing at the cost of performance.

B.5 Results

Fig. B.4a shows relative AUC (decentralized to centralized) of the method when applied to groups of $n \in [2, 20]$ robots. Recall suffers if the true match of a query is not in the same cluster as the query. It would seem that the performance of the decentralized NetVLAD method is only marginally better than the performance of the decentralized BoW method (see Fig. 7 in [39]). Consider however that as qualitatively seen in Fig.

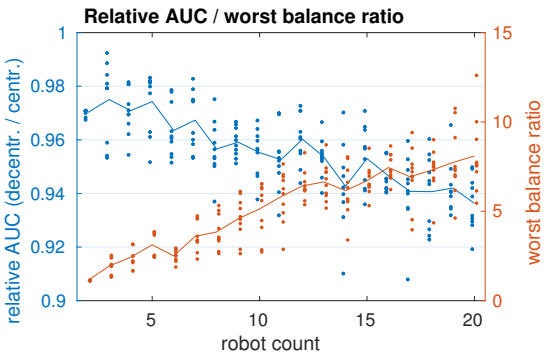


(a)

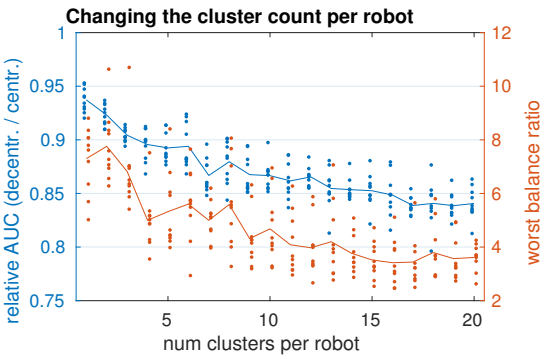


(b)

Figure B.3 – (a) We evaluate the place recognition performance using the area-under-curve measure (AUC) of the precision-recall curve, since we do not apply geometric verification. As we can see, NetVLAD exhibits excellent precision for the most part. We furthermore see how the clustering of the decentralized method results in reduced recall. This instance of decentralized place recognition has been run with 20 robots. **(b)** The subtrajectories resulting from splitting KITTI 00 into 20 parts.



(a)



(b)

Figure B.4 – (a) Relative AUC (decentralized to centralized) and worst balance ratio of our method for different robot counts. The worst balance ratio is the ratio of the busy-ness of the most queried robot compared to what it would be if the feature-to-cluster assignments were perfectly balanced. Results are averaged over 10 runs and dots indicate the results of the individual runs. **(b)** Relative AUC (decentralized to centralized) and worst balance ratio as we increase the amount of clusters per robot, with 20 robots. Results are averaged over 10 runs and dots indicate the results of the individual runs.

B.2, NetVLAD already has a higher recall than BoW in the first place. Furthermore, the method uses far less bandwidth for its distributed query than the BoW method. Recall from Table II in [39] that its distributed query size is 16 kilobytes plus overhead from sending the query to n robots. This method, when using single precision, only needs $d \times 4$ bytes per query, so 512 bytes with $d = 128$, plus overhead from only sending to one robot.

In Fig. B.4a we furthermore report the *worst balance ratio*, a measure for how much more queries the busiest robot receives compared to how much it would receive if the queries were perfectly balanced. The experiments confirm the bad balancing discussed in Section B.3.3. As we can see, the busiest robot handles up to half of all queries!

Figure B.4b shows how both load balancing improves and performance depreciates as we increase the amount of clusters in the system.

B.6 Conclusion

In this short paper, we have proposed an improvement over our previous work on decentralized place recognition [39]. The new method relies on recent, machine-learned full image descriptors and k-means clustering. We have explored how a problem of bad load balancing can arise when training and deployment feature distributions differ, and have shown how this problem can be mitigated by sacrificing some performance. Our method enables decentralized visual place recognition by sending only a lightweight query to a single other robot in the robot team. If a place is matched, a second query can be sent to the robot who observed the matching place for geometric verification.

Acknowledgement

We would like to thank Antonio Loquercio for the code reviews and helpful feedback.

C Data-Efficient Decentralized Visual SLAM

Reprinted, with permission, from:

T. Cieslewski, S. Choudhary, and D. Scaramuzza. "Data-Efficient Decentralized Visual SLAM". in: *IEEE Int. Conf. Robot. Autom. (ICRA)* (2018). DOI: 10.1109/ICRA.2018.8461155

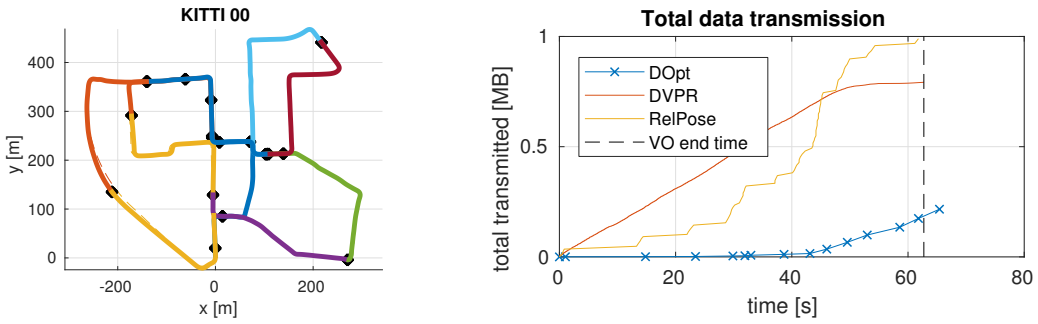
Data-Efficient Decentralized Visual SLAM

Titus Cieslewski, Siddharth Choudhary and Davide Scaramuzza

Abstract — Decentralized visual simultaneous localization and mapping (SLAM) is a powerful tool for multi-robot applications in environments where absolute positioning is not available. Being visual, it relies on cheap, lightweight and versatile cameras, and, being decentralized, it does not rely on communication to a central entity. In this work, we integrate state-of-the-art decentralized SLAM components into a new, complete decentralized visual SLAM system. To allow for data association and optimization, existing decentralized visual SLAM systems exchange the full map data among all robots, incurring large data transfers at a complexity that scales quadratically with the robot count. In contrast, our method performs efficient data association in two stages: first, a compact full-image descriptor is deterministically sent to only one robot. Then, only if the first stage succeeded, the data required for relative pose estimation is sent, again to only one robot. Thus, data association scales linearly with the robot count and uses highly compact place representations. For optimization, a state-of-the-art decentralized pose-graph optimization method is used. It exchanges a minimum amount of data which is linear with trajectory overlap. We characterize the resulting system and identify bottlenecks in its components. The system is evaluated on publicly available datasets and we provide open access to the code.

Supplementary Material

Data and code are at: https://github.com/uzh-rpg/dslam_open



(a) (b)

Figure C.1 – The proposed decentralized visual SLAM system is able to build a globally consistent map from ten sub-trajectories of KITTI 00 by exchanging only around 2MB in total – for decentralized optimization, place recognition and visual relative pose estimation, for all ten robots. **(a)** Ten sub-trajectories of KITTI 00 after running our method. Each color represents an individual trajectory, place matches are marked in black, dashed lines indicate the ground truth. **(b)** Data transmission over time for the system components: decentralized optimization (DOpt), decentralized visual place recognition (DVPR) and relative pose estimation (RelPose).

C.1 Introduction

Using several robots instead of one can accelerate many tasks such as exploration and mapping, or enable heterogeneous teams of robots, where each robot has a specialization. Multi-robot simultaneous localization and mapping (SLAM) is an essential component of any team of robots operating in an absolute positioning system denied environment, as it relates the current state estimate of each robot to all present and past state estimates of all other robots. Because cameras are cheap, light-weight and versatile sensors, we seek to implement a visual SLAM system. Visual Multi-Robot SLAM can be solved in a centralized manner, where a single entity collects all data and solves SLAM for all robots, but that relies on a central entity to always be reachable, to never fail and to scale to the size of the robot team, both in computation and bandwidth. Decentralized systems do not have this bottleneck, but are more challenging to implement.

Visual SLAM systems typically consist of three components: 1) a visual odometry algorithm which provides an initial state estimate, 2) a place recognition system which is able to relate the currently observed scene to previously observed scenes, and 3) an optimization back-end which consistently integrates the place matches from the place recognition system with the full stream of state estimates. The end product is a map, and that map feeds back to place recognition and optimization. It is this feedback which makes decentralized SLAM challenging, especially if one is concerned about communication bandwidth. Visual odometry is not involved in the feedback loop and is thus trivial to distribute – it can be run on each robot independently and feed its output to the rest of the decentralized SLAM system.

In previous work we have proposed state-of-the art decentralized place recognition [38] and optimization [32] systems separately. Both systems focus on data efficiency: they achieve performance similar to a centralized system while minimizing the data that needs to be exchanged. In this work, we integrate both systems along with a data-efficient method for visual feature association [192] into a full decentralized visual SLAM system. The system is evaluated on publicly available datasets and the code is provided open-source.

C.2 Related Work

In the following, we independently consider related work pertaining to decentralized optimization and to decentralized place recognition. We conclude with a review of existing integrated decentralized SLAM systems.

C.2.1 Decentralized Optimization

Decentralized estimation in multi-robot systems is an active field of research, with special attention being paid to communication constraints [158], heterogeneity [11, 92], consistency [10], and robust data association [55]. The literature offers distributed implementations of different estimation techniques, including Kalman filters [168], information filters [194], particle filters [29, 89], and distributed smoothers [32, 45]

In multi-robot systems, maximum-likelihood trajectory estimation can be performed by collecting all measurements at a centralized inference engine, which performs the optimization [4, 11, 55, 103, 114]. However, it is not practical to collect all measurements at a single inference engine since it requires a large communication bandwidth. Furthermore, solving trajectory estimation over a large team of robots can be too demanding for a single computational unit.

These reasons triggered interest towards *decentralized trajectory estimation*, in which the robots only exploit local communication, in order to reach a consensus on the trajectory estimate [5, 46, 68, 106, 150]. Recently, Cunnigham et al. [45, 46] used Gaussian elimination, and developed an approach, called DDF-SAM, in which robots exchange Gaussian marginals over the *separators* (i.e., the variables observed by multiple robots).

While Gaussian elimination has become a popular approach, it has two major shortcomings. First, the marginals to be exchanged among the robots are dense, hence the communication cost is quadratic in the number of separators. This motivated the use of sparsification techniques [158]. Second, Gaussian elimination is performed on a linearized version of the problem, hence these approaches require good linearization points and complex bookkeeping to ensure consistent linearization across the robots [45]. The need of a linearization point also characterizes gradient-based tech-

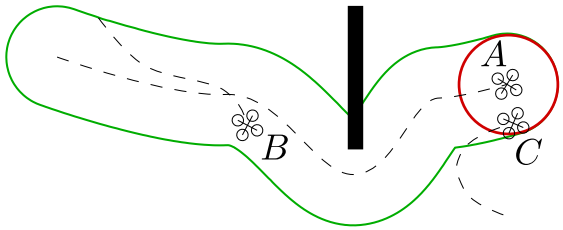


Figure C.2 – Restricting relative pose measurements to robots that are in line of sight (red circle) severely limits the recall of relative localization: Robots A and C can establish relative localization but A and B cannot. This often applies to direct relative measurements. By ensuring that relative localization can occur from all past measurements (green outline) we can increase recall and put less restrictions on the high-level application that controls the robots.

niques [106]. An alternative approach to Gaussian elimination is the Gauss-Seidel approach of Choudhary et al. [32], which requires communication linear in the number of separators. The Distributed Gauss-Seidel approach only requires the latest estimates for poses involved in inter-robot measurement to be communicated and therefore does not require consistent linearization and complex bookkeeping. This approach also avoids information double counting since it does not communicate marginals.

C.2.2 Decentralized Place Recognition and Pose Estimation

In order to solve decentralized SLAM, measurements that relate poses among different robots (*inter-robot measurements*) need to be established. A popular type of inter-robot measurements are direct measurements of the other robot [222], such as time-of-flight distance measurements [158] or vision-based relative pose estimation [103]. The latter is typically aided with markers that are deployed on the robots. To the best of our knowledge, most types of direct measurements require specialized hardware (which can precisely measure time-of-flight for example, or visual markers). Furthermore, many types of direct measurements require the robots to be in line of sight, which, in many environments, imposes a major limitation on the set of relative poses that can be established, see Figure C.2. Limiting relative measurements in this way translates into limitations for higher-level applications that would like to use decentralized SLAM as a tool.

We prefer, instead, to use indirect relative measurements. These are established through registration of observations that two robots A and B make of the same scene [6, 192]. This requires no additional hardware: the sensors already used for odometry can be re-used for inter-robot measurements. Since establishing inter-robot measurements in this way is equivalent to establishing loop closures in single-robot SLAM, indirect measurements are widely used in centralized visual SLAM algorithms [66, 164, 178]. The establishment of indirect measurements is still limited by the communication range of the robots, but in practice it is feasible to establish a communication range that

exceeds line-of-sight. Furthermore, if two robots cannot communicate directly, they might still communicate through a multi-hop network. A disadvantage of indirect measurements, then, is that they require bandwidth to communicate. If all robots share all data with every other robot, the amount of data exchanged is quadratic in the number of robots. One way to mitigate the bandwidth is to exchange data only with immediate neighbours, but this has the same reduced recall as discussed in Figure C.2. Another way to reduce bandwidth is to use compact representations that allow to establish inter-robot measurements with a minimal amount of data exchange. Visual feature-based maps can be compressed by compressing feature descriptors [127], substituting feature descriptors for corresponding cluster centers in a *visual vocabulary* [192], removing landmarks that prove not to be necessary for localization [43, 57] or using compact full-image descriptors [7]. In previous work, we have shown another way to reduce the bandwidth of place recognition: data exchange can be reduced by a factor of the robot count if the problem can be cast to key-value lookup[38]. In this work, we use [38], an advanced version of [39] which at the same time makes use of a compact, state-of-the-art place representation, NetVLAD [7]. [192] is used in association of visual features for relative pose estimation.

C.2.3 Integrated Decentralized SLAM

Several systems have previously combined decentralized optimization and relative pose estimation into decentralized SLAM. Several optimization works cited in Section C.2.1, such as [32, 158] use direct relative measurements and can thus be considered full decentralized SLAM systems with the recall limitation illustrated in Figure C.2. DDF-SAM [46] has been extended to a decentralized SLAM system in [47] by adding a data association system which matches planar landmark points. This system has been validated in the real world, but the landmarks consisted of manually placed poles that were detected by a laser range finder. [114] establishes relative poses by exchanging and aligning 2D laser scan edges. [32] has been extended to a decentralized SLAM system in [31] where data association is provided by identification and pose estimation of commonly observed objects. This approach relies on the existence of unique, pre-trained objects in the environment. The majority of decentralized SLAM systems we have found in our literature review are not vision-based. A first decentralized visual SLAM system has been proposed in [22], but it relies on exchanging the full maps among robots. Furthermore, it is only evaluated on $50m \times 50m$ L-shaped trajectory and with only two robots, with images obtained from a simulated environment. In contrast, we evaluate our system on a large scale scenario, with more robots, on real data, and our system uses state-of-the-art algorithms which exchange far less than the full maps.

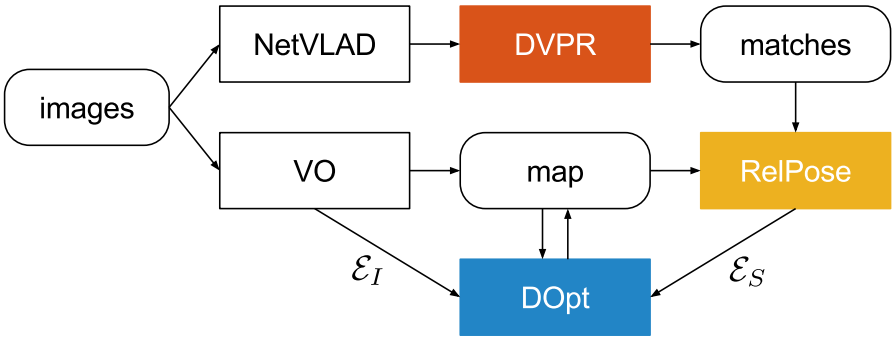


Figure C.3 – Components and interactions in our system. The diagram shows the components that run on each robot. The colored blocks indicate components that communicate with other robots. Sharp corners indicate software modules, rounded corners indicate data. *DVPR*: Decentralized visual place recognition, *VO*: visual odometry, *RelPose*: Geometric verification and relative pose estimation, *DOpt*: Decentralized optimization.

C.3 Methodology

The proposed decentralized SLAM system consists of $|\boldsymbol{\Omega}|$ robots $\boldsymbol{\Omega} = \{\alpha, \beta, \gamma, \ldots\}$ which each execute the system illustrated in Figure C.3. The images taken by a camera are fed to both NetVLAD [7] and a visual odometry (VO). NetVLAD produces a compact image representation used for decentralized visual place recognition (DVPR) [38], which itself produces candidate place recognition matches. The VO produces a sparse feature map that can be used for localization. The relative pose estimation module (RelPose) extracts the parts of the sparse map that are observed at the candidate place matches and uses them to establish relative poses between the robots trajectories, or to reject candidate matches. The decentralized optimization module (DOpt) [32] obtains the initial guess from the map, intra-robot relative pose measurements $\mathcal{E}_I$ from the VO and the inter-robot relative pose measurements $\mathcal{E}_S$ from RelPose, and updates the map. The system works continuously as new images are acquired and assumes that every robot can communicate with every other robot.

C.3.1 Intra-Robot Measurements

Intra-robot measurements are obtained from a visual odometry (VO) algorithm. Any VO algorithm can be used in our system as long as it fulfills the following requirements:

1) It produces a pose graph. The VO of robot α defines frames $\mathcal{F}_\alpha = \{(\alpha, 1), (\alpha, 2), (\alpha, 3), \ldots\}$ and provides the intra-robot measurements

$$\mathcal{E}_{I,\alpha} \doteq \{\bar{\boldsymbol{z}}^{\alpha_i}_{\alpha_{i+1}} \forall (\alpha, i), (\alpha, i+1) \in \mathcal{F}_\alpha\} \tag{C.1}$$

$$\bar{\boldsymbol{z}}^{\alpha_i}_{\alpha_{i+1}} \doteq (\bar{\boldsymbol{R}}^{\alpha_i}_{\alpha_{i+1}}, \bar{\boldsymbol{t}}^{\alpha_i}_{\alpha_{i+1}}) \in \mathrm{SE}(3) \tag{C.2}$$

describing pose transforms between subsequent frames.

2) Each pose is associated to an image to be used for place recognition. We denote this image by I_{α_i}. Our current system does not yet take full advantage of a multi-camera setup, which could be a useful future extension. The image capture time is denoted as t_{α_i} [1].

3) The VO provides absolute scale, as implied by (C.2). Monocular VO typically estimates the pose up to a scale due to scale ambiguity. Scale can be obtained from a stereo camera or by fusing with inertial measurements.

In our implementation, we use for visual odometry ORB-SLAM [148] in stereo configuration.

C.3.2 Inter-Robot Measurements

The inter-robot measurements are

$$\mathcal{E}_S \doteq \{\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} | \alpha \neq \beta\} \tag{C.3}$$

with $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ defined like (C.2). They are established in two phases: Given its latest image I_{α_i}, robot α first tries to determine whether there is any image I_{β_j} of the same scene previously captured by another robot. This is achieved using decentralized visual place recognition, detailed in Section C.3.3. Then, if $\exists I_{\beta_j}$, α tries to establish $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ using the method detailed in Section C.3.4. This two-phase approach is pursued because, as we will see, establishing relative poses consumes more data than recognizing whether $\exists I_{\beta_j}$. The first phase is capable of reducing the amount of data to be exchanged by determining which robot to send the relative pose query to, and by determining whether it is worthwhile to send it in the first place, using a minimal amount of data.

C.3.3 Decentralized Visual Place Recognition

The first phase is accomplished by matching image I_{α_i} to the images previously captured by the other robots, $\{I_{\gamma_k} | t_{\gamma_k} < t_{\alpha_i}\}$, using the full-image descriptor NetVLAD [7]. NetVLAD uses a neural network to calculate a vector $\boldsymbol{v}_{\alpha_i} \in \mathbb{R}^{D_{\text{NetVLAD}}}$ from I_{α_i}. In rough terms, it is trained such that $\|\boldsymbol{v}_{\alpha_i} - \boldsymbol{v}_{\gamma_k}\|_{\ell_2}$ is lower if I_{α_i} and I_{γ_k} observe the same scene than if they observe different scenes. For more details, see [7]. NetVLAD is well-suited for decentralized place recognition, as $\boldsymbol{v}_{\alpha_i}$ is all that robot α needs to send to robot γ to establish whether a matching image exists among $\{I_{\gamma_k}\}$. Concretely, we seek the image $I_{\beta_j}^{\star}$ captured by another robot whose NetVLAD vector $\boldsymbol{v}_{\beta_j}^{\star}$ has the shortest ℓ_2 distance

[1] A common time reference is hard to establish, and, strictly speaking, not well defined, so consider t_{α_i} a Lamport timestamp [113]. t_{α_i} is not actually used in the implementation, so this detail is not particularly important.

to $\boldsymbol{v}_{\alpha_i}$ and which satisfies

$$\left\| \boldsymbol{v}_{\alpha_i} - \boldsymbol{v}_{\beta_j} \right\|_{\ell_2} < \tau_{\text{NetVLAD}} \tag{C.4}$$

where τ_{NetVLAD} is a threshold parameter. $I^{\star}_{\beta_j}$ can be found by sending $\boldsymbol{v}_{\alpha_i}$ to all other robots, but we use a method that requires $|\boldsymbol{\Omega}|$ times less data exchange [38], which is particularly interesting for scaling to large robot teams. In this method, each robot γ is pre-assigned a cluster center $\boldsymbol{c}_\gamma$ and each robot knows the $\boldsymbol{c}$ of all other robots. The query $\boldsymbol{v}_{\alpha_i}$ is sent only to robot $\delta = \arg\min_\gamma \|\boldsymbol{v}_{\alpha_i} - \boldsymbol{c}_\gamma\|_{\ell_2}$, and δ replies with the identifier of I_{β_j},

$$(\beta, j) = \underset{(\gamma,k):\boldsymbol{v}_{\gamma_k} \in \mathcal{V}_\delta}{\arg\min} \|\boldsymbol{v}_{\alpha_i} - \boldsymbol{v}_{\gamma_k}\|_{\ell_2}, \tag{C.5}$$

if $\boldsymbol{v}_{\beta_j}$ also satisfies (C.4), where $\mathcal{V}_\delta$ is the Voronoi cell $\mathcal{V}_\delta = \{\boldsymbol{v} | \boldsymbol{c}_\delta = \arg\min_{\boldsymbol{c}_\gamma} \|\boldsymbol{v} - \boldsymbol{c}_\gamma\|_{\ell_2}\}$. The query is executed for every frame of every robot, so at t_{α_i} this query has already been executed $\forall \gamma, k : t_{\gamma_k} < t_{\alpha_i}$, and δ is aware of all $\boldsymbol{v}_{\gamma_k} \in \mathcal{V}_\delta$ by virtue of having received the previous queries. It can thus provide (β, j) without any extra data exchange. This method approximates $I^{\star}_{\beta_j} \sim I_{\beta_j}$; $I^{\star}_{\beta_j} = I_{\beta_j}$ if $\exists \delta : \boldsymbol{v}^{\star}_{\beta_j}, \boldsymbol{v}_{\alpha_i} \in \mathcal{V}_\delta$. In the method, the cluster centers are determined using k-means clustering on a training dataset, in our case the Oxford Robotcar Dataset [128]. The properties of the method are discussed in more detail in [38].

C.3.4 Relative Pose Estimation

Once and if (β, j) has been established, α sends a relative pose estimation request to β. The relative pose estimation (RelPose) can and should heavily depend on the visual odometry (VO) since it can benefit from re-using by-products of the VO such as 3D positions of landmarks. In our method, we use ORB-SLAM [148] for VO and accordingly, our RelPose implementation imitates its loop closure method. However, it does not imitate [148] exactly, since the latter makes use of the data surrounding the match, which in decentralized RelPose would translate into a lot of data exchange. We want to avoid this and thus, we combine the relative pose estimation method from [148] with the visual data association method from [192], which reduces the data to be sent from α to β, $d^{\alpha\to\beta}_{\text{RelPose}}$, to a set of landmarks and visual word identifiers:

$$d^{\alpha\to\beta}_{\text{RelPose}} = (\{(w_k, \boldsymbol{p}_k) \forall k \in K_{\alpha_i}\}, \alpha, i, j) \tag{C.6}$$

where K_{α_i} are the keypoints observed in I_{α_i}, w_k is the identifier of a visual word associated to keypoint k and $\boldsymbol{p}_k$ is the 3D position of the landmark corresponding to keypoint k, expressed in the camera frame of I_{α_i}. Visual words represent Voronoi cells in the keypoint descriptor space, here the space of ORB descriptors [170]. They provide a quantization of the high-dimensional descriptor space, and since similar descriptors

are likely to be assigned to the same word, word identifiers can be used for matching keypoints between images [192]. Since this is less precise than using raw descriptors, we provide an option to exchange full descriptors instead. Given $d^{\alpha\to\beta}$, β establishes pairs of matching keypoints $\{(k,k')|w_k = w_{k'}, \nexists\, l \in K_{\alpha_i} : w_l = w_k, \nexists\, l' \in K_{\beta_j} : w_{l'} = w_{k'}\}$. Given the corresponding pairs of landmark positions $\{(\boldsymbol{p}_k, \boldsymbol{p}_{k'})\}$, RANSAC is used to determine an initial estimate of $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ and the corresponding inlier matches $M = \{(k,k')^\star\}$. A place match is rejected if $|M| < \tau_{\text{inliers}}$. Otherwise, the inlier landmark position pairs $\{(\boldsymbol{p}_k, \boldsymbol{p}_{k'})^\star\}$ are used to refine $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ to the pose that minimizes a robust 3D registration error

$$\sum_{\{(\boldsymbol{p}_k,\boldsymbol{p}_{k'})^\star\}} \rho_{\tau_{\text{loss}}}(\|\boldsymbol{p}_k - (\bar{\boldsymbol{R}}\cdot\boldsymbol{p}_{k'} + \bar{\boldsymbol{t}})\|^2_{\ell_2}). \tag{C.7}$$

We use a robust weight loss function $\rho_{\tau_{\text{loss}}}(s) = \arctan\frac{s}{\tau_{\text{loss}}}$ to reduce the weight of remaining keypoint match outliers.

To avoid relative pose $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ outliers we use a consistency check, where $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ is only accepted if either: *a)* there is another already accepted $\bar{\boldsymbol{z}}^{\alpha_{i'}}_{\beta_{j'}}$ such that (α,i') and (α,i) are within a distance of τ_{cdist} and $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ and $\bar{\boldsymbol{z}}^{\alpha_{i'}}_{\beta_{j'}}$ are consistent; *b)* no accepted $\bar{\boldsymbol{z}}^{\alpha_{i'}}_{\beta_{j'}}$ exists, but a previous candidate which fulfills the same conditions. $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ and $\bar{\boldsymbol{z}}^{\alpha_{i'}}_{\beta_{j'}}$ are considered consistent if the two ways of calculating the relative position between (α,i') and (β,j) – with $\bar{\boldsymbol{z}}^{\alpha_{i'}}_{\beta_{j'}}$ and $\mathcal{E}_{I,\beta}$ or with $\mathcal{E}_{I,\alpha}$ and $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ – result in positions that are within a distance $< \tau_{\text{tol}}$.

RelPose still constitutes a significant amount of data, so we introduce a parameter τ_{mdg} (minimum distance geometric verifications) which allows to throttle geometric verifications. Geometric verification are skipped for matches which are close to already established relative poses. $d^{\alpha\to\beta}$ is only sent for geometric verification to β if there is no other (α,i') matched to a frame of β such that $\left\|\boldsymbol{t}_{\alpha_i} - \boldsymbol{t}_{\alpha_{i'}}\right\|_{\ell_2} < \tau_{\text{mdg}}$. This assumes that the VO and RelPose are consistent enough such that the resulting subset of retrievable $\mathcal{E}_S$ suffices to produce a globally-consistent state estimate.

C.3.5 Decentralized Optimization

In our decentralized optimization, each robot estimates its own trajectory using the available measurements, and leveraging occasional communication with other robots. The pose of frame (α,i) is denoted with $\boldsymbol{x}_{\alpha_i}$. The trajectory of robot α is then denoted as $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_1}, \boldsymbol{x}_{\alpha_2}, \ldots]$. , where $\boldsymbol{x}_{\alpha_i} = (\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}) \in \text{SE}(3)$, $\boldsymbol{R}_{\alpha_i} \in \text{SO}(3)$ represents the 3D rotation, and $\boldsymbol{t}_{\alpha_i} \in \mathbb{R}^3$ represents the 3D translation.

While our classification of the measurements (inter *vs.* intra) is based on the robots involved in the measurement process, all relative measurements can be framed within the same measurement model. Since all measurements correspond to noisy observations

of the relative pose between a pair of poses, say $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_j}$, a general measurement model is:

$$\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \doteq (\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}, \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i}), \quad \text{with:} \begin{cases} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} = (\boldsymbol{R}_{\alpha_i})^{\mathsf{T}} \boldsymbol{R}_{\beta_j} \boldsymbol{R}_{\epsilon} \\ \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} = (\boldsymbol{R}_{\alpha_i})^{\mathsf{T}} (\boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i}) + \boldsymbol{t}_{\epsilon} \end{cases} \tag{C.8}$$

where the relative pose measurement $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ includes the relative rotation measurements $\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}$, which describes the attitude $\boldsymbol{R}_{\beta_j}$ with respect to the reference frame (α, i), "plus" a random rotation $\boldsymbol{R}_{\epsilon}$ (measurement noise), and the relative position measurement $\bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i}$, which describes the position $\boldsymbol{t}_{\beta_j}$ in the reference frame (α, i), plus random noise $\boldsymbol{t}_{\epsilon}$.

Assuming that the translation noise is distributed according to a zero-mean Gaussian with information matrix $\omega_t^2 \mathbf{I}_3$, while the rotation noise follows a Von-Mises distribution with concentration parameter ω_R^2, it is possible to demonstrate [30] that the ML estimate $\boldsymbol{x} \doteq \{(\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}), \forall \alpha \in \Omega, \forall i\}$ can be computed as solution of the following optimization problem:

$$\min_{\substack{\boldsymbol{t}_{\alpha_i} \in \mathbb{R}^3, \boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3) \\ \forall \alpha \in \boldsymbol{\Omega}, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathrm{F}}^2 \tag{C.9}$$

As proposed in [31], we use a two-stage approach to solve the optimization problem in a decentralized manner: we first solve a relaxed version of (C.9) and get an estimate for the rotations $\boldsymbol{R}_{\alpha_i}$ of all robots, and then we recover the full poses and top-off the result with a Gauss-Newton (GN) iteration.

In both stages, we need to solve a linear system where the unknown vector can be partitioned into subvectors, such that each subvector contains the variables associated to a single robot in the team. For instance, we can partition the vector $\boldsymbol{r}$ in the first stage, as $\boldsymbol{r} = [\boldsymbol{r}_{\alpha}, \boldsymbol{r}_{\beta}, \ldots]$, such that $\boldsymbol{r}_{\alpha}$ describes the rotations of robot α. Similarly, we can partition $\boldsymbol{p} = [\boldsymbol{p}_{\alpha}, \boldsymbol{p}_{\beta}, \ldots]$ ($\boldsymbol{p}$ represents the linearized poses) in the second stage, such that $\boldsymbol{p}_{\alpha}$ describes the trajectory of robot α. Therefore, the linear systems in the first two stages can be framed in the general form $\boldsymbol{H}\boldsymbol{y} = \boldsymbol{g}$:

$$\begin{bmatrix} \boldsymbol{H}_{\alpha\alpha} & \boldsymbol{H}_{\alpha\beta} & \cdots \\ \boldsymbol{H}_{\beta\alpha} & \boldsymbol{H}_{\beta\beta} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_{\alpha} \\ \boldsymbol{y}_{\beta} \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_{\alpha} \\ \boldsymbol{g}_{\beta} \\ \vdots \end{bmatrix} \tag{C.10}$$

where we want to compute the vector $\boldsymbol{y} = [\boldsymbol{y}_{\alpha}, \boldsymbol{y}_{\beta}, \ldots]$ given the (known) block matrix $\boldsymbol{H}$ and the (known) block vector $\boldsymbol{g}$, partitioned according to the block-structure of $\boldsymbol{y}$.

The linear system (C.10) can be rewritten as:

$$\sum_{\delta \in \boldsymbol{\Omega}} \boldsymbol{H}_{\alpha\delta} \boldsymbol{y}_\delta = \boldsymbol{g}_\alpha \qquad \forall \alpha \in \boldsymbol{\Omega}$$

Taking the contribution of $\boldsymbol{y}_\alpha$ out of the sum, we get:

$$\boldsymbol{H}_{\alpha\alpha} \boldsymbol{y}_\alpha = - \sum_{\delta \in \boldsymbol{\Omega} \setminus \{\alpha\}} \boldsymbol{H}_{\alpha\delta} \boldsymbol{y}_\delta + \boldsymbol{g}_\alpha \qquad \forall \alpha \in \boldsymbol{\Omega} \tag{C.11}$$

The set of equations (C.11) is the same as the original system (C.10), but clearly exposes the contribution of the variables associated to each robot.

The distributed Gauss-Seidel algorithm [16] starts at an arbitrary initial estimate $\boldsymbol{y}^{(0)} = [\boldsymbol{y}_\alpha^{(0)}, \boldsymbol{y}_\beta^{(0)}, \ldots]$ and applies the following update rule, for each robot $\alpha \in \boldsymbol{\Omega}$:

$$\boldsymbol{y}_\alpha^{(k+1)} = \boldsymbol{H}_{\alpha\alpha}^{-1} \left(- \sum_{\delta \in \boldsymbol{\Omega}_\alpha^+} \boldsymbol{H}_{\alpha\delta} \boldsymbol{y}_\delta^{(k+1)} - \sum_{\delta \in \boldsymbol{\Omega}_\alpha^-} \boldsymbol{H}_{\alpha\delta} \boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha \right) \tag{C.12}$$

where $\boldsymbol{\Omega}_\alpha^+$ is the set of robots that already computed the $(k+1)$-th estimate, while $\boldsymbol{\Omega}_\alpha^-$ is the set of robots that still have to perform the update (C.12), excluding node α (intuitively: each robot uses the latest estimate). We can see that if the sequence produced by the iterations (C.12) converges to a fixed point, then such point satisfies (C.11), and indeed solves the original linear system (C.10).

C.3.6 Making Optimization Work Continuously

Since the map estimate handled by DOpt is not a consistent map state between iterations, it cannot be considered as best estimate at any time. Furthermore, the optimization is not laid out to incorporate new measurements between iterations. As a consequnce, the map estimate is concurrently modified by the VO and DOpt, which would lead to inconsistencies if unchecked. This can be solved using optimistic concurrency control (OCC). In OCC, each of the processes that concurrently operate on some data operate on their own copy. Once a process is done, its copy is merged back to the reference state. OCC is a central concept in version control tools such as SVN and GIT, and has recently been applied to distributed robotic systems in [33] and [72]. In our method, the handling of VO and place recognition is considerd one process and decentralized optimization another. We let VO and place recognition operate directly on the reference state. Decentralized optimization, as described in Section C.3.5 is executed episodically. At the beginning of each episode, the robots consent on a reference time t_e. During the episode, only the poses $\boldsymbol{x}_{\alpha_i}$ which satisfy $t_{\alpha_i} < t_e$ are optimized. Once the optimization has finished, the $\boldsymbol{x}_{\alpha_i}$ of the reference state are replaced with the newly optimized

estimates $\boldsymbol{x}'_{\alpha_i}$. The remaining $\boldsymbol{x}_{\alpha_j}$ for which $t_{\alpha_j} > t_e$ are corrected with:

$$\boldsymbol{x}_{\alpha_j} \leftarrow \boldsymbol{R} \cdot \boldsymbol{x}_{\alpha_j} + \boldsymbol{t} \tag{C.13}$$

which satisfies $\boldsymbol{x}'_{\alpha_e} = \boldsymbol{R} \cdot \boldsymbol{x}_{\alpha_e} + \boldsymbol{t}$ with (α, e) referring to the frame recorded just before t_e.

C.4 Experiments

In the experiments, we seek to find out how much data the individual components and the overall system use and how accurate the state estimate is as time progresses. This will give us an idea of the applicability and scalability of the system, and will let us identify which components are most worthwhile to be optimized in the future.

C.4.1 Data Exchange Evaluation

The components of our system that exchange data are decentralized visual place recognition, relative pose estimation and decentralized optimization. Each of these components could potentially be further developed to use less data. Thus, we record data transfer for each component individually. Data transfers for the individual components are:

$$\bigcup_{\text{episode}} d_{\text{DOpt}}^{\alpha \to \beta} = \{ \bigcup_{\text{rot. iters.}} \boldsymbol{R}_{\beta_j}^{\alpha_i}, \bigcup_{\text{pose iters.}} \bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \} \tag{C.14}$$

$$d_{\text{DVPR}}^{\alpha \to \delta} = (\alpha, i, \boldsymbol{v}_{\alpha_i}), \quad d_{\text{DVPR}}^{\delta \to \alpha} = (\beta, j) \tag{C.15}$$

$$d_{\text{RelPose}}^{\beta \to \alpha} = (\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}, \boldsymbol{x}_{\beta_{j'}}^{-1} \boldsymbol{x}_{\beta_j}) \tag{C.16}$$

with $d_{\text{RelPose}}^{\alpha \to \beta}$ from (C.6) and where $\boldsymbol{x}_{\beta_{j'}}^{-1} \boldsymbol{x}_{\beta_j}$ is sent for the relative pose consistency check. The size of data is calculated accordingly, using $6 \cdot 8$ bytes for poses $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$, $9 \cdot 8$ bytes for rotations $\boldsymbol{R}_{\beta_j}^{\alpha_i}$, $8 \cdot D_{\text{NetVLAD}}$ bytes for NetVLAD vectors, 1 byte for robot indices, 4 bytes for frame indices, 2 bytes for visual word indices w_k and $3 \cdot 4$ bytes for landmarks $\boldsymbol{p}_k$. Note that for the first part of the optimization, rotations are parametrized with 9 coefficients, because $\boldsymbol{R}_{\beta_j}^{\alpha_i} \in SO(3)$ is not enforced in order to have a linear system.

C.4.2 Accuracy Evaluation

As it is typically done for SLAM systems, we evaluate the accuracy of our system. Unlike of other SLAM systems, however, we evaluate the evolution of accuracy over time. This allows us to better characterize the system. Accuracy is measured using the average trajectory error (ATE). The ATE requires alignment to the ground truth trajectory, which is only meaningful for connected components. Thus, we report the

	Sec.	value		Sec.	value
D_{NetVLAD}	C.3.3	128	τ_{cdist}	C.3.4	$20m$
τ_{NetVLAD}	C.3.3	0.1	τ_{tol}	C.3.4	$4m$
τ_{inliers}	C.3.4	20	τ_{mdg}	C.3.4	$0m$
τ_{loss}	C.3.4	$3m$			

Table C.1 – Default parameter values

ATE for different connected components individually.

C.4.3 Parameter Studies

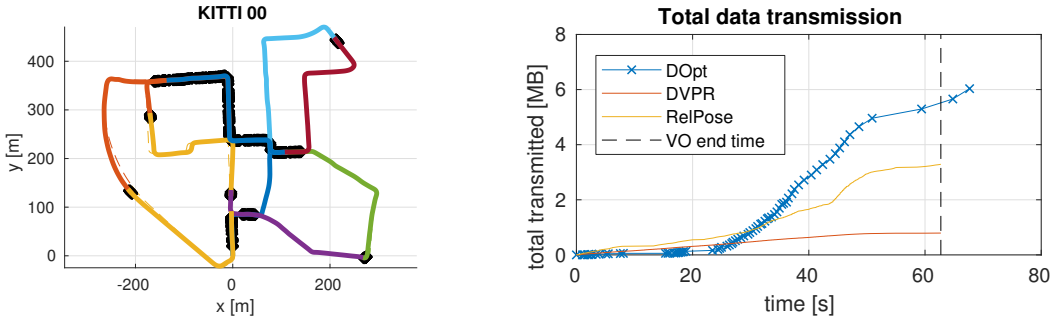
A parameter that we find of particular interest is τ_{mdg}, which determines the distance between an established relative pose $\tilde{z}^{\alpha_i}_{\beta_j}$ and the next frame of α to be sent to β for geometrical verification. τ_{mdg} can significantly reduce the data to be transmitted for RelPose and DOpt. We verify that this does not happen at the cost of accuracy.

Another interesting parameter is the used NetVLAD dimension D_{NetVLAD}. Since the last layer of NetVLAD performs principal component analysis, D_{NetVLAD} can be tuned arbitrarily up to 4096 dimensions. The lower D_{NetVLAD}, the lower the DVPR traffic, but also the lower its precision.

C.5 Results

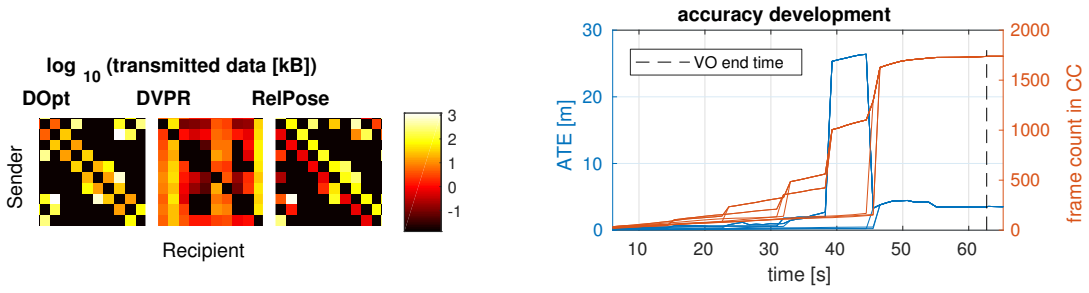
We evaluate our system on the KITTI dataset [76]. Like in [39], we split the dataset into $|\boldsymbol{\Omega}| = 10$ parts to simulate trajectories recorded by different robots. Figure C.4a shows the used sub-trajectories of KITTI 00 and the end result of running our system with the parameter values shown in Table C.1 and using the visual data association method from [192]. The corresponding data transmission is shown in Figure C.4b. Note that we plot the cumulative data transmission up to a given time. This directly exhibits the full amount of transmitted data and makes it easy to see time-varying bandwidth (slope). DOpt and RelPose require the most data transmission, but as we will see, DOpt traffic can be significantly reduced, while RelPose traffic remains high.

Figure C.5a shows the total data transmission among pairs of robots. This data transmission is highly uneven and could require consideration in multi-hop networks, but that is outside of the scope of this paper. DOpt and RelPose highly depend on the trajectory overlap between robots. The higher the overlap, the more data is exchanged by these components. Decentralized visual place recognition traffic is also uneven, but in a different way. It is due to the clustering that assigns NetVLAD cluster centers to robots. What we see in Figure C.5a is that robots 2, 6 and 7 are assigned the v that represent the majority of NetVLAD descriptors in KITTI 00. This unevenness can be



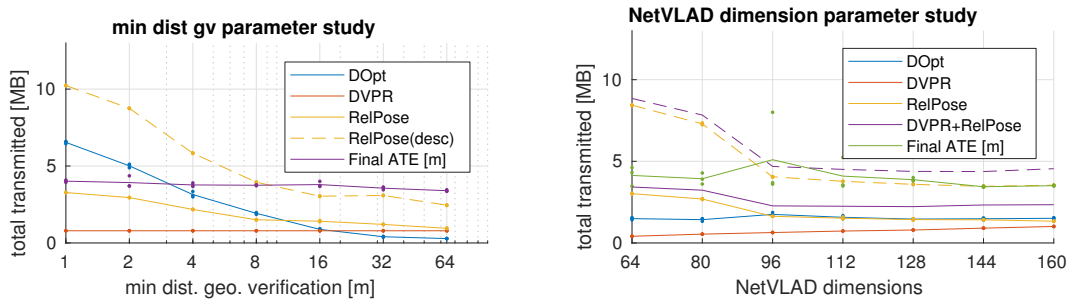
(a) (b)

Figure C.4 – (a) Ten sub-trajectories of KITTI 00 after running our method with the parameters in Table C.1. Each color represents an individual robots trajectory, place matches are marked with bold black dots. The aligned ground truth is indicated with dashed lines. **(b)** Data transmission over time for the three system components: decentralized optimization (DOpt), decentralized visual place recognition (DVPR) and relative pose estimation (RelPose), using the parameters in Table C.1. DOpt continues to run after visual odometry (VO) has ended until it has incorporated all data produced by VO.



(a) (b)

Figure C.5 – (a) Data exchanged among individual pairs of robots. Row numbers indicate sender and column numbers receiver. DOpt, DVPR and RelPose traffic is visualized separately and separated by green lines. **(b)** Average trajectory error (ATE) and size of the connected component (CC) each of the ten robots is in. As the CCs of two robots converge, so does the line representing the frame count and ATE within these CC. Note that accuracy does not change significantly in the optimizations executed after visual odometry (VO) has ended – the state estimate reaches its final accuracy already before all the data is considered for optimization.



(a) (b)

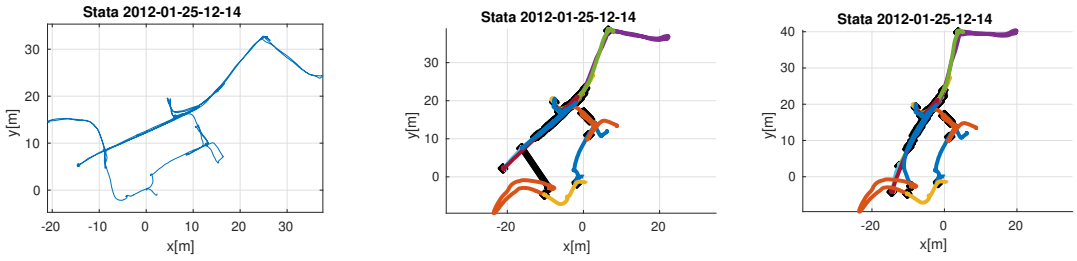
Figure C.6 – (a) Effect of changing τ_{mdg}, the parameter that controls the distance between an established relative pose $\bar{z}^{\alpha_i}_{\beta_j}$ and the next frame of α to be sent to β for geometrical verification. As we can see, τ_{mdg} can be used to significantly throttle the required bandwidth. This happens without a significant effect on the ATE. The dashed line indicates RelPose data transmission if descriptors are used for keypoint association rather than visual word indices. **(b)** Data transmission and ATE as the NetVLAD dimensionality is varied, with $\tau_{\text{mdg}} = 10$. Dashed lines indicate RelPose data transmission if keypoints are matched with descriptors, rather than words.

mitigated either by randomly assigning several clusters per robot or by training the clustering on a dataset that is very similar to the data at hand, in terms of the scope of visual appearances of keypoints. A detailed study of this problem is provided in [38].

Figure C.5b illustrates how the trajectories of the robots fuse as time progresses, and the accuracy of the resulting connected components. Seven merge events of connected components are apparent around seconds 15, 23, 32, 33, 39, 46 and 47, the remaining two merges occur at the beginning of the experiment and are not visible. At each merge event, there is a jump in connected component size and the corresponding lines in the plot fuse. In early merge events, the ATE grows as the drift of individual components is compounded with noise in relative pose measurements. This is most pronounced at the merge event around second 39. Later, as more relative measurements are added, loops are closed and the ATE decreases again, until it stabilizes around a value of 4 meters, well below 1% of the overall trajectory length.

Figure C.6a shows how increasing the minimum distance between geometric verifications τ_{mdg} can significantly reduce bandwidth requirements without negatively affecting accuracy on KITTI. Figure C.1a shows the place matches that remain and Figure C.1b data transmission over time with $\tau_{\text{mdg}} = 60m$.

Figure C.6b shows how the NetVLAD dimension affects the size of transmitted data, with $\tau_{\text{mdg}} = 10m$. Unsurprisingly, DVPR traffic is quasi-linear with respect to the NetVLAD dimension. However, using a too low dimension for NetVLAD leads to more false positives, which then have to be filtered out by the relative pose estimation. This results in a significant increase of RelPose data transmission. Conversely, increasing



(a) (b)

Figure C.7 – (a) Groundtruth trajectory for one of the sequences in the MIT Stata Center Dataset [62]. **(b)** Ten sub-trajectories of the MIT Stata Center Dataset *before* and *after the final loop closure*, estimated using our method. Each color represents an individual robots trajectory, place matches are marked with bold black lines.

NetVLAD dimensions beyond a certain point does not increase precision significantly. In Figure C.6b we can see that an efficient trade-off is reached with $D_{\text{NetVLAD}} > 100$.

We also evaluated our system on one of the sequences in MIT Stata Center Dataset [62]. Figure C.7a shows the groundtruth trajectory. Similar to the KITTI 00 dataset, we split the sequence into $|\boldsymbol{\Omega}| = 10$ parts to simulate trajectories recorded by different robots. Figure C.7b shows the sub-trajectories estimated using our method before and after the final loop closure. As we can see, the final estimate given by our method is not visually similar to the groundtruth estimate (Figure C.7a). The main reason for this mismatch is that the graph structure in the case of MIT Stata Center dataset is closer to a chain topology. Due to the chain topology, decentralized optimization requires more iterations to reach consensus among robot trajectories on either ends of the chain and therefore finds it difficult to converge to the centralized estimate. Finding sub-trajectories which follow certain graph topologies (chain vs grid) and are favorable for decentralized optimization can be a useful future extension.

C.6 Conclusion

We have presented a new integrated decentralized visual SLAM algorithm that is based on state-of-the-art components. The system has been characterized using publicly available datasets and we have explored how data transmission can be reduced to a minimum. Based on our results, we believe that future developments of decentralized visual SLAM will focus on one hand on even more data-efficient data association and on the other hand on more robust decentralized optimization.

D SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning

Reprinted, with permission, from:

T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. "SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning". In: *3D Vision (3DV)* (2019). DOI: 10.1109/3DV.2019.00072

SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning

Titus Cieslewski, Konstantinos G. Derpanis and Davide Scaramuzza

Abstract — A wide range of computer vision algorithms rely on identifying sparse interest points in images and establishing correspondences between them. However, only a subset of the initially identified interest points results in true correspondences (inliers). In this paper, we seek a detector that finds the minimum number of points that are likely to result in an application-dependent "sufficient" number of inliers k. To quantify this goal, we introduce the "k-succinctness" metric. Extracting a minimum number of interest points is attractive for many applications, because it can reduce computational load, memory, and data transmission. Alongside succinctness, we introduce an unsupervised training methodology for interest point detectors that is based on predicting the probability of a given pixel being an inlier. In comparison to previous learned detectors, our method requires the least amount of data pre-processing. Our detector and other state-of-the-art detectors are extensively evaluated with respect to succinctness on popular public datasets covering both indoor and outdoor scenes, and both wide and narrow baselines. In certain cases, our detector is able to obtain an equivalent amount of inliers with as little as 60% of the amount of points of other detectors. The code and trained networks are provided at https://github.com/uzh-rpg/sips2_open.

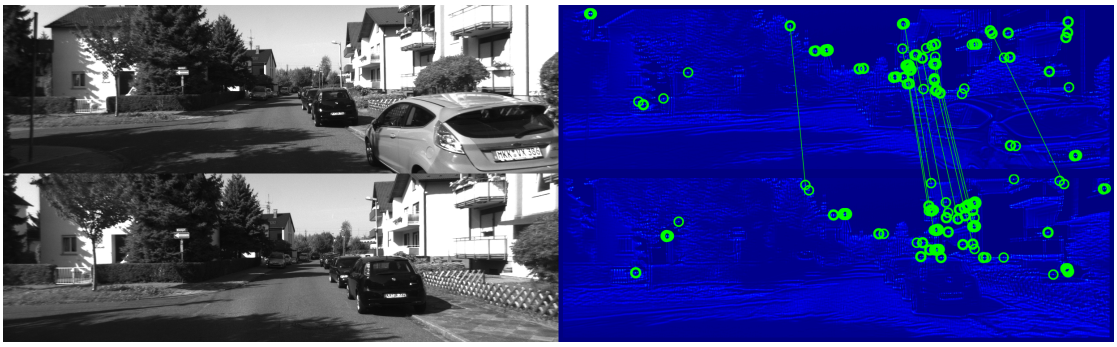


Figure D.1 – Not many inlier correspondences (green lines) are necessary to establish a good relative pose estimate between two image frames. In this paper, we look for a *succinct* interest point detector – a detector that results in sufficient inliers after extracting as little interest points (green circles) as possible. We introduce a corresponding metric, and a novel, unsupervised way to train interest point detectors. In blue, the score map of our detector. Local maxima may not be visible in print, see Figure D.3 for a close-up.

D.1 Introduction

A wide range of computer vision applications rely on establishing point correspondences between images. These correspondences can be established densely for every pixel [166], but it is often of interest to instead establish only a sparse set of correspondences. A sparse set makes many algorithms, such as visual odometry or bundle adjustment, far more tractable, both in terms of computation and memory. Furthermore, in multi-agent scenarios, using a sparse set of points to represent an agent's observation means that less data needs to be exchanged.

Typically, a set of interest points is first identified with a detector. However, only a fraction of these points will result in correct correspondences, as some of them will fail to be matched to points in the other image. Of the points that do match, some will further fail geometrical consistency, resulting in an even smaller set of inlier points. In this work, we seek to find the minimal set of points to extract in order to obtain a specific number of inliers, as shown in Figure D.1 for 10 inliers. While approaches exist to reduce an already given set of detected points [204, 217], we focus instead on directly detecting a minimal set.

To quantify this objective, we introduce the *k-succinctness* metric that builds on top of the widely used *matching score* [136]. The matching score indicates what fraction of a given set of points results in inliers, but it depends on the number of points extracted. In contrast, the k-succinctness metric answers the question of how many points need to be extracted with a given detector, and in the context of a given point matching algorithm, to yield a certain number k of inliers. The scalar k is a parameter of the metric, and depends on the application at hand. In our experiments, we consider P3P pose estimation [74] as an application. Given three 3D reference points in the object

frame and their corresponding 2D projections, P3P determines four possible solutions for the orientation and position (pose) of a calibrated perspective camera [85]. A fourth point is needed to disambiguate between the four solutions. So theoretically, only four inliers are sufficient to establish a pose estimate, but we show experimentally that instead, choosing $k > 4$ results in significantly better pose estimation performance.

Alongside the succinctness metric, we propose a new, simple training methodology for interest point detectors. The network is simply trained to predict, for every pixel, the probability of resulting in an inlier, after being subjected to non-maxima suppression and descriptor matching. Furthermore, we show for the first time how an interest point detector can be trained without the necessity of provided correspondence labels or pre-calculated depths, but merely with KLT tracking. The proposed detector and other state-of-the-art detectors are evaluated with respect to succinctness on three datasets representing both indoor and outdoor scenes, and image pairs with narrow and wide baseline.

To summarize, our contributions are as follows:

- The introduction of the *succinctness* metric, which evaluates descriptors by the minimum number of points that need to be extracted to achieve a certain number of inliers.
- A novel training methodology for interest point detectors that learns to predict the probability of a pixel being an inlier correspondence, in a self-consistent manner. The proposed method requires the least amount of training data pre-processing to date.
- An extensive evaluation of the proposed and state-of-the-art detectors with respect to succinctness on datasets covering both indoor and outdoor scenes, and wide and narrow baselines.

D.2 Related work

Traditional feature detectors Classically, interest points have been selected from distinctive image locations, that is, image locations that significantly differ from neighboring image locations. Whether an image location is distinctive can be determined explicitly [144] or using a first-order approximation, such as the Harris [84] or Shi-Tomasi [183] detector. Alternatively, distinctive interest points can be detected by convolving the image with a suitable kernel, such as the Laplacian of Gaussian (LoG), or its faster approximation, the Difference of Gaussians (DoG) kernel, prominently used in SIFT [124]. Other filter-based detectors have been proposed by [14, 130]. Finally, another method for choosing interest points is to identify image regions that explicitly resemble a sub-type of distinctive points, such as corners. Efficient ways to do this have been proposed by [81, 187, 200] and have found widespread popularity with [167] due to its

highly efficient implementation. An interesting challenge in interest point detection is ensuring that the detection is independent of scale and affine transformations. Scale invariance can be achieved using multi-scale detection [135], while affine invariance, in particular invariance to rotation is already given for most aforementioned detectors.

The problem with relying on distinctiveness as the interest point selection criterion is that it does not necessarily result in high repeatability, unless all distinctive points are selected. A more refined interest point selection criterion is needed if one wants to preserve repeatability when extracting less points. Some of the aforementioned works have attempted to derive such a criterion based on models or heuristics. An alternative and more promising method to deriving this criterion is data-driven, using machine learning. An early method based on a neural network has been proposed in [54], though it was limited to three layers due to the computational constraints of the time, and also only applied to the edge regions of an image. Subsequently, [165, 201, 206] used other types of regression functions to learn detectors.

CNN feature detectors With the recent popularity of convolutional neural networks (CNNs), CNNs have also been considered for interest point detection. They are particularly well-suited for this, as a per-pixel interest score can be directly calculated with a series of convolutional layers. LIFT [216] was the first to exploit CNNs for several components of the point correspondence process. It uses a separate network for detection, orientation estimation and description of interest points. This work is trained on patches provided by a SfM algorithm executed a priori. Lenc and Vedaldi [116] train an interest point detector using a "covariance constraint", which trains the response of a network to be invariant to viewpoint changes. Savinov et al. [175] generalize this constraint to enforce consistent ranking of responses of corresponding points between viewpoints. Superpoint [52] proposes a sophisticated training method that involves first fitting a network to detect labeled corners in a synthetic dataset, and later to train invariance on real images by warping real images. In contrast to previous work, detector and descriptor are trained jointly. Similarly, LF-Net [155] is trained on the full point correspondence algorithm. The loss acting on the LF-Net detector is similar to the covariance contraint of [116], augmented with a heuristic to favor responses that resemble narrow Gaussian kernels. Additionally, a loss requiring point descriptors to match also acts on the detector. The idea to explicitly favor sharp responses has also been presented in [220]. We find that in our approach, peakedness happens without the need for a specific loss, see Section D.5.5 and Figure D.3. Prior work relied on synthetic images, synthetic warping, ground truth homopgraphies or depth for their training. LF-Net [155], which has appeared in parallel to this work, uses COLMAP [182] to obtain depth estimates from potentially uncalibrated image sequences. In contrast, we show how to train our network using only KLT tracking performed on uncalibrated image sequences, requiring less training data pre-processing.

Interest point detector evaluation The traditional way to evaluate interest point detec-

tors is repeatability, as first introduced in [138]. Repeatability quantifies the overlap of, generally, detected affine regions in one image and the ground truth correspondence of the affine regions detected in the other image. Specifically, if a detector returns only points, the affine regions can be approximated as circles that are either uniform or scale-dependent. Repeatability was used to benchmark traditional detectors in [136, 137]. More recently, [107, 115] have used repeatability to benchmark more modern detectors. Repeatability, however, has two issues as a metric. Firstly, performance under repeatability depends on the number of extracted points. In the most extreme case, when every point is "extracted as interest point", repeatability technically reaches its maximum value. Lenc and Vedaldi [115] address this issue to some extent by evaluating repeatability at different specific point counts. Secondly, repeatability does not take descriptor and matching performance into account. Consider a detector that detects all corners of a large checkerboard. While this detector will be very repeatable, it is likely that its points will be useless, as typical feature matchers will not be able to distinguish them. The latter problem is addressed with the matching score [136]. Unlike in repeatability, affine region overlap is only considered if the corresponding interest points have also been matched. However, also the matching score performance depends on the point count. In contrast, succinctness does not depend on the point count. Rather, it reflects the point count that is necessary to obtain k inliers. While this shifts to a dependence on k, k should typically be given by the application.

A posteriori compression An alternative approach to finding minimal sets of interest points would be to first extract and describe a large set of interest points, and then reduce them to a minimal set sufficient for localization in post-processing. Several approaches exist where this reduction is based on whether specific features are consistently detected in multiple images observing the same scene [28, 57, 120, 157, 204]. Learning to predict from single images whether already-extracted features are likely to be matched has also been proposed [58, 86, 217], but again as a filtering step of already-extracted features. Instead, we directly identify succinct interest points in the detector.

D.3 Succinctness

Succinctness answers the question: *how many interest points n_k need to be extracted by a detector to achieve k inlier correspondences?* Hence, succinctness will not only depend on the detector, but also the descriptor, matching, potentially other parts of a feature matching pipeline, and whatever method is used to distinguish inlier correspondences from outlier correspondences. Specifically, while this distinction can be made based on ground truth information, we also allow for it to be achieved using another method, such as geometric verification using P3P [74] with RANSAC [64]. Furthermore, just as for other metrics, succinctness will depend on the data on which it is evaluated. In particular, we need a mechanisms to aggregate n_k across multiple image pairs, since

n_k will be different for every evaluated image pair. For this, we could use histograms, but histograms introduce a dependency on the selected bins. Instead, we propose to use a cumulative aggregation. Specifically, we propose the *succinctness curve*: the x-axis represents a specific amount of extracted points n, and the value on the y axis is $s(n)$, the fraction of pairs whose n_k is lower than n. $s(n)$ can also be thought of as the fraction of image pairs that achieve at least k inliers if n points are extracted – we assume that inlier count increases monotonically with the amount of extracted points. An example of such curves can be seen in Figure D.5.

To quantify succinctness over a whole dataset in a single number, we use the area under this curve, up to a specific $n_{\max}$. Formally:

$$\text{AUC-}n_{\max} = \frac{1}{n_{\max}} \int_{n=0}^{n_{\max}} s(n) dn. \tag{D.1}$$

This number assumes values between 0 and 1. For example, if AUC-$n_{\max} = 0.8$, this means that either (80% of image pairs obtained k inliers with 0 extracted points [1] and 20% failed to obtain k inliers with $n_{\max}$ interest points) or (all image pairs obtained k inliers with $0.2 \cdot n_{\max}$ interest points), or something in between these two extremes.

To rapidly determine n_k for a given image pair, we perform only one detection step in which we extract $n_{\max}$ interest points in both images. We then perform binary search, where the features of both images are cropped to the n features with the highest score, and the rest of the feature matching system is executed on those features, until we find the smallest n that achieves at least k inliers.

D.4 System Overview

Similar to recent learned interest point detectors, we use a fully convolutional neural network (Section D.5.7) to predict a per-pixel score map S_i from an image I_i. Figure D.2 illustrates our system used for interest point detection, relative pose estimation, and network training. Non-maxima suppression with a radius of $5px$ is used to find the n highest-scoring points $C = \{\vec{x}\}$ in S. This constitutes the forward pass of the system for a single image. For training, an inlier determination module (Section D.5.3) processes the interest points of two images and determines which of them are inliers. The loss (Section D.5.2) is sparsely applied at the interest point locations. The inlier determination module extracts and matches SURF descriptors [14] and uses KLT tracking [125] in an image sequence to determine inliers. The choice of SURF as descriptor is somewhat arbitrary – as we do not provide our own descriptor, we need to use an existing one, and SURF proved both practical to adopt and sufficiently well-performing. The presented method can easily be used with any other descriptor. Inlier

[1] This is of course an impossible example, but serves for illustration.

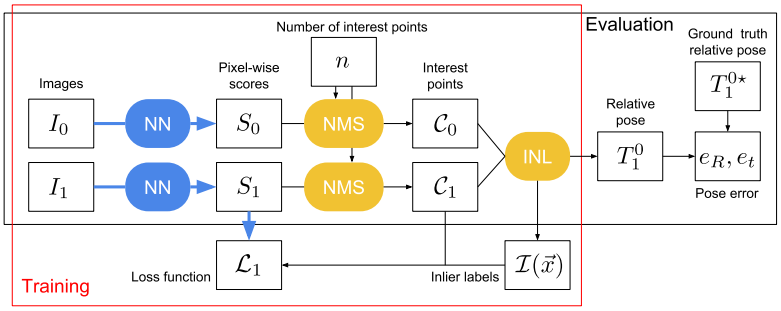


Figure D.2 – The system we propose for training and evaluation. The neural network (NN) calculates a pixel-wise score map S_i for each image I_i. The n highest scored interest points $C_i = \{\vec{x}\}$ are selected, using non-maxima suppression (NMS). During training, an inlier determination module (INL) matches the points and determines which matches are inliers $\mathcal{I}(\vec{x})$. This information, together with interest point locations is used to formulate our loss $\mathcal{L}$ on patches centered at the interest point locations, which is back-propagated during training (blue arrows). During testing and deployment, the inlier determination module is replaced with a pose estimator that uses P3P and RANSAC to find a relative pose T_1^0 that can be compared to ground truth for accuracy evaluation.

determination is replaced with P3P [74] with RANSAC [64] in the succinctness and pose estimation evaluation as illustrated on the right side of Figure D.2. In RANSAC, we use the maximum inlier set after a fixed amount of 3000 iterations. This corresponds to finding the true solution among 10% inliers with a probability of 95%.

D.5 Training Methodology

We want the selected points to have the highest probability of being inliers. To that end, we train our neural network to predict this probability. Selecting the points with highest scores thus implies selecting the points that will most likely result in inlier correspondences.

D.5.1 Inlierness Probability Model

We model point $\vec{x}$ being an inlier $\mathcal{I}(\vec{x})$ as a stochastic process conditioned on the patch around $\vec{x}$, $\mathcal{P}(\vec{x})$. In our model, this process results in $\mathcal{I}(\vec{x})$ with probability

$$P = P(\mathcal{I}(\vec{x})|\mathcal{P}(\vec{x})), \tag{D.2}$$

and in $\neg\mathcal{I}(\vec{x})$ with probability $1 - P$. This is an approximation, because $P(\mathcal{I}(\vec{x}))$ of course depends on more than just $\mathcal{P}(\vec{x})$, but, as we will show empirically, this approximation is accurate in many scenarios, and even transfers well between different environments.

Among the unmodeled conditions, there are ones that are not under our control, such as the relative pose between the cameras, and ones that are under our control, such as the interest point selection and matching method $\mathcal{M}$. For the relative pose between cameras, we assume that the two cameras point at the same scene while providing many different examples of relative poses during training to train the network in a viewpoint-invariant way. As for the method $\mathcal{M}$, we always use the method presented in Section D.4, and so P is implicitly conditioned on $\mathcal{M}$. Note that this introduces a circular dependency, as the interest point selection itself depends on the predicition of P. This means that there is no "ground truth" for P – our system will start in an arbitrary state and if trained well, it will converge to a self-consistent state.

D.5.2 Loss function

P can be learned using the cross-entropy loss:

$$\ell_{\mathcal{I}(\vec{x})}(P) = -\log(P), \quad \ell_{\neg\mathcal{I}(\vec{x})}(P) = -\log(1-P), \tag{D.3}$$

where $\ell_{\mathcal{I}(\vec{x})}(P)$ is the loss applied if $\mathcal{I}(\vec{x})$ and $\ell_{\neg\mathcal{I}(\vec{x})}(P)$ the loss applied otherwise. To provide the reader with an intuition of why this is the case, we briefly re-iterate the corresponding theory. In principle, given $\vec{x}$, the loss should move P up if it is under-estimated, suppress it if it is over-estimated, and maintain it if it is correct. However, according to our model, during training only random samples $\mathcal{I}(\vec{x})$ and $\neg\mathcal{I}(\vec{x})$ will be provided. Hence, the best thing that we can do is to make sure that this condition holds over a large quantity m of training steps involving a particular $\vec{x}$. In m such training steps, the expected amount of steps where $\mathcal{I}(\vec{x})$ holds is Pm and the expected amount of steps where $\neg\mathcal{I}(\vec{x})$ holds is $(1-P)m$. Thus, for these contradicting training steps to cancel each other out at the true P in the long term, the gradients of the loss should satisfy:

$$P\frac{\partial}{\partial P}\ell_{\mathcal{I}(\vec{x})}(P) + (1-P)\frac{\partial}{\partial P}\ell_{\neg\mathcal{I}(\vec{x})}(P) = 0. \tag{D.4}$$

Similarly, the first term of (D.4) should be larger than the second one if P is under-estimated and smaller if P is over-estimated. These conditions are satisfied by the cross-entropy loss (D.3).

D.5.3 Unsupervised inlier determination

Previous methods used for training interest point detectors either involved labeled data [52], synthetic warping of planar scenes [52, 116], or 3D scenes annotated with ground truth depth [155, 175], or ground truth correspondence labels provided in another manner. LF-Net [155], which has appeared in parallel to this work, uses COLMAP [182] to provide dense depth estimates, which allows training from uncalibrated image

sequences just as our method. Instead, we propose to use Lucas-Kanade tracking (KLT) [125] to determine correspondences. This has the benefit of lower computation and storage requirements.

Given a pair of forward passes from two images, and the intermediate images in the sequence, the interest points are tracked image-to-image, through all intermediate images, using KLT. This can be applied to wide baseline image pairs, provided the interest points can be tracked through the intermeidate frames. To discard poorly tracked points, each image-to-image tracking is verified with a bidirectional check, where a track is only considered successful if tracking forwards and backwards results in an error that is below one pixel. Using just this information would allow us to train a repeatable detector, but we also want to take the used descriptor in consideration during training. Hence, an interest point is only considered an inlier if the correspondence obtained from KLT is within 3pixels of the one obtained from descriptor matching.

Of course, this correspondence method could potentially be applied in conjunction with other losses, such as [155]. Similarly, our training method is not restricted to using this method for inlier determination. For example, for calibrated stereo image sequences, we can find the depths of the interest points in one of the two images (of the training pair, not the stereo pair) using epipolar stereo matching. Then, the depths can be used to perform P3P RANSAC with the interest points of the other images, resulting in inlier labels from P3P RANSAC, a method which even more closely mirrors our evaluation. Alternatively to these methods, any method, or labels, providing ground truth correspondence can be used in a similar fashion to inlier determination using KLT tracking. In particular, note that KLT tracking is not suitable to label correspondences when intending to train illumination invariance. Illumination invariance can be trained using images without changes in viewpoint [115] or images where true pose and depths are known [1].

D.5.4 Sparse loss application

Similar to recent work, we only apply our loss at sparse locations. Specifically, the loss is only applied at the extracted interest points. During training, 500 interest points are extracted. Ono et al. [155] argue that a loss should be applied at all correspondences to reach convergence, but we find that this is not the case in our training framework. The fact that we do not also need to train a descriptor, but use an already existing one, might facilitate convergence in our case.

As a consequence of the application of a sparse loss, it is possible that the probability prediction is only properly trained for the higher probability values, which are the values at which interest points are extracted. Since we are only interested in inlierness probability prediction for interest points, however, this is not considered an issue.

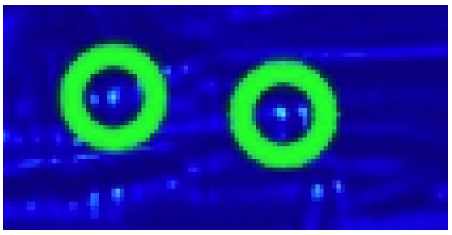


Figure D.3 – Close-up of our CNN response. Green circles indicate interest points selected in the image.

D.5.5 Avoiding degenerate relative poses

To achieve good performance in pose estimation, features need to be well-distributed in the image. There are generally two approaches to ensure such a distribution. Firstly, non-maxima suppression (NMS), which ensures that there is a minimum distance between two selected interest points. Secondly, binning, where a grid is overlayed over the image, and a specific amount of interest points is extracted in each bin. While binning has proven to be useful in several applications [67, 149], it seems to be at odds with the idea of succinctness, where repeatability at low point counts is key: if a certain point crosses a bin boundary between the two images to be matched, it is very likely that it will only be selected in one of the images, as it potentially has to compete for the highest score with completely different interest points between the bins. Instead, NMS seems to be a less intrusive way to distribute the points in the image. We use a NMS radius of $5px$. While this seems to be a relatively low radius, we find that applying it during training results in extremely peaked responses, without any further response peakedness loss as in [155, 220]. As can be seen in Figure D.3, a local maximum is typically contained within 3×3 pixels, often with a single pixel having the clearly highest response. Plenty of other examples can be seen by zooming into Figure D.1 and renderings provided in the supplementary material.

D.5.6 Image pair selection

While some training datasets like HPatches [13] provide pre-selected image pairs, we are also interested in training from uncalibrated image sequences. Here, two images need to be from the same scene for training and evaluation to be meaningful. This poses a problem, as we want to avoid manual annotation or additional labels such as ground truth pose. LF-Net [155] addresses the problem of pair selection in sequences by sampling pairs at specific time differences. However, this would not guarantee scene overlap between the two images on general image sequences. We instead determine scene overlap using KLT. For every sequence, a preprocessing step is executed where points are sampled densely in the full image, at some distance to each other, to remain tractable. These points are then tracked using KLT, and new points are detected to

keep the image densely sampled. Then, based on the tracks, we track for each image the fraction of tracked points in subsequent images ("overlap"), until no points from the original image are tracked. This information is stored and used during random pair selection to select two images such that the second image has a minimum overlap o with the first image.

D.5.7 Neural Network Architecture

Our network is a fully convolutional network with 3×3 kernels and unit strides throughout, and leaky ReLU activations, except at the final layer. The final layer activation is a sigmoid activation to constrain the output to $]0, 1[$. The only two hyperparameters we consider are the depth (layer count) and width of the network, d and w. We use $\frac{w}{2}$ channels in the first half layers and w in the second half, except for the final channel which outputs a single channel. We use two models for evaluation: firstly, a multi-scale model that calculates score maps at five different scales and uses 12 layers and a width of $w = 256$. The scale factor is $\sqrt{2}$ and all scales share the same weights. Secondly, a single-scale model that uses 10 layers and a width of $w = 128$.

D.6 Experiments

Previous work [107] has shown that learned interest point detectors tend to perform poorly on real-world datasets. This motivates our first set of evaluations, which is on robotics datasets. We then also evaluate on the wide-baseline HPatches dataset [115]. Unlike prior benchmarks, however, we evaluate succinctness and pose estimation accuracy. Finally, we evaluate the accuracy of the inlierness probability prediction.

D.6.1 Compared detectors

We evaluate succinctness for SIFT [124], SURF [14], LF-Net [155] (which has appeared in parallel to this work), SuperPoint [52], and our proposed network. For SIFT and SURF, we use the OpenCV implementations, while for LF-Net and Superpoint, we use the publicly available code and pre-trained weights. Unfortunately, LFNet does not provide the scores of the detected points, preventing us to do the binary search for n (Section D.3), so our comparison is based on extracting three specific interest point counts $n = (50, 100, 150)$.

D.6.2 Succinctness on Robotics Datasets

To evaluate succinctness on robotics datasets, we consider the outdoor/autonomous driving dataset KITTI [76], and the indoor/drone dataset EuRoC [24]. While our system

supports multi-scale extraction, experiments on a validation sequence of KITTI show that multi-scale does not improve performance on these rather narrow-baseline datasets, so we only train the system at a single scale. For training the network used in these experiments, we only use image pairs from the TUM mono [60] and Robotcars [129] datasets. For testing, we use sequence `00` of the KITTI dataset and `V1_01` of EuRoC.

From each testing sequence, we randomly select 100 image pairs according to the method outlined in Section D.5.6, with the minimum overlap o set to 0.5. These pairs are consistent for all evaluated methods. The indices of the image pairs and visualized samples are provided in the supplementary material.

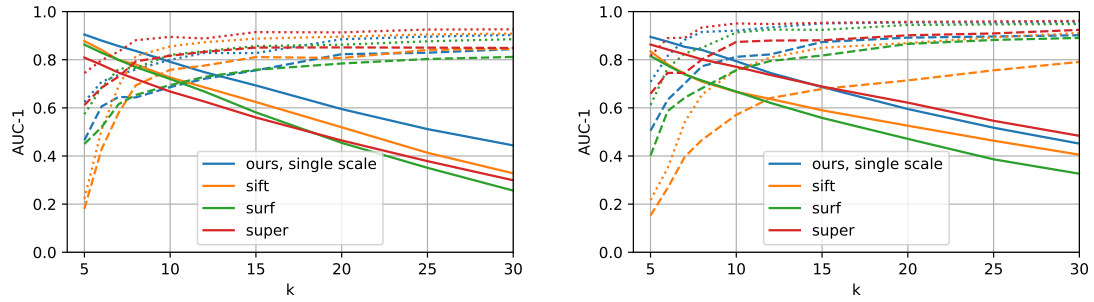
The evaluation datasets come with ground truth poses, which allows us to evaluate pose estimation error as a function of the selected minimum inlier count k. We obtain relative pose estimates using P3P [74] with RANSAC [64], where the 3D location of points of one of the images is obtained from stereo matching. This can be done as both evaluation datasets were recorded with stereo cameras. Position and rotation errors are considered separately, but their distribution over the evaluation set is summarized as one number in a way similar to succinctness. Consider a plot where the x axis represents the error and the y axis represents the fraction of image pairs that result in an error that is lower than x. Then, we use the AUC-1 (D.1) (1 meter for translation and 1 degree for rotation) to represent the distribution. For example, if the AUC-1 is 0.8, then either (80% of the pairs have a perfect estimate and 20% fail RANSAC) or (all pairs have an error of 0.2), or something in-between these extremes. For position error we use Euclidean distance in meters and for rotation error we use geodesic distance (angle of angle-axis representation) in degrees.

In Figure D.4 we inspect how pose estimation quality changes as a function of required inlier count k, and what the corresponding succinctness is. As we increase k to 10 in either dataset, significant improvements in pose accuracy can be observed. Beyond 10, accuracy further improves, but at a slower pace. In terms of succinctness, out method outperforms baselines for all evaluated k on KITTI and for $k < 15$ on EuRoC, beyond which SuperPoint slightly outperforms our method.

In Figure D.5, we show succinctness curves for the specific value of $k = 10$. In contrast to Figure D.4, this allows us to see some more nuance, such as the fact that in the EuRoC dataset, our method has the highest success rate at 75 interest points or less, but at higher interest point counts, SuperPoint has higher success rates on EuRoC.

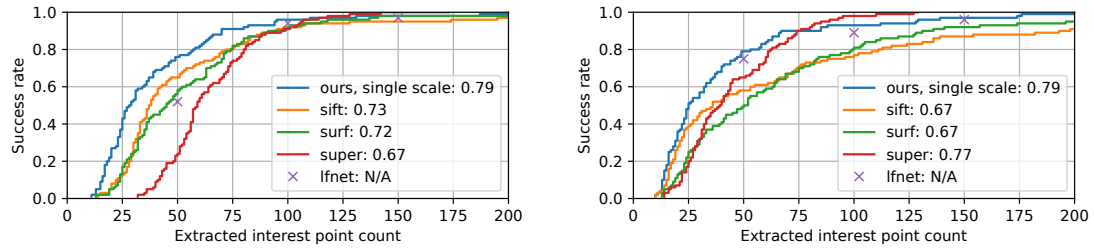
D.6.3 Pose accuracy with fixed point count

Informed by these results, we consider a practical case, where the amount of extracted interest points is fixed to 50, which, according to Figure D.5, should result in 80% of testing samples having at least 10 inliers. Figure D.6 shows the distribution of



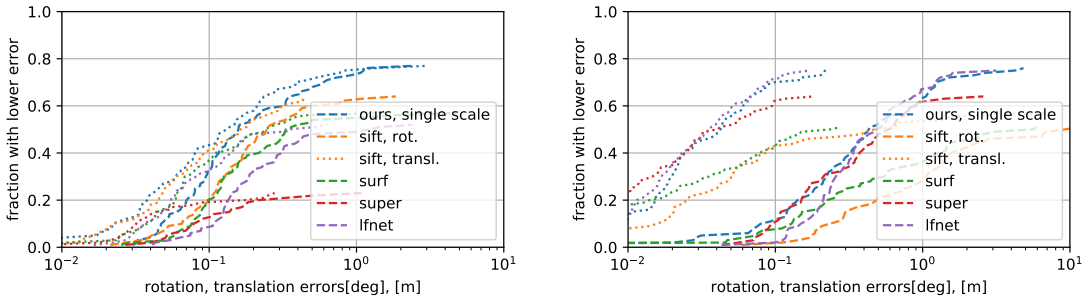
(a) KITTI `00` **(b)** EuRoC `V1_01`

Figure D.4 – The role of the required inlier count in pose estimation quality, evaluated on KITTI `00` and EuRoC `V1_01` (EuRoC uses AUC-5 for the rotation error). As we increase the required inlier count k, rotation (dashed) and translation (dotted) precision increases, but succinctness (solid) drops.



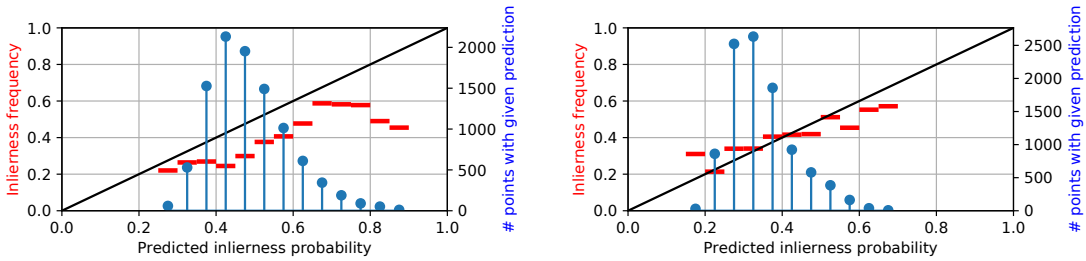
(a) KITTI `00` **(b)** EuRoC `V1_01`

Figure D.5 – Succinctness and AUC-200 values on KITTI and EuRoC with $k = 10$. At low point counts, the least amount of points need to be extracted with our detector. AUC could not be evaluated for LF-NET, as only $n \in \{50, 100, 150\}$ were sampled.



(a) KITTI 00 **(b)** EuRoC V1_01

Figure D.6 – Pose estimation errors on KITTI and EuRoC when extracting 50 points and using the best inlier set with at least 10 inliers after 3000 RANSAC iterations. The dashed lines stand for rotation errors and the dotted lines for translation errors. The curves end with the largest error obtained with 10 inliers.



(a) KITTI 00 **(b)** EuRoC V1_01

Figure D.7 – Actual inlierness frequency versus predicted inlierness probability after extraction, matching, and P3P + RANSAC verification of 50 interest points. Inlierness frequency is calculated within bins defined by the predicted probability, bin ranges are indicated by the width of the red lines, and their cardinality by the blue lines. The black line indicates perfect inlierness probability prediction.

rotation and translation errors of the relative pose estimates obtained when using the best inlier set with at least 10 inliers after 3000 RANSAC iterations. As we can see, our method generally performs best at this low interest point number, with the other learned detectors performing similarly on EuRoC, but worse on KITTI.

D.6.4 Inlierness probability prediction

We have trained our network to predict the probability of a pixel resulting in an inlier. Besides resulting in a succinct interest point detector, Figure D.7 shows that the ability to predict this probability generalizes well, even if tested with a smaller number of extracted points, here 50 instead of 500. On KITTI, the predicted probability is somewhat overconfident, especially at the higher end. However, only few points

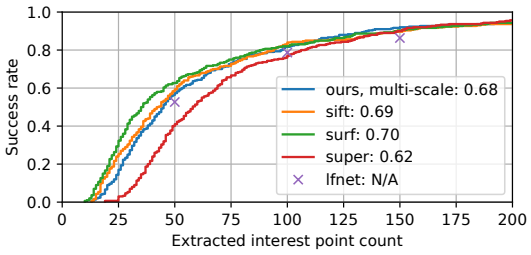


Figure D.8 – Succinctness and AUC-200 on HPatches with $k = 10$. Wide baselines seem to be challenging for learned interest point detectors, in terms of succinctness. Among the learned detectors, our multi-scale detector trained on the HPatches training set performs best, almost as good as SIFT and SURF. AUC could not be evaluated for LF-NET, as only $n \in \{50, 100, 150\}$ were sampled.

have a predicted probability beyond 75%. Consequently, few samples were available to calculate the inlier frequency that the probability prediction is compared to.

D.6.5 Succinctness on Wide Baseline

While the previous outdoor and indoor datasets represent scenes that are likely encountered in robotic applications, such as autonomous driving and autonomous drone flight, they exhibit a relatively narrow baseline. To evaluate our method on more extreme viewpoint changes, we consider the viewpoint sets of HPatches [13, 115], with the training and testing split as suggested by the authors. For HPatches, we find that we need to train multi-scale, as several pairs in our validation sequence exhibit strong scale changes. The network is trained on the HPatches training sets only. For performance, we downscale the HPatches images such that the largest dimension is at most $1000px$, and convert the images to grayscale. The results are shown in Figure D.8. Consistent with the observation of [115], SIFT and SURF do very well here. When trained on wide baselines, our network can achieve similar performance, as does LF-NET. Here, Superpoint performs worst, but remarkably well considering that multi-scale is not explicitly expressed in its architecture.

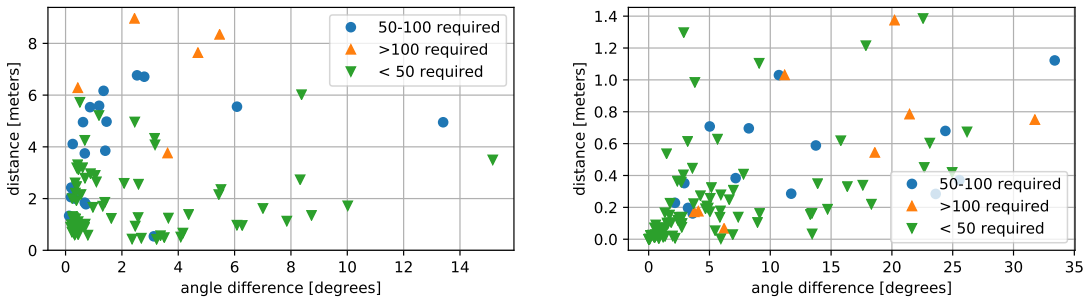
D.7 Conclusion

In this work, we set out to find a detector that has good performance at low numbers of extracted interest points. Extracting a minimum number of interest points is attractive for many applications, because it can reduce computational load, memory, and data transmission. To quantify this goal, we have introduced the *succinctness* metric, which measures the miminum required number of interest points to achieve a certain number k of inliers. In our experiments, the choice of k has been driven by relative pose estimation

using P3P and RANSAC. We have evaluated several state-of-the-art detectors, along with a novel detector which can be trained with the least amount of training data labeling and pre-processing to date. Our detector consistently exhibits top performance in terms of succinctness. Additionally, it boasts an interpretable score which can be used to predict the probability that an interest point will yield an inlier.

Acknowledgments

This work was supported by the National Centre of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant. The Titan Xp used for this research was donated by the NVIDIA Corporation. Konstantinos G. Derpanis is supported by a Canadian NSERC Discovery grant. He contributed to this work in his personal capacity as an Associate Professor at Ryerson University.



(a) KITTI `00` **(b)** EuRoC `V1_01`

Figure D.9 – True pose difference plots for KITTI and EuRoC. Different markers indicate different numbers of interest points that are required using our method to obtain 10 inliers.

D.8 Supplementary Material

D.8.1 CSV files

As written in the paper, a detailed CSV file is attached for each of the two robotic testing datasets. This file contains the columns:

- `name`: sequence name and the indices of the two images that form the pair.
- `dR, dt`: ground truth difference in rotation and translation between the two image frames.
- `nmin`: number of points that needs to be extracted to obtain 10 inliers.
- `eR, et`: rotation and translation *error* of the relative pose estimate.

Both `dR` and `eR` are measured with the geodesic distance, in degrees, which corresponds to the angle of the angle-axis representation of the relative rotation.

D.8.2 True pose difference plots

To provide an intuition for the robotic testing sets, we plot the distribution of true relative poses in Figure D.9. Note the difference between the two datasets: KITTI exhibits larger translation distances, while EuRoC exhibits larger orientation differences. Recall that given the first image, the second image is randomly sampled among subsequent images with a scene overlap of at least 50%.

Besides showing the distribution, this plot also indicates how many interest points need to be extracted to obtain 10 inliers for each pair. While generally, more points are required at larger pose differences, there are no very clear boundaries — relative pose is not a strong predictor of succinctness for an image pair.

D.8.3 Match renderings

Several match renderings akin to Figure 1 in the paper are attached. They show the input image, score output and the inliers when detecting 50 interest points. Probability prediction is indicated with the thickness of the circle, and inlier matches are connected with lines. The plots demonstrate the peakedness of our response and validates that our method works as expected.

D.8.4 Sequence videos

We visualize the stability of our interest points as well as the correlation of that stability with predicted inlierness probability with a video composed of the score maps extracted over the full testing sequences.

To highlight succinctness, only 50 points are extracted in each frame. Note that the interest points are independently extracted in each frame. Still, many of the interest points behave as if they were directly tracked, especially the ones with high scores, as indicated by thicker circles. This corroborates our results about succinctness — few points are needed to obtain sufficient inliers. Furthermore, the natural peakedness of our score response can be easily discerned in the videos.

E Matching Features without Descriptors: Implicitly Matched Interest Points

Reprinted, with permission, from:

T. Cieslewski, M. Bloesch, and D. Scaramuzza. "Matching Features without Descriptors: Implicitly Matched Interest Points". In: *British Mach. Vis. Conf. (BMVC)*. 2019

Matching Features without Descriptors: Implicitly Matched Interest Points

Titus Cieslewski, Michael Bloesch and Davide Scaramuzza

Abstract — The extraction and matching of interest points is a prerequisite for many geometric computer vision problems. Traditionally, matching has been achieved by assigning descriptors to interest points and matching points that have similar descriptors. In this paper, we propose a method by which interest points are instead already implicitly matched at detection time. With this, descriptors do not need to be calculated, stored, communicated, or matched any more. This is achieved by a convolutional neural network with multiple output channels and can be thought of as a collection of a variety of detectors, each specialised to specific visual features. This paper describes how to design and train such a network in a way that results in successful relative pose estimation performance despite the limitation on interest point count. While the overall matching score is slightly lower than with traditional methods, the approach is descriptor free and thus enables localization systems with a significantly smaller memory footprint and multi-agent localization systems with lower bandwidth requirements. The network also outputs the confidence for a specific interest point resulting in a valid match. We evaluate performance relative to state-of-the-art alternatives.

Multimedia Material

Source code and data for this work are available at https://github.com/uzh-rpg/imips_open.

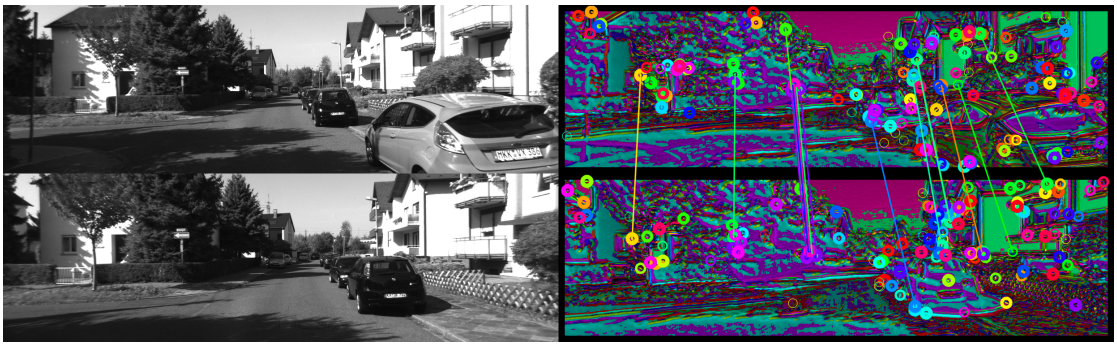


Figure E.1 – We propose a CNN interest point detector which provides implicitly matched interest points: descriptors are not needed for matching. This image illustrates the output of the network. Hue indicates which channel has the strongest response for a given pixel, and brightness indicates that response. Circles indicate the interest points, which are the global maxima of each channel. Lines indicate inlier matches after P3P localization.

E.1 Introduction

Many applications of computer vision, such as structure from motion and visual localization, rely on the generation of point correspondences between images. Correspondences can be found densely [63, 88, 166], where a correspondence is sought for every pixel, or with sparse feature matching, where correspondences are only established for a few distinctive points in the images. While dense correspondences capture more information, it is often of interest to establish them only sparsely. Sparse correspondences make algorithms like visual odometry or bundle adjustment far more tractable, both in terms of computation and memory.

Sparse feature matching used to be solved with hand-crafted descriptors [14, 124, 170], but has more recently been solved using learned descriptors [52, 155, 216]. In this paper, we propose a novel approach that exploits convolutional neural networks (CNNs) in a new way. Traditionally, features are matched by first detecting a set of interest points, then combining these points with descriptors that locally describe their surroundings, to form visual features. Subsequently, correspondences between images are formed by matching the features with the most similar descriptors. This approach, which has been developed with hand-crafted methods, has been directly adopted in newer methods involving CNNs. As a result, different CNNs have been used for the different algorithms involved in this pipeline, such as interest point detection, orientation estimation, and descriptor extraction.

We propose a method that uses only a single, convolution-only neural network that subsumes all of these algorithms. This network can be thought of as an extended interest point detector, but instead of outputting a single channel that allows the selection of interest points using non-maximum suppression, it outputs multiple channels (see Figure E.1). For each channel only the global maximum is considered an interest point,

and the network is trained in such a way that the points extracted by the same channel from different viewpoints are correspondences. As with traditional feature matching, geometric verification is required to reject outlier correspondences. The network can alternatively be thought of as a dense point descriptor, but instead of expressing descriptors along the channel axis of the output tensor, each channel represents the response to a function defined in the descriptor space.

An important benefit of our method is that descriptors do not need to be calculated, stored, communicated or matched any more. Previously, the minimal representation of an observation for relative pose estimation consisted of point coordinates and their associated descriptors. With our method, the same observation can be represented with as little as the point coordinates (3 bytes for up to 4096×4096 images), ordered consistently with the channel order.

We provide an evaluative comparison of our method with other state-of-the-art methods on indoor, outdoor and wide-baseline datasets. As our evaluations show, it is a viable alternative to methods involving explicit descriptors particularly with narrow baselines, achieving a similar pose estimation performance.

E.2 Related Work

Modern feature matching can be best understood by considering the sub-problems and their historical context. Calculating dense correspondences for images used to be prohibitive, so instead early work concentrated on finding sparse sets of points that could be used for correspondence generation. In a first interest point detector, [144] identified points which are explicitly *distinct* from the points that surround them—thus increasing the likelihood to be unambiguously matched in another image. Subsequently, faster approximations for distinctiveness have been found, whether using first-order approximations [84, 183], or convolutional filters [14, 124]. Alternatively, a detector can explicitly target a subset of distinctive points, such as corners and dots [167]. All of these methods calculate a response for every pixel in the image, and the n pixels with the largest response are selected as interest points. This process typically involves non-maximum suppression in order to prevent directly neighboring points from being selected.

Once a set of interest points is extracted in the images, they need to be matched between each other to establish correspondences. One can match points based directly on the surrounding image patches, but this is fragile to slight changes in illumination and viewpoint. Instead, descriptors can be used, which are functions of patches and whose output is typically lower-dimensional, but invariant to some amount of illumination and viewpoint change, yet still distinctive enough to differ between the different points extracted in one image. A popular class of traditional descriptors is histograms of

gradients (HoG) [124]. Another example are binary descriptors, which are particularly efficient to calculate [27, 117, 170].

Most descriptors, however, are still sensitive to large affine transformations such as changes in scale and orientation. Consequently, modern feature matchers use multi-scale detection [117, 135], and orientation estimation [124, 170]. A wide variety of traditional feature matching systems comprising these components exist, see the survey in [147].

Recent success of convolutional neural networks (CNNs), however, has led the community to revisit these systems and replacing their components with CNN-based methods. For detection, the traditionally handcrafted "featureness" responses can directly be replaced with a fully convolutional neural network. Rather than just imitating traditional interest point detectors, CNN-based detectors can be trained to be invariant across different viewpoints [116], to present consistent ranking in the images in which they are extracted [175] and to provide particularly sharp and thus unambiguous responses [220]. A majority of these methods is compared in [115]. CNNs are furthermore proven function approximators for image patches and thus well suited for descriptor calculations. The output channels of a CNN can simply be interpreted as the coefficients of a descriptor [82, 101, 123, 141, 185, 210, 219]. A comparison of traditional and learned descriptors is provided in [181]. While the results of comparing CNN-based methods to traditional methods do not yet suggest absolute superiority of CNN-based methods [181], an advantage of CNN-based methods is that they are malleable: firstly, they can adapt to and learn from new data: Consider an application where the type of environment is known beforehand—CNN-based methods can be trained to work particularly well on that particular type of environment. Secondly, they can adapt to or be trained together with other components of a larger system.

Two recent systems that fully integrate CNN-based methods and do this kind of joint training are LF-Net [155] and SuperPoint [52]. LF-Net [155] builds on top of a previous method by the same authors, LIFT [216], the first such system, in which the method was trained on a set of pre-extracted patches. [155], instead, is trained in a self-supervised and more unconstrained manner, only requiring an image sequence with ground truth depths and poses. Like [216], [155] uses separate CNNs for multi-scale interest point detection, feature orientation estimation and feature description. In contrast, SuperPoint [52] only contains an interest point detector and a feature descriptor network, both of which are sharing several encoder layers. It also does not explicitly express multi-scale detection, but rather trains multi-scale detection implicitly. It is first pre-trained on labeled synthetic images, then fine tuned on artificially warped real images. Both [155] and [52] still consider the traditional components of feature detection as separate functional units, even if the whole system is trained end-to-end.

In contrast, we offer a novel approach in which all components are subsumed into a

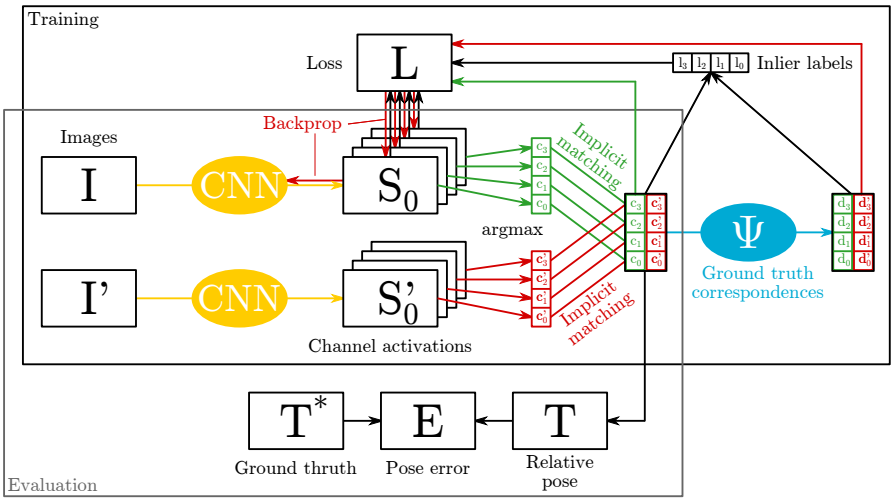


Figure E.2 – Overview of our approach. Given an image, a CNN computes n activations, the argmax of which are considered the n interest points. The interest points from two different images are matched by channel. During training, these matches are labeled as *inliers*, *outliers*, or *unassigned* using ground truth correspondences. Our loss promotes inliers, penalizes outliers, and suppresses redundancy. For evaluation, the correspondences are used to compute a relative pose between two images which is compared against ground truth.

single network. Beside having the benefit that all components can be jointly trained (from scratch) and thus tailored to one another, we also get rid of explicit descriptors. Instead, interest points are implicitly matched by the CNN output channel from which they originate. In practice, this results in memory, computation and potentially data transmission savings, as descriptors do not need to be stored, matched or communicated any more.

There have been some previous attempts to significantly reduce the amount of data associated with descriptors. In [192], the authors replace descriptors with word identifiers of the corresponding visual word in a Bag-of-Words visual vocabulary [186]. This can be used jointly with Bag-of-Words place recognition in order to facilitate multi-agent relative pose estimation with minimal data exchange. In [127], the authors propose highly compressed maps for visual-inertial localization in which binary descriptors are projected down to as little as one byte. In contrast, our approach circumvents the use of any explicit descriptor, by implicitly embedding a form of descriptor in the learned detection algorithm itself.

E.3 System Overview

In analogy to other state-of-the-art approaches for interest point detection, we employ a neural network to predict a per-pixel response from an input image. But instead of only predicting a single output score for every pixel, we predict n different activations, see

Figure E.2. The neural network consists of 14 layers of 3×3 convolutions with stride 1 and leaky ReLU activations, except for the final activation, which is a sigmoid. The first half layers output 64 channels, the second half 128. From each final output channel i, we extract the argmax as i-th interest point with coordinates $\boldsymbol{c}_i$. The key concept is that we then *implicitly* match the interest point from the same output channel across multiple frames. This has the advantage of inherently solving the data association problem, without the need to use descriptors explicitly. Formally, point $\boldsymbol{c}_i$ from image I is matched with point $\boldsymbol{c}'_i$ from image I'. At test time, a relative pose between both images can then be computed based on the corresponding interest point coordinates.

During training, an inlier determination module (Section E.3.1) processes the matches $(\boldsymbol{c}_i, \boldsymbol{c}'_i)$ and determines which of them are inliers. This relies on ground truth correspondences $\Psi(\boldsymbol{c}_i)$. Interest points, correspondences and inlier labels shape mini-batches that contribute to the loss for a given training step as described in Section E.4. During evaluation and deployment, the inlier determination module is replaced with an application-specific geometric verifier, such as a perspective-n-point (PnP) localizer. In our experiments, we evaluate our system with P3P [74] localization using RANSAC [64], which produces a relative pose estimate. We compare this pose estimate to the ground truth relative pose to assess the viability of our method for visual pose estimation.

E.3.1 Inlier and True Correspondence Determination

Our training methodology requires inlier labels and correspondences in order to calculate the loss. Given a method Ψ to calculate true correspondences between the two images provided in a training step, we label an interest point $\boldsymbol{c}_i$ as *inlier* if the correspondence of the matched interest point from the other image $\Psi^{-1}(\boldsymbol{c}'_i)$ is within $3px$ of the matched interest point in the other image and vice versa. Otherwise, the match is either labeled an *outlier* if the correspondence lies somewhere else within the respective image, or *unassigned* if it is outside the image frame. In some datasets, a method to compute ground truth correspondence is provided. If such a method is not provided, correspondence can be calculated from ground truth depth and pose [175]. In case these are not provided either, the correspondence can be estimated using an SfM algorithm [155] given image sequences, or by using direct tracking such as KLT [125]. With this only image regions with sufficient texture can be tracked, which is acceptable since image regions without texture are unlikely to be of interest.

E.4 Training Methodology

We use standard iterative training using Adam updates [104]. In each iteration, a training sample, which consists of two images of the same scene is forwarded through the system to obtain a set of matches $\{(\boldsymbol{c}_i, \boldsymbol{c}'_i), i \in \{0, \dots, n-1\}\}$ and an associated set

chn.	label	$\boldsymbol{P}(\boldsymbol{c}_0)$	$\boldsymbol{P}(\boldsymbol{c}_1)$	$\boldsymbol{P}(\boldsymbol{c}_2)$	$\boldsymbol{P}(\boldsymbol{d}'_0)$	$\boldsymbol{P}(\boldsymbol{d}'_1)$	$\boldsymbol{P}(\boldsymbol{d}'_2))$
0	inlier	↑					n/a
1	outlier	↓	↓			↑	n/a
2	unass.	↓					n/a

Table E.1 – Mini-batch toy example to illustrate losses. For inliers, the activation of the maxima is strengthened while suppressing the activation in the other channels. For outliers, the activation of the maxima is weakened while promoting the response of the true correspondence.

of true correspondences $\{(\boldsymbol{d}_i, \boldsymbol{d}'_i) = (\Psi(\boldsymbol{c}_i), \Psi^{-1}(\boldsymbol{c}'_i))\}$ according to Figure E.2. During training, the loss is only applied at these sparse locations. In order to allow for efficient gradient backpropagation, patches are gathered from these locations. In image I, two mini-batches are formed: one from stacking *interest point patches* $\boldsymbol{P}(\boldsymbol{c}_i)$ centered around $\boldsymbol{c}_i$ and shaped according to the receptive field, $r \times r$, of a single pixel at the output. The other from stacking *correspondence patches* $\boldsymbol{P}(\boldsymbol{d}'_i)$ centered around the correspondences $\boldsymbol{d}'_i$. Both batches have a shape $[n, r, r, 1]$. The network transforms both of them into output tensors of shape $[n, 1, 1, n]$. The training loss is now applied to these tensors. Since they are flat along the height and width dimensions, we can conceptualize them as square matrices along the batch and channel dimensions, and visualize how the loss is applied to them in Table E.1 for a toy example with $n = 3$. Note that the diagonal in the first tensor contains the responses of patch $\boldsymbol{P}(\boldsymbol{c}_i)$ at channel i, which is the maximum value in channel i and the value that caused $\boldsymbol{c}_i$ to be selected as interest point. Similarly, the diagonal in the second tensor contains the responses that *should* be the maximum in the given channel considering the correspondence from the interest point selected in the other image. The training loss that is applied to these tensors has three components with their specific purpose:

- *Inlier reinforcement* reinforces interest points that are inliers in a given training sample, and suppresses interest points that are outliers.
- *Redundancy suppression* ensures that different channels do not converge to the same points.
- *Correspondence reinforcement* reinforces true correspondences of all points which are outliers in a given training sample.

The entire loss formulation is symmetrically applied to the other image I'.

Let p_{ij} be the scalar response of channel j to patch $\boldsymbol{P}(\boldsymbol{c}_i)$ and l_i the inlier label of that patch. Then, the inlier reinforcement loss is simply the cross-entropy loss according to that label, applied where $i = j$:

$$\mathcal{L}_{inl}(p_{ij}, l_i) = \begin{cases} -\log(p_{ij}), & l_i = \text{inlier}, i = j, \\ -\log(1 - p_{ij}), & l_i = \text{outlier}, i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{E.1}$$

This is the loss responsible for learning a high response for points that are likely to result in inliers. In Table E.1 the inlier reinforcement effect is observed for channel 0 on $\boldsymbol{P}(\boldsymbol{c}_0)$ and the outlier suppression effect is present for channel 1 on $\boldsymbol{P}(\boldsymbol{c}_1)$.

To prevent channels from converging to the same interest points, a loss is applied on inlier patches that suppresses the response on all channels, except the one which gave rise to the inlier:

$$\mathcal{L}_{red}(p_{ij}) = \begin{cases} -\log(1-p_{ij}), & l_i = \text{inlier}, i \neq j, \\ 0 & \text{otherwise.} \end{cases} \tag{E.2}$$

In Table E.1 the redundancy suppression is present on channels 1 & 2 on $\boldsymbol{P}(\boldsymbol{c}_0)$. We found that without redundancy suppression, all channels tend to converge to a single feature, all selecting the same interest point in every image.

Finally, we have found that our network does not converge with the above losses alone. Thus, for channels with outliers, we promote p'_{ii}, the response of channel i to patch $\boldsymbol{P}(\boldsymbol{d}'_i)$:

$$\mathcal{L}_{cor}(p'_{ii}) = \begin{cases} -\log(p'_{ii}), & l_i = \text{outlier} \\ 0 & \text{otherwise.} \end{cases} \tag{E.3}$$

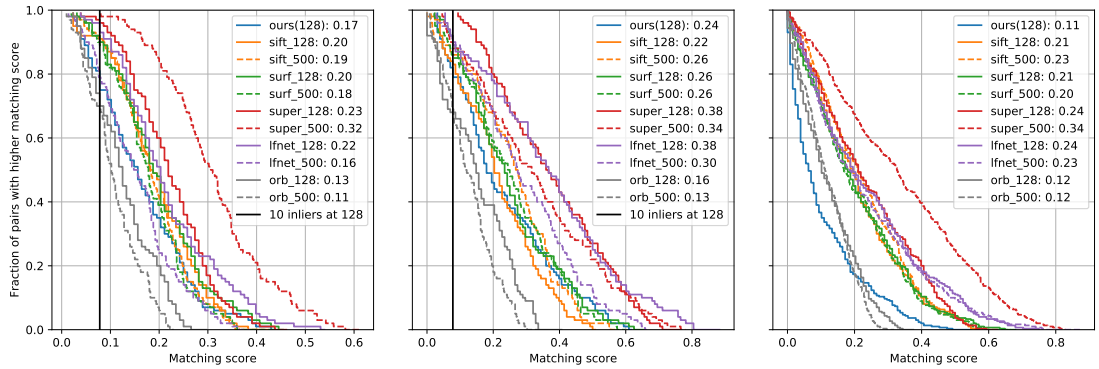
In Table E.1 the correspondence reinforcement can be observed in channel 1 on $\boldsymbol{P}(\boldsymbol{d}'_1)$, which is the patch extracted at the correspondence of the maximum activation of the paired image.

E.4.1 Training Pair Selection

In datasets such as HPatches [13], pairs are pre-selected. For image sequences, we extract pairs as follows: Given one image, points are densely sampled and subjected to KLT tracking for as far into subsequent images as possible. A pair is then formed between this initial image and a random subsequent image in which at least a fraction o of the initial points is still tracked. o thus reflects the minimum scene overlap between the two images. The benefit of this method is that it can be applied to uncalibrated image sequences, while providing good guarantees regarding minimum scene overlap. For training, we use $o = 0.3$, for selecting pairs during evaluation $o = 0.5$.

E.5 Experiments

We subject our method to evaluation and comparison to state-of-the-art on three datasets: The viewpoint-variant part of the wide-baseline HPatches benchmark [13, 115], sequence `00` of the outdoor autonomous driving dataset KITTI [76] and sequence `V1_01` of the indoor drone dataset EuRoC [24]. For the two sequences, 100 image pairs



(a) KITTI (b) Euroc (c) HPatches

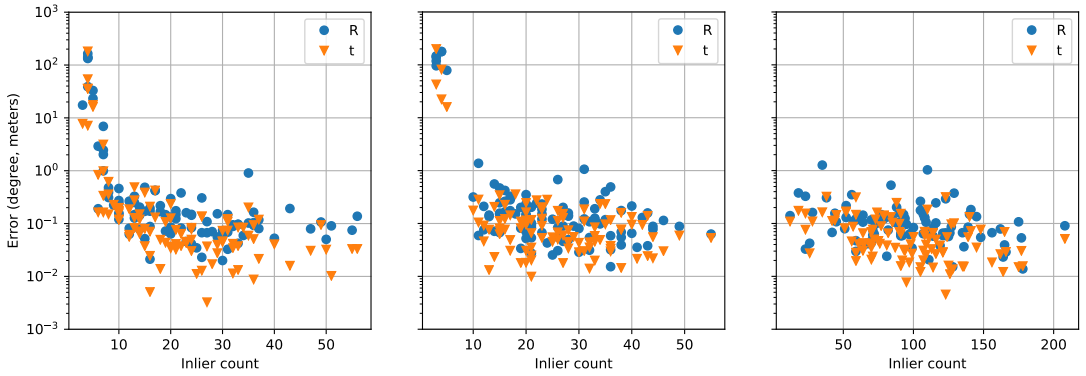
Figure E.3 – Matching score distributions (higher is better).

are randomly selected using the method in Section E.4.1. KITTI and EuRoC are stereo datasets, and we only use the left camera for the image pairs. Rotation and translation differences within these pairs are plotted in the supplementary material. For the HPatches benchmark, we train our network using the provided training split, while for KITTI and EuRoC, we train our network on a separate dataset, TUM mono [60], where we use the rectified images of sequences `01`, `02`, `03` (indoors) and `48`, `49`, `50` (outdoors). We found that adding sequences `04` to `47` does not result in better performance. For the sequences, we use KITTI `05` as validation dataset. Ground truth correspondences are given by ground truth homographies in the HPatches training data. In TUM mono, they are instead established using KLT as discussed in Section E.3.1.

We compare our method to SIFT [124], SURF [14], ORB [170], LF-Net [155] and SuperPoint [52]. For SIFT, SURF and ORB, we use the OpenCV implementation, while for LF-Net and Superpoint, we use the publicly available code and pre-trained weights. All baselines are evaluated both at 128 interest points, like our method, and at a more native interest point count of 500.

E.5.1 Matching score

For all datasets, we evaluate matching score, which is the fraction of inlier matches among all matches. In HPatches, the ground truth homography is used to distinguish inlier from outlier matches. In KITTI and EuRoC we instead use the inliers that are determined during pose estimation, see Section E.5.2. As a consequence, the matching score here more closely reflects the matching score that is achieved in practice when doing relative pose estimation, whether in SLAM or in map localization. Figure E.3 shows the matching score obtained in the three datasets. Both on KITTI and on EuRoC, our method performs slightly worse than all baselines except ORB, especially if the



(a) ours (b) SIFT, 128 points (c) SIFT, 500 points

Figure E.4 – Pose error versus inlier count on KITTI. R: rotation error, t: translation error.

latter are permitted to extract more points than our method. Note, however, that SURF, SIFT, SuperPoint and LF-NET use 64, 128, 256 and 256 floating points to describe each interest point, respectively, in addition to point locations. Instead, our method represents a visual location using only 128 properly ordered point locations. The results on HPatches show that our method is still not very well suited to the significant viewpoint and scale changes present in that dataset. Nevertheless, we believe that implicit interest point matching opens a new research direction in terms of more closely integrated detector training. Future work to address strong viewpoint and scale changes could include considerations such as scale and rotation invariance, whether by modeling inside the neural network [155] or by data augmentation and curricular learning [52].

E.5.2 Pose Estimation Accuracy

To understand how matching score translates to pose estimation accuracy, we compare the pose estimated using our method and SIFT with the ground truth relative pose on both KITTI and EuRoC. Both in SLAM and map localization absolute relative pose is typically obtained from interest points matched to 3D point locations by P3P localization [74] with RANSAC [64]. To obtain 3D point locations in one of the two images, we use epipolar stereo matching of the interest points extracted using our method or a baseline with the corresponding image from the right camera. Rotation and translation error are measured with the geodesic (angle in angle-axis) and euclidean distances. In Figure E.4 these errors are compared to the inlier count for each pair in the KITTI testing set, using both our interest points and SIFT. The same plot for EuRoC can be found in the supplementary material. We can see that both with our method and with SIFT, an inlier count above 10 indicates a good relative pose (rotation error below 1°, translation error below 30cm). Estimates get slightly better up to an

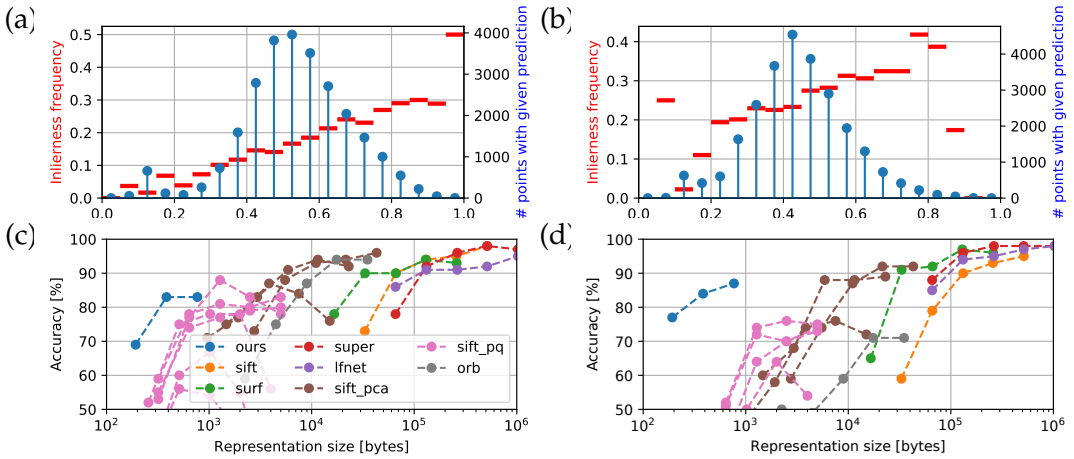


Figure E.5 – (a), (b) Empiric "inlierness" frequency versus response, and histogram of interest point responses. Inlierness frequency is calculated per histogram bin by dividing the amount of inliers within a response range by the total amount of interest points within that range. (c), (d) Accuracy versus representation size for our method and baselines, including compression schemes for SIFT. Note the logarithmic scale. Left: KITTI, right: EuRoC.

inlier count of 20, beyond which they do not improve by much, even if much more interest points are extracted. We transfer this insight to Figure E.3, where we indicate the matching score corresponding to 10 inliers with a vertical line. From this, we can see that our method results in good relative poses in 80% of the tested image pairs.

E.5.3 Other results

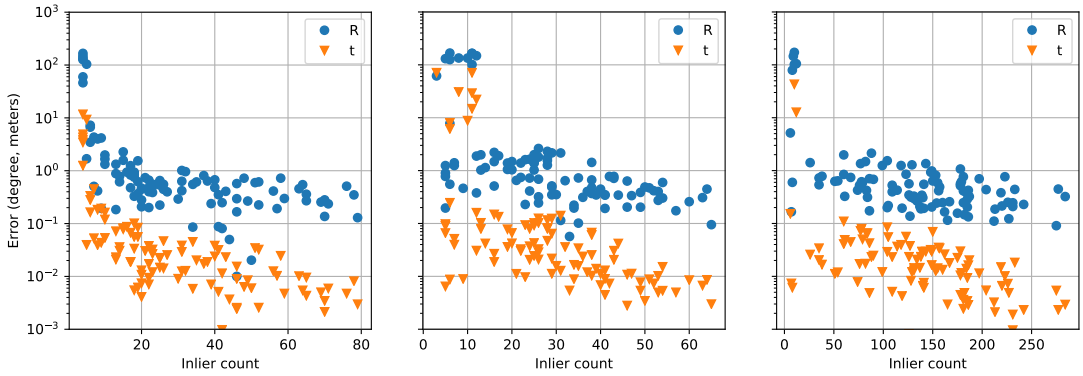
Some additional insights are shown in Figure E.5. The parts (a) and (b) show how the probability of a point being an inlier is correlated with the response of the corresponding channel at that point, using our network. As can be seen, there is a good correlation, which indicates that the response has some predictive power regarding the probability of a point resulting in an inlier. This could potentially be exploited, for example in RANSAC model sampling. In (c) and (d), we show how pose estimation accuracy compares to the amount of data needed to represent a visual frame for our method, the previously used baselines and two compression methods applied to SIFT descriptors. Here, accuracy represents the fraction of pose estimates with rotation and translation errors below 1° and 30cm (KITTI) and 3° and 10cm (EuRoC). The two compression methods are principle component analysis (PCA) projection of SIFT descriptors and product quantization [96, 127], see the supplementary material for details. For our method, we evaluate networks with output channel counts $\in \{64, 128, 256\}$. For all other methods, we evaluate $\{64, 128, 256, 500, 1000\}$ interest points per image. As can be seen, our method outperforms baselines in the trade-off between accuracy and representation size.

E.6 Conclusion

In this paper, we have introduced a descriptor-free approach for detection and matching of sparse visual features. We rely on a convolutional neural network to predict multiple activation layers and we define the location of the maximal response in each layer to be an interest point. The key novelty is that instead of relying on descriptors for matching, interest points are uniquely associated to the activation layer they are extracted from. This setup allows us to train the traditionally modularized interest point detection, description, and matching processes jointly in a simple setup, while at the same time getting rid of the requirement for explicit descriptors. Without descriptors, visual features can be stored, communicated and matched at a highly reduced cost. For this system, we have devised a self-supervised training methodology that reinforces interest points that result in inliers, ensures that each channel specializes on different features, and uses ground truth correspondences to ensure that channels not resulting in inliers find good candidate features. This training can be achieved with uncalibrated image sequences. Albeit achieving slightly lower matching scores when compared to other approaches that do use descriptors, we demonstrated the applicability of our descriptor-free approach in a visual pose estimation setup.

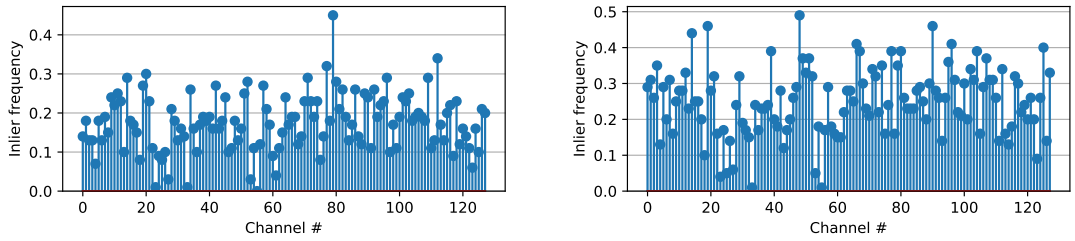
E.7 Acknowledgments

This work was supported by the National Centre of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant. The Titan Xp used for this research was donated by the NVIDIA Corporation.



(a) ours (b) SIFT, 128 points (c) SIFT, 500 points

Figure E.6 – Pose error versus inlier count on EuRoC. R: rotation error, t: translation error.



KITTI EuRoC

Figure E.7 – Distribution of inliers among channels in the testing datasets.

E.8 Supplementary Material

E.8.1 Additional results

Figure E.6 shows the equivalent of Figure E.4 for the EuRoC dataset. The result is similar except that the rotation error is larger and the translation error smaller. The latter is likely due to the smaller scene depth of the indoor EuRoC dataset versus the outdoor KITTI dataset.

Figure E.7 shows how inliers are distributed among channels. As can be seen, the empiric frequency of resulting in an inlier on testing data is well-distributed among the channels, meaning that the training manages to evenly distribute the feature space of good interest points among the channels.

E.8.2 CSV files

A detailed CSV file is attached for each of the two robotic testing dataset. They contain the columns:

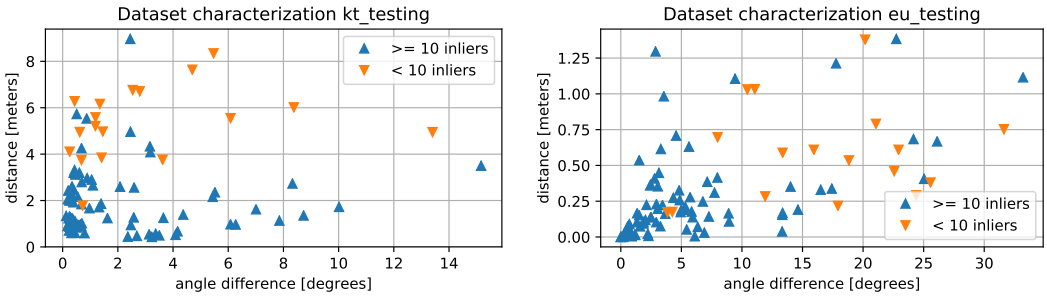


Figure E.8 – True pose difference plots for KITTI (left) and EuRoC (right). Markers indicate whether or not more than 10 P3P RANSAC inliers have been achieved for a given pair. As shown in the paper, this is a good inlier threshold to accept or reject relative pose estimates.

- `name`: sequence name and the indices of the two images that form the pair.
- `dR, dt`: ground truth difference in rotation and translation between the two image frames.
- `matching score`: The matching score of our method for the given pair.
- `eR, et`: rotation and translation *error* of the relative pose estimate.

Both `dR` and `eR` are measured with the geodesic distance, in degrees, which corresponds to the angle of the angle-axis representation of the relative rotation.

E.8.3 True relative pose in testing sets

To provide an intuition for the robotic testing sets, we plot the distribution of true relative poses in Figure E.8 Note the difference between the two datasets: KITTI exhibits larger distances, while EuRoC exhibits larger orientation differences. Recall that given the first image, the second image is randomly sampled among subsequent images with a scene overlap of at least 50%.

Besides showing the distribution, this plot also indicates which pairs result in 10 or more P3P RANSAC inliers using our method. As shown in the paper, these pairs result in a good relative pose estimate. While the pairs that fail to obtain a good relative pose estimate are typically the ones with larger pose differences, there is no clear boundary between pairs that succeed and pairs that fail.

E.8.4 Accuracy versus representation size figures

In Figure E.5 (c), (d) we show results for two compression schemes applied to SIFT, principal component analysis (PCA) projection and product quantization [96, 127]. For each method, several lines are plotted for several parameter choices. For PCA projection, these correspond to projections to $\{3, 5, 10\}$ dimensions, while for prod-

uct quantization, these correspond to $(m,k) \in \{(1,256),(2,16),(4,4)\}$ (one byte per feature), and $(m,k) \in \{(2,256),(4,16),(8,4)\}$ (two bytes per feature), where m is the amount of quantizations/dimension subdivisions and k the amount of clusters in each quantization. Both compression methods are trained on the same dataset as our method, TUM mono sequences `01`, `02`, `03` (indoors) and `48`, `49`, `50` (outdoors).

Furthermore, these figures contain results for our network with 64 and 256 dimensions. The 64 network has the same intermediate channel counts as the standard 128 network described in Section E.3. The 256 network, instead, has all intermediate channel counts doubled.

E.8.5 Sequence video

We visualize the stability of our interest points as well as the correlation of that stability with predicted inlierness probability with a video composed of the score maps extracted over the testing sequences. Due to size constraints, only 180 frames of KITTI and 250 frames of EuRoC are shown. Note that the interest points are independently extracted in each frame. Still, several and in particular the high-score interest points behave as if they were directly tracked. This corresponds to points that would result in correct matches, and gives an idea of what the matching score means. One observation that can be made in our video is that the response is not always very well-localized. Future work to address this issue could be to extend the training losses with a loss that favours peakedness [155, 220].

F Exploration Without Global Consistency Using Local Volume Consolidation

Reprinted, with permission, from:

T. Cieslewski, A. Ziegler, and D. Scaramuzza. "Exploration Without Global Consistency Using Local Volume Consolidation". In: *Proc. Int. Symp. Robot. Research (ISRR)* (2019)

Exploration Without Global Consistency Using Local Volume Consolidation

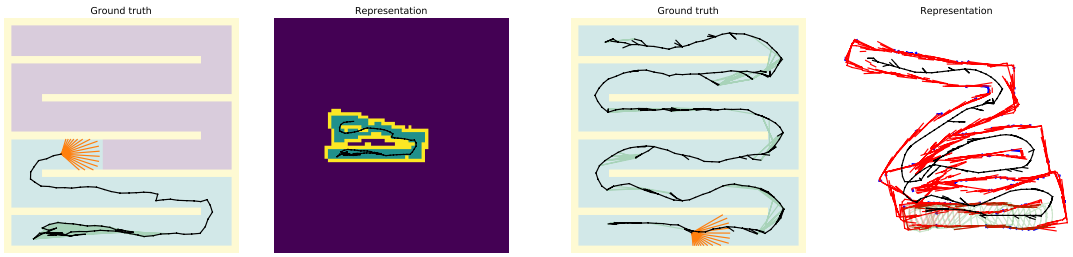
Titus Cieslewski, Andreas Ziegler and Davide Scaramuzza

Abstract — In exploration, the goal is to build a map of an unknown environment. Most state-of-the-art approaches use map representations that require drift-free state estimates to function properly. Real-world state estimators, however, exhibit drift. In this paper, we present a 2D map representation for exploration that is robust to drift. Rather than a global map, it uses local metric volumes connected by relative pose estimates. This pose-graph does not need to be globally consistent. Overlaps between the volumes are resolved locally, rather than on the faulty estimate of space. We demonstrate our representation with a frontier-based exploration approach, evaluate it under different conditions and compare it with a commonly-used grid-based representation. We show that, at the cost of longer exploration time, using the proposed representation allows full coverage of space even for very large drift in the state estimate, contrary to the grid-based representation. The system is validated in a real world experiment and we discuss its extension to 3D.

Video: A video is available at https://youtu.be/s4Xnet_h4ss

F.1 Introduction

Exploration is a fundamental task in autonomous robotics. The goal is to map an unknown environment [212], for example to establish how a robot can navigate through it. Exploration can be used in various scenarios from search and rescue, oil and gas exploration, 3D reconstruction to inspection tasks. Different applications have different



(a) Global, grid-based **(b)** Ours

Figure F.1 – In a maze-like environment where loop closures cannot be established to account for pose estimate drift, grid-based representations build an inconsistent map, which can lead to premature termination of exploration (a). The purple region in the ground truth of explored space, in the leftmost image, indicates unexplored space. Our method, in contrast, only resolves overlaps between volumes locally, which allows to fully explore the environment (b). See Figure F.2 for a detailed legend.

requirements. In search and rescue, for instance, it is important to find survivors rapidly. Other requirements could be that the generated map has a high accuracy or that the path travelled by the robot is as short as possible. In exploration, a map representation is used to describe the space that the robot perceived so far. Based on this description, a planning algorithm decides where to move next. In this work, we focus on the map representation.

Most map representations currently used for exploration assume perfect state estimates. However, on-board state estimators, such as visual-inertial odometry, are prone to drift. Since these are the estimators most likely to be used in unknown environments in the real world, these map representations can in practice lead to incorrect maps, as illustrated in Figure F.1.

To some extent, drift can be removed when a robot revisits a place and a correction of the trajectory estimate can be performed with a loop closure. However, as Figure F.1 illustrates, not all drift error can be removed in this way. Furthermore, such a correction causes heavy computational load in typical grid-based maps, as every depth measurement ray needs to be re-cast with the new trajectory estimate [211]. Sub-mapping approaches can mitigate this computational cost [20], [179], [140].

F.1.1 Contributions

In this paper, we propose a map representation for exploration that is robust to state estimate drift. We build our approach using ideas from [91], where free space is represented with local polygons connected by relative poses. Our contributions are as follows:

- We apply the ideas from [91] to exploration by adding semantics to the local polygons, which incorporate information from other nearby polygons. Nearby polygons are determined based on vicinity in a pose graph, as opposed to vicinity in an estimate of space that is faulty due to drift.
- We show that with this, global map consistency is not required for the robot to know when exploration is complete. With this, global map optimization is not needed, which is particularly interesting for multi-robot exploration.
- We show how to apply our representation to an exploration algorithm that previously used a grid-based representation. Local navigation is performed in local polygons, while navigation between remote locations in the map is performed using teach-and-repeat [61].
- Finally, we simulate our proposed map representation under different conditions, compare it to grid-based representations, and validate it in the real world. As we show, our method comes at a cost of longer exploration times, but given reliable place recognition, is able to deal with large drift in cases where grid-based representations fail.

F.2 Related Work

While a map representation is used to represent the space that the robot perceived so far, a planning algorithm has to plan where the robot should move next. In the following, we review these two aspects of exploration separately.

F.2.1 Map Representations

Exploration can be formalized as the problem of discovering all free space within a bounded volume. Thus, the map representation needs to represent the spatial information necessary to achieve this. Map representations are commonly divided into three categories [208]: metric representations, topological representations and hybrid, topometric representations.

Metric Representations Metric representations express locations as coordinates in a global reference frame. The predominant type of metric representations used in exploration are grid-based representations. In a grid-based representation, the volume is quantized into voxels, arranged in a grid. The voxels are then assigned appropriate labels, such as “free”, “unknown” or “obstacle” [212]. Voxels typically carry further information, such as occupancy probabilities [145, 211] or a signed distance to the closest obstacle [94, 154]. In two dimensions, an alternative way of representing known free space is to represent it with a polygon [25, 78]. Here, the inside of the polygon represents free space, while its boundaries can either represent obstacles or the interface

to unknown space. Because the location of all entities in fully metric representations are expressed in a global reference frame, they are highly reliant on accurate pose estimates: wrong pose estimates would lead to depth measurements being inconsistent with the true scene geometry, and with each other. As a consequence, the robot could wrongly believe occupied space to be free or inversely, free space to be occupied, which could have disastrous consequences for navigation.

Topological Representations Unlike metric representations, topological representations do not express locations as coordinates in a reference frame. Instead, locations are represented as vertices in a graph. Relationships between locations, such as adjacency, are expressed as edges. In order to be navigable, these edges should carry enough information to allow a robot to move between adjacent vertices. Vision-based navigation in such maps has been demonstrated using Visual Teach and Repeat [61, 71]. While topological maps can also be derived from volumetric maps [**Bloechliger17arxiv**, 208], we are not aware of any work that uses purely topological maps to directly represent volumetric information. Pure topological maps are thus rarely used for exploration. A notable exception to this are [2, 20]. Rather than explicitly keeping track of free space, their robots build a pose graph similar to [61]. At every vertex of this graph, the robot adds to the graph a finite amount of adjacent candidate locations where the robot could move to. Subsequently, these candidate locations are all visited and the process is repeated recursively unless a candidate turns out to correspond to a previously visited location. However, candidate robot locations are only a very crude representation of free space, which makes it hard to estimate the actual coverage.

Hybrid Representations Hybrid, or *topometric*, representations are a combination of metric and topological representations, aimed at combining their advantages. Strictly speaking, the aforementioned pose graphs were already topometric, since they contained metric positions and transformations. Nonetheless, in this section we focus on representations which augment topological graphs with volumetric information. An early topometric representation has been proposed in [91]. This representation consists of a set of polygons, each representing the free space measured around a specific pose of the robot. The locations of these polygons are expressed relative to each other. However, [91] does not consolidate the information of neighbouring polygons in a way that would be helpful for exploration. As a consequence, in [90], where [91] is used in an exploration system, the authors fall back on local occupancy grids for exploration. In contrast, we extend [91] with a consolidation procedure that allows its direct application to exploration. Furthermore, while [90] ends up building a globally consistent map, we show that global consistency is not required for exploration. The use of local occupancy grid *submaps* is also proposed in [140, 179], where the authors aim at producing a globally consistent map of the environment just like [90]. Achieving global consistency requires map optimization, which in turn leads to costly re-calculation of the occupancy grid submaps, even if it can sometimes be avoided for small loop closure corrections [179]. We show that all of this can be avoided since global consistency is

not necessary for exploration.

F.2.2 Path Planning for Exploration

In order to understand how to build a good representation, we need to understand how it is used. In path planning for exploration, after every observation the robot has to decide where to move next. In order to achieve fast exploration, the robot should perceive unknown space as quickly as possible. There are two main planning approaches in the literature: Next-best-view (NBV) planning and frontier-based planning.

Next-best-view (NBV) Planning NBV planners try to choose the next position in such a way that the perception is optimal according to a *utility* function. This problem has also been studied in the computer vision community [42]. Most NBV planners determine the next best view by sampling candidate views, predicting their utility [78, 156], and picking the view with the highest utility. The advantage is that the utility function can be adapted to contain arbitrary terms and constraints, such as the motion model of the robot, or terms describing map accuracy. The disadvantage is that finding the view with optimal utility is usually intractable, hence a sampling method has to be applied. Furthermore, and somewhat as a consequence, NBV planning commonly requires heavy computational load.

Frontier-Based Exploration Planning Frontier-based exploration planners follow a simple and efficient scheme. The strategy is to navigate to the closest boundary between known free and unknown space. These boundaries are referred to as *frontiers*. Frontier-based exploration methods assume that navigating to a frontier will result in the exploration of new space. The original idea was introduced in [212]. Unlike NBV planners, frontier-based planners do not natively support arbitrary terms or constraints. However, it was shown that they cover the space faster than their NBV counterparts [37, 87]. In this work, we will evaluate our representation using a modified version of the frontier-based exploration method presented in [37]. Frontiers are explicitly expressed in our method, so it is natural to adopt frontier-based exploration.

F.3 Problem Statement

The goal of exploration is to build a map of an unknown environment. Consider a bounded region of space $\mathcal{V} \in \mathbb{R}^2$ that represents the unknown environment that we would like to explore. $\mathcal{V}$ consists of free and occupied space $\mathcal{V} = \mathcal{V}_{\text{free}} \cup \mathcal{V}_{\text{occ}}$. With a robot that can measure the free space around itself, we can explore $\mathcal{V}_{\text{free}}$. We denote a robot pose in the world frame W as $\mathtt{T}_{W,R}$ with orientation $\mathtt{R}_{W,R}$ and translation $t_{W,R}$. The field of view (FOV) of the robot at pose $\mathtt{T}_{W,R}$ is denoted as $\mathcal{V}_{\text{fov}}(\mathtt{T}_{W,R}) \subset \mathcal{V}_{\text{free}}$. While a robot is moving on a trajectory $\mathtt{T}_{W,R}(t)$, it will measure $\mathcal{V}_{\text{fov}}$ at consecutive

sampling times $t_0, t_1, \ldots$. We denote the corresponding poses as $\mathrm{T}_{W,R_0}, \mathrm{T}_{W,R_1}, \ldots$. The space explored up to time t_k (*known free space*) is the union of all the FOV measured so far:

$$\bar{\mathcal{V}}_{\text{free}}(t_k) = \bigcup_k \mathcal{V}_{\text{fov}}(\mathrm{T}_{W,R_k}). \tag{F.1}$$

We consider two goals for our representation: 1) represent the robots' estimate of $\bar{\mathcal{V}}_{\text{free}}$. 2) Allow to determine when exploration is complete, that is, when the whole free space is covered:

$$\bar{\mathcal{V}}_{\text{free}} = \mathcal{V}_{\text{free}}. \tag{F.2}$$

Without knowing the ground truth $\mathcal{V}_{\text{free}}$, this condition can be established given sufficient knowledge about the boundary of $\bar{\mathcal{V}}_{\text{free}}$, $\partial\bar{\mathcal{V}}_{\text{free}}$. Generally, $\partial\bar{\mathcal{V}}_{\text{free}}$ either consists of interfaces to occupied space or interfaces to unknown free space. As can be shown by contradiction, (F.2) holds if and only if $\partial\bar{\mathcal{V}}_{\text{free}}$ consists only of interfaces to occupied space[1] ("no frontiers left"). Hence, the second goal can be reached by correctly representing $\partial\bar{\mathcal{V}}_{\text{free}}$.

F.4 Proposed Representation

As in [91], the proposed map representation is based on polygons. More specifically, local polygons individually represent the FOV of every pose, $\mathcal{V}_{\text{fov}}(\mathrm{T}_{W,R})$. The inside of a polygon represents known free space $\bar{\mathcal{V}}_{\text{free}}$ while its edges represent the boundary $\partial\bar{\mathcal{V}}_{\text{free}}$. Unlike [91], $\partial\bar{\mathcal{V}}_{\text{free}}$ is labeled, which allows us to reach the second goal of the problem statement. The labels are *obstacle* for parts of the boundary that are common with occupied space, *frontier* for parts adjacent to unknown space, and *free* for parts that are considered to lie within another polygon. See Figure F.2 for a visualization of our representation.

F.4.1 Local Volumes from Depth Measurements

We assume a depth sensor as source to build the polygon of the robot's field of view (FOV). Each measurement consists of a set of simultaneously acquired depth samples. In our simulations, these samples are modelled as equally distributed within the FOV, as shown in Figure F.3. We build the polygon from these samples by taking the position of adjacent samples and the robot's position as vertices and connecting them with edges. Some samples correspond to measured depths while others correspond to samples where the closest obstacle is beyond the sensor range. If two adjacent depth samples are not beyond range, and have values with a difference below a threshold δ,

[1] Trivial exceptions, such as disconnected free space, are omitted for brevity.

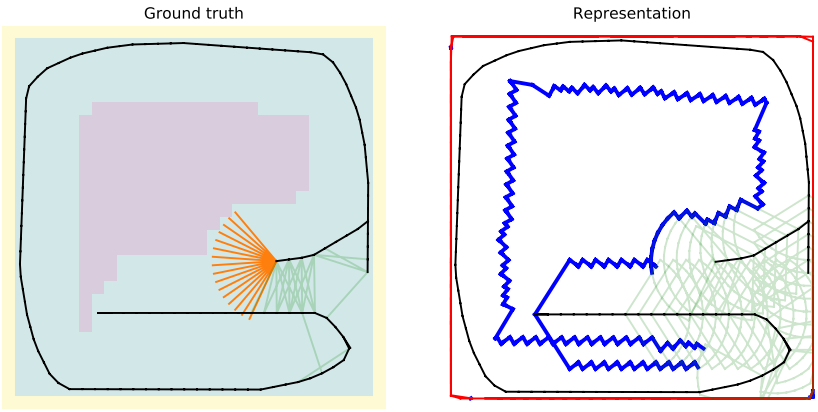


Figure F.2 – Our representation, visualized for a simulation without drift in the pose estimate. Left image (ground truth): black line: robot trajectory, green lines: place recognitions, orange lines: depth sensor rays at current pose; in the background, yellow squares indicate true occupied space, green squares true known free space and blue squares true unknown free space. Right image (our representation): black line: pose graph without loop closures, red lines: known obstacles, blue lines: frontiers, green lines: local volumes in consolidation scope (see Section F.4.3). Note how frontiers may overlap, as they are not resolved globally but only within the consolidation scope.

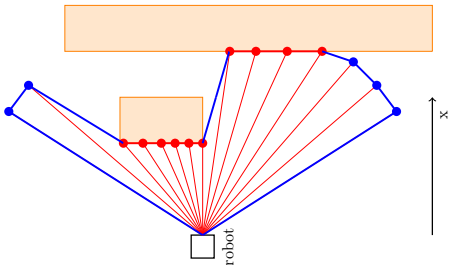


Figure F.3 – A local volume obtained from one measurement. The red lines starting at the robot position represent the depth measurement rays. Blue points represent samples at maximum sensor range and red points represent samples that hit an obstacle. Blue edges are the frontier edges and red edges obstacle edges.

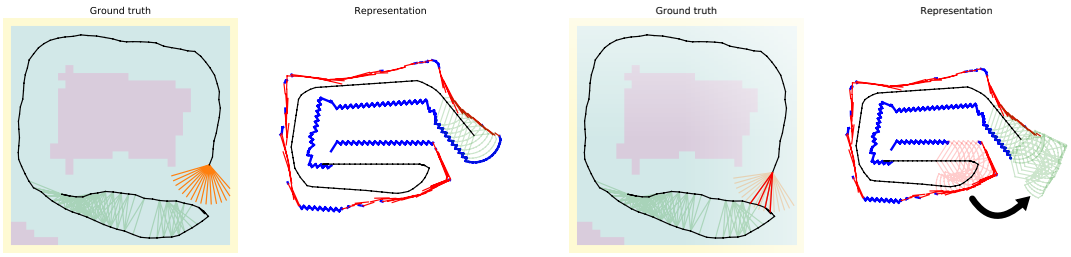
the edge connecting them is considered an obstacle edge. In all other cases, the edge is considered a frontier edge. The threshold δ is intended to account for occlusions (see Figure F.3). Sometimes, these cannot be distinguished from obstacle surfaces that are nearly parallel to the measurement rays [78]. These incorrectly labeled frontiers can be resolved by approaching that obstacle from another angle. Furthermore, the two edges adjacent to the robot position are also considered frontiers.

F.4.2 Pose Graph Representation

In our representation, we store separate local volumes for each set of depth measurement samples, which are built as described in the previous section. These polygons are referenced in the vertices of a *pose graph*. Each vertex of this graph represents a robot pose at which a measurement was taken. The vertices are connected with edges that carry the relative transformation $\mathtt{T}_{R_{k-1},R_k}$ between the two poses. The edges are either established between subsequent measurements, in which case $\mathtt{T}_{R_{k-1},R_k}$ is estimated using odometry, or they are established due to place recognition, in which case $\mathtt{T}_{R_{k-x},R_k}$ comes from a relative pose estimation algorithm. The pose graph and our representation can be constructed incrementally and on-line. The relative transformation between non-adjacent poses can be estimated by integrating relative transformations along a path connecting the two corresponding vertices. Considering odometry drift, this estimate becomes less accurate as the path length increases. Without global consistency, estimates integrated along different paths can differ [20, 61]. Thus, for our purposes, we use the estimate resulting from integrating along the shortest path.

F.4.3 Frontier Consolidation

When adding new depth measurements, we need to update the local polygons. New measurements can turn unknown space into known free space, and the corresponding frontier boundaries need to be converted into free boundaries. Similarly, frontiers might need to be re-assessed after place recognition. This process, which we call *frontier consolidation*, is illustrated in Figure F.4. For every new depth measurement $\mathcal{V}_{\text{fov}}(\mathtt{T}_{W,R_k})$, a *consolidation scope* comprising the local volumes of nearby poses is established using Dijkstra's algorithm. All poses within a distance of $\mathcal{R}$ are added to that scope, where the norm of the estimated translation $t_{A,B}$ between two adjacent poses is used as the weight of the edge between them. Consequently, $\mathcal{R}$ reflects the distance over which the pose estimation (odometry and relative pose estimation) has small drift and can be used to consolidate local volumes. Inside the consolidation scope, all local volumes are then consolidated pairwise among all possible pairs. To that end, all local volumes are temporarily expressed relative to $\mathcal{V}_{\text{fov}}(\mathtt{T}_{W,R_k})$ by using the pose resulting from pose integration along the shortest path from $\mathcal{V}_{\text{fov}}(\mathtt{T}_{W,R_k})$, see Figure F.4. Given two local volumes transformed in this way, any frontier edge of one volume that lies inside



(a) Before loop closure **(b)** After loop closure

Figure F.4 – Frontier consolidation prompted by place recognition. See Figure F.2 for a detailed legend. (a) Before place recognition, the consolidation scope contains most recent poses. (b) After place recognition (red lines in ground truth), local volumes across the place recognition edges (light red) are added to the consolidation scope, temporarily transformed into the local frame of the current pose (arrow, light green volumes). This affects frontiers (blue): They frontiers above the red volumes and the "range arc" of the current pose are both resolved.

the other volume is re-labeled as free edge, and vice versa. If a frontier edge is only partially inside, it is subdivided accordingly.

Consolidation is triggered for every new depth measurement. In our experiments, we assume that previously visited places are recognized at the same time as a vertex is inserted into the polygon. Otherwise, polygon consolidations would also need to be triggered as new edges are inserted into the pose graph due to place recognition.

So while our approach does not rely on odometry accuracy, it relies on good place recognition performance. An increasing rate of false negatives could potentially be dealt with by methods that expand existing matches, or by increasing the volume in which polygons are consolidated. An increasing rate of false positives (perceptual aliasing) would be harder to deal with. However, false positives have also been shown to be catastrophic for optimization-based approaches [190] – remedies to false positives are equally applicable to both approaches.

F.4.4 Extension to 3D

While for simplicity, we focus on two dimensions in this paper, we believe that the presented approach is readily applicable in 3D. The main challenge here is the 3D representation of local volumes, and the intersection of those local volumes, where volume boundaries become surfaces, rather than edges. One could use local occupancy grids, or local coarse meshes [80, 193] for memory efficiency.

F.5 Application to Frontier-Based Exploration

To demonstrate how our map representation can be fitted to exploration approaches that previously used metric representations, we adapt the state-of-the-art method proposed in [37] to use our solution. Exploration in [37] is performed using a state machine with the following three states:

1. **Reactive:** While frontiers are present in the current FOV, navigate towards the frontier that incurs the least change to the current velocity.

2. **Deliberate:** If no frontiers are left in the FOV, but frontiers are left globally, plan a path to the closest frontier.

3. If no frontiers are left overall, exploration is considered **complete**.

Since the **reactive** state is based on the current field of view, it is trivial to adapt to our representation. In the original implementation, information was extracted from OctoMap [211] updates to determine frontier voxels in the current FOV. With our representation, frontier candidates are simply selected from the edges of the current local polygon. The **deliberate** state cannot be adapted directly. In the original implementation, a path to the closest frontier was found using a Dijkstra search across adjacent free space voxels. With our representation, adjacency of free space is only meaningful among polygons that are also close to each other in the pose graph. Hence, instead of looking for the closest frontier in 2D space, we instead look for the closest vertex in the pose graph that has a local volume with unconsolidated frontiers. We then let the robot navigate to that vertex using the teach and repeat navigation method proposed in [61], as it does not require global consistency of the map. Once the robot arrives at that vertex, it switches back to the reactive state. **Completion** in the original method is assumed once there are no frontier voxels left. In our approach, this corresponds to none of the polygons having frontier edges left.

F.6 Experiments

Using [37] as exploration algorithm, we compare our representation to a metric grid-based representation, with and without loop closure capability, in simulated 2D environments under varying conditions. The representations are compared in different environments, with simulated odometry drift of varying intensity and with simulated place recognition of varying recall.

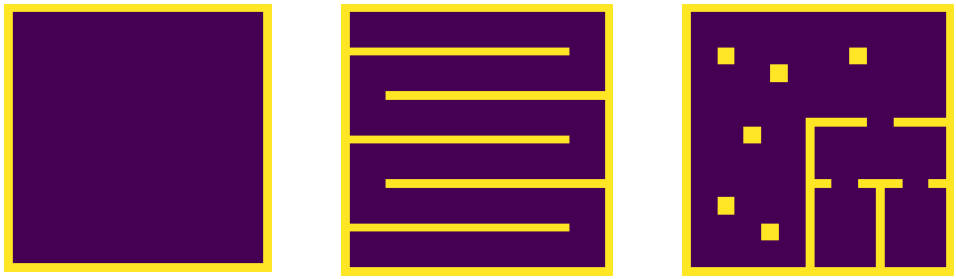


Figure F.5 – The simulated environments: "Open", "Maze" and "Forest house". All have a size of 30×30 meters.

F.6.1 Experimental Setup

We simulate a robot with a forward-looking depth sensor operating in either of the $30m \times 30m$ environments shown in Figure F.5. The depth sensor has a FOV of 115° in horizontal direction and a range d_{FOV} of $5m$, similar to the sensor used in [37]. The simulation is performed in time steps $k \in \mathbb{N}$, for each of which the robot's true pose $\mathbf{T}_{W,R_k}$ and pose estimate $\bar{\mathbf{T}}_{W,R_k}$ are updated as follows:

$$\mathbf{T}_{W,R_k} = \mathbf{T}_{W,R_{k-1}} \bar{\mathbf{T}}_{R_{k-1},R_k} \eta_{\mathbf{T}_{R_{k-1},R_k}}, \tag{F.3}$$

$$\bar{\mathbf{T}}_{W,R_k} = \bar{\mathbf{T}}_{W,R_{k-1}} \bar{\mathbf{T}}_{R_{k-1},R_k}, \tag{F.4}$$

where $\bar{\mathbf{T}}_{R_{k-1},R_k}$ is the relative pose command output provided by the exploration planner. In the true pose update, a pose increment $\eta_{\mathbf{T}_{R_{k-1},R_k}}$ is applied to $\bar{\mathbf{T}}_{R_{k-1},R_k}$. As it is the robot that executes the desired pose increment, it believes it has executed $\bar{\mathbf{T}}_{R_{k-1},R_k}$, and so noise needs to be applied to the true pose update, rather than the estimate. The translation of $\eta_{\mathbf{T}_{R_{k-1},R_k}}$ has coefficients sampled from a zero-mean Gaussian $\mathcal{N}(0, f(\sigma_{\text{pos}}, \bar{\mathbf{T}}_{R_{k-1},R_k}))$ and its rotation angle is sampled from $\mathcal{N}(0, f(\sigma_{\text{rot}}, \bar{\mathbf{T}}_{R_{k-1},R_k}))$. Here, f is a function that ensures that the variance of the noise is proportional with the distance travelled:

$$f(\sigma, \bar{\mathbf{T}}_{R_{k-1},R_k}) = \|\bar{\mathbf{t}}_{R_{k-1},R_k}\| \cdot \sigma^2. \tag{F.5}$$

We furthermore simulate place recognition. Whenever the current pose $\mathbf{T}_{W,R_k}$ falls within a threshold distance d_{pr} to a previous pose $\mathbf{T}_{W,R_l}$, the identifier of that pose, l, as well as $\mathbf{T}_{R_l,R_k}$ are provided to the simulated robot. To prevent self-matches, all poses within $1.5 \cdot d_{\text{pr}}$ of pose graph traversal are excluded from place recognition. Also, place recognition is only provided if $\mathbf{T}_{W,R_l}$ is in line of sight from $\mathbf{T}_{W,R_k}$, to prevent place recognition across occlusions.

F.6.2 Baseline Implementation

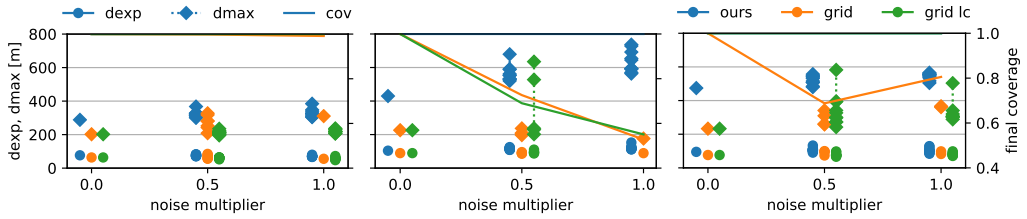
As a baseline, we implement a simplified version of the grid representation [211] using a regular grid instead of octrees, as we do not take computational performance into account. According to [145, 211], the occupation probability $P(n|z_{1:k})$ of each cell is updated using the most recent measurement z_k with

$$L(n|z_{1:k}) = L(n|z_{1:k-1}) + L(n|z_k), \quad L(n) = \lim_{x \to P(n)} \log(\frac{x}{1-x}). \tag{F.6}$$

We assume a precise depth sensor and let $P(n|z_k) = 1$ if a ray hits an obstacle inside n at time k, $P(n|z_k) = 0$ if cell n is fully contained inside the polygon spanned by the sensor rays as defined in Section F.4.1, and $P(n|z_k) = 0.5$ otherwise. Consequently, $P(n)$ can only assume values $\in \{0, 0.5, 1\}$, which we accordingly label as *free*, *unknown* and *occupied*. For cases where (F.6) is undefined, later measurements override earlier measurements. This baseline is implemented in two ways: a naive one ("*grid*"), which uses the state estimate $\bar{\mathsf{T}}_{W,R_k}$ as-is, and one with loop closure capability ("*grid-lc*"). The latter exploits place recognition events to optimize its pose-graph according to all relative pose estimates present in the pose graph using g2o [110]. Every place recognition triggers an optimization after which all updates according to (F.6) are executed anew with the optimized pose estimates $\bar{\mathsf{T}}^{\star}_{W,R_k}$. We use a voxel length of 1m for all maps.

F.6.3 Evaluation Metrics

To quantify the performance of exploration, we measure the distance traveled until the robot *believes* full coverage has been achieved, d_{max}, and the expected distance to be traveled to discover an arbitrary location in free space, d_{exp}. d_{exp} reflects the general speed of exploration. It is not severely affected if the robot takes a very long time at the end to navigate to the last couple of frontiers. Depending on the application, one metric is more important than the other. For more insight into the progress of exploration, we track the ratio between the volume of known free space $\bar{\mathcal{V}}_{\text{free}}$ and the full free space, $\mathcal{V}_{\text{free}}$, $|\bar{\mathcal{V}}_{\text{free}}|/|\mathcal{V}_{\text{free}}|$, which we call *coverage ratio*. It is initially 0 and reaches 1 when full coverage is achieved. Note that $\bar{\mathcal{V}}_{\text{free}}$ refers to the ground truth volume covered by the robot, not the estimate of that space by the robot. $\mathcal{V}_{\text{free}}$ is represented using a grid – in our simulations we use this grid to define the environment in the first place, see Figure F.5. Unlike in the baseline grid representation, occupied cells are known beforehand, and only free cells $n \in \mathcal{V}_{\text{free}}$ are labeled as *known free* or *unknown free*. Known free space $\bar{\mathcal{V}}_{\text{free}}$ is approximated as the set of cells labeled *known free*. An unknown free cell is relabeled *known free* as soon as any sensor ray intersects it and remains *known free* until the end of the simulation. Given this approximation of $\bar{\mathcal{V}}_{\text{free}}$, the coverage ratio is calculated by dividing the count of cells in $\bar{\mathcal{V}}_{\text{free}}$ by the count of cells in $\mathcal{V}_{\text{free}}$. Since full coverage will not be reached in all experiments, even if the



(a) Open **(b)** Maze **(c)** Forest house

Figure F.6 – Expected distance until discovery d_{exp} and distance until termination d_{max} versus final coverage for the evaluated approaches, with different noise intensity, on different maps. For fair comparison, d_{exp} and d_{max} are omitted for samples where the final coverage is below 1. Ten samples are collected for every setting.

robot believes this to be the case, we also report the final coverage ratio. In these cases, d_{exp} is undefined.

F.6.4 Experiments

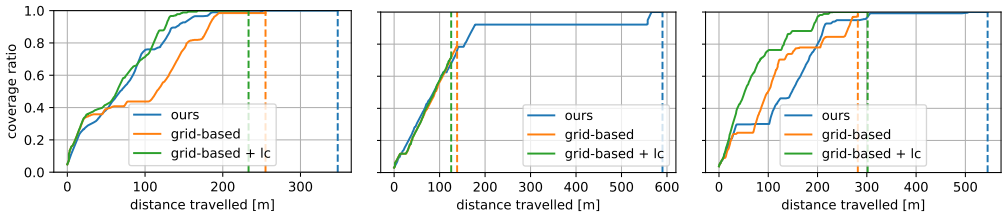
We perform experiments with the following parameter combinations: simulation in either of the environments shown in Figure F.5; pose estimate noise simulated with

$$\sigma_{\text{pos}} = \alpha \cdot 0.1m, \quad \sigma_{\text{rot}} = \alpha \cdot 5^{\circ}, \tag{F.7}$$

with the noise multiplier $\alpha \in \{0, 0.5, 1\}$; and place recognition radius d_{pr} of $0.5, 1, 1.5$ or 2 times the sensor range d_{FOV}. For every parameter setting, ten different runs have been performed (due to the stochastic nature of the simulated noise). The place recognition radius is varied because a larger radius should result in the robot having to travel less in order to consolidate frontiers. It is limited to two times the sensor range as beyond that, polygon intersection between $\mathcal{V}_{\text{fov}}(\mathsf{T}_{W,R_k})$ and $\mathcal{V}_{\text{fov}}(\mathsf{T}_{W,R_l})$ is impossible.

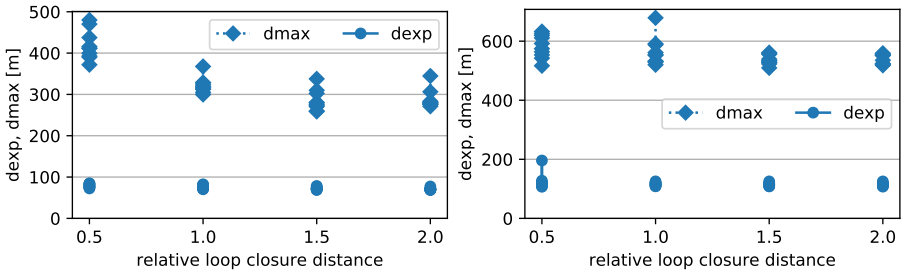
F.7 Results

Figure F.6 shows the performance of the evaluated methods as pose estimation noise is increased. As we can see, only the proposed representation always reaches full coverage in all environments. The other approaches perform particularly poorly in the maze environment, where the robot has to travel long distances in close proximity without the ability to close loops, see Figure F.1. We can also see that this comes at a cost of longer distances covered until the robot believes coverage is achieved. d_{max} is between 1.5 and 3 times larger for our approach, d_{exp}, however, only up to 1.5 larger. This happens because the proposed approach needs to consolidate frontiers of all spatially close polygons through place recognition, while the grid-based representation is able to



(a) Open **(b)** Maze **(c)** Forest house

Figure F.7 – True coverage as a function of distance travelled for individual runs at noise multiplier $\alpha = 1$. Vertical dashed lines indicate the distance at which the robot believes coverage to be complete according to its own representation.



(a) Open **(b)** Maze

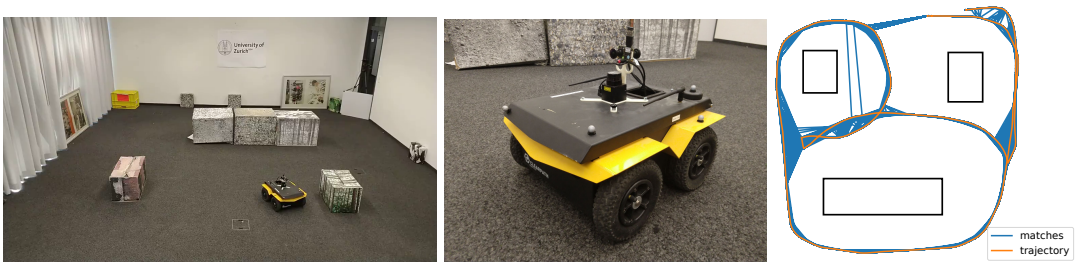
Figure F.8 – d_{exp} and d_{max} when using our representation in the *open* and *maze* environments with noise multiplier $\alpha = 0.5$, for different relative loop closure distances β. Place recognition is provided by the simulator for any two poses when the distance between them is below β times the depth sensor range.

consolidate frontiers from measurements even if the relative pose established between them comes from an estimate integrated over a very long path. A typical evolution of the coverage ratios for one sample of each method and noise multiplier $\alpha = 1$ is shown in Figure F.7.

Figure F.8 shows how our method is affected by the loop closure distance d_{pr}. As can be seen, a larger distance at which places can be recognized leads to faster exploration times in the open environment. This makes sense, as a larger loop closure distance leads to generally larger frontier consolidation scopes, allowing frontiers to be removed faster. Exploration speed in the maze environment, however, is not significantly affected, as there are generally not many place recognition events happening in that environment.

F.8 Validation in the Real World

We validate that our approach works in the real world with the experimental setup shown in Figure F.9. The platform is a ClearPath *Jackal* equipped with a Hokuyo



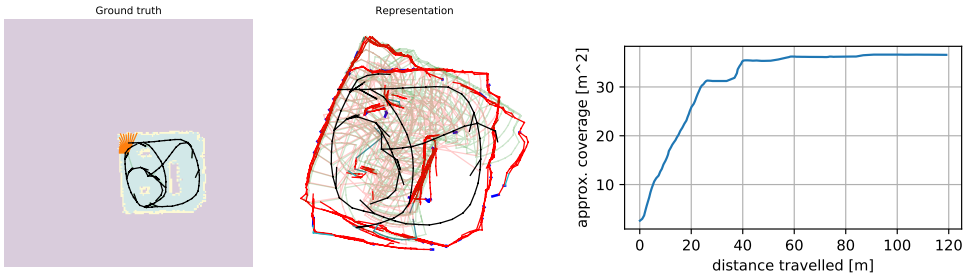
(a) (b) (c)

Figure F.9 – (a) Real environment in which we have validated our approach. (b) Close-up of our experimental platform. (c) NetVLAD matches on a trajectory circumnavigating all obstacles. Obstacle locations are approximate.

laser scanner as depth sensor and two fisheye cameras that provide a 360° view of the environment. Ground truth is obtained with a motion tracking system using reflective IR markers. For the state estimate, we use the wheel odometry provided by the Jackal robot. We intentionally do not use the best available state estimate, to demonstrate robustness to drift.

The most important assumption that we have made in our simulations is that sufficient place recognition and relative pose estimation can be provided. To this end, we use visual place recognition from a panoramic image stitched from the two fisheye cameras. The vertical field of view is restricted to prevent place recognition from structure that is visible from everywhere in the room. Using a 360° view allows place recognition independent of the orientation of the robot between the two matched places. First, the CNN full image desciptor NetVLAD [7] is used to quickly determine the visually most similar previously recorded image. Figure F.9 (c) shows NetVLAD matches in a trajectory where the robot drives once around each obstacle. As can be seen, all place recognitions occur locally, with only two matches across a longer distance, but still within line of sight. Given a NetVLAD match, we match ORB features [170] and use P3P [74] and RANSAC [64] for geometric verification and relative pose estimation. For P3P, the 3D locations of features in each frame are triangulated using KLT tracks [125] of those features in subsequent frames. The only component that we have not implemented, as it is out of the scope of this project, is teach-and-repeat [61]. To simulate teach-and-repeat, we instead use the motion tracking system to let the robot backtrack its trajectory. This, and ground truth for evaluation, are the only things for which the motion tracking system is used.

Figure F.10 (a) shows the final state of exploration in our experiment. As we can see, there is drift in the estimated trajectory, yet our approach still manages to fully cover the environment. Note that there are parts of the environment where trajectories overlap. These are locations where our visual relative pose estimator fails to obtain enough feature matches. However, as long as relative pose estimations are obtained within



(a) (b)

Figure F.10 – (a) Ground truth and representation once exploration is complete. (b) Corresponding coverage over time, obtained from the ground truth grid.

the distance of the consolidation scope (see polygons shown in Figure F.10 (a)), this does not pose a significant problem. Figure F.10 (b) shows the corresponding coverage over the travelled distance. The behaviour is consistent with the results obtained in simulation, wherein the robot first quickly covers most of the map, after which it does a lot of backtracking to seek out frontiers that remain in the map.

F.9 Conclusion

In this paper, we present a map representation that allows exploration in spite of large drift in the pose estimate. We show that global consistency in the pose graph is not required to determine if exploration is complete. This alleviates the need for map optimization, which is particularly interesting for multi-robot exploration. In addition, the proposed method can be adapted to algorithms that currently use different representations. Using a state-of-the-art exploration algorithm, we compare our representation to a grid-based representation. In contrast to the latter, and at a cost of longer exploration time, all of the free space can be fully covered with our representation, even with large drift in the state estimate.

Acknowledgments

This work was supported by the National Centre of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant.

Bibliography

[1] H. Aanæs, A. Dahl, and K. Steenstrup Pedersen. "Interesting Interest Points". In: *Int. J. Comput. Vis.* (2012), pp. 18–35.

[2] B. C. Akdeniz and H. I. Bozma. "Exploration and topological map building in unknown environments". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1079–1084. DOI: 10.1109/icra.2015.7139310.

[3] M. Aly, M. Munich, and P. Perona. "Distributed kd-trees for retrieval from very large image collections". In: *British Mach. Vis. Conf. (BMVC)*. 2011.

[4] L. A. A. Andersson and J. Nygards. "C-SAM: Multi-Robot SLAM using square root information smoothing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2008, pp. 2798–2805.

[5] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues. "Multi-agent localization from noisy relative pose measurements". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011, pp. 364–369. DOI: 10.1109/ICRA.2011.5979799.

[6] R. Aragues, E. Montijano, and C. Sagues. "Consistent data association in multi-robot systems with limited communications". In: *Robotics: Science and Systems (RSS)*. 2010. DOI: 10.15607/RSS.2010.VI.013.

[7] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. "NetVLAD: CNN Architecture for Weakly Supervised Place Recognition". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2016, pp. 5297–5307. DOI: 10.1109/CVPR.2016.572.

[8] R. Arandjelović and A. Zisserman. "All About VLAD". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2013, pp. 1578–1585. DOI: 10.1109/CVPR.2013.207.

[9] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. ACM press New York, 1999.

[10] A. Bahr, M. R. Walter, and J. J. Leonard. "Consistent cooperative localization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2009, pp. 3415–3422. DOI: 10.1109/ROBOT.2009.5152859.

[11] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. F. Durrant-Whyte. "Decentralised cooperative localisation for heterogeneous teams of mobile robots". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011, pp. 2859–2865. DOI: 10.1109/ICRA.2011.5979850.

[12] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *Int. J. Comput. Vis.* 56.3 (2004), pp. 221–255.

[13] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. "SURF: Speeded up robust features". In: *Comput. Vis. Image. Und.* 110.3 (2008), pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014.

[15] J. L. Bentley. "Multidimensional Binary Search Trees in Database Applications". In: *IEEE Transactions on Software Engineering* SE-5.4 (1979), pp. 333–340. DOI: 10.1109/TSE.1979.234200.

[16] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[17] A. Bhowmik, S. Gumhold, C. Rother, and E. Brachmann. "Reinforced Feature Points: Optimizing Feature Detection and Description for a High-Level Task". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2020. DOI: 10.1109/CVPR42600.2020.00500.

[18] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez. "The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario". In: *Int. J. Robot. Research* 33.2 (2014), pp. 207–214. DOI: 10.1177/0278364913507326.

[19] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015.

[20] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. "An Atlas framework for scalable mapping". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 2. 2003, pp. 1899–1906. DOI: 10.1109/robot.2003.1241872.

[21] M. Bosse and R. Zlot. "Keypoint design and evaluation for place recognition in 2D lidar maps". In: *J. Robot. and Auton. Syst.* (2009).

[22] G. Bresson, R. Aufrère, and R. Chapuis. "Consistent multi-robot decentralized SLAM with unknown initial positions". In: *Int. Conf. Inf. Fusion (FUSION)*. July 2013, pp. 372–379.

[23] R. Brooks. "Visual map making for a mobile robot". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 2. 1985, pp. 824–829. DOI: 10.1109/ROBOT.1985.1087348.

[24] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. "The EuRoC micro aerial vehicle datasets". In: *Int. J. Robot. Research* 35.10 (2015), pp. 1157–1163. DOI: 10.1177/0278364915620033.

[25] A. Caccavale and M. Schwager. "Wireframe Mapping for Resource-Constrained Robots". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8594057.

[26] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Trans. Robot.* 32.6 (2016), pp. 1309–1332.

[27] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. "BRIEF: Computing a Local Binary Descriptor Very Fast". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.7 (2012), pp. 1281–1298.

[28] S. Cao and N. Snavely. "Minimal Scene Descriptions from Structure from Motion Models". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2014, pp. 461–468. DOI: 10.1109/CVPR.2014.66.

[29] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri. "Simultaneous Localization and Mapping Using Rao-Blackwellized Particle Filters in Multi Robot Systems". In: *J. Intell. Robot. Syst.* 63.2 (Aug. 2011), pp. 283–307. DOI: 10.1007/s10846-010-9457-0.

[30] L. Carlone, D. M. Rosen, G. Calafiore, J. J. Leonard, and F. Dellaert. "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015, pp. 125–132. DOI: 10.1109/IROS.2015.7353364.

[31] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert. "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models". In: *Int. J. Robot. Research* 36.12 (2017), pp. 1286–1311. DOI: 10.1177/0278364917732640.

[32] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert. "Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed Gauss-Seidel approach". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016, pp. 5261–5268. DOI: 10.1109/ICRA.2016.7487736.

[33] T. Cieslewski, L. Simon, M. Dymczyk, S. Magnenat, and R. Siegwart. "Map API - Scalable Decentralized Map Building for Robots". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015.

[34] T. Cieslewski, M. Bloesch, and D. Scaramuzza. "Matching Features without Descriptors: Implicitly Matched Interest Points". In: *British Mach. Vis. Conf. (BMVC)*. 2019.

[35] T. Cieslewski, S. Choudhary, and D. Scaramuzza. "Data-Efficient Decentralized Visual SLAM". In: *IEEE Int. Conf. Robot. Autom. (ICRA)* (2018). DOI: 10.1109/ICRA.2018.8461155.

[36] T. Cieslewski, K. G. Derpanis, and D. Scaramuzza. "SIPs: Succinct Interest Points from Unsupervised Inlierness Probability Learning". In: *3D Vision (3DV)* (2019). DOI: 10.1109/3DV.2019.00072.

[37] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. "Rapid exploration with multi-rotors: A frontier selection method for high speed flight". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017, pp. 2135–2142. DOI: 10.1109/IROS.2017.8206030.

[38] T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition From Full-Image Descriptors". In: *IEEE Int. Symp. on Multi-Robot and Multi-Agent Sys.* (Dec. 2017), pp. 78–82. DOI: 10.1109/MRS.2017.8250934.

[39] T. Cieslewski and D. Scaramuzza. "Efficient Decentralized Visual Place Recognition Using a Distributed Inverted Index". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 640–647. DOI: 10.1109/LRA.2017.2650153.

[40] T. Cieslewski, E. Stumm, A. Gawel, M. Bosse, S. Lynen, and R. Siegwart. "Point cloud descriptors for place recognition using sparse visual information". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016, pp. 4830–4836.

[41] T. Cieslewski, A. Ziegler, and D. Scaramuzza. "Exploration Without Global Consistency Using Local Volume Consolidation". In: *Proc. Int. Symp. Robot. Research (ISRR)* (2019).

[42] C. Connolly et al. "The determination of next best views". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 2. 1985, pp. 432–435.

[43] L. Contreras and W. Mayol-Cuevas. "O-POCO: Online point cloud compression mapping for visual odometry and SLAM". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 4509–4514. DOI: 10.1109/ICRA.2017.7989523.

[44] Y. Cui and S. S. Ge. "Autonomous vehicle positioning with GPS in urban canyon environments". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2001, pp. 1105–1110. DOI: 10.1109/ROBOT.2001.932759.

[45] A. Cunningham, V. Indelman, and F. Dellaert. "DDF-SAM 2.0: Consistent Distributed Smoothing and Mapping". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2013.

[46] A. Cunningham, M. Paluri, and F. Dellaert. "DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2010, pp. 3025–3030. DOI: 10.1109/IROS.2010.5652875.

[47] A. Cunningham, K. M. Wurm, W. Burgard, and F. Dellaert. "Fully distributed scalable smoothing and mapping with robust multi-robot data association". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012, pp. 1093–1100. DOI: 10.1109/ICRA.2012.6225356.

[48] Z. Danping and T. Ping. "CoSLAM: Collaborative Visual SLAM in Dynamic Environments". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2012).

[49] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2019. DOI: 10.1109/ICRA.2019.8793887.

[50] J. Delmerico and D. Scaramuzza. "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2018.

[51] D. DeTone, T. Malisiewicz, and A. Rabinovich. "Self-Improving Visual Odometry". In: *arXiv e-prints* (Dec. 2018). URL: https://arxiv.org/abs/1812.03245.

[52] D. DeTone, T. Malisiewicz, and A. Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2018.

[53] G. Di Caro and M. Dorigo. "AntNet: Distributed stigmergetic control for communications networks". In: *Journal of Artificial Intelligence Research* (1998).

[54] P. Dias, A. A. Kassim, and V. Srinivasan. "A neural network based corner detection method". In: *IEEE Int. Conf. Neural Netw.* Vol. 4. 1995, pp. 2116–2120.

[55] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert. "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 5807–5814. DOI: 10.1109/ICRA.2015.7140012.

[56] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[57] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale. "The gist of maps - Summarizing experience for lifelong localization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015.

[58] M. Dymczyk, E. Stumm, J. Nieto, R. Siegwart, and I. Gilitschenski. "Will it last? Learning Stable Features for Long-Term Visual Localization". In: *3D Vision (3DV)*. 2016, pp. 572–581. DOI: 10.1109/3DV.2016.66.

[59] J. Engel, J. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2014, pp. 834–849.

[60] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.3 (Mar. 2018), pp. 611–625. DOI: 10.1109/TPAMI.2017.2658577.

[61] S. K. van Es and T. D. Barfoot. "Being in Two Places at Once: Smooth Visual Path Following on Globally Inconsistent Pose Graphs". In: *Conf. Comput. Robot Vis. (CRV)*. 2015. DOI: 10.1109/crv.2015.17.

[62] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard. "The MIT Stata Center dataset". In: *Int. J. Robot. Research* 32.14 (2013), pp. 1695–1699. DOI: 10.1177/0278364913509035.

[63] G. Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conf. on Im. Analysis (SCIA)*. 2003, pp. 363–370.

[64] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (1981), pp. 381–395. DOI: 10.1145/358669.358692.

[65] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza. "AlphaPilot: Autonomous Drone Racing". In: *Robotics: Science and Systems (RSS)*. July 2020. DOI: 10.15607/RSS.2020.XVI.081.

[66] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. "Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013, pp. 3962–3970. DOI: 10.1109/IROS.2013.6696923.

[67] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems". In: *IEEE Trans. Robot.* 33.2 (2017), pp. 249–265. DOI: 10.1109/TRO.2016.2623335.

[68] M. Franceschelli and A. Gasparri. "On agreement problems with gossip algorithms in absence of common reference frames". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2010, pp. 4481–4486. DOI: 10.1109/ROBOT.2010.5509788.

[69] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. "The sensor-based random graph method for cooperative robot exploration". In: *IEEE/ASME Transactions on Mechatronics* (2009).

[70] F. Fraundorfer and D. Scaramuzza. "Visual Odometry. Part II: Matching, Robustness, Optimization, and Applications". In: *IEEE Robot. Autom. Mag.* (2012). DOI: 10.1109/MRA.2012.2182810.

[71] P. Furgale and T. D. Barfoot. "Visual teach and repeat for long-range rover autonomy". In: *J. Field Robot.* 27.5 (2010), pp. 534–560. DOI: 10.1002/rob.20342.

[72] M. Gadd and P. Newman. "Checkout my map: Version control for fleetwide visual localisation". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 5729–5736. DOI: 10.1109/IROS.2016.7759843.

[73] D. Gálvez-López and J. D. Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Trans. Robot.* 28.5 (Oct. 2012), pp. 1188–1197. DOI: 10.1109/TRO.2012.2197158.

[74] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. "Complete solution classification for the perspective-three-point problem". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 25.8 (Aug. 2003), pp. 930–943. DOI: 10.1109/TPAMI.2003.1217599.

[75] M. Gassner, T. Cieslewski, and D. Scaramuzza. "Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 5196–5202. DOI: 10.1109/ICRA.2017.7989609.

[76] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: The KITTI dataset". In: *Int. J. Robot. Research* 32.11 (2013), pp. 1231–1237. DOI: 10.1177/0278364913491297.

[77] C. Godard, O. M. Aodha, M. Firman, and G. Brostow. "Digging Into Self-Supervised Monocular Depth Estimation". In: *Int. Conf. Comput. Vis. (ICCV)*. 2019, pp. 3827–3837. DOI: 10.1109/ICCV.2019.00393.

[78] H. H. González-Baños and J.-C. Latombe. "Navigation Strategies for Exploring Indoor Environments". In: *Int. J. Robot. Research* 21.10–11 (Oct. 2002), pp. 829–848. DOI: 10.1177/027836402021010834.

[79] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. "Depth From Videos in the Wild: Unsupervised Monocular Depth Learning From Unknown Cameras". In: *Int. Conf. Comput. Vis. (ICCV)*. 2019, pp. 8976–8985. DOI: 10.1109/ICCV.2019.00907.

[80] W. N. Greene and N. Roy. "FLaME: Fast Lightweight Mesh Estimation Using Variational Smoothing on Delaunay Graphs". In: *Int. Conf. Comput. Vis. (ICCV)*. 2017, pp. 4696–4704. DOI: 10.1109/ICCV.2017.502.

[81] A. Guiducci. "Corner characterization by differential geometry techniques". In: *Pattern Recognition Letters* 8.5 (1988), pp. 311–318.

[82] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. "MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2015.

[83] B. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle. "Review and analysis of solutions of the three point perspective pose estimation problem". In: *Int. J. Comput. Vis.* 13.3 (1994), pp. 331–356.

[84] C. Harris and M. Stephens. "A combined corner and edge detector". In: *Proc. Fourth Alvey Vision Conf.* Vol. 15. 1988, pp. 147–151. DOI: 10.5244/C.2.23.

[85] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd Edition. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511811685.

[86] W. Hartmann, M. Havlena, and K. Schindler. "Predicting matchability". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2014, pp. 9–16.

[87] D. Holz, N. Basilico, F. Amigoni, and S. Behnke. "Evaluating the Efficiency of Frontier-based Exploration Strategies". In: *Int. Symp. Robotics (ISR)*. 2010, pp. 1–8.

[88] B. K. Horn and B. G. Schunck. "Determining optical flow". In: *J. Artificial Intell.* 17.1 (1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

[89] A. Howard. "Multi-robot Simultaneous Localization and Mapping using Particle Filters". In: *Int. J. Robot. Research* 25.12 (2006), pp. 1243–1256.

[90] A. Howard, L. E. Parker, and G. S. Sukhatme. "Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection". In: *Int. J. Robot. Research* 25.5–6 (2006), pp. 431–447. DOI: 10.1177/0278364906065378.

[91] A. Howard, G. S. Sukhatme, and M. J. Matarić. "Multi-Robot Mapping using Manifold Representations". In: *Proc. IEEE* 94.9 (July 2006), pp. 1360–1369.

[92] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. "Graph-based distributed cooperative navigation for a general multi-robot measurement model". In: *Int. J. Robot. Research* 31.9 (2012), pp. 1057–1080. DOI: 10.1177/0278364912446325.

[93] V. Indelman, E. Nelson, J. Dong, N. Michael, and F. Dellaert. "Incremental Distributed Inference from Arbitrary Poses and Unknown Data Association: Using Collaborating Robots to Establish a Common Reference". In: *IEEE Control Syst. Mag.* (2016).

[94] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. Davison, and A. Fitzgibbon. "KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction". In: *SIGGRAPH*. Aug. 2011, p. 23.

[95] A. Jacobson, W. Scheirer, and M. Milford. "Deja vu: Scalable Place Recognition Using Mutually Supportive Feature Frequencies". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)* (2017).

[96] H. Jegou, M. Douze, and C. Schmid. "Product Quantization for Nearest Neighbor Search". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.1 (Jan. 2011), pp. 117–128. DOI: 10.1109/TPAMI.2010.57.

[97] H. Jegou, M. Douze, C. Schmid, and P. Pérez. "Aggregating local descriptors into a compact image representation." In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2010, pp. 3304–3311. DOI: 10.1109/CVPR.2010.5540039.

[98] B.-S. Jeong and E. Omiecinski. "Inverted file partitioning schemes in multiple disk systems". In: *IEEE Transactions on Parallel and Distributed Systems* (1995).

[99] R. Ji, L.-Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao. "Learning to distribute vocabulary indexing for scalable visual search". In: *IEEE Transactions on Multimedia* (2013).

[100] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *Int. J. Robot. Research* 31.2 (Feb. 2012), pp. 217–236.

[101] M. Keller, Z. Chen, F. Maffra, P. Schmuck, and M. Chli. "Learning Deep Descriptors With Scale-Aware Triplet Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[102] A. Kendall, M. Grimes, and R. Cipolla. "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization". In: *Int. Conf. Comput. Vis. (ICCV)*. 2015, pp. 2938–2946. DOI: 10.1109/ICCV.2015.336.

[103] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. "Multiple relative pose graphs for robust cooperative mapping". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2010, pp. 3185–3192. DOI: 10.1109/ROBOT.2010.5509154.

[104] D. P. Kingma and J. L. Ba. "Adam: A Method for Stochastic Optimization". In: *Int. Conf. Learn. Representations (ICLR)* (2015).

[105] L. Kneip, D. Scaramuzza, and R. Siegwart. "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2011, pp. 2969–2976. DOI: 10.1109/CVPR.2011.5995464.

[106] J. Knuth and P. Barooah. "Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2013, pp. 1534–1539. DOI: 10.1109/ICRA.2013.6630774.

[107] J. Komorowski, K. Czarnota, T. Trzcinski, L. Dabla, and S. Lynen. "Interest point detectors stability evaluation on ApolloScape dataset". In: *Eur. Conf. Comput. Vis. Workshops (ECCVW)* (Sept. 2018).

[108] D. Koutsonikolas, Y. C. Hu, and C.-C. Wang. "Pacifier: High-throughput, reliable multicast without crying babies in wireless mesh networks". In: *IEEE/ACM Transactions on Networking (TON)* (2012).

[109] E. Kruppa. "Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung". In: *Sitzungsberichte der Akademie der Wissenschaften, Wien, Mathematisch-Naturwissenschaftlichen Klasse, Abteilung IIa*. Vol. 122. 1913, pp. 1939–1948.

[110] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A General Framework for Graph Optimization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011.

[111] P.-Y. Lajoie, S. Hu, G. Beltrame, and L. Carlone. "Modeling Perceptual Aliasing in SLAM via Discrete–Continuous Graphical Models". In: *IEEE Robot. Autom. Lett.* 4.2 (2019), pp. 1232–1239. DOI: 10.1109/LRA.2019.2894852.

[112] P.-Y. Lajoie, B. Ramtoula, Y. Chang, L. Carlone, and G. Beltrame. "DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams". In: *IEEE Robot. Autom. Lett.* 5.2 (2020), pp. 1656–1663.

[113] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". In: *Commun. ACM* 21.7 (July 1978), pp. 558–565. DOI: 10.1145/359545.359563.

[114] M. T. Lázaro, L. M. Paz, P. Piniés, J. A. Castellanos, and G. Grisetti. "Multi-robot SLAM using condensed measurements". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013, pp. 1069–1076. DOI: 10.1109/IROS.2013.6696483.

[115] K. Lenc and A. Vedaldi. "Large scale evaluation of local image feature detectors on homography datasets". In: *British Mach. Vis. Conf. (BMVC)*. 2018.

[116] K. Lenc and A. Vedaldi. "Learning Covariant Feature Detectors". In: *Eur. Conf. Comput. Vis. Workshops (ECCVW)*. 2016, pp. 100–117.

[117] S. Leutenegger, M. Chli, and R. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *Int. Conf. Comput. Vis. (ICCV)*. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[118] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization". In: *Int. J. Robot. Research* (2015).

[119] R. Li, S. Wang, Z. Long, and D. Gu. "UnDeepVO: Monocular Visual Odometry Through Unsupervised Deep Learning". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2018, pp. 7286–7291. DOI: 10.1109/ICRA.2018.8461251.

[120] Y. Li, N. Snavely, and D. P. Huttenlocher. "Location Recognition Using Prioritized Feature Matching". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2010, pp. 791–804. DOI: 10.1007/978-3-642-15552-9_57.

[121] X. Lin, Y. Shen, L. Cai, and R. Ji. "The Distributed System for Inverted Multi-index Visual Retrieval". In: *Neurocomputing* (2016).

[122] H. C. Longuet-Higgins and K. Prazdny. "Fast, robust, continuous monocular egomotion computation". In: *Proc. R. Soc. Lond.* July 1980, pp. 385–397. DOI: 10.1098/rspb.1980.0057.

[123] A. Loquercio, M. Dymczyk, B. Zeisl, S. Lynen, I. Gilitschenski, and R. Siegwart. "Efficient descriptor learning for large scale localization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 3170–3177. DOI: 10.1109/ICRA.2017.7989359.

[124] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *Int. J. Comput. Vis.* 60.2 (Nov. 2004), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94.

[125] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Int. Joint Conf. Artificial Intell. (IJCAI)*. 1981, pp. 674–679.

[126] S. Lynen, M. Bosse, P. Furgale, and R. Siegwart. "Placeless Place-Recognition". In: *3D Vision (3DV)*. Dec. 2014, pp. 303–310. DOI: 10.1109/3DV.2014.36.

[127] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart. "Get Out of My Lab: Large-scale, Real-Time Visual-Inertial Localization". In: *Robotics: Science and Systems (RSS)*. 2015. DOI: 10.15607/RSS.2015.XI.037.

[128] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. "1 Year, 1000km: The Oxford RobotCar Dataset". In: *Int. J. Robot. Research* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498.

[129] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. "1 Year, 1000 km: The Oxford RobotCar dataset." In: *Int. J. Robot. Research* 36.1 (2017), pp. 3–15.

[130] P. Mainali, G. Lafruit, Q. Yang, B. Geelen, L. V. Gool, and R. Lauwereins. "SIFER: scale-invariant feature detector with error resilience". In: *Int. J. Comput. Vis.* 104.2 (2013), pp. 172–197.

[131] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan. "Pairwise Consistent Measurement Set Maximization for Robust Multi-Robot Map Merging". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2018, pp. 2916–2923. DOI: 10.1109/ICRA.2018.8460217.

[132] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon. "Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment". In: *Science Robotics* 4.35 (2019). DOI: 10.1126/scirobotics.aaw9710.

[133] S. Melink, S. Raghavan, B. Yang, and H. Garcia-Molina. "Building a distributed full-text index for the web". In: *ACM Transactions on Information Systems* (2001).

[134] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas. "Maintaining connectivity in mobile robot networks". In: *Int. Symp. Experimental Robotics (ISER)*. 2009.

[135] K. Mikolajczyk and C. Schmid. "Scale & Affine Invariant Interest Point Detectors". In: *Int. J. Comput. Vis.* 60.1 (2004), pp. 63–86. DOI: 10.1023/B:VISI.0000027790.02288.f2.

[136] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. "A Comparison of Affine Region Detectors". In: *Int. J. Comput. Vis.* 65.1 (2005), pp. 43–72. DOI: 10.1007/s11263-005-3848-x.

[137] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.10 (2005), pp. 1615–1630.

[138] K. Mikolajczyk and C. Schmid. "An Affine Invariant Interest Point Detector". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2002, pp. 128–142. DOI: 10.1007/3-540-47969-4_9.

[139] M. J. Milford and G. F. Wyeth. "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012, pp. 1643–1649. DOI: 10.1109/ICRA.2012.6224623.

[140] A. Millane, Z. Taylor, H. Oleynikova, J. I. Nieto, R. Siegwart, and C. Cadena. "TSDF Manifolds: A Scalable and Consistent Dense Mapping Approach". In: *arXiv e-prints* (Oct. 2017). URL: http://arxiv.org/abs/1710.07242.

[141] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. "Working hard to know your neighbors margins: Local descriptor learning loss". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2017, pp. 4826–4837.

[142] E. Montijano, R. Aragues, and C. Sagüés. "Distributed data association in robotic networks with cameras and limited communications". In: *IEEE Trans. Robot.* (2013).

[143] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T. Au, and S. J. Kim. "Challenges and implemented technologies used in autonomous drone racing". In: *Springer: Intelligent Service Robotics* 12 (2019), pp. 137–148. DOI: 10.1007/s11370-018-00271-6. URL: https://link.springer.com/article/10.1007/s11370-018-00271-6.

[144] H. P. Moravec. "Obstacle Avoidance and Navigation in the Real World by Seeing Robot Rover". PhD thesis. Pittsburgh, Pennsylvania: Carnegie-Mellon University, Sept. 1980.

[145] H. Moravec and A. Elfes. "High resolution maps from wide angle sonar". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 2. 1985, pp. 116–121. DOI: 10.1109/ROBOT.1985.1087316.

[146] J. G. Morrison, D. Gálvez-López, and G. Sibley. "MOARSLAM: Multiple Operator Augmented RSLAM". In: 2016.

[147] D. Mukherjee, Q. M. J. Wu, and G. Wang. "A comparative experimental study of image feature detectors and descriptors". In: *Mach. Vis. and Applications* 26.4 (May 2015), pp. 443–466. DOI: 10.1007/s00138-015-0679-9.

[148] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: a Versatile and Accurate Monocular SLAM System". In: *IEEE Trans. Robot.* 31.5 (2015), pp. 1147–1163. DOI: 10.1109/TRO.2015.2463671.

[149] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Trans. Robot.* 33.5 (Oct. 2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[150] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli. "Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2009.

[151] D. Nister and H. Stewenius. "Scalable Recognition with a Vocabulary Tree". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. Vol. 2. 2006, pp. 2161–2168. DOI: 10.1109/CVPR.2006.264.

[152] D. Nistér. "An efficient solution to the five-point relative pose problem". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.6 (2004), pp. 756–777. DOI: 10.1109/TPAMI.2004.17.

[153] A. Oertel, T. Cieslewski, and D. Scaramuzza. "Augmenting Visual Place Recognition With Structural Cues". In: *IEEE Robot. Autom. Lett.* 5.4 (2020), pp. 5534–5541. DOI: 10.1109/LRA.2020.3009077.

[154] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. "Signed distance fields: A natural representation for both mapping and planning". In: *RSS Workshop: Geometry and Beyond - Representations, Physics, and Scene Understanding for Robotics*. 2016.

[155] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. "LF-Net: Learning Local Features from Images". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2018, pp. 6234–6244.

[156] C. Papachristos, S. Khattak, and K. Alexis. "Uncertainty-aware receding horizon exploration and mapping using aerial robots". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 4568–4575. DOI: 10.1109/ICRA.2017.7989531.

[157] H. S. Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen. "3D point cloud reduction using mixed-integer quadratic programming". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2013, pp. 229–236.

[158] L. Paull, G. Huang, M. Seto, and J. J. Leonard. "Communication-constrained multi-AUV cooperative SLAM". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 509–516. DOI: 10.1109/ICRA.2015.7139227.

[159] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Object retrieval with large vocabularies and fast spatial matching." In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2007. DOI: 10.1109/CVPR.2007.383172.

[160] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Trans. Robot.* 34.4 (2018), pp. 1004–1020. DOI: 10.1109/TRO.2018.2853729.

[161] A. Quraishi, T. Cieslewski, S. Lynen, and R. Siegwart. "Robustness to connectivity loss for collaborative mapping". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 4580–4585. DOI: 10.1109/IROS.2016.7759674.

[162] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A Scalable Content-Addressable Network". In: *ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. 2001, pp. 161–172. DOI: 10.1145/383059.383072.

[163] P. Reynolds and A. Vahdat. "Efficient peer-to-peer keyword searching". In: *ACM/IFIP/USENIX Int. Conf. on Middleware*. 2003.

[164] L. Riazuelo, J. Civera, and J. M. M. Montiel. "C2TAM: A Cloud framework for cooperative tracking and mapping". In: *J. Robot. and Auton. Syst.* 62.4 (2014), pp. 401–413. DOI: https://doi.org/10.1016/j.robot.2013.11.007.

[165] A. Richardson and E. Olson. "Learning convolutional filters for interest point detection". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2013, pp. 631–637. DOI: 10.1109/ICRA.2013.6630639.

[166] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. "Neighbourhood Consensus Networks". In: *Conf. Neural Inf. Process. Syst. (NIPS)* (2018), pp. 1651–1662.

[167] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2006, pp. 430–443. DOI: 10.1007/11744023_34.

[168] S. I. Roumeliotis and G. A. Bekey. "Distributed multirobot localization". In: *IEEE Trans. Robot. Autom.* 18.5 (2002), pp. 781–795.

[169] A. Rowstron and P. Druschel. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In: *Middleware*. Springer Berlin Heidelberg, 2001, pp. 329–350. DOI: 10.1007/3-540-45518-3_18.

[170] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An Efficient Alternative to SIFT or SURF". In: *Int. Conf. Comput. Vis. (ICCV)*. 2011, pp. 2564–2571.

[171] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk. "From Coarse to Fine: Robust Hierarchical Localization at Large Scale". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[172] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 8601–8610. DOI: 10.1109/CVPR.2018.00897.

[173] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. "Image Retrieval for Image-Based Localization Revisited". In: *British Mach. Vis. Conf. (BMVC)*. 2012, pp. 76.1–76.12.

[174] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. "Understanding the Limitations of CNN-Based Absolute Camera Pose Regression". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019. DOI: 10.1109/CVPR.2019.00342.

[175] N. Savinov, A. Seki, L. Ladický, T. Sattler, and M. Pollefeys. "Quad-Networks: Unsupervised Learning to Rank for Interest Point Detection". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017.

[176] D. Scaramuzza and F. Fraundorfer. "Visual Odometry [Tutorial]. Part I: The First 30 Years and Fundamentals". In: *IEEE Robot. Autom. Mag.* 18.4 (Dec. 2011), pp. 80–92. DOI: 10.1109/MRA.2011.943233.

[177] P. Schmuck and M. Chli. "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams". In: *J. Field Robot.* 36.4 (2019), pp. 763–781. DOI: 10.1002/rob.21854.

[178] P. Schmuck and M. Chli. "Multi-UAV collaborative monocular SLAM". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 3863–3870.

[179] P. Schmuck, S. A. Scherer, and A. Zell. "Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping". In: *IFAC-PapersOnLine* 49.15 (2016), pp. 230–235. DOI: 10.1016/j.ifacol.2016.07.738.

[180] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. "maplab: An Open Framework for Research in Visual-inertial Mapping and Localization". In: *IEEE Robot. Autom. Lett.* (2018). DOI: 10.1109/LRA.2018.2800113.

[181] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys. "Comparative Evaluation of Hand-Crafted and Learned Local Features". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017, pp. 1482–1491.

[182] J. L. Schönberger and J.-M. Frahm. "Structure-from-Motion Revisited". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2016.

[183] J. Shi and C. Tomasi. "Good features to track". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[184] O. Simeoni, Y. Avrithis, and O. Chum. "Local Features and Visual Words Emerge in Activations". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[185] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. "Discriminative Learning of Deep Convolutional Feature Point Descriptors". In: *Int. Conf. Comput. Vis. (ICCV)*. 2015, pp. 118–126.

[186] J. Sivic and A. Zisserman. "Video Google: a text retrieval approach to object matching in videos". In: *Int. Conf. Comput. Vis. (ICCV)*. 2003, pp. 1470–1477. DOI: 10.1109/ICCV.2003.1238663.

[187] S. M. Smith and J. M. Brady. "SUSAN – a new approach to low level image processing". In: *Int. J. Comput. Vis.* 23.1 (1997), pp. 45–78.

[188] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications". In: *IEEE/ACM Trans. Netw.* 11.1 (Feb. 2003), pp. 17–32. DOI: 10.1109/TNET.2002.808407.

[189] L. von Stumberg, P. Wenzel, N. Yang, and D. Cremers. "LM-Reloc: Levenberg-Marquardt Based Direct Visual Relocalization". In: *arXiv e-prints* (Oct. 2020). URL: https://arxiv.org/abs/2010.06323.

[190] N. Sünderhauf and P. Protzel. "Switchable Constraints for Robust Pose Graph SLAM". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2012.

[191] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford. "On the performance of ConvNet features for place recognition". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015.

[192] D. Tardioli, E. Montijano, and A. R. Mosteo. "Visual data association in narrow-bandwidth networks". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015, pp. 2572–2577. DOI: 10.1109/IROS.2015.7353727.

[193] L. Teixeira and M. Chli. "Real-time mesh-based scene estimation for aerial inspection". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 4863–4869. DOI: 10.1109/iros.2016.7759714.

[194] S. Thrun and Y. Liu. "Multi-robot SLAM with Sparse Extended Information Filters". In: *Proc. Int. Symp. Robot. Research (ISRR)* 15 (2005), pp. 254–266.

[195] Y. Tian, A. Koppel, A. S. Bedi, and J. P. How. "Asynchronous and Parallel Distributed Pose Graph Optimization". In: *arXiv e-prints* (2020). URL: https://arxiv.org/abs/2003.03281.

[196] Y. Tian, V. Balntas, T. Ng, A. Barroso-Laguna, Y. Demiris, and K. Mikolajczyk. "D2D: Keypoint Extraction with Describe to Detect Approach". In: *arXiv e-prints* (May 2020). URL: https://arxiv.org/abs/2005.13605.

[197] G. Tolias, R. Sicre, and H. Jégou. "Particular object retrieval with integral max-pooling of CNN activations". In: *Int. Conf. Learn. Representations (ICLR)* (2016).

[198] A. Tomasic and H. Garcia-Molina. "Performance of inverted indices in distributed text document retrieval systems". In: (1992).

[199] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. "Visual Place Recognition with Repetitive Structures". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2013.

[200] M. Trajković and M. Hedley. "Fast corner detection". In: *Image and vision computing* 16.2 (1998), pp. 75–87.

[201] L. Trujillo and G. Olague. "Using Evolution to Learn How to Perform Interest Point Detection". In: *IEEE Int. Conf. Pattern Recog. (ICPR)*. 2006, pp. 211–214. DOI: 10.1109/ICPR.2006.1153.

[202] P. Truong, S. Apostolopoulos, A. Mosinska, S. Stucky, C. Ciller, and S. D. Zanet. "GLAMpoints: Greedily Learned Accurate Match Points". In: *Int. Conf. Comput. Vis. (ICCV)*. Oct. 2019. DOI: 10.1109/ICCV.2019.01083.

[203] P. Truong, M. Danelljan, L. V. Gool, and R. Timofte. "GOCor: Bringing Globally Optimized Correspondence Volumes into Your Neural Network". In: *arXiv e-prints* (Sept. 2020). URL: https://arxiv.org/abs/2009.07823.

[204] P. Turcot and D. G. Lowe. "Better matching with fewer features: The selection of useful features in large database recognition problems". In: *Int. Conf. Comput. Vis. Workshops (ICCVW)*. 2009, pp. 2109–2116.

[205] M. J. Tyszkiewicz, P. Fua, and E. Trulls. "DISK: Learning local features with policy gradient". In: *arXiv e-prints* (June 2020). URL: https://arxiv.org/abs/2006.13566.

[206] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. "TILDE: A Temporally Invariant Learned DEtector". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2015, pp. 5279–5288.

[207] M. Volkov, G. Rosman, D. Feldman, J. W. Fisher, and D. Rus. "Coresets for visual summarization with applications to loop closure". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015.

[208] J. O. Wallgrun. *Hierarchical Voronoi Graphs: Spatial Representation and Reasoning for Mobile Robots*. Springer Publishing Company, Incorporated, 2010. DOI: 10.1007/978-3-642-10345-2.

[209] S. Wang, R. Clark, H. Wen, and N. Trigoni. "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 2043–2050. DOI: 10.1109/ICRA.2017.7989236.

[210] X. Wei, Y. Zhang, Y. Gong, and N. Zheng. "Kernelized Subspace Pooling for Deep Local Descriptors". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[211] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. "OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems". In: *Proc. ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*. 2010.

[212] B. Yamauchi. "A frontier-based approach for autonomous exploration". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 1997, pp. 146–151.

[213] T. Yan, D. Ganesan, and R. Manmatha. "Distributed image search in camera sensor networks". In: *Proceedings of the 6th ACM Conf. on Embedded Network Sensor Systems*. 2008.

[214] N. Yang, R. Wang, J. Stuckler, and D. Cremers. "Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018.

[215] Y. Ye, T. Cieslewski, A. Loquercio, and D. Scaramuzza. "Place Recognition in Semi-Dense Maps: Geometric and Learning-Based Approaches". In: *British Mach. Vis. Conf. (BMVC)*. 2017, pp. 74.1–74.13. DOI: 10.5244/C.31.74.

[216] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. "LIFT: Learned invariant feature transform". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2016, pp. 467–483.

[217] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. "Learning to Find Good Correspondences". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[218] J. Yick, B. Mukherjee, and D. Ghosal. "Wireless sensor network survey". In: *Computer networks* (2008).

[219] S. Zagoruyko and N. Komodakis. "Learning to Compare Image Patches via Convolutional Neural Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2015, pp. 4353, 4361. DOI: 10.1109/CVPR.2015.7299064.

[220] L. Zhang and S. Rusinkiewicz. "Learning to Detect Features in Texture Images". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[221] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. "Tapestry: a resilient global-scale overlay for service deployment". In: *IEEE Journal on Selected Areas in Communications* 22.1 (2004), pp. 41–53.

[222] X. S. Zhou and S. I. Roumeliotis. "Multi-robot SLAM with Unknown Initial Correspondence: The Robot Rendezvous Case". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2006, pp. 1785–1792. DOI: 10.1109/IROS.2006.282219.

Titus Cieslewski

PhD in Computer Vision and Robotics

September 23rd 1990
Nationality: Swiss

ACADEMIC

PhD Program in Computer Science, Robotics and Perception Group, University of Zürich — *Prof. Dr. Davide Scaramuzza*

January 2016 - February 2021

Thesis: **Data-Efficient Decentralized Visual SLAM,** Video

Research Assistant, Autonomous Systems Lab, ETH Zürich — *Prof. Dr. Roland Siegwart*

November 2014 - December 2015

M.S. in Microengineering, Micro- and Nanosystems, EPFL Lausanne

September 2012 - October 2014

Exchange Year, Electrical and Computer Engineering, Carnegie Mellon University

August 2011 - May 2012

B.S. in Microengineering, EPFL Lausanne

August 2009 - July 2011

PROFESSIONAL EXPERIENCE

Research Postdoc, Intel Munich — *Vladlen Koltun*
April 2021 - November 2021

Teaching Assistant, ETH Zürich
2017 - 2018
Vision Algorithms for Mobile Robots

Intern, NVIDIA, Sunnyvale
June 2012 - August 2012

RESEARCH INTERESTS

Computer Vision, Multi-Robot SLAM, Image Features, Place Recognition and Localization, Exploration

SKILLS

Programming: C++, Python

ML: TensorFlow