

INFORMATION AND COMMUNICATION TECHNOLOGIES
COLLABORATIVE PROJECT

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	3.3
Deliverable Name	Vehicle Localisation, Local 3D Mapping and Dynamic Object Detection
WP Number	3
Date of Preparation	16.05.2014
Date of Submission	31.05.2014
Editor/Beneficiary	Bürki/ETH Zürich

Contents

1	Introduction	2
1.1	Task Description	2
2	Overview of the contributions	2
2.1	Task 3.1: Pose Estimation using Vision (ETHZ (Lead), VW, TUBS)	2
3	Conclusion	5
A	Minimal Solutions for Pose Estimation of a Multi-Camera System	6
B	OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision	23
C	Motion Estimation Using Rolling-Shutter Cameras for an Autonomous Car	32
D	Long-term 3D map maintenance in dynamic environments	35
E	Scene Signatures: Localised and Point-less Features for Localisation	44
F	Dense Reconstruction of Outdoor Scenes from Sparse Range images with Adaptive Model Selection	54
G	Unified Temporal and Spatial Calibration for Multi-Sensor Systems	56
H	Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems	64

1 Introduction

1.1 Task Description

The overall objective of WP3 is to develop novel approaches for perceiving the local environment from vision (mono & stereo), ultrasonic data and potentially radar, and to reliably classify and track dynamic obstacles (other cars, people). The output of this WP is then used for motion planning and vehicles control in WP4. In particular, the following system modules will be provided in WP3:

- vehicle localization
- detection of dynamic objects
- tracking and prediction of dynamic objects
- object classification (car, pedestrian, etc.)
- learning parameter control policies for persistent localisation

2 Overview of the contributions

Work on all tasks in WP3 has continued since deliverable D3.2. To prevent major overlaps with D3.4, we have decided to only report on progress on the localization and mapping pipeline (task 3.1) and to treat all progress from depth measurements over dynamic object detection, extraction, tracking and prediction separately in deliverable D3.4. All the documents referred to in the following subsections can be found in the appendix.

2.1 Task 3.1: Pose Estimation using Vision (ETHZ (Lead), VW, TUBS)

	Contributions	From	Link
1	Minimal Solutions for Pose Estimation of a Multi-Camera System, <i>Lee, Li, Pollefeys, Fraundorfer</i>	ETHZ-CVG	A
2	OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision, <i>Kneip, Furgale, ICRA 2014 Conference</i>	ETHZ-ASL	B
3	Motion Estimation Using Rolling-Shutter Cameras for an Autonomous Car, <i>Bürki, Master Thesis</i>	ETHZ-ASL	C
4	Long-term 3D map maintenance in dynamic environments, <i>Pomerleau, Krüsi, Colas, Furgale, Siegwart, ICRA 2014 Conference</i>	ETHZ-ASL	D
5	Scene Signatures: Localised and Point-less Features for Localisation, <i>McManus, Upcroft, Newman, Robotics Science and Systems</i>	UOXF.DF	E
6	Dense Reconstruction of Outdoor Scenes from Sparse Range images with Adaptive Model Selection, <i>Pinies, Paz, Newman, British Machine and Vision Conference</i>	UOXF.DF	F
7	Unified Temporal and Spatial Calibration for Multi-Sensor Systems, <i>Furgale, Rehder, Siegwart, IROS 2013 Conference</i>	ETHZ-ASL	G
8	Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems, <i>Barfoot, Furgale, IEEE Transactions on Robotics</i>	ETHZ-ASL	H

Our efforts on task 3.1 have focused on the following aspects: solving the pose estimation problem from image correspondences (contribution A and B), visual odometry (contribution C), mapping and long-term map maintenance (contribution D), long-term localization and scene understanding (contribution E and F) as well as fundamental research on calibration (contribution G) and pose uncertainty (contribution H).

Both contribution A and B focus on pose estimation techniques for non-central multi-camera systems as they are present on the V-Charge cars. Light rays being captured by the cameras of such multi-camera systems do not meet at a single focal point as would be the case for traditional single-camera systems. As a consequence of this, classic algorithms to solve for the perspective pose are not applicable and special approaches for pose estimation of such non-central multi-camera systems are required. In “Minimal Solutions for Pose Estimation of a Multi-Camera System”, Gim Hee Lee, Bo Li, Marc Pollefeys and Friedrich Fraundorfer propose a novel approach to estimating the pose of a multi-camera system wrt. a world frame using a minimal set of 3D points observed by the camera system. With no prior required on the location of the camera system, these techniques are vital in robotics for initial localization against a known map (i.e. after boot-up), as well as geometric verification and place recognition for finding loop closures. Transforming a minimal set of three light rays to Plücker coordinates allows solving for the unknown depth of the corresponding 3D points by evaluating the solutions of an eight degree polynomial. This leads to a maximum of eight real solutions for the unknown depths, with each of them corresponding to a single solution for the pose of the camera system. For each of these (maximally) eight rigid transformation candidates between the camera system and the world coordinate frame, the depth of the other 3D points observed in the images can be computed and verified in a RANSAC step. This allows robustly finding the correct pose solution out of the one to eight candidates. Traditional solvers for finding the absolute orientation in a perspective pose estimation problem with a minimal number of point correspondences rely on an SVD step, which becomes computationally expensive when executed numerous times within the RANSAC procedure. Therefore, Gim Hee Lee et al. propose a closed-form solution for finding the absolute orientation, therewith getting rid of the SVD. Apart from simulated tests, their algorithm is evaluated on extensive real-world data collected by the non-central four-camera system of the V-Charge cars.

In contribution B, Laurent Kneip and Paul Furgale present and document their effort in providing a new C++ library called “OpenGV” for calibrated real-time 3D geometric vision algorithms. While OpenGV covers traditional central camera systems, its special focus on algorithms for absolute and relative camera pose computation of non-central multi-camera systems makes it unique with respect to completeness and directly applicable to the four-camera system used for localization and mapping on the V-Charge cars. OpenGV mainly contains implementations of purely geometric algorithms. Hence, potential applications include 3D image matching or reconstruction with no prior information on the temporal or topographical ordering of the images involved. Even in a classic visual-odometry system, such as the one used on the V-Charge cars for localization and mapping, these pure geometric algorithms become interesting and necessary in special tasks where no prior knowledge on the pose of the robot is available such as bootstrapping and localization recovery, as well as for loop-closure detection. In addition to that, a major contribution of OpenGV lies in its extensive collection of state-of-the-art algorithms with some of them covering identical problems. This provides an unprecedented playground for geometric vision algorithms to be compared against each other.

Rolling-shutter cameras have a number of advantages over global-shutter cameras that make them interesting especially for outdoor, all-day application such as in V-Charge. For example, temporal overlapping frame integration allows for longer exposure times at same frame-rates compared to global-shutter cameras, therewith covering a higher dynamic light range. However, these benefits come at the cost of image distortion effects whenever there is motion between the camera and the observed scene during the frame integration time. These distortions will lead to biases in the motion estimation when they are not accounted for in the underlying motion models. The work by Mathias Bürki on “Motion Estimation Using Rolling-Shutter Cameras for an Autonomous Car” (contribution C) deals with integrating rolling-shutter cameras into a visual odometry framework by using a continuous time state space representation which allows modeling the rolling-shutter dynamics. The resulting sliding-window visual odometry algorithm is evaluated

on a four-camera rolling-shutter system - mounted onto an Toyota Prius - that is virtually identical to the global-shutter camera system on the V-Charge cars.

The work presented by François Pomerleau, Philipp Krüsi, Francis Colas, Paul Furgale and Roland Siegwart on “Long-term 3D map maintenance in dynamic environments” (contribution [D](#)) focuses on generating and curating maps for long-term localization and navigation in urban environments. Building upon the extensive work on single-session SLAM, their approach extends localization and mapping techniques for mobile robots to allow reusing and updating maps of a dynamic environment over multiple work assignments on an long-term time-scale. This is achieved by separating dynamic from static environment, enabling consistent updates to and localization against the static parts of the map, as well as accurate velocity and trajectory estimation of dynamic objects which is of high value to subsequent path planning algorithms. An extensive experimental evaluation in a highly dynamic urban environment around the ETH main campus in Zurich over the time span of seven months shows the applicability of their work to mobile robots subject to long-term localization and mapping tasks. Although their concrete implementation and experimental evaluation is based on 3D laser range data, the underlying principles involved of segregating static from dynamic environment, as well as estimating the velocities and flow patterns of dynamic objects, are independent of the sensor setup and applicable to a wide range of mobile robot platforms including vision based localization and mapping systems as they are present on the V-Charge cars.

The work of Collin McManus, Ben Upcroft and Paul Newman about “Scene Signatures: Localised and Point-less Features for Localisation” (contribution [E](#)) addresses the problem of localising an autonomous car across extreme lighting and weather conditions. It departs from the traditional pointfeature-based approach since matching under dramatic appearance changes is a brittle and hard task. Point-feature detectors are rigid procedures which pass over an image examining small, low-level structure such as corners or blobs. They apply the same criteria to all images of all places. McManus et al. take a contrary view and ask what is possible if instead a bespoke detector is learned for every place. The localisation task then turns into curating a large bank of spatially indexed detectors. This strategy yields vastly superior performance in terms of robustness in exchange for a reduced but tolerable metric precision. The core of the algorithm is an unsupervised system that produces broad-region detectors for distinctive visual elements, called scene signatures, which can be associated across almost all appearance changes. It has been extensively shown, using 21km of data collected over a period of 3 months, that the proposed system is capable of producing metric localisation estimates from night-to-day or summer-to-winter conditions. In the context of the V-Charge project, the current localisation system can leverage of the recent technique which naturally provides a parameter control policy, learnt over time, to increase the localisation performance (especially in varying light conditions).

In the work “Dense Reconstruction of Outdoor Scenes from Sparse Range images with Adaptive Model Selection” (contribution [F](#)) the authors Pinies, Paz and Newman have successfully shown another example where a learning control parameter policy is applied with very positive impact on the 3D reconstruction accuracy. They address the problem of dense depth map estimation from sparse noisy range data to reconstruct large heterogeneous outdoor scenes. A solution for surface interpolation is proposed through energy minimization with an adaptive selection of surface regularisers among a set of well known convex and non-convex norms. Recent research in this topic has shown that the selection of norm is pivotal with respect to the intrinsic surface characteristics. The goal of this work is to show how dense interpolation of sparse range data can be leveraged of more exotic and non-convex norms which can better capture the scene geometry. In contrast to state of the art solutions, the authors do not restrict themselves to this set of norms, instead their approach searches for the most apt norm for each semantically segmented part of the scene. As a result, they provide a novel energy model selection method that uses Bayesian optimisation to learn the best choice of free parameters. This results in an adaptive model selection and the generalization of well studied regularisation norms. An exhaustive and detailed experimental analysis supports the approach using a basis of four norms over ten challenging outdoor scenes.

Paul Furgale. Joern Rehder and Roland Siegwart’s work on “Unified Temporal and Spatial Calibration for Multi-Sensor Systems” (contribution G) tackles the fundamental problems of robot platforms where information from more than one sensor is fused together. For the localization and mapping system on the V-Charge cars, this is the case for the wheel odometry sensors, which provide a 3D pose estimate that is combined the visual cues provided by the multi-camera system. Measurements from different sensors are recorded at globally unknown times, as the different sensor units are usually not time-synchronized or globally trigger. This results in deterministic time offsets between the involved sensor units, introducing biases into the fused state estimation. These time offsets, however, can be determined and later accounted for by a temporal calibration process. While there exist previous solution to both calibrate spatial as well as temporal parameters of a multi-sensor system individually, the work by Paul Furgale and Roland Siegwart offer a novel approach to estimate both the temporal as well as spatial calibration parameters jointly in a single calibration attempt. With their continuous time state space representation based on B-Splines, an initially unknown time offset parameter between the different sensor units can directly be integrated into the conventional spatial calibration process. Their proposed solution is evaluated both in simulation and with real data using an IMU/camera setup, but readily extends to an arbitrary multi-sensor platform.

With “Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems” (contribution H), we include a paper by Timothy Barfoot and Paul Furgale that approaches another very fundamental problem in robotics: Namely the problem of how to associate uncertainty with 4x4 transformation matrices. The difficult aspect of this stems from the fact that transformation matrices, as members of $SE(3)$, do not form a vector space. Tim Barfoot and Paul Furgale describe how probability distributions on the underlying 6 degree parameter vector can be transformed to a probability distribution over the transformation matrices and how small disturbances in the parameter space affect the corresponding transformation matrices. This allows dealing with a series of common problems in robotics involving three-dimensional poses, of which the following three concrete examples are explicitly dealt with by the authors: 1) propagating uncertainty through a compound pose change, 2) fusing multiple measurements of a pose and 3) propagating poses and landmarks through a non-linear camera model. In V-Charge, uncertainty with poses has to be dealt with extensively in the mapping and localization pipeline, as uncertain pose estimates from the wheel odometry are used in propagating the pose of the car forward in time, influencing the uncertainty of the position of the landmarks during mapping as well as the uncertainty of the pose estimate during localization.

3 Conclusion

As can be seen from the reports attached to this document, Work Package 3 of the V-Charge project is well on track with its scientific, engineering, and integration goals at the time of writing.

A Minimal Solutions for Pose Estimation of a Multi-Camera System

Authors: Lee, Li, Pollefeys, Fraundorfer

Link to [Table of Contents](#).

Minimal Solutions for Pose Estimation of a Multi-Camera System

Gim Hee Lee, Bo Li, Marc Pollefeys, and Friedrich Fraundorfer

Abstract In this paper, we propose a novel formulation to solve the pose estimation problem of a calibrated multi-camera system. The non-central rays that pass through the 3D world points and multi-camera system are elegantly represented as Plücker lines. This allows us to solve for the depth of the points along the Plücker lines with a minimal set of 3-point correspondences. We show that the minimal solution for the depth of the points along the Plücker lines is an 8 degree polynomial that gives up to 8 real solutions. The coordinates of the 3D world points in the multi-camera frame are computed from the known depths. Consequently, the pose of the multi-camera system, i.e. the rigid transformation between the world and multi-camera frames can be obtained from absolute orientation. We also derive a closed-form minimal solution for the absolute orientation. This removes the need for the computationally expensive Singular Value Decompositions (SVD) during the evaluations of the possible solutions for the depths. We identify the correct solution and do robust estimation with RANSAC. Finally, the solution is further refined by including all the inlier correspondences in a non-linear refinement step. We verify our approach by showing comparisons with other existing approaches and results from large-scale real-world datasets.

1 Introduction

The pose estimation problem of a multi-camera system refers to the problem of determining the rigid transformation between the world frame and multi-camera frame, given a set of 3D points defined in the world frame and its corresponding 2D image points. In contrast with a single camera that has a single center of projection,

Gim Hee Lee, Bo Li and Marc Pollefeys

Department of Computer Science, ETH Zürich, Universitätstrasse 6 CH-8092 Zürich. e-mail: {glee@student, libo@student, marc.pollefeys@inf}.ethz.ch

Friedrich Fraundorfer

Faculty of Civil Engineering and Surveying, Technische Universität München, Arcisstrasse 21 80333 München. e-mail: friedrich.fraundorfer@tum.de



Fig. 1 (a) Our robotic car platform with a multi-camera system made up of 4 separate fish-eye cameras looking front, rear, left and right (cameras are embedded in the car logos and side mirrors). (b) Sample images from the 4 cameras.

the multi-camera system is an imaging sensor where light rays passing through the 3D world points and camera are non-central, i.e. the light rays do not meet at a single center of projection. An advantage of the multi-camera system is that it provides the flexibility to be set in a configuration which gives a maximum coverage of the environment. The solution to the pose estimation problem of a multi-camera system has important applications in robotics such as finding the initial camera pose estimates in structure-from-motion (SfM) / visual Simultaneous Localization and Mapping (SLAM), geometric verification and place recognition for loop-closures, and visual localization of a robot with respect to a given map that contains visual descriptors. Figure 1 shows our robotic car platform and the images from the multi-camera system mounted on it.

The fact that the light rays from a multi-camera system do not meet at a single center of projection means that all the classical approaches [6, 17, 13] for solving the perspective pose problem cannot be used. An alternative approach has to be proposed to handle the non-central nature of the multi-camera system. In addition, it is important that the proposed approach is a minimal solution and requires minimal correspondences that makes it efficient to be used within robust estimators such as RANSAC [5] (see Section 5 for more detail).

In this paper, we proposed a novel formulation to solve the pose estimation problem of a multi-camera system. In particular, we adopt the representation of non-central light rays from a multi-camera system with the Plücker line coordinates from existing works [16, 12, 10, 11] for relative motion estimation of the multi-camera system. We show that this allows us do a two-step approach for solving the pose estimation problem - (a) solve for the unknown depth of the points along the Plücker lines and (b) compute the multi-camera pose from the known depths with absolute orientation [6, 8]. We show that with a minimal number of 3-point correspondences, it leads to an 8 degree polynomial minimal solution that yields up to a maximum of 8 real solutions for the unknown depths. Each of these possible solutions of the depth is used to compute the coordinates of the 3D world points in the multi-camera

frame. The known 3D points in the multi-camera frame are used to compute the pose of the multi-camera system using absolute orientation.

The standard approach for solving the absolute orientation requires an expensive step of SVD and it is inefficient to perform the SVD multiple times to evaluate all the possible solutions of the depths. We circumvent this problem by deriving an efficient minimal solution for the absolute orientation, which allows us to compute the rigid transformation between the world and multi-camera frames from 3-point correspondences in closed-form without the need for SVD. Once we have obtained all the possible solutions for the rigid transformation, we compute the depths of all the other 3D world points. This allows us to choose the correct solution within a robust estimator such as RANSAC. Finally, the solution is further refined by including all the inlier correspondences in a non-linear refinement step that minimizes the reprojection errors (see Section 6 for more detail). We verify our approach by showing comparisons with other existing approaches and results from large-scale real-world datasets.

2 Related Works

The method proposed by Chen *et al.* [2] is most related to our method. In this work, they proposed a 3-point minimal solution and N-point solution to the multi-camera pose estimation problem. Similar to our method, their proposed solution is also a two-step approach. First, the coordinates of the 3D points in the multi-camera frame are determined. The 3D points in the multi-camera frame are determined by solving three distance parameters defined on the rays that passes through the 3D points. Next, the rigid transformation between the 3D points in the world and multi-camera frames is solved by absolute orientation. The formulation in the first step resulted in two 8 degree polynomials where a total of up to 16 real solutions are computed by root finding. In comparison, our method resulted in only one 8 degree polynomial that gives up to 8 real solutions, which has the advantage of less computational time needed to identify the correct solution. Another drawback of [2] is that the representations of the rays used to define the distance parameters breaks down when the three rays are respectively lying on parallel planes and in the case of linear pushbroom cameras [7] (see Section 4.3 for more detail). As a result, an alternative representation has to be made. In contrast, our representation of the rays as the Plücker lines is holistic and does not require any alternative formulation in any case. In addition, we also derive an efficient closed-form minimal solution for absolute orientation.

In [14], Nister proposed a formulation that directly solves for the rotation and translation parameters. His formulation gives an 8 degree polynomial minimal solution. This method is of special interest as the coefficients for the equation system can be computed with a low number of computations making it a fast method. He also proposed the use of Sturm sequencing for root finding and stated that the execution times is in the order of microseconds. The method is evaluated with simulations and compared to the single camera case. Similar to Nister’s method, our method also ends up with an 8 degree polynomial minimal solution, which can also be solved

with the Sturm sequencing to achieve the same execution time. Despite the computational efficiency, as also noted in [9], the derivation of Nister’s method is not intuitive and requires laborious geometry and algebraic reasonings.

Kneip *et al.* presented that most recent work on pose estimation using a multi-camera system in [9]. In this work, the authors presented a 3-point minimal solution and N-point solution. They first solved for the rotations and point depths with a Gröbner Basis [3] solver followed by solving for the translation. They showed simulation experiments, comparisons to single camera perspective pose methods and a real-world visual odometry experiment using a two-camera setup. The exact process of solving the pose estimation problem with the Gröbner Basis approach is a black-box process which is not described in detail in [9]. Hence, Kneip’s method cannot be reproduced easily. In comparison, our method is based on several algebraic equations which are intuitive and easy to implement. They mentioned that the generated solution from the Gröbner Basis solver has a length of 8000 lines of code and the execution time in the order of milliseconds. This makes it slower than Chen’s, Nister’s and our methods which solve an 8 degree polynomial that can be done in the order of microseconds as noted in Nister’s paper [14].

In contrast to the minimal solvers for the pose estimation problem of the multi-camera system, there also exist linear [4] and iterative N-point [18, 19] solutions. The linear solution needs 6 or more point correspondences and thus less efficient in RANSAC [5] compared to our minimal solution which requires only 3 point correspondences. Since the iterative N-point solutions involves computationally expensive iterations, they are usually used to refine the pose estimation after all the inlier point correspondences have been found by RANSAC coupled with a minimal solution.

We adopt the Plücker lines representation for a multi-camera system from existing works on motion estimation [16, 12, 10, 11]. However, it is important to note that we adopt the Plücker lines representation to solve the multi-camera pose estimation problem, which is a completely different problem from the multi-camera motion estimation problem in [16, 12, 10, 11]. The objective of multi-camera motion estimation is to compute the relative transformation between two multi-camera frames given the 2D-2D image point correspondences, while the multi-camera pose estimation problem ask for the rigid transformation between a given world frame and the multi-camera frame given the 2D image point to 3D world point correspondences. To the best of our knowledge, no other work has adopted the Plücker lines representation to solve the multi-camera pose estimation problem.

3 Problem Definition

Figure 2 shows an illustration of the pose estimation problem of the multi-camera system. It is made up of multiple cameras denoted by (C_1, C_2, C_3) that are rigidly fixed onto a single body. Note that we show only 3 cameras in Figure 2 but our proposed method works for any multi-camera system that has any number of cameras. Our method also works even if there was only 1 single camera (see perspective case in Section 4.3). We denote the reference frame of the multi-camera system and the

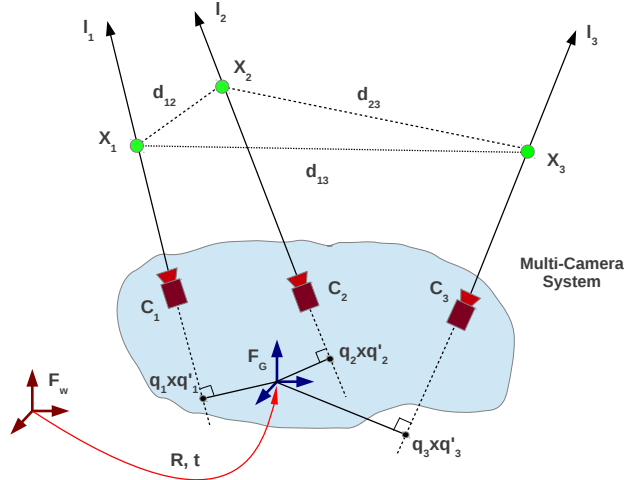


Fig. 2 Illustration of the pose estimation problem for a multi-camera system.

world frame as F_G and F_w . The intrinsics and extrinsics of the respective cameras are assumed to be known from calibration and are denoted by K_i and $T_{C_i} = [R_{C_i} \ t_{C_i}; 0 \ 1]$ with respect to the multi-camera frame F_G , where $i = 1, 2, 3$. The pose estimation problem of a multi-camera system is defined as follows:

Definition 1. Given a set of three 3D points defined in F_w denoted by (X_1, X_2, X_3) that are seen by arbitrary cameras on the multi-camera system and their corresponding 2D image coordinates denoted by (x_1, x_2, x_3) , find the rigid transformation R and t that brings the multi-camera frame F_G into the world frame F_w .

4 Multi-Camera Pose Estimation

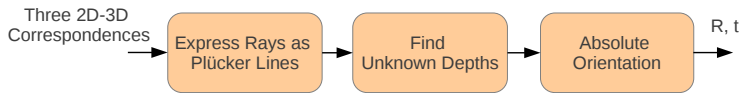


Fig. 3 Our formulation for the pose estimation of the multi-camera system.

Figure 3 shows an illustration of our formulation for pose estimation of the multi-camera system. We first express the rays that join the respective three 2D-3D correspondences as Plücker line coordinates with respect to the multi-camera frame F_G (see Section 4.1 for more detail). Next, we solve for the unknown depths associated with each of the Plücker line using our minimal solution that leads to an 8 degree polynomial giving up to 8 real solutions (see Section 4.2 for more detail). Lastly,

we compute the coordinates of the 3D points in the multi-camera frame F_G with the known depths and solve for the rigid transformation R and t between the world and multi-camera frames using our efficient minimal solution of absolute orientation in closed-form (see Section 4.4 for more detail).

4.1 Plücker Line Representation

We saw in Section 1 that the main problem with a multi-camera system is the absence of a single projection center for the camera. Following [16], we remove the need for a single projection center by representing the rays that pass through the 3D world points and the multi-camera system as Plücker line coordinates expressed in the multi-camera frame F_G . The Plücker line is a 6-vector $l_i = [q_i^T \ q_i'^T]^T$ where $i = 1, 2, 3$ as shown in Figure 2. $q_i = R_{C_i} \hat{x}_i$ is the unit direction of the ray expressed in the multi-camera frame F_G where $\hat{x}_i = K_i^{-1} x_i$ is the normalized image coordinate of the point x_i . The closest point from the Plücker line to F_G is given by $q_i \times q_i'$ as shown in Figure 2 and it is the point that forms a perpendicular intersection on the Plücker line from the multi-camera frame F_G . q_i' is defined as the cross product $q_i' = t_{Ci} \times q_i$. Any point X_i^G that is expressed in the multi-camera frame F_G is given by

$$X_i^G = q_i \times q_i' + \lambda_i q_i \quad (1)$$

where λ_i is the depth of the point X_i^G along the Plücker line, i.e. the signed distance from $q_i \times q_i'$ to X_i^G . Note that λ always has to be positive for the 3D point to appear in front of the camera.

4.2 Minimal Solution for Depths

The distances d_{ij} where $(i, j) \in \{(1, 2), (1, 3), (2, 3)\}$ between the 3D points X_i in the world frame F_W shown in Figure 2 have to be the same as the distances between the 3D points X_i^G in the multi-camera frame F_G , i.e.

$$\|X_i - X_j\|^2 = \|X_i^G - X_j^G\|^2 \quad (2)$$

where $(i, j) \in \{(1, 2), (1, 3), (2, 3)\}$. By making use of the preservation of the 3D point distances given by Equation 2 and the Plücker line equation from Equation 1, we get three constraints

$$\|X_i - X_j\|^2 = \|(q_i \times q_i' + \lambda_i q_i) - (q_j \times q_j' + \lambda_j q_j)\|^2 \quad (3)$$

where $(i, j) \in \{(1, 2), (1, 3), (2, 3)\}$ with three unknown depths λ_1 , λ_2 and λ_3 from the Plücker lines. Expanding and rearranging the unknowns in Equation 3, we get

$$k_{11}\lambda_1^2 + (k_{12}\lambda_2 + k_{13})\lambda_1 + (k_{14}\lambda_2^2 + k_{15}\lambda_2 + k_{16}) = 0 \quad (4a)$$

$$k_{21}\lambda_1^2 + (k_{22}\lambda_3 + k_{23})\lambda_1 + (k_{24}\lambda_3^2 + k_{25}\lambda_3 + k_{26}) = 0 \quad (4b)$$

$$k_{31}\lambda_2^2 + (k_{32}\lambda_3 + k_{33})\lambda_2 + (k_{34}\lambda_3^2 + k_{35}\lambda_3 + k_{36}) = 0 \quad (4c)$$

where k are the coefficients made up from the known Plücker line coordinates q_i and q'_i , and 3D world points X_i . We drop the full expressions of the coefficients for brevity. Using the Sylvester Resultant [3] to eliminate λ_1 from Equations 4a and 4b, we get a polynomial $f(\lambda_2, \lambda_3) = 0$ which is a function of only λ_2 and λ_3 . We do another Sylvester Resultant on $f(\lambda_2, \lambda_3) = 0$ and Equation 4c to eliminate λ_2 , we get an univariate 8 degree polynomial dependent on only λ_3 .

$$A\lambda_3^8 + B\lambda_3^7 + C\lambda_3^6 + D\lambda_3^5 + E\lambda_3^4 + F\lambda_3^3 + G\lambda_3^2 + H\lambda_3 + I = 0 \quad (5)$$

where A, B, C, D, E, F, G, H and I are coefficients made up from k from Equation 4. The roots of Equation 5 can be obtained from the eigen-values of the Companion matrix [3] made up of the coefficients. A maximum of up to 8 real roots can be obtained for λ_3 . As suggested in [14], a more efficient way to solve for the roots of the 8 degree polynomial is by using the Sturm sequences.

λ_2 can be found by back-substituting λ_3 in Equation 4c. After completing the square on Equation 4c and making λ_2 the subject, we get

$$\lambda_2 = \frac{1}{2a}(-b \pm \sqrt{b^2 - 4ac}) \quad (6)$$

where $a = k_{31}$, $b = k_{32}\lambda_3 + k_{33}$, $c = k_{34}\lambda_3^2 + k_{35}\lambda_3 + k_{36}$. Similarly, λ_1 can be found by back-substituting λ_2 into Equation 4a which takes similar form as Equation 6 after completing the square and making λ_1 the subject. A total of up to 32 (i.e. $8 \times 2 \times 2$) solution triplets of λ_1 , λ_2 and λ_3 can be obtained. A solution triplet is discarded if any one of the λ s is an imaginary or negative value. A further step to disambiguate the solutions is by doing a redundancy check on λ_1 using Equation 4b. The solution pairs of λ_2 and λ_3 should produce consistent λ_1 from both Equations 4a and 4b. Any solution pair of λ_2 and λ_3 which produces λ_1 with discrepancy from Equations 4a and 4b is discarded. In our simulations, we observed that these disambiguation checks are capable of reducing the maximum number of solutions to two for most of the time. All the other existing 2D-3D point correspondences are used to identify the correct solution within RANSAC, i.e. the correct solution yields the highest number of inliers in RANSAC.

4.3 Special Cases

In this section, we look at five special cases for the multi-camera pose estimation problem mentioned in [2]. In particular, we compare the similarities and differences between the existing methods and our method under these five different cases.

Case 1: Partially Parallel. Two out of the three light rays are parallel in this case as illustrated in Figure 4. This means that two of the unit directions must be equal, i.e. $q_1 = q_2 \neq q_3$. From Figure 4, we can see that $\lambda_2 = \lambda_1 + c_{12}$, where c_{12} is a known value from the Plücker line coordinates and distance between the 3D points (X_1, X_2) . Applying the Sylvester Resultant for variable elimination together with Equations 4b and 4c, we get a 4 degree polynomial minimal solution that can be

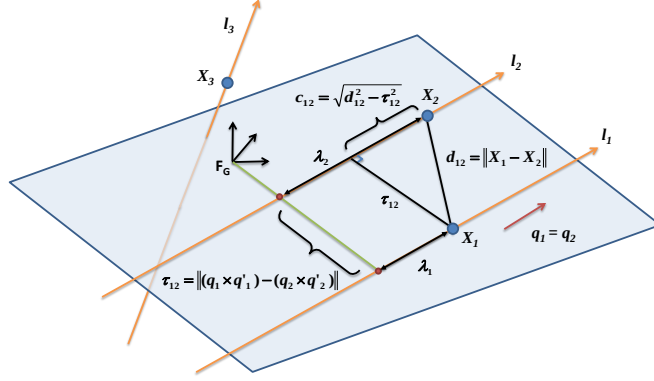


Fig. 4 Illustration of the partially parallel case.

solved in closed-form. Similar 4 degree polynomial minimal solution was obtained for Chen’s and Nister’s methods.

Case 2: Perspective. The three light rays pass through a common center of projection in the perspective case, i.e. all the 2D-3D correspondences are from one single camera in the multi-camera system. Let us choose the camera reference frame to be the center of projection. This implies that the camera extrinsics become $t_{C1} = t_{C2} = t_{C3} = 0$ and $R_{C1} = R_{C2} = R_{C3} = I$. Substituting these values into Equation 3 and applying the Sylvester Resultant for variable elimination, we get a 4 degree polynomial minimal solution that can be solved in closed-form. This result is similar to Chen’s and Nister’s method, and the P3P solution for a perspective camera [6]. Note that a 4 degree polynomial is obtained even if the reference frame was not chosen as the center of projection of the camera.

Case 3: Parallel Plane. This is the case where the three light rays lie on three different planes that are parallel to each other. It is important to note that these light rays however do not have the same unit direction, i.e. $q_1 \neq q_2 \neq q_3$ from the Plücker lines. It can be observed that the constraints from our method in Equation 3 does not break down. In contrast, the representations of the rays used by Chen *et al.* [2] to define the distance parameters cannot be used in the case where all the three rays respectively lie on parallel planes. As a result, an alternative representation has to be made.

Case 4: Linear Pushbroom. There is only one camera in the case of linear pushbroom [7]. Here, the camera moves through a straight line of motion with a known speed and takes images at regular intervals. Hence, the transformations between any three camera locations (similar to the extrinsics of a multi-camera system) are known and the rays that observed unique 3D world points from these locations lie on parallel planes. This implies that the linear pushbroom case is the same as the parallel plane case where our method does not break down. In comparison, an alternative representation has to be made for Chen’s method.

Case 5: Orthographic. For orthographic projection, all the light rays are parallel. Hence, all the unit directions of the Plücker lines are equal, i.e. $q_1 = q_2 = q_3$. Similar to the partially parallel case, we have the following 3 constraints $\lambda_2 =$

$\lambda_1 + c_{12}$, $\lambda_3 = \lambda_1 + c_{13}$ and $\lambda_3 = \lambda_2 + c_{23}$, where infinite solutions exist for λ_1 , λ_2 and λ_3 . Intuitively, we can move the multi-camera system anywhere along the direction of the parallel light rays and the constraints are still fulfilled, hence infinite solutions. This degeneracy is independent of the formulation and holds for all works [2, 14, 9, 4, 18, 19] on pose estimation for the multi-camera system.

4.4 Minimal solution for Absolute Orientation

Absolute Orientation can be solved using the methods from [8, 6]. However, these methods require a computationally inefficient step of SVD which becomes an overhead when it is used numerous times within RANSAC to compute all the hypothesis solutions. We present a minimal solution which allows us to compute the absolute orientation in closed-form without the need for SVD. The proposed method computes the transformation R and t to align the two point sets P and Q consisting of three correspondence 3D points as shown in Equation 7.

$$P_i = RQ_i + t, \quad i = 1, 2, 3 \quad (7)$$

First, two local frames F_M and F_N are defined on the point sets P and Q respectively. The origins of the local frames are defined on the first points, i.e. P_1 and Q_1 . We can now write the transformed points in the newly defined local frames F_M and F_N as $M_i = P_i - P_1$ and $N_i = Q_i - Q_1$. Next, we define the x-axis of each local frame to pass through the second point respectively, i.e. M_2 and N_2 . The x-axis of F_M and F_N can be aligned by applying the following transformations

$$M_2 = \begin{bmatrix} M_{2x} \\ M_{2y} \\ M_{2z} \end{bmatrix} = R_M \begin{bmatrix} \|M_2\| \\ 0 \\ 0 \end{bmatrix}, \quad N_2 = \begin{bmatrix} N_{2x} \\ N_{2y} \\ N_{2z} \end{bmatrix} = R_N \begin{bmatrix} \|N_2\| \\ 0 \\ 0 \end{bmatrix} \quad (8)$$

where R_M and R_N are unknown rotation matrices that align the two x-axis. Here, we only describe the steps to solve for R_M since R_N can be computed in an analogous fashion. Since the alignment of the x-axis only involves two rotations around the y- and z-axis, R_M can be written as

$$R_M = R_{Mz}R_{My} = \begin{bmatrix} ce & -f & de \\ cf & e & df \\ -d & 0 & c \end{bmatrix} \quad (9)$$

where c and d are sine and cosine of the rotation angle around the y-axis, and e and f are sine and cosine of the rotation angle around the z-axis. Putting Equation 9 into Equation 8, we get the following three constraints

$$\|M_2\|ce - M_{2x} = 0 \quad (10a) \quad \|M_2\|cf - M_{2y} = 0 \quad (10b) \quad -M_{2z} - \|M_2\|d = 0 \quad (10c)$$

where d can be calculated directly from Equation 10c and c can then be computed with the Pythagoras identity. e and f can be solved by substituting c into Equations 10a and 10b. The full expressions for solving a, b, c and d are given as follows

$$d = \frac{-M_{2z}}{\|M_2\|}, c = \sqrt{1-d^2}, e = \frac{M_{2x}}{\|M_2\|c}, f = \frac{M_{2y}}{\|M_2\|c} \quad (11)$$

Finally, we apply R_M and R_N to align the x-axis of both point sets. The new sets of transformed points are given by

$$U_i = R_M^T M_i, V_i = R_N^T N_i, \quad i = 1, 2, 3 \quad (12)$$

The last step is to find the remaining rotation R_V around the x-axis which would complete the alignment of the two local frames F_M and F_N . This gives the constraint in Equation 13 which can be expanded into three independent constraints in Equations 14a-14c.

$$U_3 = R_V V_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & -b \\ 0 & b & a \end{bmatrix} V_3 \quad (13)$$

$$V_{3x} - U_{3x} = 0 \quad (14a)$$

$$V_{3y}a - U_{3y} - V_{3z}b = 0 \quad (14b)$$

$$V_{3z}a - U_{3z} - V_{3y}b = 0 \quad (14c)$$

where a and b are sine and cosine of the rotation angle. $U_3 = [U_{3x} \ U_{3y} \ U_{3z}]^T$ and $V_3 = [V_{3x} \ V_{3y} \ V_{3z}]^T$. We do variable elimination on Equations 14b and 14c to solve for a which can be back-substituted to solve for b . The full expressions for a and b are

$$a = \frac{U_{3y}V_{3y} + U_{3z}V_{3z}}{V_{3y}^2 + V_{3z}^2}, \quad b = \frac{-U_{3y} + V_{3y}a}{V_{3z}} \quad (15)$$

Finally, the full transformation R and t is given by

$$R = R_N^T R_V R_M, \quad t = -R P_1 + Q_1 \quad (16)$$

5 Robust Estimation

Outlier 2D-3D point correspondences are rejected from our proposed method using RANSAC [5]. We compute the reprojection errors of all the 2D-3D point correspondences based on the hypotheses generated from random sets of unique 3-point correspondences. The hypothesis that yields the highest inlier count, i.e. highest number of 2D-3D point correspondences with the respective reprojection error lower than a given threshold, is chosen as the correct solution. As defined in [5], the number of RANSAC iterations needed is given by $\eta = \frac{\ln(1-p)}{\ln(1-w^n)}$, where p is the probability that all selected correspondences are inliers, w is the probability that any selected correspondence is an inlier and n is the number of correspondences needed for the hypothesis. Assuming that $p = 0.99$ and $w = 0.5$, a total of 35 iterations are needed

for our 3-point algorithm, i.e. $n = 3$. In contrast, the linear 6-point algorithm [4] where $n = 6$ would require 293 iterations. The efficiency in having less iterations within RANSAC highlights the importance of using the minimal number of point correspondences.

Each hypothesis generated by RANSAC often give rise to more than one real solution from solving the polynomial equation in Section 4.2. We do additional iterations within RANSAC to check the inlier count for each of these solutions from each hypothesis, where the correct solution gives the highest inlier count. It is therefore desirable to have the minimal solution to keep the number of additional RANSAC iterations low. The number of additional RANSAC iterations for our method is halved compared to Chen’s method [2] since our method has a minimal solution up to 8 real solutions while Chen’s method yields up to 16 real solutions.

6 Non-Linear Refinement

We further refine the estimated pose R and t by minimizing the total reprojection errors from all the inlier point correspondences found from RANSAC. The cost function is given by

$$\operatorname{argmin}_{R,t} \sum_i \sum_j \|\pi(P_i, X_j) - \mathbf{x}_{ij}\|^2 \quad (17)$$

where $\mathbf{x}_{ij}$ is the 2D image point with X_j as its 3D point correspondence and seen by the i^{th} camera C_i that makes up the multi-camera system. $\pi(\cdot)$ is the camera projection function that projects a 3D point onto the 2D image. P_i is the camera projection matrix given by

$$P_i = K_i [R_{C_i}^T R^T - R_{C_i}^T (R^T t + t_{C_i})] \quad (18)$$

where K_i is the camera intrinsics, R_{C_i} and t_{C_i} are the camera intrinsics as defined in Section 3. The minimization of Equation 17 is done with the Google Ceres solver¹ using the Levenberg-Marquardt algorithm.

7 Results

We evaluate our proposed multi-camera pose estimation algorithm with both simulations and large-scale real-world datasets.

7.1 Simulations

We compare the accuracy and stability of our algorithm with Nister’s [14], Chen’s [2] and Kneip’s [9] algorithms based on the simulation setup suggested in [17]. The simulated multi-camera system is made up of 4 separate cameras looking front,

¹ <http://code.google.com/p/ceres-solver/>

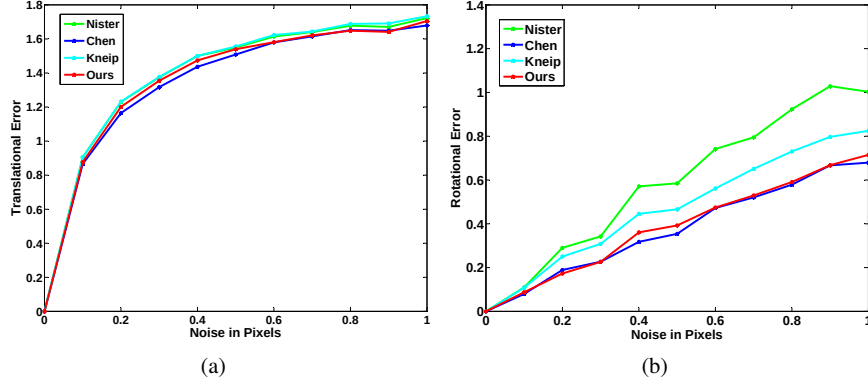


Fig. 5 Average (a) translational and (b) rotational errors from 500 random trials at different pixel noise levels using Nister’s [14], Chen’s [2], Kneip’s [9] and our algorithms. Note that a large part of the translational error for Nister’s method (green line) is hidden behind the translational error for Kneip’s method (cyan line).

right, left and right with no overlapping field-of-views. Note that the chosen camera configuration and simulated rays are free from the parallel ray degeneracy mentioned in Section 4.3. The absolute orientation used in Chen’s method is from [6] while the minimal solution proposed in Section 4.4 is used in our method.

We randomly generate a ground truth camera pose within a given range of $[-1 \ 1]$ m for (x,y,z) and $[-0.1 \ 0.1]$ rad for all angles, i.e. roll, pitch and yaw. We also randomly generate three 3D world points within a given range of $[-10 \ 10]$ m for (x,y,z) . The image coordinates are found by reprojecting the 3D points into the respective camera where it is visible. We corrupt the image coordinates with noise ranging from 0.1 to 1 pixel with a 0.1 pixel interval. The pose of the camera in the world frame is computed based on the corrupted image coordinates using the four algorithms. Following [17], we compute the relative translational error as $2\|t_{est} - t_{gt}\| / (\|t_{est}\| + \|t_{gt}\|)$ where t_{est} and t_{gt} are the estimated and ground truth translations. The relative rotational error is computed as the norm of the Euler angles from $R_{est}R_{gt}^T$ where R_{est} and R_{gt} are the estimated and ground truth rotation matrices.

Figures 5(a) and 5(b) shows the plots of the average relative translational and rotational errors from 500 random trials per image coordinate noise level. It can be seen that Chen’s and our algorithms produced very similar errors for all noise levels. The errors from both Chen’s and our algorithms are also significantly lower than Nister’s and Kneip’s algorithms. The results imply that the two-step approaches, i.e. Chen’s and our algorithms, that solves for the depths and absolute orientation are less susceptible to the influences of noise compared to Nister’s direct approach and Kneip’s Gröbner basis method.

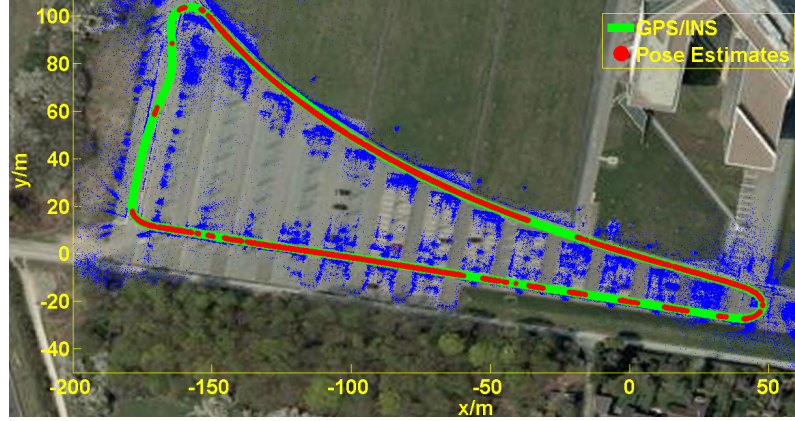
Time Efficiency of Minimal Solution for Absolute Orientation: We compare the time efficiency of our minimal solution for absolute orientation proposed in Section 4.4 with the standard approach that requires SVD [6, 8]. We randomly generate 500

camera poses in the world frame. For each of these poses, we randomly generate 3 points in the world frame and compute the coordinates of these points in the camera frame. The rigid transformation between the camera and world frames is computed with both approaches. Note that the poses estimated from both methods are always the same as the groundtruth and there is no need for RANSAC in this comparison since the points are noise-free. We obtain the time needed for each trial with the respective method and compute the efficiency of our minimal solution for absolute orientation method over the SVD method as the ratio of the mean time taken by our method to the mean time taken by the SVD method for all the 500 trials. The efficiency ratio is found to be 1.23 and this means that on the average our proposed method is 1.23 times faster than the standard SVD approach.

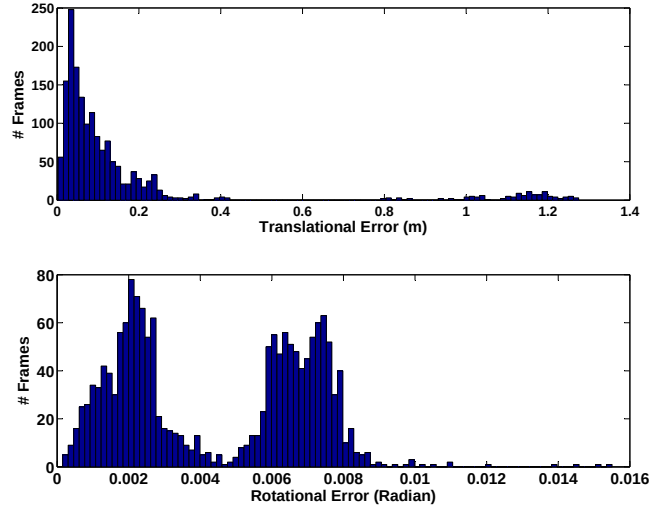
7.2 Real Datasets

Figure 1(a) shows our car platform with 4 fish-eye cameras looking front, rear, left and right with minimal overlapping field-of-views used to collect the datasets for testing our algorithm. The GPS/INS system is also available for ground truth. Figure 1(b) shows 4 sample images from the respective cameras. Figures 6(a) and 7(a) shows two areas for testing our algorithm. TestArea01 and TestArea02 are car parks besides an office building and a supermarket, and covers an area of approximately $140 \times 280\text{m}$ and $160 \times 150\text{m}$ respectively. We collect two datasets separately from each of the test area, i.e. 2×2 datasets - one for building a map and the other for testing our pose estimation algorithm on the map in each test area. To build the maps, we extract the SURF [1] features, and triangulate the 3D points based on the GPS/INS readings. We apply bundle adjustment (implemented with Google Ceres solver) on the GPS/INS poses and triangulated 3D points to get the final maps. The maps also contains all the 2D-3D correspondences of the SURF and 3D points. The blue dots on Figures 6(a) and 7(a) are the 3D points from the maps after bundle adjustment.

The green trajectories on Figures 6(a) and 7(a) are the GPS/INS ground truth readings from the second datasets for testing our pose estimation algorithm on both areas. A total of 2500 and 2100 frames are used for testing. We first create a vocabulary-tree [15] with all the SURF features from the map. For every frame from the test dataset, we extract the SURF features, and query for the frame from the map with the highest similarity score with the vocabulary-tree. We obtain the 2D-3D correspondences of the test and map frames by matching the SURF features. Finally, we compute the pose of the test frame in the map with our multi-camera pose estimation algorithm. Note that a frame refers to a set of 4 images from all the cameras. The red dots on Figures 6(a) and 7(a) are the estimated poses with our algorithm with at least 20 2D-3D correspondences. It can be seen that the poses estimated from our algorithm follows the GPS/INS ground truth closely. Figures 6(b) and 7(b) show the distributions of the translational and rotational errors. We can see that the error distributions are sufficiently low. The translational error is computed as $\|t_{est} - t_{gt}\|$ where t_{est} and t_{gt} are the translations from the pose estimation and GPS/INS ground truth. The rotational error is computed as the norm of the Eu-



(a)



(b)

Fig. 6 (a) Localization results for TestArea01. Results from frames with < 20 correspondences are discarded. (b) Plots showing the distribution of the translational and rotational errors against GPS/INS ground truths.

ler angles from $R_{est}R_{gt}^T$ where R_{est} and R_{gt} are the rotation matrices from the pose estimation and GPS/INS ground truth.

8 Conclusion

We showed a new formulation to solve the pose estimation problem of a multi-camera system. Our formulation is intuitive and easy to implement. It is based on the Plücker line coordinates which solves the pose estimation problem in two steps

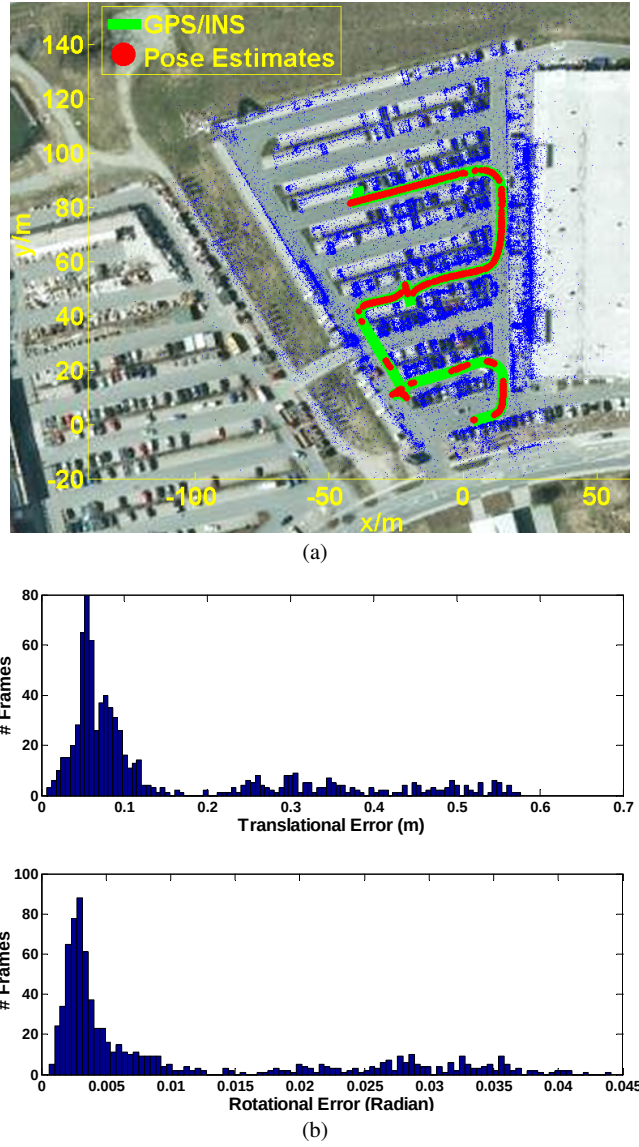


Fig. 7 (a) Localization results for TestArea02. Results from frames with < 20 correspondences are discarded. (b) Plots showing the distribution of the translational and rotational errors against GPS/INS ground truths.

- (a) solve for the depth and (b) solve for the rigid transformation with absolute orientation. We showed that the depths can be solved with a minimal number of 3-point correspondences and leads to an 8 degree polynomial minimal solution. We identified a degenerated case for our method in the case of orthographic projection. We also derived an efficient analytical closed-form minimal solution for the absolute

orientation. Our method is verified with both simulations and large-scale real-world datasets from a robotic car platform.

9 Acknowledgement

This work is supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (v-charge) and 4DVideo ERC Starting Grant Nr. 210806.

References

1. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). In *Computer Vision and Image Understanding*, volume 110, pages 346–359, June 2008.
2. C. S. Chen and W. Y. Chang. On pose recovery for generalized visual sensors. In *Pattern Analysis and Machine Intelligence*, volume 26, pages 848–861, 2004.
3. D. A. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (2. ed.)*. Springer, 1997.
4. A. Ess, A. Neubeck, and L. Van Gool. Generalised linear pose estimation. In *British Machine Vision Conference*, September 2007.
5. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, volume 24, pages 381–395, June 1981.
6. R.M. Haralick, D. Lee, K. Ottenburg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition*, pages 592–598, 1991.
7. R. I. Hartley and R. Gupta. Linear pushbroom cameras. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 963–975, 1994.
8. B.K.P. Horn. Closed form solutions of absolute orientation using unit quaternions. In *JOSA-A*, volume 4, pages 629–642, 1987.
9. L. Kneip, P. Furgale, and R. Siegwart. Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In *International Conference on Robotics and Automation*, 2013.
10. G. H. Lee, F. Fraundorfer, and M. Pollefeys. Motion estimation for a self-driving car with a generalized camera. In *Computer Vision and Pattern Recognition*, May 2013.
11. G. H. Lee, F. Fraundorfer, and M. Pollefeys. Structureless pose-graph loop-closures with a multi-camera system on a self-driving car. In *International Conference on Intelligent Robots and Systems*, 2013.
12. H. D. Li, R. Hartley, and J. H. Kim. A linear approach to motion estimation using generalized camera models. In *Computer Vision and Pattern Recognition*, pages 1–8, June 2008.
13. F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative $\mathcal{O}(n)$ solution to the pnp problem. In *International Conference on Computer Vision*, pages 1–8, 2007.
14. D. Nistér. A minimal solution to the generalised 3-point pose problem. In *Computer Vision and Pattern Recognition*, volume 1, pages 560–567, 2004.
15. D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, pages 2161–2168, 2006.
16. R. Pless. Using many cameras as one. In *Computer Vision and Pattern Recognition*, volume 2, pages 587–93, June 2003.
17. L. Quan and Z. D. Lan. Linear n-point camera pose determination. In *Pattern Analysis and Machine Intelligence*, volume 21, pages 774–780, 1999.
18. G. Schweighofer and A. Pinz. Globally optimal $\mathcal{O}(n)$ solution to the pnp problem for general camera models. In *British Machine Vision Conference*, pages 1–10, 2008.
19. S. Tariq and F. Dellaert. A multi-camera 6-dof pose tracker. In *International Symposium on Mixed and Augmented Reality*, pages 296–297, 2004.

B OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision

Authors: Kneip, Furgale

Link to [Table of Contents](#).

OpenGV: A Unified and Generalized Approach to Real-Time Calibrated Geometric Vision

Laurent Kneip and Paul Furgale
CECS, Australian National University and ASL, ETH Zurich

Abstract—OpenGV is a new C++ library for calibrated real-time 3D geometric vision. It unifies both central and non-central absolute and relative camera pose computation algorithms within a single library. Each problem type comes with minimal and non-minimal closed-form solvers, as well as non-linear iterative optimization and robust sample consensus methods. OpenGV therefore contains an unprecedented level of completeness with regard to calibrated geometric vision algorithms, and it is the first library with a dedicated focus on a unified real-time usage of non-central multi-camera systems, which are increasingly popular in robotics and in the automotive industry. This paper introduces OpenGV’s flexible interface and abstraction for multi-camera systems, and outlines the performance of all contained algorithms. It is our hope that the introduction of this open-source platform will motivate people to use it and potentially also include more algorithms, which would further contribute to the general accessibility of geometric vision algorithms, and build a common playground for the fair comparison of different solutions.

I. INTRODUCTION

Even in times of powerful and freely available localization frameworks that rely on temporal model predictions, the purely geometric computation of a camera pose remains a fundamental problem that requires robust and accurate real-time solutions. The most prominent class of applications is formed by large-scale 3D reconstruction frameworks [1], which are capable of transforming sets of unordered images into 3D models of entire cities. The absence of temporal or topological information linking the images demands a purely geometric solution of camera poses prior to photometric error minimization. However, even classical visual-SLAM algorithms [2], [3] or modern visual-inertial solutions [4] that typically rely on temporal predictions given by a motion model require geometric vision algorithms for any of the “not so normal” tasks, such as bootstrapping, loop-closure, pose recovery, or relocalization. Further potential applications exist for multi-robot localization and mapping. In summary, we can easily identify a persistent need for geometric pose computation algorithms in all domains where structure-from-motion plays a role, such as localization and mapping, object tracking, object modelling, geolocalization, robot extrinsic calibration, augmented reality, and photogrammetry, to name just a few.

Although the need for geometric vision algorithms is ubiquitous within robotics and computer vision, no existing software library for camera pose computation covers all interesting cases. There are many generic structure-from-motion libraries available, such as OpenCV [5], VXL (derived from TargetJR and IUE), Gandalf (mainly homogra-

phy estimation), LibMV, Bundler, CMVS (using bundler), and OpenMVG. These mostly aim at offering complete pipelines for 3D reconstruction or dense model generation, covering all processing steps from interest point extraction down to large-scale non-linear optimization. However, the geometric camera pose computation algorithms provided by these packages never go beyond the standard single-camera problems of space resectioning, homography estimation, or the computation of the fundamental or essential matrix¹.

The goal of the OpenGV project is to introduce a novel common open-source platform for accessing, comparing, and collecting all types of efficient camera pose computation algorithms. In contrast to many other libraries, OpenGV is a pure 3D library that aims at highly efficient solutions to calibrated problems, thus mainly targeting applications in the robotics, automotive, and consumer electronics industries that require real-time solutions to absolute and relative camera pose computation². Besides a more focussed scope, the main difference to existing frameworks is that OpenGV is not limited to single camera applications. The palette of algorithms is completed by including solutions for both *central* and *non-central* camera systems. Central camera systems are characterized by a single viewing origin—all rays from the camera originate at a single point—whereas non-central

¹VXL additionally contains methods for solving the orthogonal procrustes problem, and Libmv also contains methods for pose estimation with unknown focal length.

²By “real-time”, we understand here the capability of an algorithm to—even if embedded into an iterative RANSAC scheme—support processing at common camera frame-rates (>10 Hz).

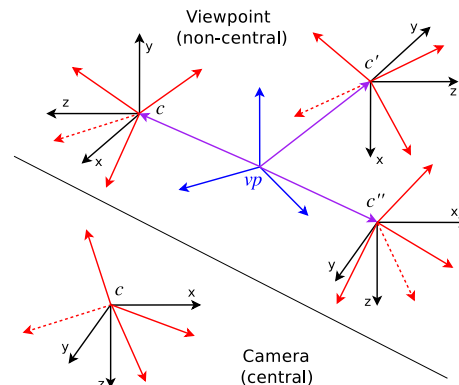


Fig. 1. Illustration of a camera c and a viewpoint vp (in blue). The camera is a single reference frame with bearing vector measurements (in red). The viewpoint has multiple cameras, each one containing its own bearing vector measurements and an individual transformation to the viewpoint frame.

camera systems have rays originating from multiple points. Figure 1 illustrates the difference between these two cases. Non-central algorithms are more robust than central algorithms in the case of multi-camera systems [6], and thus are of utmost importance for real-time 3D vision tasks in modern robotics and automotive applications where multi-sensor or multi-camera systems are ubiquitous. OpenGV contains a flexible interface designed for painless interoperability with existing libraries, and therefore is preferably understood as a complementary extension to existing frameworks rather than a competition.

OpenGV is the first library to present a unified approach to geometric vision not depending on the specifics of the employed imaging system. The library operates directly in 3D and abstracts the camera projection system by using transformed image information only. It employs a generalized description of multi-camera systems, and is therefore applicable to almost any—either central or non-central—camera. It is the first library to provide implementations that solve the highly complicated problems of computing the absolute or relative pose of a generalized camera. Although the benefits of the latter in multi-camera structure from motion are regularly praised in the literature, those algorithms have so far only been presented without accompanying implementations. OpenGV now makes it possible to put the credo of *using many cameras as one* [7] into practice by providing the required building blocks to migrate well-known single camera structure-from-motion concepts to multi-camera scenarios. OpenGV is written in C++, and provides multiple efficient solutions to identical calibrated absolute and relative camera pose computation problems as well as a Matlab interface. The framework makes it easy to compare algorithms against each other and provides a suitable benchmark-tool for future algorithm development. Our goal is for OpenGV to become a commonly accepted geometric vision library for camera pose computation and automatic benchmarking, thus making it easy for future algorithms to be included and avoiding the problem that algorithms posted as independent code fragments on laboratory web-pages may have different interfaces and therefore be difficult to compare.

The remainder of this paper is structured as follows. Section II highlights OpenGV’s generalized approach towards multi-camera systems as well as all contained algorithms. In Section III, we highlight some important implementation details. In Section IV, we present the comparative evaluation of all contained algorithms obtained by our automatic Matlab-based benchmarking tool.

Access: The entire library is hosted as an open-source project on github and can be forked under

<https://github.com/laurentkneip/opengv>

The library also contains its own webpage with an exhaustive documentation of the interface and numerous examples. It can be accessed under

<http://laurentkneip.github.io/opengv>

All results in this paper can be easily reproduced by installing the library and executing the respective benchmark files.

II. OPENGV

This section introduces OpenGV. It starts by outlining the employed generic description of multi-camera systems used by all algorithms within the library. It then gives a brief introduction to all the problems that can be solved, and finally introduces some important details and novelties about OpenGV’s sample-consensus functionalities.

A. Unified description of input data

OpenGV employs a generalized description of camera systems allowing an application to nearly any optical system. The description is centered around an elegant representation of a multi-camera system, without losing the ability to represent fully generalized camera models. An example for the latter is for instance given by a catadioptric camera looking at an arbitrarily shaped mirror.

The specifics of particular camera models are hidden from the interface by representing each image measurement as a 3D bearing vector: a unit vector originating at the camera center and pointing toward the landmark. This can be accomplished easily for most practically relevant cameras—including perspective, dioptric, and catadioptric imaging devices—given that the calibration parameters of a suitable central camera model are known. Each bearing vector has only two degrees of freedom, which are the azimuth and elevation inside the camera reference frame. 3D bearing vectors are—next to normalized image coordinates—a standard choice in computer vision. They provide the advantage of being able to represent omnidirectional landmark observations.

As OpenGV assumes calibrated cameras, landmark observations are always given in form of bearing vectors expressed inside a camera frame. In the present context, a *camera* therefore denotes a camera frame with a set of bearing vectors, all pointing from the origin to landmarks. The possibility of also describing multi-camera systems that cannot be represented by a single camera center is then given by the introduction of *viewpoints*, which allow OpenGV to transparently handle both central and non-central cameras. A viewpoint can contain an arbitrary number of central cameras each one having its own landmark observations (e.g. bearing vectors). Figure 1 illustrates a viewpoint with 3 cameras. A practical example of a viewpoint would be the set of images and related measurements captured by a fully-calibrated, rigid multi-camera rig with synchronized cameras. The viewpoint represents a single (multi-image) snapshot captured by multiple rigidly connected cameras at the same time and hence it can be parametrized by only one single pose (e.g., the viewpoint). Each camera has a known transformation to the viewpoint frame. In the central case the viewpoint simply contains a single camera with an identity transformation. The most general case—the generalized camera—can also be described by the viewpoint; each bearing vector would then have its own camera and related transformation. In fact, the bearing vectors along with the individual camera-to-viewpoint transformations are nothing but alternative representations of Plücker line vectors.

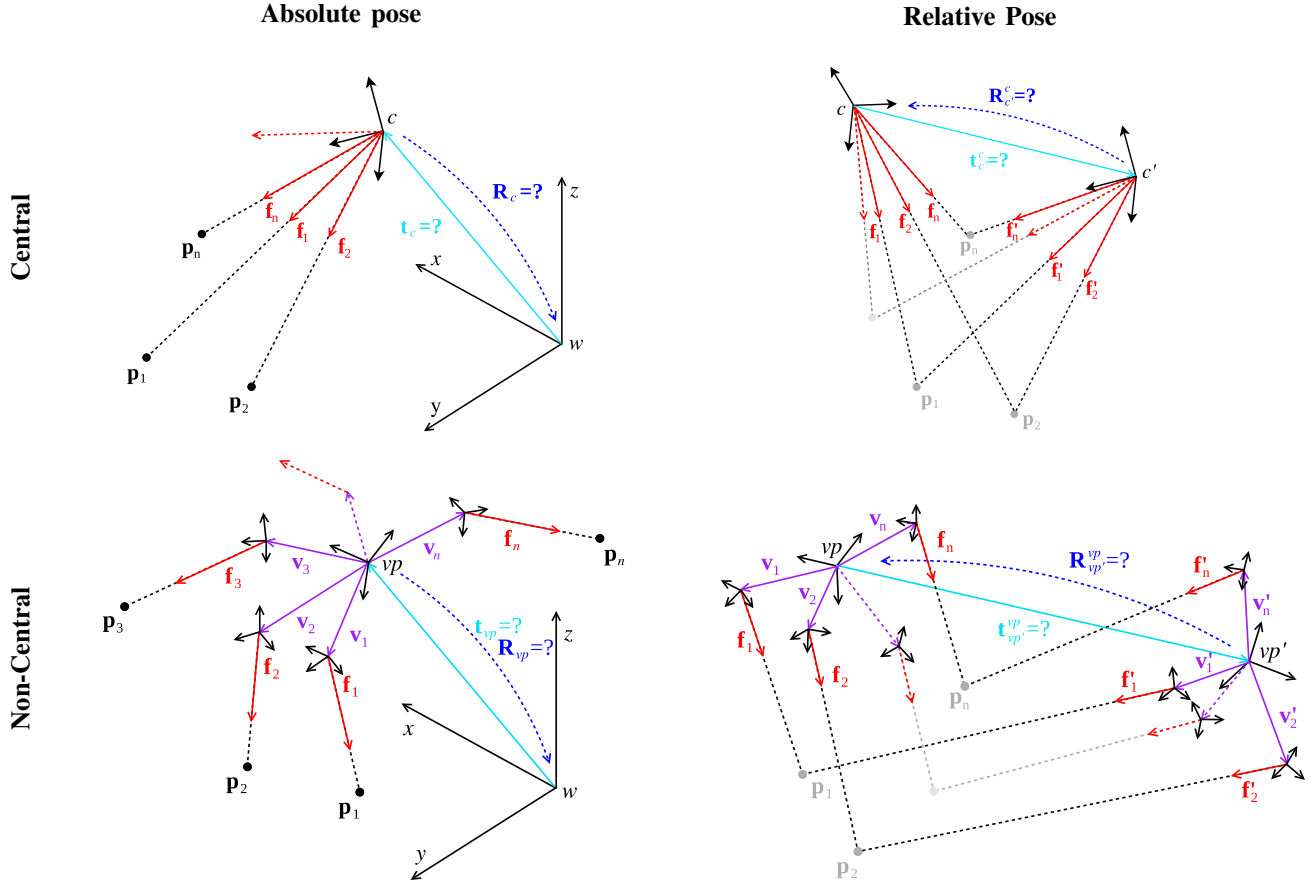


Fig. 2. An overview of the absolute and relative pose computation problems included in OpenGV. All problems can be tackled with either minimal or non-minimal closed-form solvers, as well as within non-linear optimization and robust estimation frameworks. Known variables are bearing vectors $\mathbf{f}_i$ in red, world points $\mathbf{p}_i$ in black, or transformations between the cameras in multi-camera scenarios (the position of cameras inside the viewpoint are given by $\mathbf{v}_i$). Each algorithm solves for either absolute or relative transformation parameters ($\mathbf{t}$ and $\mathbf{R}$). w represents the world frame, c a camera frame, and vp a viewpoint frame containing multiple cameras, as outlined in Figure 1.

As further outlined in Section III, OpenGV implements the “adapter pattern” [8] to define a single interface for accessing viewpoints and bearing vectors according to the above description. Users can leave their internal datastructures unchanged, and interact with the OpenGV algorithms through the implementation of a single adapter class.

B. Overview of the included problem solvers

The main purpose of OpenGV is to provide a set of algorithms to compute the pose of a viewpoint, meaning its position and orientation. OpenGV is able to handle both the absolute and the relative case. In the absolute pose situation, OpenGV computes the pose of a viewpoint in the world reference frame from a number of correspondences between bearing vectors and 3D points expressed in a world frame (2D-3D correspondences). In the relative pose situation, OpenGV computes the pose of a viewpoint with respect to another viewpoint given a number of correspondences between bearing vectors in two different viewpoints (2D-2D correspondences). As summarized in Figure 2, the library provides central as well as non-central solutions to each of these problems, as well as minimal and non-minimal variants. The library therefore provides an unprecedented

completeness and unification of calibrated geometric camera pose computation algorithms.

The central absolute pose problem consists of finding the pose of a viewpoint with a single camera given a number of 2D-3D correspondences between bearing vectors in the camera frame and points in the world frame. The problem seeks the transformation encoded by the position $\mathbf{t}_c$ of the camera seen from the world frame and the rotation $\mathbf{R}_c$ from the camera to the world frame. OpenGV currently hosts a minimal variant using only two points in case the rotation is known (P2P [9]), the minimal P3P solvers presented in [10] and [11], and the n -point solver presented in [12].

The non-central absolute pose problem consists of finding the pose of a viewpoint given a number of 2D-3D correspondences between bearing vectors in multiple camera frames and points in the world frame. The problem seeks the transformation encoded by the position $\mathbf{t}_{vp}$ of the viewpoint seen from the world frame, and the rotation $\mathbf{R}_{vp}$ from the viewpoint to the world frame. OpenGV includes the state-of-the-art minimal and non-minimal algorithms presented in [6], denoted gP3P and gPnP.

The central relative pose problem then consists of finding the pose of a viewpoint with a single camera with respect

to a different single camera viewpoint given a number of 2D-2D correspondences between bearing vectors expressed in the respective camera frames. The problem seeks the transformation encoded by the position $\mathbf{t}_c^c$ of the second camera seen from the first one and the rotation $\mathbf{R}_c^c$ from the second camera back to the first camera frame. OpenGV currently has its greatest variety of solutions in the central relative pose computation domain. The extent reaches from a minimal variant using only two points in case the rotation is known (2pt [9]), a two-point solver for the case of a pure rotation change, an n -point solver for relative rotation, the five-point algorithms presented in [13], [14], and [15]—the latter one solves for rotation immediately—the seven-point algorithm [16], and the eight-point algorithm [17], to the very recent n -point solution presented in [18], which shows for the first time the ability to solve relative pose as an eigenvalue-minimization problem. Note that [13], [14], the seven-point algorithm, as well as [17] can be applied for more than 5, 7, or 8 points as well.

Finally, the non-central relative pose problem consists of finding the pose of a viewpoint with respect to a different viewpoint given a number of 2D-2D correspondences between bearing vectors in multiple camera frames. The problem seeks the transformation encoded by the position $\mathbf{t}_{vp'}^{vp}$ of the second viewpoint seen from the first one and the rotation $\mathbf{R}_{vp'}^{vp}$ from the second viewpoint back to the first viewpoint frame. There is currently only one method for solving this complicated problem in the library, which is the 17-point algorithm presented in [19]. It can be used with an arbitrary number of points, and has the advantage of remaining robust in certain degenerate situations that previous linear generalized relative pose algorithms ignored.

The set of available methods is completed by n -point non-linear optimization methods for both central and non-central absolute and relative pose computation, as well as a few triangulation and point-cloud alignment methods (e.g. [20]). All algorithms—except the triangulation methods—are also made available inside a generic sample consensus framework, which currently includes the most basic and popular RANSAC approach presented in [21].

Since the entire library operates in 3D, we also need a way to compute and threshold reprojection errors in 3D within RANSAC. OpenGV looks at the angle θ between the original bearing-vector $\mathbf{f}_{meas}$ and the reprojected one $\mathbf{f}_{repr}$. By adopting a certain threshold angle $\theta_{threshold}$, we constrain $\mathbf{f}_{repr}$ to lie within a cone of axis $\mathbf{f}_{meas}$ and of opening angle $\theta_{threshold}$, as illustrated in Figure 3. The threshold-angle $\theta_{threshold}$ can be easily approximated from classical reprojection error-thresholds ψ expressed in pixels by using the focal length l , and $\theta_{threshold} = \arctan \frac{\psi}{l}$. OpenGV uses an efficient implementation of this threshold, which we will describe here. The most efficient way to compute the angle between bearing vectors is given by taking the scalar product of $\mathbf{f}_{meas}$ and $\mathbf{f}_{repr}$, which equals to $\cos \theta$. Since this value is between -1 and 1, and we actually want an error that minimizes to 0, we use $\epsilon = 1 - \mathbf{f}_{meas}^T \mathbf{f}_{repr} = 1 - \cos \theta$ to express a reprojection error. The threshold error

is therefore given by

$$\epsilon_{threshold} = 1 - \cos \theta_{threshold} = 1 - \cos(\arctan \frac{\psi}{l}). \quad (1)$$

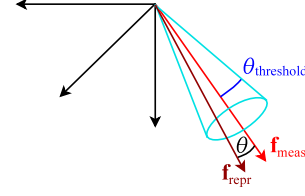


Fig. 3. OpenGV computes reprojection errors in 3D by considering the angle θ between the measured bearing vector $\mathbf{f}_{meas}$, and the reprojected one $\mathbf{f}_{repr}$.

C. RANSAC with multi-camera systems

OpenGV contains a special adapter that is able to hold multiple sets of bearing-vector correspondences originating from pairs of cameras originating from *different* viewpoints. The correspondences in this adapter are accessed via a composite-index built from a pair-index (referring to a specific pair of cameras), and a correspondence-index (referring to the correspondence within that camera-pair). This formulation of pairwise correspondences arises in many practical problems such as:

- non-central relative pose problems that involve two viewpoints originating from motion-estimation with a multi-camera rig. In the situation where the cameras are pointing in different directions, and where the motion between the viewpoints is not too big (a practically relevant case), the correspondences typically originate from the same camera in both viewpoints. We therefore can do a camera-wise grouping of the correspondences in the multi-camera system. Figure 4 illustrates this situation with four camera-pairs.
- central multi-viewpoint problems. By multi-viewpoint we understand here problems that involve more than two viewpoints. As indicated in Figure 5, a problem involving three viewpoints for instance allows to identify three camera-pairs as well. The number of camera-pairs in an

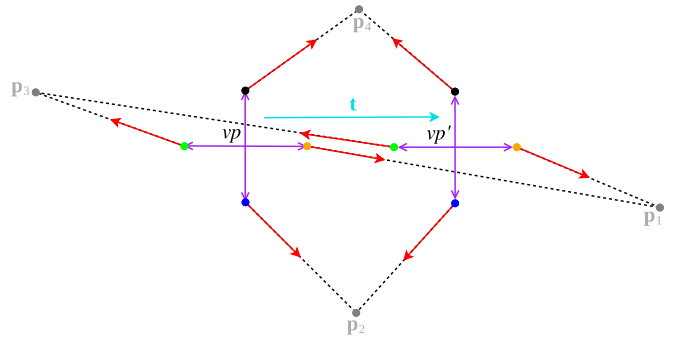


Fig. 4. Correspondences originating from 4 camera-pairs in two viewpoints (black, green, blue, and orange camera).

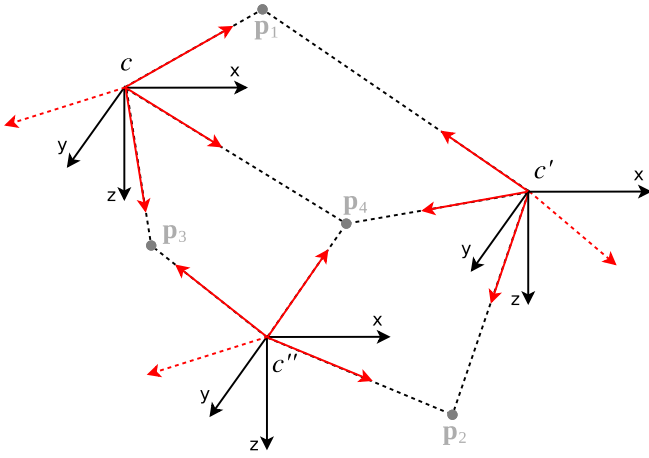


Fig. 5. Multi-viewpoint example with three central viewpoints. It gives rise to the three camera pairs (c, c') , (c', c'') , and (c'', c) . The first pair has a set of correspondences originating from points p_1 and p_4 , the second one from p_2 and p_4 , and the third one from p_3 and p_4 .

n -view problem amounts at most to the combination of 2 out of n , meaning $n(n-1)/2$.

The benefit of keeping track of the pair-wise correspondences between cameras becomes clear when considering a random sample-consensus scheme. When computing the relative pose of a non-overlapping multi-camera system with small viewpoint-change, the most straightforward solution consists of simply applying RANSAC and drawing random samples from a single set containing all feature correspondences. The downside of this approach is that we might not necessarily sample a balanced number of points from each camera pair. This is especially bad in situations where the cameras are pointing into different directions, because we potentially lose the benefit of computing the pose with omni-directional landmark observations and—in the worst case—the solution collapses back to the single camera case. It is well known that the latter is actually an ill-conditioned problem, where the disambiguation between rotation and translation induced disparity in the image plane is far from trivial. It moreover renders the scale of the problem unobservable, and thus ultimately renders multi-camera solutions degenerate. By keeping track of the groups, we can easily ensure homogenous sampling of correspondences over the different camera-pairs. Concerning the multi viewpoint case, one could of course solve a central relative pose problem for each camera-pair individually. However, the simultaneous access of correspondence groups becomes useful here when attempting a joint solution of multiple relative poses by exploiting constraints over loops of Euclidean transformations, which should have identity rotation and zero translation.

OpenGV therefore includes a custom RANSAC mechanism and dedicated adapters that are able to operate on multiple sets of grouped correspondences. It ensures an evenly distributed sampling over all cameras. While this might at first appear as a minor implementational technicality, a deeper reflection quickly shows that there is also a potential trap behind this approach, namely in the situation

where one of the images is actually more affected by outliers than another one. The multi-sample consensus mechanism is therefore preferably extended by an adaptive part that modifies the number of samples drawn from each set towards best increment in performance. This sampling strategy certainly shares similarities with the GroupSAC approach [22], however differs in the fact that it maintains sampling within all groups rather than identifying dominant groups and restricting the sampling to them.

From a theoretical point of view, the approach is best explained by first recalling the original functionality of the RANSAC algorithm. Let p denote the desired probability of the algorithm choosing at least one set containing only inliers during one of all the iterations. Let n denote the number of samples needed for instantiating a hypothesis. Now let w denote the probability of one sample being an inlier. We have $w = \frac{\text{number of inlier points}}{\text{total number of points}} = \frac{y}{z}$. When drawing n samples, the probability of all points being inliers is w^n , and the probability that at least one of the points is an outlier is therefore given by $1 - w^n$. After k iterations, the probability that in each iteration at least one outlier has been sampled goes down to $(1 - w^n)^k$, and by definition this needs to be equal to $1 - p$. This finally leads to the well-known formula introduced by Fischler and Bolles [21] for computing k , the number of iterations required to satisfy a given p :

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}. \quad (2)$$

The situation changes when considering samples drawn from multiple sets of correspondences. In this case, we define a per-set probability of drawing an inlier, which is $w_i = \frac{y_i}{z_i}$, with $i \in \{1, \dots, m\}$ and m being the number of sets (e.g. camera-pairs). If we draw n_i samples from correspondence set i , the final law for computing the remaining number of iterations changes to

$$k = \frac{\log(1 - p)}{\log(1 - \prod_{i=1..m} w_i^{n_i})}. \quad (3)$$

The question now is, which strategy—depending on the n_i —should we choose? The equation arising from (3) considering pairwise correspondence sets and including the term

$$\prod_{i=1..m} w_i^{n_i} = \prod_{i=1..m} \left(\frac{y_i}{z_i} \right)^{n_i}, \quad (4)$$

or the vanilla version in (2) that includes the term

$$w^n = \left(\frac{\sum_{i=1..m} y_i}{\sum_{i=1..m} z_i} \right)^{\sum_{i=1..m} n_i}. \quad (5)$$

In general, we want this term to be maximal, which causes the required number of iterations to go down. Some simple testing reveals that—in case of homogeneous sampling—(5) is bigger or equal to (4), which suggests that the best way of sampling the points is completely random. It is also easy to see that (4) becomes bigger than (5) when pushing the sampling towards the set with highest w_i , in case the latter are not equal.

The problem is that this statistical analysis does not consider the fact that the accuracy of computing a pose with homogeneously distributed points is in general higher, which in turn increases the probability of finding the correct inliers, and thus also reduces the number of iterations.

While a further investigation of the sampling strategy would certainly go beyond the scope of this paper, we restrict ourselves here to an experimental proof of the benefit of homogeneous sampling in balanced situations. The results are provided in Section IV.

III. IMPLEMENTATION

OpenGV depends on the open-source linear algebra toolbox Eigen, as well as the standard library extension boost. Both are header-only libraries, and are readily available for most architectures. All methods in OpenGV accept a variable called an *adapter* as a function-call parameter. Algorithm calls in OpenGV are based on the adapter pattern [8]. Adapters hold the algorithm input (e.g. landmarks, bearing-vectors, poses, multi-camera configuration, and correspondences) in any user specific format (or references to the alike) and they are responsible for converting the data from this format into the format required by OpenGV. This happens either on demand, or—better—when constructing the adapter. The unified access outlined in Section II is then enforced by deriving all adapters from a base class that defines the methods used by the algorithms to access the data in OpenGV format. There are three adapter base-classes:

- `AbsoluteAdapterBase`, the base-class for adapters holding 2D-3D correspondences for absolute-pose methods,
- `RelativeAdapterBase`, the base-class for adapters holding 2D-2D correspondences for relative-pose methods, and
- `PointCloudAdapterBase`, the base-class for adapters holding 3D-3D correspondences for point-cloud alignment methods.

The adapter pattern gives the library great flexibility. Users only have to implement the above adapters for the specific data-format they are using, and can then access the full functionality of the library. Adapters for Matlab mex-arrays are already included, and further adapters such as for instance an adapter for OpenCV keypoint and match-types and a camera model are already planned. A mex-function that exposes all algorithms to Matlab is provided as well. The mentioned benchmark-tool is implemented in Matlab, and uses this interface in order to apply the compiled C++-functions to simulation data created in Matlab. The results are provided in the following section.

IV. BENCHMARK

The algorithms in OpenGV have been used in several projects and perform all reasonably well. Since multiple solutions to identical problems are comfortably made available, OpenGV provides an unprecedented playground for geometric vision algorithms to be compared against each other. In the following, we present the results generated

by our automated benchmark tool. Anyone can reproduce and modify them by just installing the library and using the corresponding Matlab-files, which eases the choice of algorithm for future users. Note that the results generated by OpenGV algorithms are not guaranteed to be totally equivalent to the results presented in the original papers as code for many published algorithms is not available. The present results are based on our best-effort reimplementations of the algorithms included. As this is an open-source project, we invite others to contribute by improving a given algorithm in the library or extending the library with new algorithms.

Figure 6 shows the accuracy of most algorithms. It indicates the mean and median error of the computed rotation (norm of difference between the axis-angle representation of the ground truth and the recomputed rotation) as a function of measurement noise. The value for each noise-level and algorithm is averaged over 5000 random problems. Each problem consists of randomly chosen landmarks having a distance between 4 and 8 to the world frame origin, and a randomly chosen viewpoint with a maximum distance of 2 to the origin, as well as a random orientation. In the relative case this will define the second viewpoint, with the first one being kept at the world frame origin. In the non-central case, we additionally create a random multi-camera system with 4 cameras each one having a maximum distance of 0.5 to the viewpoint origin. Noise is added to the measurements by extracting the orthogonal plane of each bearing vector, normalizing the current noise-level using a focal length of 800, adding random noise in the orthogonal plane, and renormalizing the resulting perturbed bearing vector. The analyzed noise-levels reach from 0 to 5 pixels. All random numbers are drawn from uniform distributions. The analyzed solutions are restricted to those solving for the entire pose, because the accuracy of other algorithms (e.g. P2P, 2pt) depends additionally on the accuracy of the “known” part. If an algorithm has a solution multiplicity, the benchmarks currently do an automatic selection of the best solution based on comparison to groundtruth. As expected, non-linear iterative minimization of the reprojection error outperforms all other solutions in each type of problem.

Regarding central absolute pose, the direct approach P3P (Kneip) presented in [10] shows a minor improvement in robustness compared to traditional two-step approaches such as P3P (Gao) [11]. The EPnP algorithm [12] has superior robustness too, and returns valuable results even in case of using only 6 correspondences (minimum). In the non-central case, the GP3P algorithm performs very well, however can easily face a situation with almost parallel bearing vectors, which renders the position unobservable. The GPnP algorithm has poor robustness, especially when using only a few points. The behavior in the position estimate is a bit better (not shown), but it can still be concluded that—despite of the important recent progress in computational efficiency presented in [6]—the GPnP problem still leaves room for further investigations.

In the central relative pose situation, it can be observed that the most efficient algorithms—the 7pt [16] and the 8pt [17]

TABLE I
EXECUTION TIME OF ALL CAMERA POSE COMPUTATION METHODS

Algorithm	Type	Ref.	Execution time [ms]
P2P	Abs. central	[9]	0.0003
P3P (Kneip)	Abs. central	[10]	0.0033
P3P (Gao)	Abs. central	[11]	0.0115
EPnP (50 points)	Abs. central	[12]	0.1015
2-point	Rel. central	[9]	0.0002
5-point (Stewenius)	Rel. central	[13]	0.1045
5-point (Nister)	Rel. central	[14]	0.2479
5-point (Kneip)	Rel. central	[15]	0.9432
7-point	Rel. central	[16]	0.0261
8-point	Rel. central	[17]	0.0264
Eigensolver	Rel. central	[18]	0.0831
GP3P	Abs. noncentral	[6]	0.0837
GPnP (50 points)	Abs. noncentral	[6]	0.5270
17-point	Rel. noncentral	[19]	0.0912

solvers—are less resilient to noise than the minimal solvers, even if using more points. Since working on calibrated cameras—and thus “normalized” data—the algorithms currently dispose of any explicit data normalization techniques. 5pt (Stewenius) [13] is the most accurate non-iterative solver, and returns almost identical accuracy than the iterative eigensolver [18] when using the same number of points (not shown). The latter, however, returns a unique solution, and potentially outperforms the polynomial solvers in terms of computational efficiency. Note that not all iterations were taken into account. The sanity checks performed by the 5pt (Kneip) algorithm [15] are quite strict, and therefore—in some rare ill-posed experiments—might reject all found solutions. In the non-central relative pose case, the linear 17pt algorithm [19] again performs far from optimal under noise. It is also potentially affected by a singularity, namely when the rotation between the viewpoints approaches identity, which renders the scale unobservable. One of the next goals in OpenGV therefore is an implementation of the polynomial solver presented in [23].

Table I summarizes the execution times of all non-iterative algorithms. All experimental results in this paper have been generated on an Intel Core 2 Duo with 2.8 GHz. P3P (Kneip) out-performs P3P (Gao) because the latter is a two-step approach with an iterative SVD of a 3x3 matrix in the second step. This could be done in closed-form as well, however only at the cost of reduced accuracy. EPnP and GPnP as well as all other n -point methods in OpenGV have at most linear complexity in the number of points. The 7pt and the 8pt solvers are far more efficient than the polynomial solvers. The relatively high execution time of 5pt (Nister) [14] is mainly related to OpenGV’s implementation of the Sturm root bracketing approach, which can certainly be improved. The only iterative algorithm in the list is the eigensolver, which is included because—compared to plain non-linear optimization—it converges comparably easy to the global minimum. The indicated number, however, is for a low number of iterations only. It is only a qualitative statement of the efficiency of the eigensolver, which has constant iteration time independently of the number of features, and therefore drastically outperforms nonlinear optimization.

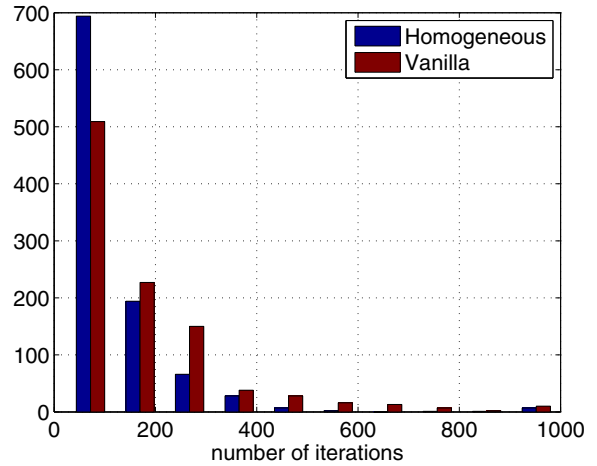


Fig. 7. Histogram of RANSAC-iterations in non-central relative pose computation depending on the sampling strategy. The experiment considers the performance of different sampling strategies in the presence of camera-wise grouping of correspondences. The *homogeneous* sampling strategy samples a balanced number of correspondences arising from each camera, whereas the *vanilla* strategy does completely random sampling.

Figure 7 finally shows the performance improvement resulting from homogeneous sampling in non-central relative pose computation. The experiment analyzes a balanced situation with similar outlier fractions in all images. 1000 random experiments are effectuated for each sampling strategy with 0.5 pix noise and an outlier fraction chosen uniformly between 10% and 20%. Although probabilistics say that completely random sampling is at least as good if not better than homogeneous sampling, the results prove that the gain in accuracy by drawing a balanced number of correspondences from each camera practically leads to higher accuracy, higher probability of finding the correct inliers, and thus a reduced number of iterations. The mean number of RANSAC iterations is 113 for homogeneous sampling, and 164 for completely random sampling.

V. OUTLOOK

The present paper introduced OpenGV, a novel platform for calibrated real-time geometric vision algorithms. We provided a comprehensive overview of the goals behind the OpenGV-project, including use-cases, basic properties, as well as a comparative evaluation of the contained algorithms. OpenGV is the first library to unify state-of-the-art real-time solutions to the outlined types of problems in a single place, but there obviously remains space for extensions. Our future efforts consist of adding more algorithms to the library, thus extending the comparative benchmarks for the different problem classes. We also hope that we can convince people to add their algorithms to the library—no matter if they are old or new—which could ultimately make OpenGV *the* place where algorithms can be exposed to identical conditions, and thus easily and fairly compared against each other.

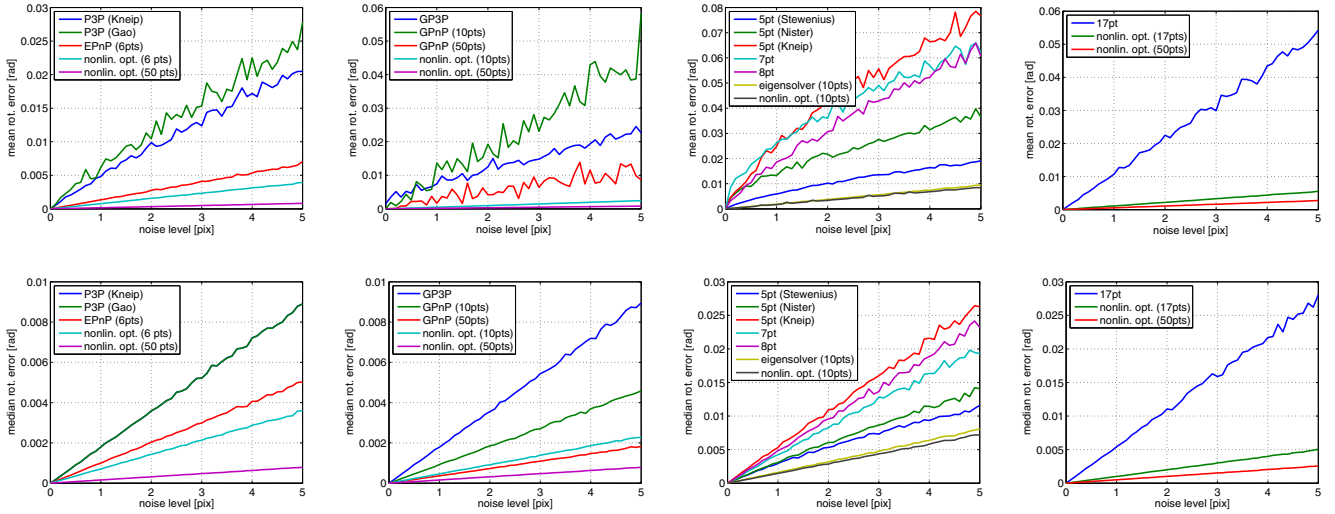


Fig. 6. Benchmark of multiple algorithms showing rotation error in function of measurement noise. The plot reflects the current status of OpenGV by including the main closed-form and iterative optimization methods. The value for each algorithm/noise level is averaged over 5000 random experiments. Noise levels are expressed in pixels, but internally applied on the bearing vectors' orthogonal plane using a focal length of 800 pixels.

ACKNOWLEDGMENT

The research leading to these results has received funding from ARC grants DP120103896 and DP130104567, and the EU project FP7-269916 (*Vcharge*).

REFERENCES

- [1] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building rome in a day. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 72–79, 2009.
- [2] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, 2007.
- [3] A. Davison, D. Reid, D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):1052–1067, 2007.
- [4] J. Kelly and G.S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *International Journal of Robotics Research (IJRR)*, 30(1):56–79, 2011.
- [5] G. Bradski. The OpenCV Library. *Dr. Dobbs' Journal of Software Tools*, 2000.
- [6] L. Kneip, P. Furgale, and R. Siegwart. Using Multi-Camera Systems in Robotics: Efficient Solutions to the NPNP Problem. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.
- [7] R. Pless. Using many cameras as one. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 587–593, Madison, WI, USA, 2003.
- [8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
- [9] L. Kneip, M. Chli, and R. Siegwart. Robust real-time visual odometry with a single camera and an IMU. In *Proceedings of the British Machine Vision Conference (BMVC)*, Dundee, Scotland, 2011.
- [10] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, USA, 2011.
- [11] X.S. Gao, X.R. Hou, J. Tang, and H.F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.
- [12] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate $O(n)$ solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):578–589, 2009.
- [13] H. Stewenius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.
- [14] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777, 2004.
- [15] L. Kneip, R. Siegwart, and M. Pollefeys. Finding the Exact Rotation Between Two Images Independently of the Translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Firenze, Italy, 2012.
- [16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, second edition, 2004.
- [17] H.C. Longuet-Higgins. *Readings in computer vision: issues, problems, principles, and paradigms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.
- [18] L. Kneip and S. Lynen. Direct Optimization of Frame-to-Frame Rotation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Sydney, Australia, 2013.
- [19] H. Li, R. Hartley, and J.-H. Kim. A Linear Approach to Motion Estimation using Generalized Camera Models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, Alaska, USA, 2008.
- [20] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(5):698–700, 1987.
- [21] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [22] K. Ni, H. Jin, and F. Dellaert. GroupSAC: Efficient consensus in the presence of groupings. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Kyoto, Japan, 2009.
- [23] H. Stewenius and D. Nistér. Solutions to Minimal Generalized Relative Pose Problems. In *Workshop on Omnidirectional Vision (ICCV)*, Beijing, China, 2005.

C Motion Estimation Using Rolling-Shutter Cameras for an Autonomous Car

Author: Bürki

Link to [Table of Contents](#).



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Motion Estimation Using Rolling-Shutter Cameras for an Autonomous Car

Master Thesis

Mathias Buerki

Advisors

Dr. Paul Furgale

Prof. Roland Siegwart

May 28, 2014

1 Introduction

Rolling-shutter cameras have become predominant in a broad field of consumer electronic devices, such as cell phones and digital photo cameras. This is mainly due to the fact that rolling-shutter sensor chips are cheaper in production than their global-shutter counterpart. However, in addition to that, rolling-shutter cameras often also come with additional benefits that make them interesting for applications depending on good image quality and high photo sensitivity rather than mere cost. One of these benefits is the artefact-free images produced by rolling-shutter cameras. In contrast to global-shutter cameras, which often suffer from smearing artefacts caused by low global-shutter efficiency, the electronic charge from exposed sensor rows on a rolling-shutter chip can immediately be moved off the light sensitive area, resulting in sharp and artefact-free pictures even in situations with very low exposure times and/or the presence of bright light spots such as sun reflections. This greatly simplifies image processing, especially in outdoors conditions, as no extra care has to be taken to handle these artefacts common to global-shutter images. In addition to that, the fact that the rolling-shutter mechanism allows overlapping the exposure and read-out phase of multiple frames, higher frame-rates at the same exposure time, or longer exposure at the same frame rate can be achieved compared to global-shutter cameras. This makes rolling-shutter cameras especially interesting for the use in all-day, outdoor applications, such as autonomous cars and driver assistant systems - an area of research going through rapid development in the past few years.

However, despite this wide range of applications and benefits from rolling-shutter cameras, relatively little effort has been put into the development of algorithms taking the rolling-shutter dynamics into account. The focus on previous work on rolling-shutter cameras has mainly been set on the rectification of rolling-shutter images to allow post-processing them further with global-shutter image processing algorithms. The goal of this work, on the other hand, is to develop an online image processing pipeline that takes the rolling-shutter dynamics fully into account by the means of an appropriate model. We achieve this by employing a continuous time state-space approximation with cubic B-Splines embedded in a sliding-window estimator for online application. The resulting estimation framework is then tested in the problem-context of online estimation of the motion of a car, equipped with rolling-shutter cameras and rear-wheel odometry sensors.

To our best knowledge, the sliding-window estimator developed in this thesis, using cubic B-Spline as a continuous state space approximation, presents a novel approach in continuous time online filtering.

The remainder of this thesis are structured as follows: After an overview of how this thesis relates to previous work, we motivate and describe the functionality of our continuous-time sliding-window estimator in detail, focusing on the integration of rolling-shutter images and the marginalization step required to move the sliding-window forward in time. We then describe how the developed framework is integrated into our experimental set-up involving a car with rolling-shutter cameras and rear-wheel odometer, before we present some experimental results.

D Long-term 3D map maintenance in dynamic environments

Author: Pomerleau, Krüsi, Colas, Furgale, Siegwart

Link to [Table of Contents](#).

Long-term 3D map maintenance in dynamic environments

François Pomerleau, Philipp Krüsi, Francis Colas, Paul Furgale and Roland Siegwart

Abstract—New applications of mobile robotics in dynamic urban areas require more than the single-session geometric maps that have dominated simultaneous localization and mapping (SLAM) research to date; maps must be updated as the environment changes and include a semantic layer (such as road network information) to aid motion planning in dynamic environments. We present an algorithm for long-term localization and mapping in real time using a three-dimensional (3D) laser scanner. The system infers the static or dynamic state of each 3D point in the environment based on repeated observations. The velocity of each dynamic point is estimated without requiring object models or explicit clustering of the points. At any time, the system is able to produce a most-likely representation of underlying static scene geometry. By storing the time history of velocities, we can infer the dominant motion patterns within the map. The result is an online mapping and localization system specifically designed to enable long-term autonomy within highly dynamic environments. We validate the approach using data collected around the campus of ETH Zurich over seven months and several kilometers of navigation. To the best of our knowledge, this is the first work to unify long-term map update with tracking of dynamic objects.

Index Terms—Long-term mapping, dynamic obstacles, ICP, kd-tree, registration, scan matching, robot, SLAM.

I. INTRODUCTION AND RELATED WORK

The success of SLAM has been a major enabler of robot autonomy. Until recently, the majority of research focused on increasing the accuracy and robustness of single-session SLAM. Now that robotic hardware and software are becoming more widespread, applications such as navigation in dense crowds [1], or autonomous driving in cities [2] are demanding new algorithms that can maintain maps over time. A SLAM system should not only be able to build a map of the geometry of an environment, but it should also be capable of updating this map over time and of encoding useful semantic information for planning. For example, a map that includes lane and average speed information may be used to calculate the probable trajectories of cars tracked by a robot, so planning a collision-free path may be focused on likely trajectories and thus, can be computed faster. Prior knowledge of motion patterns within a map may also be used to detect anomalous behavior of other agents so that extra attention may be paid to these agents. Furthermore, it is important that dynamic objects are not included in static environment maps as they may degrade localization accuracy and cause motion planning to fail. Motivated by these applications, we therefore seek to develop a mapping



Fig. 1. ARTOR, a search and rescue robot specialized for outdoor applications navigating in a highly dynamic urban environment. Typical mobile elements include pedestrians, bikers, cars, trucks and trams.

and localization system that is able to build and maintain these hybrid geometric/semantic maps.

This paper presents an algorithm that performs SLAM, map updating, classification of map points as dynamic or static, and estimation of the velocity of dynamic points, all in real time and using only data from a 3D laser scanner. The approach is bottom up; it does not rely on prior information, such as object models. The only prior used is a weak smoothness assumption on the velocity of dynamic scene points. By saving the time history of point-wise velocity estimates, we are able to infer the dominant motion patterns in the map. The algorithm is validated using data collected over seven months with the field robot shown in Fig. 1 in the highly dynamic urban area around the main campus of ETH Zurich.

The work presented here is related to two research areas: map updating for lifelong navigation and segmentation of dynamic obstacles during localization and mapping. As the techniques used in laser processing and vision processing have not yet converged on common solutions for either of these problems, we will restrict our survey of prior work to papers that use lidar data.

The work of Biber et al. [3] describes a long-term map updating scheme that shares the same goals as our framework. They describe a long-term two-dimensional (2D) SLAM system based on scan matching and odometry. The mapping system maintains a set of local maps (2D scans at fixed positions) with multiple hypotheses for the range values at different timescales. Specifically, they highlight four requirements for any long-term mapping strategy: (1) map adaptation should not depend on the wall-clock time, (2) mapping should be resilient to outliers, (3) multiple

hypotheses should be maintained until there is enough data for inference, and (4) the map should only contain measured values (not interpolated or smoothed quantities). Our method fulfills the same requirements, but we extend the capability to 3D space, remove the dependence on viewpoint and, rather than maintaining multiple hypotheses, we probabilistically segment samples into static and dynamic. Our segmentation strategy is similar to the dynamic mapping technique of Burgard et al. [4] who use an expectation maximization scheme to differentiate between dynamic and static cells in a 2D grid map. They show that removing dynamic points from the data during mapping increases localization accuracy. However, they make no attempt to cluster or track dynamic objects throughout a scene. Theoretically, it should be possible to extend their method to process 3D laser data by adopting an efficient 3D grid representation such as Octomap [5]. However, the processing of data with long sensor beams in large outdoor areas can become prohibitively slow due to the requirement to ray-trace through the grid. In contrast, our method works in real time on raw 3D laser data in expansive environments. The trade-off is that we do not explicitly model free space throughout the entire volume of the mapped area—only in places where our laser has returned a previous reading. While the explicit modeling of free space may be required for mobile manipulation or flying robots navigating in tight spaces, it is not strictly necessary for a large class of mobile robots with local sensing that is accurate enough to infer the drivable area directly around the robot.

There have been a number of other algorithms developed specifically for updating map over several passes through an environment. Aijazi et al. [6] use highly accurate localization based on a differential global positioning system to resolve points from a single pass into a fixed grid. Grids from multiple passes can then be directly compared to infer the static parts of the scene. The algorithm produces excellent segmentation results, but the reliance on highly accurate localization makes it unsuitable for the general case of a moving robot. Ryde and Hillier [7] also use a grid-based representation to detect changes in 3D laser maps. They sidestep the need for accurate localization by matching the latest point cloud with an existing voxel grid. A change is detected after alignment by finding points that do not lie within an occupied voxel.

Another class of work has ignored map update to focus on the segmentation and tracking of dynamic objects from laser data. Several studies have produced notable results using a static 3D laser [8], [9], [10]. However, it is not immediately clear how to extend them to a moving sensor. Wang et al. describe an impressive system for 2D laser-based SLAM with moving object tracking [11]. The system segments the incoming laser scan, matches segments with the predicted locations of tracked objects, and accumulates point-based models of the dynamic objects and the static scene. Moosman and Fraichard have developed a very similar system for 3D laser data [12]. However, both approaches rely on the initial range-image segmentation step, and neither one addresses the difficult problem of splitting tracks that can appear when

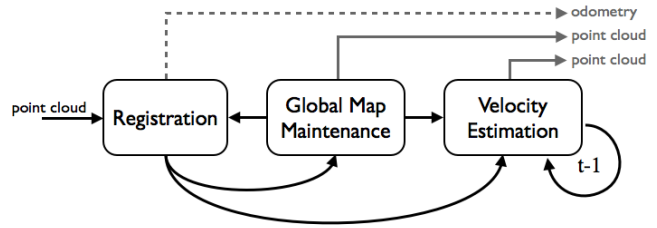


Fig. 2. Block diagram of the processing pipeline. Boxes represent separate processes running at different frequencies. Solid arrows represent point clouds being communicated to each module and dashed line, odometry. The gray arrows show the output of each module.

the initial segmentation step clusters objects with similar positions and speeds. In contrast, our approach is entirely point-based with a weak smoothness prior on velocities. The lack of reliance on segmentation allows us to seamlessly handle clumps of agents that merge and diverge without relying on specialized detection and handling of these cases.

To the best of our knowledge, this is the first work to unify map update with tracking of dynamic obstacles, and we believe it is a significant step toward the unified geometric and semantic mapping needed for robot navigation in highly dynamic environments.

II. SYSTEM OVERVIEW

A general overview of the proposed system is presented in Fig. 2. Point clouds from the sensor are provided to the *Registration* module, which corrects the odometry of the robot and provides the point clouds in global coordinates. The *Global Map Maintenance* module evaluates if past information were dynamic or not, and concatenate the new registered point cloud in order to keep the global map at a constant density. The *Velocity Estimation* module takes the newly registered point cloud and segments it based on the information of the global map. What is currently considered as dynamic at time t is then evaluated against the last dynamic elements at time $t - 1$. The module outputs a point cloud with all the points having a speed larger than zero. The 3D registration module is based on the already published *libpointmatcher* [16], therefore this article will focus on the two other modules.

III. DYNAMIC ELEMENT IDENTIFICATION

We infer dynamic parts of the scene based on visibility assumptions; if we observe a laser point behind a point that was previously observed, that previous point might be dynamic. For such process, the standard approach is ray-tracing. While widely used in 2D, scaling up to 3D is expensive in memory as it requires a dense representation of both the occupied and free space. To ensure online computation, we propose to directly use the same representation as for the localization: sparse point clouds. First, we transform a local subset of the point cloud map Q into the reference frame of the current point cloud P . Then, using spherical coordinates, we associate the points q of the map to each single reading point p in the same small conical aperture

of size θ_{max} . This can be done quickly using an efficient kd-tree implementation, `libnabo` [13]. All the points of the map that are further than the point of the current sensor reading in each cone are left untouched. However, the points of the map that are closer than the reading need to be updated as they should have intercepted the ray. Based on our field observations, this update should fulfill the following criteria:

- 1) the greater the angle between the beam producing p and q , the less we change the knowledge on q ,
- 2) the greater the angle between the beam producing q and its surface normal n , the less we change the knowledge on q ,
- 3) if p and q are spatially close, q is more probably static; otherwise more probably dynamic,
- 4) a point has more chances to become dynamic knowing that it is static than the inverse,
- 5) most of the new points observed are static.

Most of those criteria are easy to motivate, except maybe criterion 2), which require more explanations. Indeed, the sensor produces readings of environmental elements that can be located at up to 80m from the sensor itself. At that distance, many points from the ground can be in the cone of a reading point due to the big incidence angle, but they should not be considered as dynamic. This problem was also observed by Wurm et al. [5] but not explicitly addressed in their updated equations.

In order to give a formal expression to those criteria, we observe that the problem is to update some knowledge state based on uncertain information. Therefore, we use a Bayesian approach to update the knowledge on each map point q . We consider the following notations for the variables, while the parameters are defined in Table I:

- p : point from a newly acquired reading expressed from the center of the sensor,
- q : point of the global map expressed from the center of the sensor and associated to p ,
- Dyn : binary variable indicating whether q is now dynamic or not,
- $Odyn$: binary variable indicating whether q was dynamic or not,
- U : binary variable indicating whether we need to update the point or not,
- θ : angle between p and q defined as $\arccos\left(\frac{q \cdot p}{\|q\| \cdot \|p\|}\right)$,
- ϕ : incidence angle on q based on its surface normal n defined as $\arccos\left|n \cdot \frac{q}{\|q\|}\right|$,
- δ : distance between q and p defined as $\|p - q\|$.

We can write:

$$P(Dyn|\theta, \phi, \delta) \propto \sum_{U, Odyn} \left| \frac{P(Odyn)P(U|\theta, \phi)}{\times P(Dyn|Odyn)P(\delta|Dyn)} \right| \quad (1)$$

Where:

- $P(Odyn)$: is either a prior (80% chance to be static) or the result of a previous inference,
- $P(U|\theta, \phi) = \begin{cases} \frac{\theta}{\theta_{max}} \left(1 - \frac{2\phi}{\pi}\right) & \text{if } \|p\| \geq \|q\| \\ 0 & \text{otherwise} \end{cases}$

TABLE I

DEFINITION OF THE PARAMETERS AND THEIR VALUES USED FOR OUR TARGETED SCENARIOS.

	Values	Descriptions
θ_{max}	1°	Angle around which all points in Q are associated to a point p .
α	0.99	Probability of staying static given that the point was static.
β	0.90	Probability of staying dynamic given that the point was dynamic.
ϵ_d	0.1 m	Fixed noise on depth measurement of a point p .
ϵ_a	0.2	Ratio of noise based on depth measurement of a point p .
γ_{max}	0.9	Probability at which a point is considered permanently dynamic.

is the probability to update based on separation and incidence angles,

$$\begin{aligned} \bullet P(Dyn|Odyn) &= \begin{bmatrix} \alpha & 1 - \beta \\ 1 - \alpha & \beta \end{bmatrix} \\ &\text{is a decay matrix to allow points to change,} \\ \bullet P(\delta|Dyn) &\propto \begin{cases} \max(0, \min(1, \epsilon_d + \|p\|\epsilon_a - \delta)) & \text{if } Dyn \text{ is false} \\ 1 - P(\delta|Dyn = false) & \text{if } Dyn \text{ is true} \end{cases} \end{aligned}$$

is the observation probability distribution.

With this model, we are able to compute whether a point is dynamic or static based on multiple observations. In the results section, we will demonstrate that this works well for highly dynamic objects. However, it can have issues with dynamic objects that are periodic (i.e., object that comes back at the same location often). This applies particularly to trams that are constrained to their tracks and to cars parked in well-defined parking spaces. The definition of a static object becomes therefore ambiguous, and higher-level models of objects are needed. Here, we retain a bottom-up approach by deciding that if an object is sufficiently dynamic—meaning that it was seen and disappeared—then it cannot go back to being static if $P(Dyn = true) \geq \gamma_{max}$.

IV. VELOCITY ESTIMATION

Building on top of the dynamic object classification, one can estimate the velocity of moving objects. To be useful for dynamic obstacle avoidance algorithm, like the one proposed by Ruffi et al. [14], velocity must be extracted at high rate. Most approaches rely on the clustering of the points into objects for which the velocity is then estimated looking, for example, at the change in position of the center of mass. In this section, we briefly introduce our fast and generic approach as a complement to the dynamic element classification.

From a newly acquired point cloud P_t at time t , we associate all of its points to the global map. A subset of mobile points M_t is generated from P_t , fulfilling the requirement of being a dynamic obstacle. This can be based on the dynamic element identification (as described in the previous section) of the global map and on the definition of obstacles for a given platform. Those dynamic obstacles M_t

can then be compared to the last subset M_{t-1} to extract velocity vectors. We based our approach on point-cloud registration using iterative closest point (ICP), where M_t is the *reading* point cloud and M_{t-1} is the *reference*. Having different transformation parameters for each point is known as non-rigid ICP [15]. We reuse the underlying principles but extracted only translation components instead of the full 6 degrees of freedom (DOF) transformation. In essence, we propose to do dual non-rigid ICP—both from reading to reference and from reference to reading—and, given that we have a timestamp per point, divide the alignment error with the difference of acquisition time to estimate the velocity vectors. We use neighboring constraints to harmonize the velocities across close points.

As shown in Fig. 3, the measurements received are sparser than typical full point clouds used with ICP. Moreover, a high ratio of noise is possible, especially during exploration of new areas where not enough information has been acquired to accurately classify points. To cope with those challenges, we vary the number of nearest neighbors inversely proportional to the number of iterations. More precisely, for every p_{read} in M_t , we assign k nearest neighbors p_{ref} from the reference point cloud M_{t-1} . Then, we compute the average velocity vector produced by the k matched points and assign it to p_{read} . This augments the robustness of the association phase against noisy matches when using a large k , while keeping the accuracy of a single match at the last iteration. To ensure locally coherent velocities, we apply a windowed-mean filter in the Euclidean space for all p_{read} . The iterative process reuses velocity estimates of the latest iteration to project points before association with the speed of new points initialized to zero. In the current instantiation of the algorithm, five iterations were sufficient for convergence.

Often, objects change rapidly in term of shape and density between two scans, which lead to an asymmetry between the velocity estimation from the reading to the reference, and vice-versa. To cope with this situation, we compute both directions of matching in parallel. Fig. 3 shows an example of this process on a pedestrian at different viewing distances with points projected in both directions. As it can be seen from the projected points (in green and blue), our process is able to correctly estimate the velocity of the points by re-projecting them properly to the other point cloud taken at a different time. It also shows the importance of the iterative process; with point clouds 1 m apart, points from one point cloud would all initially match a few points of the other, like an extended foot or arm.

Having several iterations that average over matches and neighbors fosters local consistency in the velocity estimate while allowing deformations. Although multiple nearest neighbor searches are used, only one kd-tree generation is required for every new scan, which gives a very fast computation. As opposed to cluster-based approaches, all dynamic objects can be treated in parallel as each point has its own velocity. Timing will be discussed in the results section.

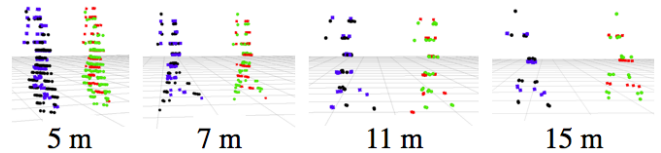


Fig. 3. Registration example of a pedestrian over different distances from the robot. Points in black are the current readings and the points in red are the references, taken roughly 0.5 s before. The points in green are the projections of the readings to the reference, while the points in blue are the projections of the reference to the readings.

V. EXPERIMENTAL RESULTS

Although we could evaluate our approach in a simulated environment, we decided to go for real environments leading to a richer set of events at the expense of a direct access to precise ground truth information. We validated our approach with two scenarios in controlled environments and two large scale (i.e., kilometer range) experiments with daily challenges caused by dynamic elements. The robot assigned to the task was ARTOR (see Fig. 1), a platform based on the LandShark system by Black-I Robotics (USA), with custom modifications realized by RUAG Land Systems (Thun, Switzerland). The robot has a maximum speed of 3.5 m/s but is typically driven at 1 m/s in crowded environments. Equipped with a large sensor suite, the final platform is suitable for applications ranging from Search & Rescue to surveillance and reconnaissance. The main sensor used for the experiments is the Velodyne HDL-32E. It produces roughly 70'000 points per 360° scan at a rate of 11 Hz and with a maximum range of 80 m. The map representation is a set of sparse points with information about their surface normals, timestamps, probability of being dynamic, etc. A new point is only added to the map if the distance to its nearest neighbor in the global map is larger than 0.3 m. This keeps the point density of the global map constant, keeping the computation time close to real time. All the following results were obtained by running the input data at the same rate at which they were recorded.

A. Dynamic Element Segmentation

The goal of this experiment is to evaluate the capability of the system to identify dynamic elements. We selected the visitor parking lot of a hospital, which means that cars do not typically stay overnight. The section of the parking lot we surveyed has 50 dedicated parking places. Moreover, the middle of the parking lot is also a busy bike path during the day. Thus, this environment presents two kinds of dynamic obstacles: cars that come and go in between experimental runs, and bikes and pedestrians that move during the runs.

The robot was driven around to survey the area at different times during three consecutive days. The first survey was considered as the exploration phase, while the following missions are built upon the prior map. Points are kept in the map independently of their categorization and only split at the end for evaluation with the threshold $P(Dyn = true) < 0.5$. The environment and the path the robot did is depicted in

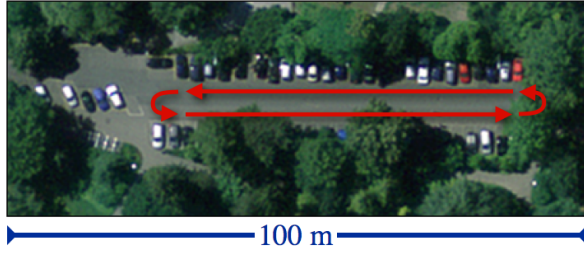


Fig. 4. Aerial view of the parking lot used for the segmentation experiment. In red, the survey path realized by the robot. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000.

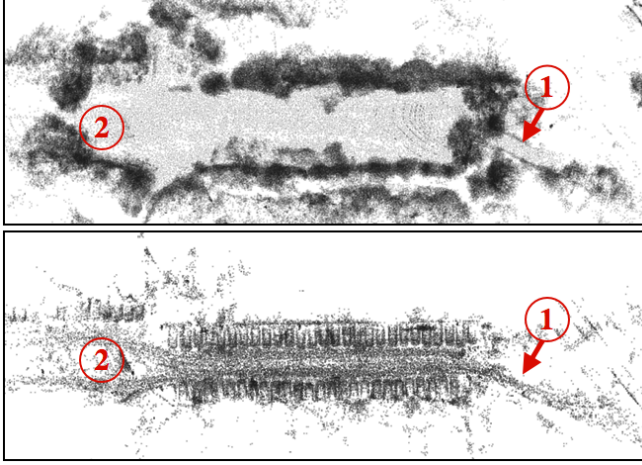


Fig. 5. Result of the segmentation after 9 surveys over the course of 3 days. *Top*: Reconstruction with $P(Dyn = true) < 0.5$. *Bottom*: Reconstruction with $P(Dyn = true) > 0.5$. The flow of pedestrians and bikes can be seen using a path (1) and splitting to avoid another row of parked cars (2).

Fig. 4. During the experiment, 61 pedestrians, 27 bikes and 10 cars were in motion through the surveyed area, without considering the multiple punctual changes in parked cars that appeared or disappeared through the days.

In addition to those surveys, we produced a ground truth map of the environment with a density that was three times higher than our regular maps. We recorded this map at night in order to have neither cars parked nor bikes or pedestrians. This map constitutes our ground truth for static elements and is not part of the evaluation set. In the following experiment, a point in the survey map is considered static if there is a point in the ground truth map within a radius of 0.15m; otherwise, it is considered dynamic.

Fig. 5 shows the resulting segmentation between static and dynamic points. In the static map (top panel), the trees and the ground are clearly visible, whereas in the dynamic map (bottom panel) the cars are well highlighted as well as many points in the middle belonging to bikes or pedestrians. Moreover, the comparison of both maps in position (1) shows a path in the static map and trails from pedestrians and bikes in the dynamic map. The same comparison in position (2) shows the flow of pedestrians and bikes splitting around parked cars.

Using our ground truth map, we can evaluate the error

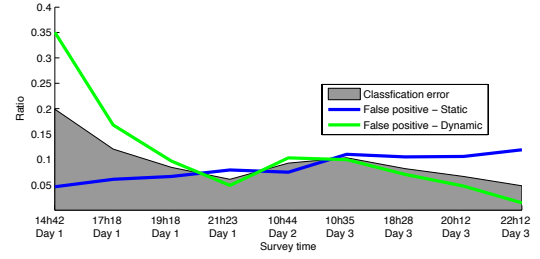


Fig. 6. Evolution of the segmentation results for each surveyed time. Blue line: static points wrongly identified with $P(Dyn = false) < 0.5$. Green line: dynamic points wrongly identified with $P(Dyn = true) < 0.5$. Shaded area: total classification error expressed as the ratio of wrong classification over the total number of points.

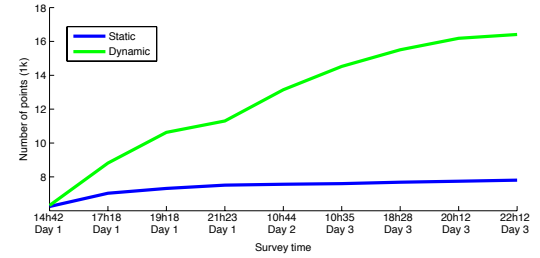


Fig. 7. Evolution of the static (in blue) and dynamic (in green) points added to the map for each surveyed time. The classification is done by comparison with our ground truth map.

rate of our classification. Fig. 6 presents both the global classification error, as well as the error for each class. If we distinguish static points from dynamic points, we can see two different evolutions. On the one hand, the error on the dynamic points decreases over time. This is expected because we need to observe that the point is missing to provide a classification. On the other hand, we see a steady increase in the rate of static points. Those points are mainly points that were close to the ground and were classified as dynamic; often, they are the lower part of cars. Finally, we observe an overall decrease of the error, from 20% to around 5%. This shows that there are more points that are dynamic than static.

Fig. 7 illustrates the evolution of the number of static and dynamic points in the map. This graph confirms that the number of static points is smaller than of dynamic points and that the difference increases with time. As expected, the number of new points added into the map at each visit decreases with each run as the environment gets to be better known. Moreover, the static information gets added into the map faster than dynamic elements and at the end of the first day, most static points are there. The plateaus in the dynamic points at the end of both day 1 and day 3 show the decrease in activity in the evening for this parking lot. Finally, with each successive run, dynamic points are added into an almost uniform 2m-thick layer above the ground. These points represent all the pedestrians and bikes that have crossed the area during all the runs. The main issue about those dynamic points, greatly outnumbering static points, is that they can prevent ICP localization to properly align the

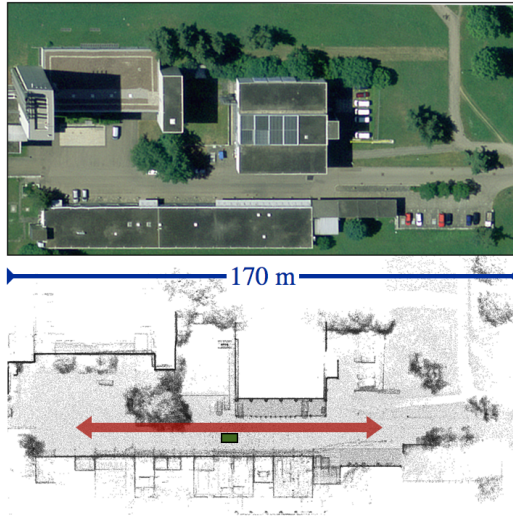


Fig. 8. Experimental setup for the velocity estimation. *Top*: Aerial view of the street used for the tests. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Bottom*: 3D reconstruction of the street with the zone reserved for our controlled dynamic elements in red and the position of the robot in green.

current sensor reading with the map, causing a slow drift in the localization of the robot. With our approach, we can use $P(Dyn)$ as a weight in the ICP algorithm in order to not trust dynamic points for the computation of the transformation between the point clouds. This way, we were able to solve the issue of localization drifting due to the dynamic points, by providing both a more precise and cleaner map of the environment. This is consistent to what Burgard et al. [4] have already demonstrated in their 2D experiments.

B. Velocity Estimation

In this experiment, we aim at assessing the capability of the system to estimate different velocities while minimizing the noise induced by the dynamic element segmentation and the localization. The experiment was conducted in a controlled environment consisting of a remote street without traffic. This allowed us to add only one moving obstacle at a time. First, we drove the robot along the street in absence of any dynamic elements, yielding a 170 m wide 3D map of the static part of the environment. Fig. 8 shows an aerial view of the test area, the 3D reconstruction of the street, and the zone where we moved objects. We parked the robot at the indicated position (in green), and then let different dynamic objects pass by in linear motion. We tested two kinds of dynamic objects: a pedestrian and a minibus. The pedestrian was asked to cross the scene once by *Walking* and once by *Jogging*. The driver of the minibus was asked to drive at three different speeds: the lowest speed the vehicle could go (approximately 2 m/s, *Slow*), at 5.5 m/s (*Medium*) and at 11 m/s (*Fast*).

Fig. 9 shows the resulting velocity estimation of our system. The median values of the pedestrian’s estimated speed were 1.7 m/s and 4.1 m/s for *Walking* and *Jogging*, respectively. In the experiments with the minibus, the speed

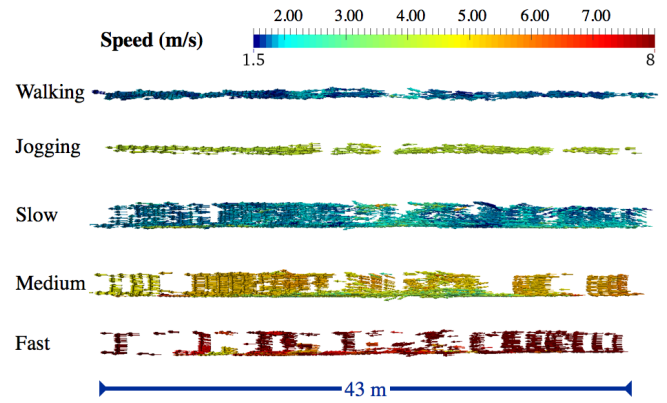


Fig. 9. Top view of the 3D dynamic trails. The 2 first lines represent the trails of a pedestrian, and the 3 last lines represent the ones of a minibus. The color indicates the estimated velocity in m/s.

of the vehicle could be roughly controlled using the car’s speedometer. The velocities listed above represent the *maximum* values reached in each of the three runs. However, the acceleration phase was included in the test track, and is therefore in the estimation. This explains why the estimated median values—1.9 m/s (*Slow*), 4.6 m/s (*Medium*), and 8.1 m/s (*Fast*)—are considerably lower than the targeted speeds. One can also observe that velocities can be estimated at up to 22 m from the sensor.

C. Applications

Finally, we conducted two experiments in a dynamic urban environment. The aim was to demonstrate the performance of our system in real world scenarios, and to give an idea of the range of possible applications. In the first experiment, we drove the robot three times along a 1.3 km long route in the city of Zurich, spread over seven months (March 12, May 23 and September 9, 2013). The first pass was used as the exploration phase, with all subsequent passes building upon the prior map. Fig. 10 shows the results of the survey; the bottom graph mapped the count of dynamic elements that were removed to produce the static map. This graph can be used to identify zones of interest. Two construction sites, that partially occupied the streets, are marked with (1) in red. Marked with (2), is a very large tree (i.e., 1 m diameter trunk) that has been chopped between March and May. Finally, the zone marked with (3) is a busy intersection with cars, trucks, bike paths, trams and many pedestrians (see Fig. 1). The experiment shows that our system can be employed to extract zones that are potentially dangerous to navigate (i.e., places that contain plenty of dynamic objects) or that exhibit large seasonal changes.

The second experiment was conducted in front of the main building of ETH Zurich. It took place during the *information day*, which meant that many young students gathered in the streets, with sometimes as many as 15 persons in the vicinity of the robot. The main street consists of two large sidewalks, two lines for cars, and two lines for trams. The robot surveyed the area twice within 20 minutes, each time

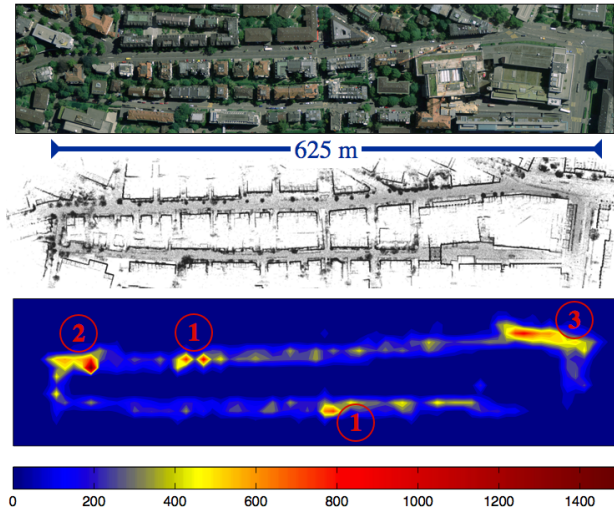


Fig. 10. Long range survey over a 1.3 km long path. The environment was monitored over a period of seven months. *Top*: Aerial view of surveyed area. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Middle*: 3D reconstruction after dynamic elements removal. *Bottom*: Occurrence of dynamic elements. The graph highlights the position of (1) construction sites, (2) a large tree and (3) a busy street intersection. Color represent to number of dynamic points over a cell size of 10 m.

driving on the sidewalks on both sides of the street. Fig. 11 presents the results of the experiment. The two lower graphs show the extracted dynamic objects over the course of the survey, with their estimated speed and direction of motion. In the speed graph (left), blue corresponds to the range of typical walking speeds of pedestrians. The sidewalks and the pedestrian crossings (the latter marked with red arrows) can be clearly identified by looking at the blue objects. Furthermore, there are two lines of faster objects (yellow to red), which designate the car lanes. Note that the velocities are lower in the vicinity of the pedestrian crossings, which comes from the fact that drivers stop to let people cross the street. In the orientation graph (right), the two main directions of the cars are clearly visible. On the sidewalks the situation is naturally more chaotic, as pedestrians do not walk on distinct lanes. Trams were less detected because their speed can only be detected if the robot sees the front or the rear of the wagons. Otherwise, they look like large walls appearing and disappearing from the laser perspective.

The result of this experiment is a first, yet significant, step towards automatic road graph extraction: our system can correctly identify regions of low speed (sidewalks and pedestrian crossings) and road lanes, including the direction of traffic. Enough data to identify crowd and traffic behaviors were collected only by surveying the environment twice and in continuous motion. The velocity extraction is robust enough to estimate a significant number of measurements, even when exploring the environment for the first time.

D. Computation Time

All computations were realized on a single laptop with a four-core Intel Core i7 and 4 GB of RAM. As explained earlier with Fig. 2, all modules run at different speeds. The

registration module runs between 6 and 11 Hz by down-sampling the input points and using the wheel odometry as prior alignment. The map maintenance, including the concatenation of the new information and identification of dynamic elements, runs in average at 2 Hz, even with maps as large as 600'000 for the 1.3 km long survey. The computation time for the velocity estimation depends on the number of dynamic elements in the scan. In average, it is 0.03 s (≈ 30 Hz) in the single dynamic object experiment. The laser sensor produces scans at 11 Hz, which means that we can follow the sensor rate, with some margin. However, a typical pedestrian would only move by 15 cm between two scans. This poses a problem for the speed estimation, as the velocity vectors become noisy at very small distances. On the other hand, if the object is too fast, the assumption of the closest point will fail. As a compromise, we slowed down the data rate to 8 Hz, which we found to be a good compromise to handle velocities from 1.5 to 10 m/s.

VI. CONCLUSION AND FUTURE WORK

Our paper presents an online approach for computing both the probability of a 3D point to be dynamic or static and the velocities of dynamic points. Based on 3D point clouds as a sparse representation, we use a Bayesian model for assessing whether a point is dynamic and static, which leads to a cleaner map and a better localization. We also use dual non-rigid ICP to simultaneously compute the velocity of all dynamic points. This approach is cluster- and model-free, with only a weak smoothness assumption, and is able to successfully evaluate the velocity of dynamic objects.

We have shown that identifying dynamic objects produces an accurate map of the static scene geometry. This is especially important for difficult path-planning tasks in highly cluttered 3D environments. In both cases, it is detrimental for dynamic objects to be wrongly classified as obstacles, as it could invalidate the only feasible path to a goal.

We have also shown that the integration of the velocity information in the map reveals the main characteristics of traffic or pedestrian flow. This could be used to extract higher-level semantic information, which is necessary for more advanced path-planning techniques in dynamic environments. For example, a lot of advanced collision avoidance techniques require the ability to predict the trajectory of all other dynamic objects. This is usually not feasible, unless those dynamic objects are other robots tracked by an external localization system [17], restraining the use of such techniques. Our approach provides velocity estimates for dynamic obstacles as well as the aggregated knowledge of past observations, which may be used to predict future behavior, like the possibility that cars slow down at the pedestrian crossing. In future work, we would like to use the time history of dynamic objects to build probabilistic models of motions within the map, as the 2D camera model in [18], and then use these models to perform safe real-time motion planning and navigation in crowded urban settings. We will also evaluate a larger range of parameters to define their impact on the robustness of the system and give more

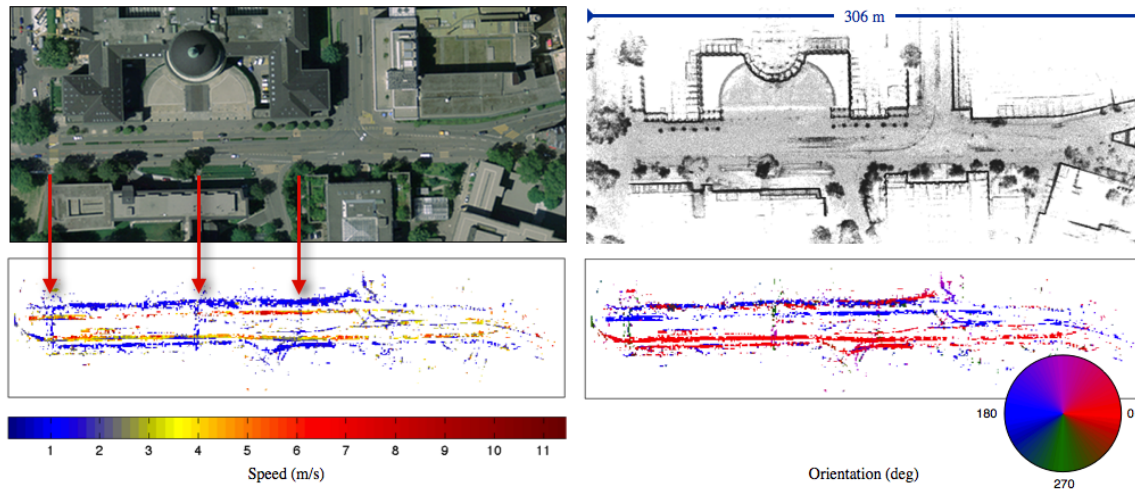


Fig. 11. Extraction of velocity information at a global scale. *Top left*: Aerial view of the street in front of ETH Zurich. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Top right*: 3D reconstruction after dynamic element removal. *Bottom left*: Average speed of the moving objects. *Bottom right*: Average orientation of the moving objects. The red arrows highlight the pedestrian crossings.

insides on how others could tune them given their specific applications. We believe that with our approach, we have brought advanced path-planning techniques closer to field robotics.

VII. ACKNOWLEDGMENTS

This work was supported by the EU FP7 IP projects Natural Human-Robot Cooperation in Dynamic Environments (ICT-247870), TRADR (ICT-609763), the armasuisse S+T UGV research program, and the EU FP7 2007-2013 Program, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No 269916, V-Charge. The authors are grateful to M. Ruffli, U. Schwesinger and M.-È. Garneau for useful comments.

REFERENCES

- [1] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "A navigation system for robots operating in crowded urban environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.
- [2] P. Furgale, U. Schwesinger, M. Ruffli, W. Derendarz, H. Grimmer, P. Mühlheller, S. Wonneberger, J. T. S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brünig, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart, "Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project," in *IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, Australia, 23–26 June 2013, pp. 809–816.
- [3] P. Biber and T. Duckett, "Experimental analysis of sample-based maps for long-term slam," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 20–33, 2009.
- [4] W. Burgard, C. Stachniss, and D. Hhnel, "Mobile robot map learning from range data in dynamic environments," in *Autonomous Navigation in Dynamic Environments*, ser. Springer Tracts in Advanced Robotics, C. Laugier and R. Chatila, Eds. Springer Berlin Heidelberg, 2007, vol. 35, pp. 3–28.
- [5] 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 *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010 (ICRA)*, 2010.
- [6] A. Aijazi, P. Checchin, and L. Trassoudaine, "Automatic Removal of Imperfections and Change Detection for Accurate 3D Urban Cartography by Classification and Incremental Updating," *Remote Sensing*, vol. 5, no. 8, pp. 3701–3728, Aug. 2013.
- [7] J. Ryde and N. Hillier, "Alignment and 3D scene change detection for segmentation in autonomous earth moving," in *Robotics and Automation, 2011. Proceedings of the IEEE International Conference on*, 2011, pp. 1484–1490.
- [8] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, "Generative object detection and tracking in 3D range data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 3075–3081.
- [9] J. Shackleton, B. VanVoorst, and J. Hesch, "Tracking People with a 360-Degree Lidar," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 2010, pp. 420–426.
- [10] P. Anderson-Sprecher, R. Simmons, and D. Huber, "Background subtraction and accessibility analysis in evidence grids," in *Robotics and Automation, 2011. Proceedings of the IEEE International Conference on*, 2011, pp. 3104–3110.
- [11] C. C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous Localization, Mapping and Moving Object Tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, Sept. 2007.
- [12] F. Moosmann and T. Fraichard, "Motion estimation from range images in dynamic outdoor scenes," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 142–147.
- [13] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration," *Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 2–12, Mar. 2012.
- [14] M. Ruffli, J. Alonso-Mora, and R. Siegwart, "Reciprocal Collision Avoidance With Motion Continuity Constraints," *Robotics, IEEE Transactions on*, vol. 29, no. 4, pp. 899–912, 2013.
- [15] J. Feldmar and N. Ayache, "Locally affine registration of free-form surfaces," in *Computer Vision and Pattern Recognition, 1994. Proceedings of the IEEE Computer Society Conference on*, 1994, pp. 496–501.
- [16] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, Feb. 2013.
- [17] J. Alonso-Mora, A. Breitenmoser, M. Ruffli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, May 2012.
- [18] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, November 2011.

E Scene Signatures: Localised and Point-less Features for Localisation

Authors: Colin McManus, Ben Upcroft and Paul Newman

Link to [Table of Contents](#).

Scene Signatures: Localised and Point-less Features for Localisation

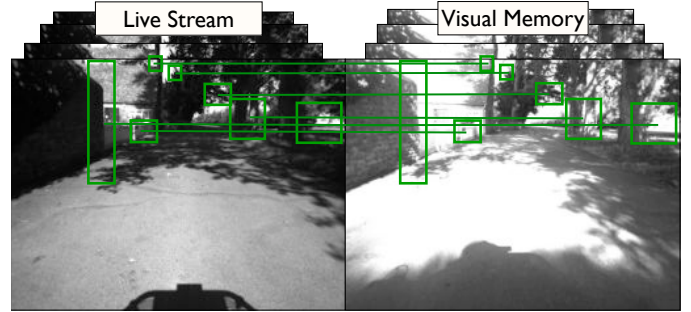
Abstract—This paper is about localising across extreme lighting and weather conditions. We depart from the traditional point-feature-based approach since matching under dramatic appearance changes is a brittle and hard thing. Point-feature detectors are rigid procedures which pass over an image examining small, low-level structure such as corners or blobs. They apply the same criteria to all images of all places. This paper takes a contrary view and asks what is possible if instead we learn a bespoke detector for every place. Our localisation task then turns into curating a large bank of spatially indexed detectors and we show that this yields vastly superior performance in terms of robustness in exchange for a reduced but tolerable metric precision. We present an unsupervised system that produces broad-region detectors for distinctive visual elements, called *scene signatures*, which can be associated across almost all appearance changes. We show, using 21 km of data collected over a period of 3 months, that our system is capable of producing metric localisation estimates from night-to-day or summer-to-winter conditions.

I. INTRODUCTION

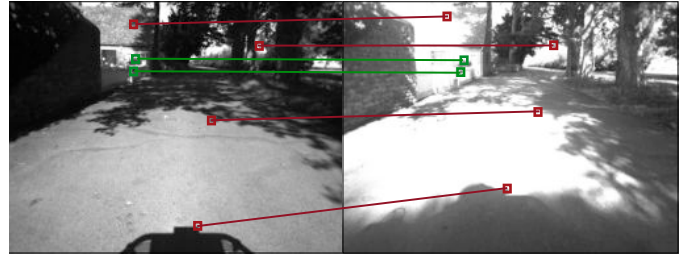
Matching point features between different images has been the standard approach in visual motion estimation for decades and has led to a number of impressive systems for pose estimation and/or mapping over large scales (e.g., Visual Teach & Repeat (VT&R) [1], and Visual Simultaneous Localisation And Mapping (VSLAM) [2]). Matching low-level features such as edges, blobs, or corners works well when observing the same scene under similar conditions (e.g., for on line ego-motion estimation); however, when trying to match images taken at different times of day or in different seasons, these low-level features often look utterly different. Larger image structures on the other hand, (think: windows, signs, or doors), offer more hope as they capture shape and texture on a broader scale. We will show that if we are careful about how we identify suitable structures – which we shall refer to as *scene signatures* – then they can be reliably matched under large variations in lighting, thus opening the door to robust localisation.

Low-level point features can be thought of as being on one extreme of a localisation paradigm, while using whole-image information, such as SeqSLAM [3], can be thought of as being on the opposite end of that spectrum. However, it should be noted that SeqSLAM just provides an estimate on the topological location of the vehicle and not a metric pose estimate. If we wish to provide a vehicle controller with a metric pose estimate, we need something in between these two approaches.

In this paper, we present an unsupervised approach to find distinctive visual elements, such as windows, signs, doors, or



(a) By matching *scene signatures* from a live stream (left) to a memory (right), we are able to *successfully* localise our vehicle.



(b) By matching point features from a live stream (left) to a memory (right), we are unable to successfully localise our vehicle.

Fig. 1. An illustration of the benefits of matching scene signatures, which are distinctive visual elements such as fences, windows, tree lines, etc., versus the traditional point-feature approach. Using point features for data association under extreme appearance changes often fails because point features only consider low-level structure, like edges, corners, or blobs. Scene signatures are more robust since they are large, distinctive elements.

tree silhouettes (see Figure 1), in a given place, π_p , which is a node in a hybrid topological/metric map. These are distinctive signatures specific to the scene, and so we refer to them as *scene signatures*. We wish to stress that the benefit of using scene signatures over point features is that we can associate these scene signatures across extreme appearance changes, such as night to day or sunny to winter.

We shall constrain ourselves to the task of teaching a vehicle, for example an autonomous car, to localise using vision. We assume that the vehicle has or will be driven through the environment on multiple occasions and so we have many examples of the appearance of the places the vehicle drives through. Rather than building a map of point features against which to match point features detected at run time, we will construct, in an unsupervised way, a large set of spatially indexed classifiers, which are associated with topological locations in the world. Each of these classifiers

is carefully constructed to fire on a particular and distinctive aspect of the environment at that particular place, π_p . As the vehicle progresses through its environment, we will retrieve the classifiers, $\{c_i\}_p$, relevant to its location, π_p , and use them to identify known structure in the live image feed. These broad level features are used to create a “weak localiser” of sufficient accuracy to provide coarse local, metric information about the vehicle’s pose.

Immediately we should ask, “for what tasks is such precision adequate?” We envision a hierarchical system in which at the top level we have very crude topological localiser which outputs the gross location of the vehicle. This output drives the localiser described in this work which takes a topological hint and returns a metric position accurate in orientation but with perhaps tens of centimeters in translational error. We assert that for autonomous road-vehicle navigation and control, we only need a coarse metric estimate of the vehicle’s pose, after which, lower level lane following and/or curb detection algorithms can be applied to refine the estimate for a vehicle controller. This is a shift from the traditional methods that try and obtain centimeter-level accuracy. For a road vehicle with on-board obstacle avoidance and lane following software, global localisation accuracy to the half metre is sufficient.

The novel contributions of this paper are the following: (i) the introduction of “weak localisers,” which use scene signatures to perform metric estimation, (ii) an unsupervised method which finds distinctive scene signatures, and (iii) the validation on challenging datasets displaying extreme appearance changes, from full light to deep darkness.

II. BACKGROUND

Decades of work have been focused on designing interest-point detectors and descriptors that can identify repeatable features and describe them using unique, compact representations. The output of these systems has enabled efficient feature correspondence across images taken at different viewpoints. Popular corner detectors include Harris Corners [4] and FAST [5], while blob detection can be performed with the Laplacian of Gaussian or MSER [6]. A range of image-point descriptors also exist, for example SIFT [7], SURF [8], BRIEF [9], and ORB [10] to name a few. However, all of these interest-point detectors/descriptors operate on small image patches, which can look entirely different under different lighting and/or weather conditions. We will show that scene signatures enable matching across extreme changes in appearance because they contain large, distinctive elements in the image. Note that our approach is very different from the localisation and mapping systems of Davison et al. [11], [12], which use image patches as their landmarks. These methods still rely on interest-point detection to find the patches and they use small patches (e.g., 11×11 pixels in size). By construction, scene signatures are large distinctive elements in the scene that can be matched across extreme appearance changes.

Recently, there has been a number of attempts to shift away from the traditional, straight forward approach of blindly applying an out-of-the-box point-feature detector/descriptor

for egomotion estimation and/or localisation. Richardson et al. [13] present a method for learning an optimal feature detector for Visual Odometry (VO) tasks. Their method searches the space of convolution filters to find the detector that minimises reprojection error. Although this method is aimed at improving standard detection methods for an application specific task, it still focused on using point features, which works well for VO, but not for localisation (e.g., matching a sunny day against a rainy day).

Lategahn et al. [14] present a method for learning an optimal whole-image descriptor for place recognition. They use a genetic optimisation approach to find the optimal combination of fundamental feature blocks to construct their optimal descriptor. However, as with other methods, such as SeqSLAM [3], this can only inform the system of the topological position of the vehicle; it does not provide a metric pose estimate, which is important for us because we are interested in controlling a vehicle.

Rublee et al. [10] developed a new feature called ORB, which builds upon the FAST [15] detector and the BRIEF [9] descriptor. They use a greedy learning algorithm for de-correlating BRIEF features under rotational invariance. However, as this is still based on low-level structure, data association remains hard under extreme appearance change. Hundelshausen et al. [16] present a noteworthy descriptor that goes beyond point features and instead constructs a network of nodes and directed edges, where each edge is a descriptor in the network, referred to as a “d-token”. However, because these descriptors directly sample pixel intensities, this would not be suitable for the types of extreme appearance changes we are considering.

Ultimately, we are concerned with the problem of long-term, robust localisation in outdoor environments, which experience a great deal of appearance changes (e.g., time of day and/or time of year). One approach to this problem would be a system like experience-based navigation [17], which records distinct visual experiences of the environment as the vehicle traverses. If the live video stream cannot be matched to a prior experience, it means the appearance of the world has changed enough to warrant the creation of a new experience. Although this is a feasible approach, we offer an alternative that tries to learn what elements in the environment are stable across all appearances. In this way, localisation is not done against numerous experiences, but rather just a collection of distinctive scene elements.

Recently, Doersch et al. [18] presented a method for extracting geo-distinctive image patches from a collection of images of London and Paris. Their method was able to find image patches, or *visual elements*, of windows, balconies, and street signs which clearly distinguished the Parisian streets from the London streets. The method is, in principle, very simple and relies on a large amount of data and a cross-validation training scheme. This will be discussed further in the next section. We have applied this idea to the localisation problem to find distinctive visual elements that are stable across a wide range of appearance changes, such as lighting differences and/or

seasonal changes. We call these scene signatures. The benefit of using scene signatures instead of low-level point features, which look for corners, edges, or blobs, is that the data association problem becomes less challenging, since scene signatures are very distinctive (e.g., doors, signs, windows, etc). As we use a stereo camera as the primary sensor, we can perform left-to-right matching and obtain 3D position information for each scene signature. This allows us to swap out point features for scene signatures in a VO framework, in order to produce metric pose estimates.

III. SYSTEM OVERVIEW

Here we describe the two main components of our system: (i) the notion of a “weak localiser” that uses scene signatures for pose estimation, and (ii) the offline training algorithm which produces the scene signatures. As we will show, using scene signatures instead of point features offers vast improvements in robustness to extreme appearance changes, resulting in a more robust localisation system.

At a high level, the steps involved in our localisation system work as follows:

- 1) Initialisation in the map (e.g., place recognition system),
- 2) Use dead reckoning (e.g., wheel odometry) to predict what place, π_p , the vehicle is close to and load the bank of SVM classifiers, $\{c_i\}_p$, associated with that place,
- 3) Provided that the vehicle is sufficiently close to that place (e.g., within several meters), we use each SVM classifier at multiple scales to search for associations in the live image,
- 4) For each association, we compute the 3D stereo landmarks and solve for the optimal transformation estimate against the map.

An illustration of our system is shown in Figure 2.

A. WEAK LOCALISERS

For a given place, π_p , scene signatures represent distinctive visual elements, such as buildings, trees, or distinctive structure boundaries in the scene. Examples of scene signatures can be seen in Figure 3. Note that each place is associated with a set of SVM classifiers trained on distinctive scene signatures. We will show in the next section how these scene signatures can be learned offline in an unsupervised manner. However, let us assume that for now we have access to a bank of spatially indexed (for example by distance along a road) SVM classifiers of scene signatures, $\{c_i\}_p$.

Although these scene signatures represent large areas in the image, they can still provide a good metric idea of where the vehicle is locally. Additionally, because we can perform left-to-right matching between the stereo pair, each scene signature has an associated 3D point, allowing us to produce metric estimates local to each place.

In order to obtain sensible solutions, sensible handling of the measurement uncertainties is required. The positional uncertainty of a visual element in image space, $\mathbf{P}_{z_i}$, will be a function of the scale, s , at which it was detected, the area

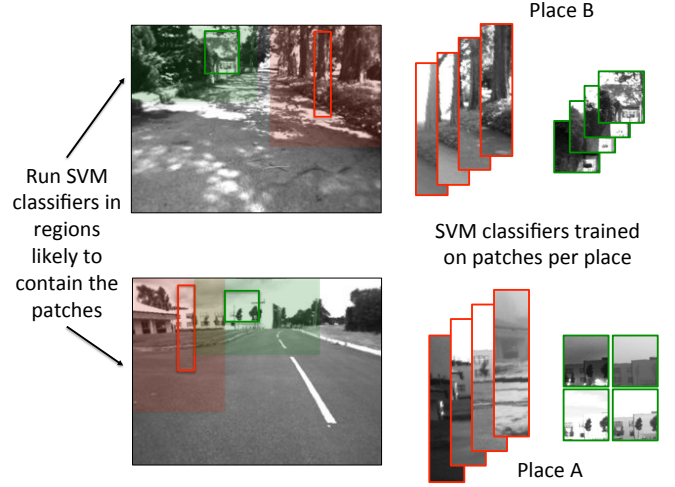


Fig. 2. Offline, we learn scene signatures in the form of SVM classifiers. Each classifier is associated with a particular place, π_p , and spatial region in the image (i.e., the region that it’s most likely to fire in). At run-time, we use the bank of pretrained classifiers associated with π_p to perform data association and then localisation. By using larger, distinctive visual elements, we are able to localise in regions with extreme appearance change, where the point-feature-based counterpart fails.

of the patch, a , the search resolution used when detecting the feature, r , and the SVM detection probability, λ :

$$\mathbf{P}_{z_i} = \mathbf{f}(a, r, s, \lambda). \quad (1)$$

The relationship between the scale and search resolution is given by,

$$\mathbf{P}_{z_i} \propto \frac{1}{s} \mathbf{P}_r, \quad (2)$$

where $\mathbf{P}_r$ is the noise covariance on the search resolution, which is scaled according to the pyramid level at which the detector fires. The relationship with the other parameters, however, is less clear. Intuitively, we expect that the lower the probability of being a scene signature and the larger the area of the patch, the less certain the keypoint position should be. Thus, as a heuristic, we assume that the covariance takes the following form,

$$\mathbf{P}_{z_i} := \frac{a}{\lambda s} \mathbf{P}_r. \quad (3)$$

Since each patch feature, $\mathbf{z}^j$, has an associated 3D landmark, $\mathbf{p}^j$, we can use the standard stereo model, $\mathbf{h}(\cdot)$, to predict the location of a landmark in frame b relative to some other frame a , according to the transformation matrix, $\mathbf{T}_{a,b}$:

$$\mathbf{z}_a^j = \mathbf{h}(\mathbf{T}_{a,b}, \mathbf{p}_b^j) + \mathbf{n}_a^j, \quad \mathbf{n}_a^j \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_{z_a^j}). \quad (4)$$

Additionally, we use a strong prior, $\hat{\mathbf{T}}_{a,b}$, with a small uncertainty in the vertical offset between the live frame and the map, as well as roll and pitch, since we know that these positional differences would be small for a road vehicle. The prior is important because the translational component of the localisation estimate will not be very accurate owing to the low number of patches in the foreground. However, we note that

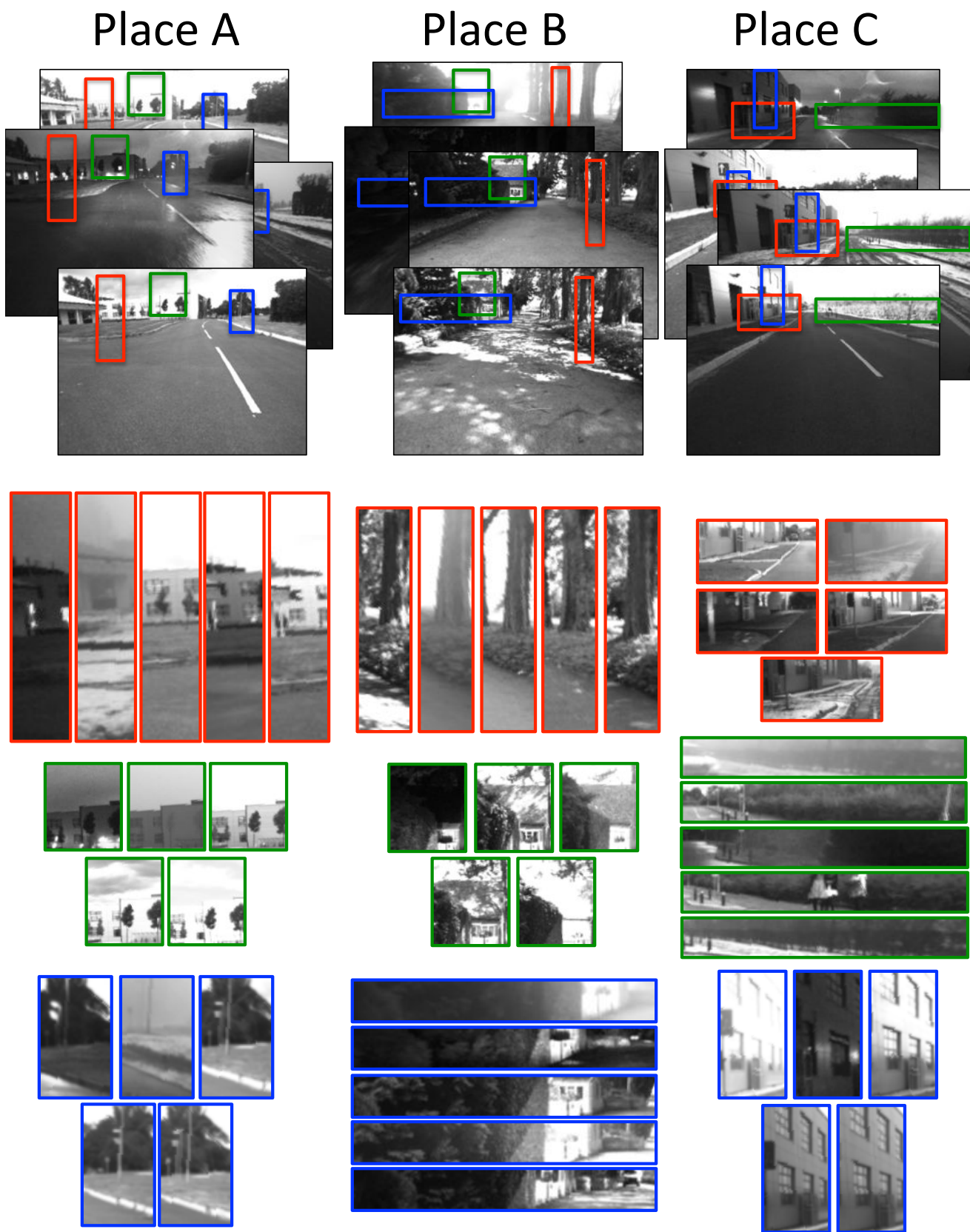


Fig. 3. Example scene signatures learned by our algorithm. Image sets from the same place with varying appearances (represented by run-times in this figure) are used offline to learn these distinctive scene signatures. SVM classifiers are trained for each cluster of scene signatures and can be used at run-time on the live image stream to perform data association, followed by metric pose estimation. Note that the shapes vary in size and dimension and tend to pick up things like changes in structure boundaries, as these are very distinctive (e.g., from road, to grass, to building, to sky).

we obtain very good orientation estimates, so combined with a reasonable egomotion estimate from, say, wheel odometry or VO, the weak localisers are sufficient in providing pose estimates with similar accuracy as our INS system (i.e., sub meters) in large outdoor environments. Including the prior estimate, $\hat{\mathbf{T}}_{a,b}$, the final least-squares system we seek to optimize is given by the following:

$$\mathcal{O}(\mathbf{T}_{a,b}) = \frac{1}{2} \begin{bmatrix} \mathbf{q}(\mathbf{T}_{a,b}, \hat{\mathbf{T}}_{a,b}) \\ \mathbf{z}_a - \mathbf{g}(\mathbf{T}_{a,b}, \mathbf{p}_b) \end{bmatrix}^T \begin{bmatrix} \mathbf{P}_x^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_z^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{q}(\mathbf{T}_{a,b}, \hat{\mathbf{T}}_{a,b}) \\ \mathbf{z}_a - \mathbf{g}(\mathbf{T}_{a,b}, \mathbf{p}_b) \end{bmatrix}, \quad (5)$$

where

$$\mathbf{z}_a := \begin{bmatrix} \mathbf{z}_a^0 \\ \vdots \\ \mathbf{z}_a^M \end{bmatrix}, \quad \mathbf{p}_a := \begin{bmatrix} \mathbf{p}_a^0 \\ \vdots \\ \mathbf{p}_a^M \end{bmatrix}, \quad \mathbf{P}_z := \text{diag}(\mathbf{P}_{z_a^0}, \dots, \mathbf{P}_{z_a^M}), \quad (6)$$

and $\mathbf{q}(\cdot)$ is a function that takes two SE3 transformation matrices and computes a 6×1 error vector, which depends on the choice of the orientation parameterisation. Note that we also use the Geman-McClure [19] robust cost function, which leaves us with the following objective function:

$$\mathcal{O}(\mathbf{T}_{a,b}) = \frac{1}{2} \sum_i \frac{\mathbf{e}_i^T \mathbf{P}_i^{-1} \mathbf{e}_i}{\sigma_i^2 + \mathbf{e}_i^T \mathbf{P}_i^{-1} \mathbf{e}_i}, \quad (7)$$

where σ_i are the M-estimator parameters and each $\mathbf{e}_i$ represents an error term (e.g., the prior or measurement). This has the effect of scaling the covariance to down weight the contribution of potential outliers during the iterative solve, which is done using Levenberg Marquardt [20].

B. UNSUPERVISED LEARNING OF SCENE SIGNATURES

This section will describe our unsupervised approach to learning scene signatures, which are locally distinctive and stable visual elements. *Locally distinctive* means that the visual element is distinct in a local region in image space. Given that we have a reasonable prior on the motion of the vehicle, it does not matter if the visual element occurs elsewhere in the image, it need only be locally distinctive for data association. *Stable* means that the visual element can be identified across multiple images of the same area, under a variety of appearances.

The training algorithm can be divided into the following steps.

- 1) Collect a set of N images with a variety of appearances at a particular location in the world (we have $N=31$ in our experiments).
- 2) For each image, partition it into M tiles (see Figure 4; we have $M=4$ in our experiments).
- 3) For all M tiles, do the following (we refer to each iteration here as a “round”):
 - a) Set the same tile in all N images as the “positive” tile, which means all patches sampled from this tile are put in the positive set (the light brown tile in Figure 4).

- b) Sample K patches of varying dimension from the positive tile (these are in the positive set for this round).
- c) Select a band around the positive tile; patches sampled in these regions are put in the negative set (the blue band in Figure 4).
- d) Sample K patches of varying dimension from the negative band (these are in the negative set for this round).
- e) Split the positive and negative patches sampled from all the tiles into l positive/negative datasets (we use $l = 3$ in our method).
- f) For each patch, compute a HOG descriptor [21] and compute the top 20 nearest neighbours for each of the positive patches. Note that each grouping of 20 patches represents the seed for a candidate scene signature cluster.
- g) Prune the candidates by discarding candidates where more than half of the nearest neighbours belong to the negative set, or if there are more than 2 overlaps with any of the other clusters. This reduces the number of candidate clusters to approximately 200.
- h) Perform cross-validation training on l datasets to produce a set of SVM classifiers for the scene signatures.

This last step requires some detail to be given. In each iteration, we train an SVM detector using the top k nearest neighbours for each candidate cluster and all of the negative-set patches as negative examples ($k = 5$ in our experiments). Then, the newly trained detectors are applied to one of the other datasets to select the top k detections for retraining the SVM. After retraining the SVM, the process continues: applying the detectors trained on the previous round to the other datasets. If the top k detections for each dataset stop changing, then the detector has converged and the image patches are deemed sufficiently distinctive and stable. As in [21], we use a maximum of three iterations for convergence. Note that the output of this process is a set of bespoke SVM classifiers tuned to a particular place, $\{c_i\}_p$ per region in our map, π_p .

Figure 3 shows example clusters that were generated from this algorithm (note that this is a small subset; typically we find around 30-40 patches for each image). Each scene signature has been colour coded to show which cluster it belongs to. As can be seen, many of the clusters are things like distinctive tree silhouettes, corners of doors, fences, windows, etc. Another interesting observation to make is that the road is typically ignored, which is to be expected as it is mainly a homogeneous texture and not very distinctive between datasets.

IV. EXPERIMENTS AND RESULTS

In this section, we compare our weak-localiser approach to a typical point-feature-based approach (SURF [8] in our case) for the task of localisation. We trained our system with 31 datasets of a 750m outdoor loop. Places were defined

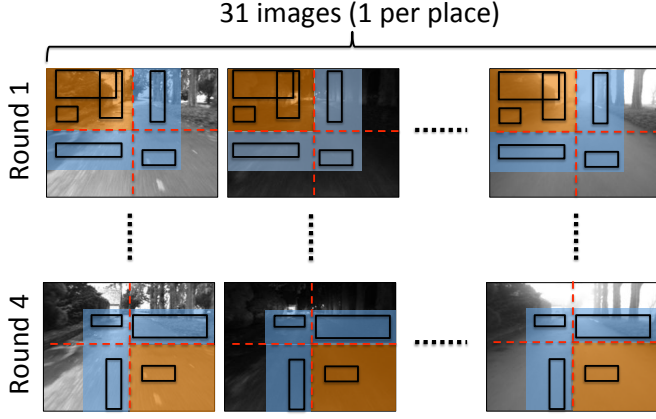


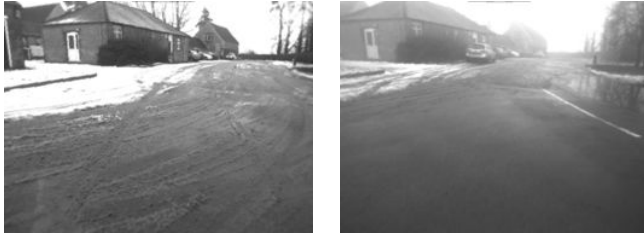
Fig. 4. Our strategy for partitioning the data to produce scene signatures. We take a collection of images at a particular location in the world and partition each image into a number of tiles. In this example, patches (black rectangles) drawn from the light brown regions are placed in the positive set and patches drawn from the blue regions are placed in the negative set. Since we have the same pattern for every image and each image has roughly the same viewpoint, we are able to seed the training algorithm with elements subject to varying appearance changes.



(a) Canonical map image: clear and sunny; noon. (b) Test image: clear and sunny; morning.

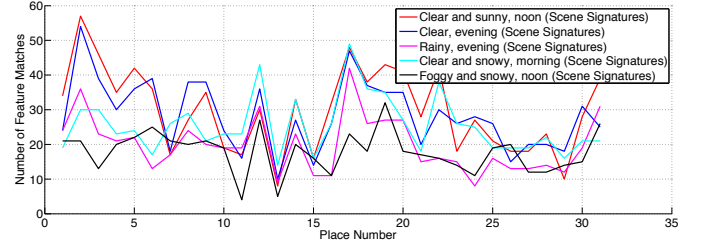


(c) Test image: clear; evening. (d) Test image: rainy; evening.

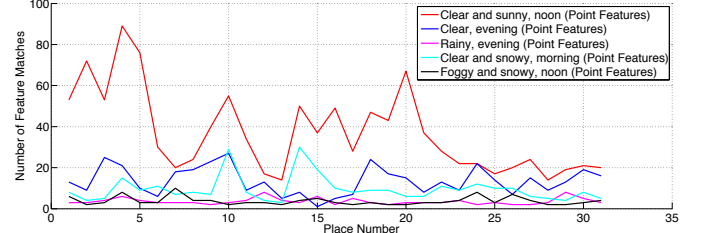


(e) Test image: clear and snowy; morning. (f) Test image: foggy and snowy; noon.

Fig. 5. Example test images used in our localisation experiments. These were chosen due to their large visual variability.



(a) Feature matches using scene signatures.



(b) Feature matches for use point features.

Fig. 7. Feature matches for each place over all 5 runs for both the point-feature system and our scene-signature system. Note how the number of matches using scene signatures stays relatively consistent across all 5 runs while point features struggle greatly for the foggy and rainy runs.

every 20 m along the route according to an INS system and images from all 31 datasets at each place were used to generate spatially indexed feature classifiers, according to the method described in Section III-B. We note however, that places can be defined by other means, either manually or by using place recognition techniques. The only important factor is that the training images for a particular place have roughly the same viewpoint in the area.

Our code was implemented in MATLAB and takes approximately one hour of learning per place. We note that we did not exploit any parallel processing, which would significantly reduce processing times as the training in each image tile can be done independently. As each place is represented by a collection of SVM classifiers, the total storage size per place is $10 \sim 15$ MB.

For our test data, we used 5 separate datasets that included a sunny day run, an evening run, a rainy evening run, a snowy run, and a snowy and foggy run. Examples of some test images are shown in Figure 5. The goal of these experiments is to show that we can use scene signatures and a weak localiser to produce metric estimates regardless of the appearance changes.

To reiterate, our localisation strategy is as follows. After initialising in the map, we use dead reckoning to predict when we are within a couple meters of a place, after which we load the SVM classifiers associated with that place and run them on the live image to detect the scene signatures. Once we have associated these scene signatures, we can perform local, metric, pose estimation. This approach of predicting where the nearest topological node is and then localising against the map is similar to teach-and-repeat systems such as McManus et al. [22] and Furgale and Barfoot [1], except that our map keyframes are separated by larger distances.

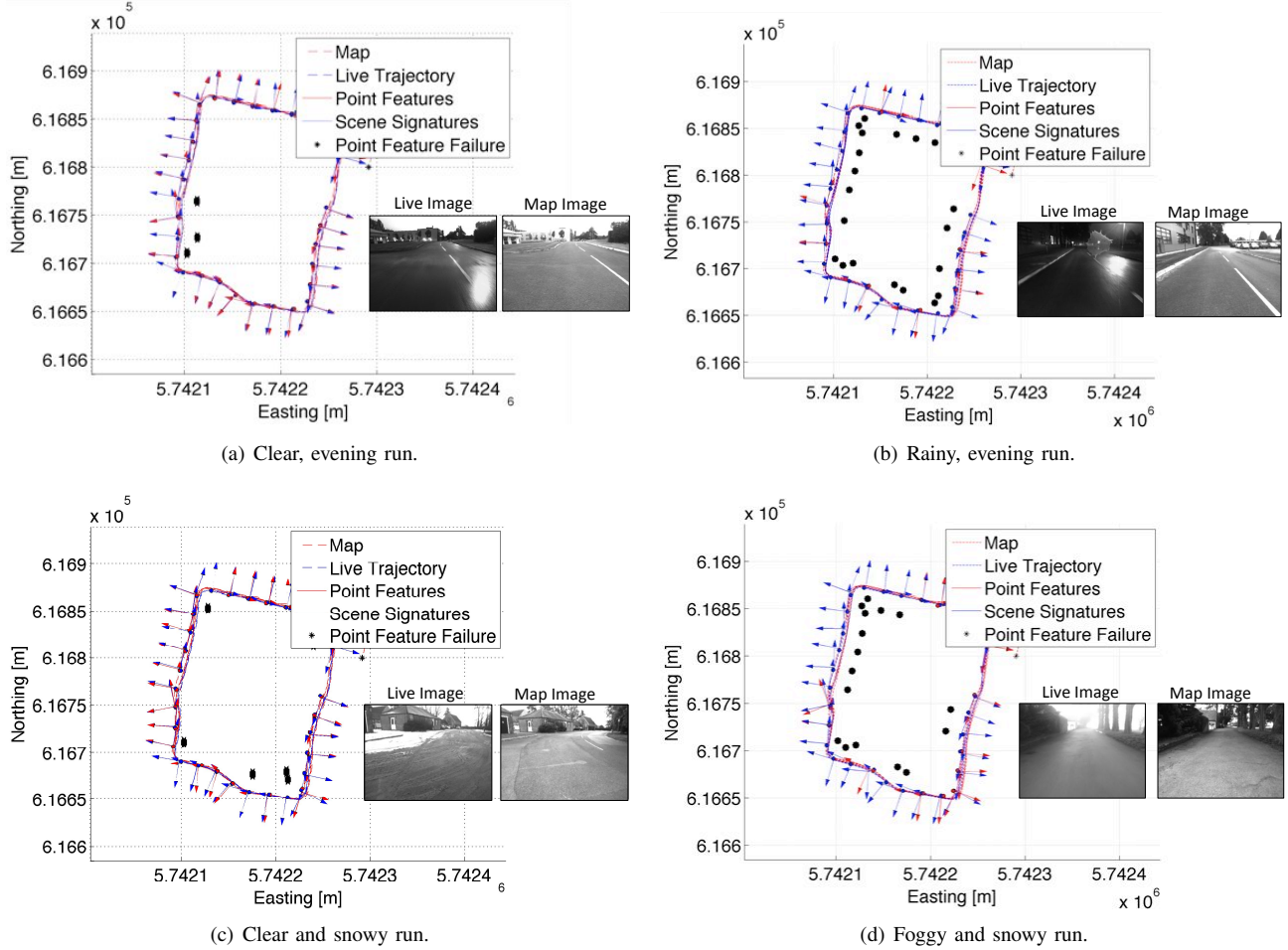


Fig. 6. Localisation results. Note that the arrows represent 2D projections of the vehicle coordinate frames for the point-feature and patch methods. As both the point-feature and our scene-signature method were able to localise all frames for the first dataset (due to the similarity between the live run and the map), we omitted the plots here and turn to the more challenging cases. Our scene-signature approach was able to localise all frames, whereas the point-feature-based system failed on 33% of the places. These failures have been indicated on the plots with a large black circle. Note that almost all estimates agree to the INS ground truth within meters.

Figure 6 presents the localisation results for the 5 live runs against our map that contains a bank of trained classifiers per place. The results show the live and reference INS trajectories as well as the localisation estimates for both our system and the baseline. Defining the ground truth 6×1 pose as $\mathbf{x} := [\mathbf{t}^T, \boldsymbol{\theta}^T]^T$ and the estimated pose as $\hat{\mathbf{x}} := [\hat{\mathbf{t}}^T, \hat{\boldsymbol{\theta}}^T]^T$, localisation failures were declared if $\|\mathbf{t} - \hat{\mathbf{t}}\|_2 > \alpha$ or $\|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}\|_2 > \beta$. In our experiments, we chose $\alpha = 4\text{m}$ and $\beta = 30^\circ$. Note that we were generous in the translational/rotational estimates, but this is because our INS system can experience large drift from datasets collected months apart.

Table I shows the number of frames localised against for each run (according to INS ground truth), where we see that our system was able to localise all frames in all 5 datasets, despite extreme variations in appearance. The point-feature-based system was unable to localise a majority of the frames for the rainy evening run and the foggy snowy run. Figure 7 shows the number of feature matches for each place over all 5 runs. Figure 8 shows examples of our system succeeding

TABLE I
NUMBER OF FRAMES LOCALISED AGAINST OUT OF 31 PLACES.

Live Run	Scene Signatures	Point Features
Clear and sunny morning	31	31
Clear evening	31	28
Rainy evening	31	9
Clear and snowy morning	31	24
Foggy and snowy afternoon	31	12

where the point-feature system failed. Note that we only show a subset of the matches for visual clarity.

V. DISCUSSION AND CONCLUSION

We have demonstrated a new approach to the localisation task, which departs from the traditional point-feature system by learning spatially indexed classifiers of distinctive visual elements called *scene signatures*. Although we are unable to obtain accuracy on the order of centimeters, we are more robust to extreme appearance change and obtain the same type of coverage as a topological localisation system, like

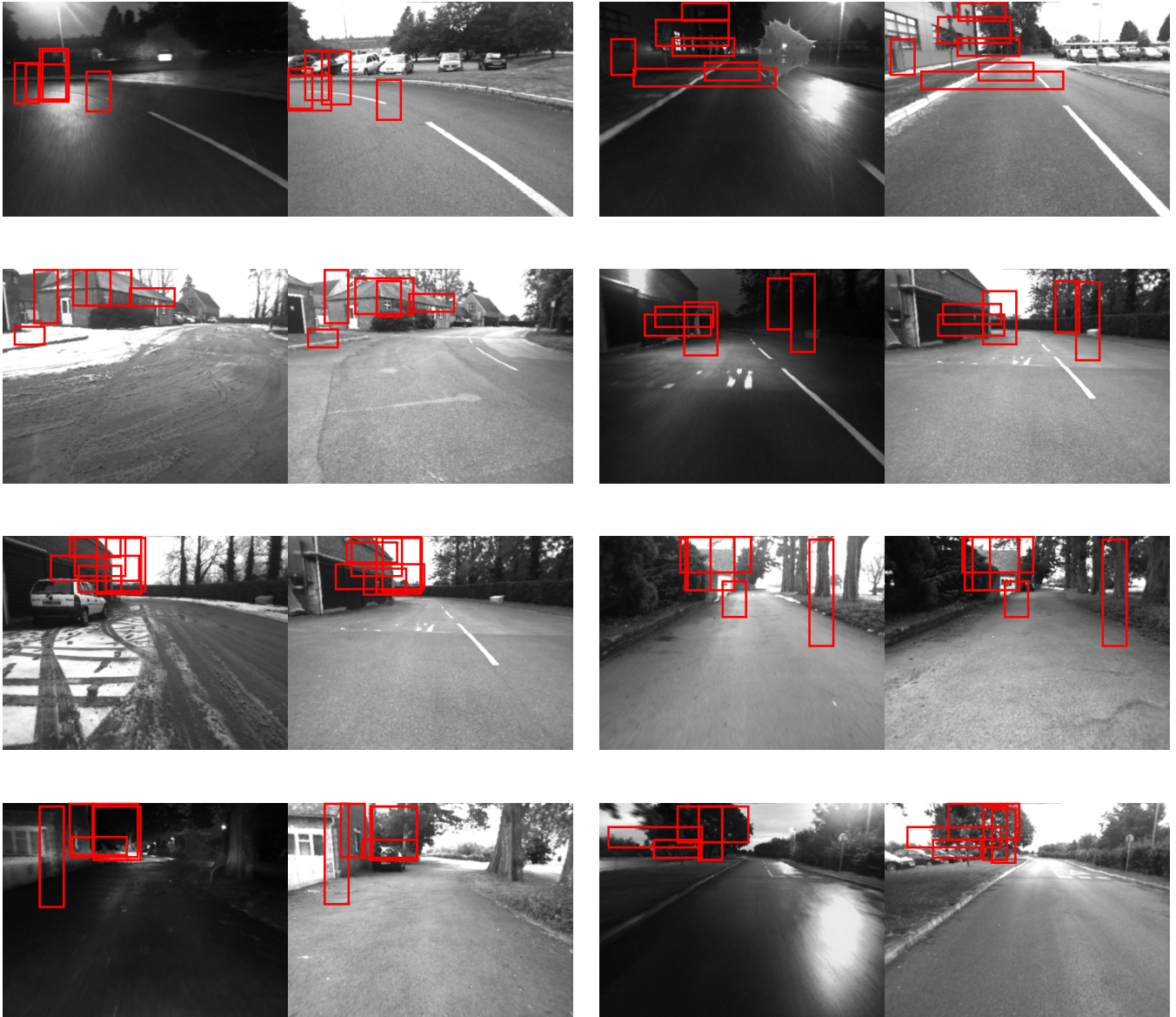


Fig. 8. Examples of where our method was able to successfully localise the live run (left image of each pair) with the map (right image of each pair), while the point-feature method failed. Note that only a subset of the matches are being shown for clarity. On average, we obtained 24 matches per place across all five datasets using scene signatures.

SeqSLAM [3], but with the added benefit of metric pose. Accounting for the uncertainties of each patch and including a prior on the motion of the vehicle is important in constraining the translational estimate. However, because the patches tend to be far away, our orientation estimates are accurate.

Scene signatures enable robust, metric localisation where traditional systems simply fail. Each bank of place-dependant SVM classifiers is run on the live image stream to perform the data association, and a standard frame-to-frame localisation framework is used to obtain the metric pose estimate. We have shown that our approach can successfully localise the vehicle across very challenging lighting and/or weather conditions. We believe that point features alone are simply not enough for

robust, long-term localisation systems and that our approach is a step in the right direction.

REFERENCES

- [1] P. Furgale and T. Barfoot, “Visual teach and repeat for long-range rover autonomy,” *Journal of Field Robotics, special issue on “Visual mapping and navigation outdoors”*, vol. 27, no. 5, pp. 534–560, 2010.
- [2] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, “View-based maps,” *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 941–957, 2010.
- [3] M. Milford and G. Wyeth, “Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, Minnesota, USA, 14–18 May 2012.
- [4] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

- [5] E. Rosten, G. Reitmayr, and T. Drummond, "Real-time video annotations for augmented reality," in *Advances in Visual Computing*, 2005.
- [6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust Wide Baseline Stereo from Maximally Stable Extremal Regions," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2002, pp. 36.1–36.10, doi:10.5244/C.16.36.
- [7] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.
- [8] H. Bay, A. Ess, T. Tuytelaars, and L. Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.
- [9] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.
- [10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.
- [11] A. Davison, I. Reid, N. Motlon, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, 2007.
- [12] A. Davison and D. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, 2002.
- [13] A. Richardson and E. Olson, "Learning convolutional filters for interest point detection," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [14] H. Lategahn, J. Beck, B. Kitt, and C. Stiller, "How to learn an illumination robust image feature for place recognition," in *IEEE Intelligent Vehicles Symposium*, Gold Coast, Australia, 2013.
- [15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006.
- [16] F. von Hundelshausen and R. Sukthankar, "D-Nets: Beyond Patch-Based Image Descriptors," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.
- [17] W. Churchill and P. Newman, "Practice makes perfect? managing and leveraging visual experiences for lifelong navigation," in *Proceedings of the International Conference on Robotics and Automation*, Saint Paul, Minnesota, USA, 14–18 May 2012.
- [18] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros, "What makes paris look like paris?" *ACM Transactions on Graphics*, 2012.
- [19] S. Geman and D. McClure, "Statistical method for tomographic image reconstruction," in *Proceedings of the 46th Session of the International Statistical Institute, Bulletin of the ISI*, vol. 52, 1987, pp. 5–21.
- [20] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *The Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.
- [21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, San Diego, California, USA, 2005, pp. 886–893.
- [22] C. McManus, P. Furgale, B. Stenning, and T. D. Barfoot, "Visual Teach and Repeat Using Appearance-Based Lidar," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

F Dense Reconstruction of Outdoor Scenes from Sparse Range images with Adaptive Model Selection

Authors: Pedro Pinies, Lina Paz and Paul Newman

Link to [Table of Contents](#).

Dense Reconstruction of Outdoor Scenes from Sparse Range images with Adaptive Model Selection

BMVC 2014 Submission # ID 000

Abstract

In this paper we address the problem of dense depth map estimation from sparse noisy range data to reconstruct large heterogeneous outdoor scenes. We propose a surface inpainting solution through energy minimization with an adaptive selection of surface regularisers among a set of well known convex and non-convex norms. In fact, the selection of norm is pivotal with respect to the intrinsic surface characteristics. Our goal is to show how dense interpolation of sparse range data can be leveraged of more exotic and non-convex norms such as the log and logTGV [1] which can better capture the scene geometry. In contrast to state of the art solutions, we do not restrict ourselves to this set of norms, instead we search for the most apt norm for each semantically segmented part of the scene. Our energy model selection use Bayesian optimisation to learn the best choice of free parameters. This results in an adaptive model selection and the generalization of well studied regularisation norms. We conclude with a detailed experimental analysis of our approach using a basis of four norms over ten challenging outdoor scenes.

References

[1] Peter Ochs, Alexey Dosovitskiy, Thomas Brox, and Thomas Pock. An iterated l1 algorithm for non-smooth non-convex optimization in computer vision. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 0:1759–1766, 2013. ISSN 1063-6919. doi: <http://doi.ieeecomputersociety.org/10.1109/CVPR.2013.230>.

G Unified Temporal and Spatial Calibration for Multi-Sensor Systems

Authors: Furgale, Rehder, Siegwart

Link to [Table of Contents](#).

Unified Temporal and Spatial Calibration for Multi-Sensor Systems

Paul Furgale, Joern Rehder, and Roland Siegwart

Abstract—In order to increase accuracy and robustness in state estimation for robotics, a growing number of applications rely on data from multiple complementary sensors. For the best performance in sensor fusion, these different sensors must be spatially and temporally registered with respect to each other. To this end, a number of approaches have been developed to estimate these system parameters in a two stage process, first estimating the time offset and subsequently solving for the spatial transformation between sensors.

In this work, we present on a novel framework for jointly estimating the temporal offset between measurements of different sensors and their spatial displacements with respect to each other. The approach is enabled by continuous-time batch estimation and extends previous work by seamlessly incorporating time offsets within the rigorous theoretical framework of maximum likelihood estimation.

Experimental results for a camera to inertial measurement unit (IMU) calibration prove the ability of this framework to accurately estimate time offsets up to a fraction of the smallest measurement period.

I. INTRODUCTION

Most methods for state estimation that fuse data from multiple sensors assume and require that the timestamps of all measurements are accurately known with respect to a single clock. Consequently, the time synchronization of sensors is a crucial aspect of building a robotic system.

The most desirable method of determining the timestamps of all measurements is with the support of hardware, using interrupts on a single processor to detect signal lines that change state at each measurement time. This is the method used for high-accuracy photogrammetry [1] but it requires specialized hardware and increases the complexity of integration.

When hardware support is not available, the next best option is to use software-supported time synchronization. For example, the mapping between clocks can be learned using the Network Time Protocol (NTP) [2] or the TICSyn algorithm [3]. This mapping then allows one to resolve the device timestamps with respect to a common clock. However, these methods require software support on each device and very few (if any) off-the-shelf sensors provide this.

Our approach is targeted toward the majority of robotic systems, where individual sensors do not support the hardware or software synchronization methods described above. In such systems, a central unit either triggers or polls sensor readings or receives a continuous stream of fixed-rate measurements from a sensor. Using a single clock, the central unit timestamps this data either on arrival or when initiating a

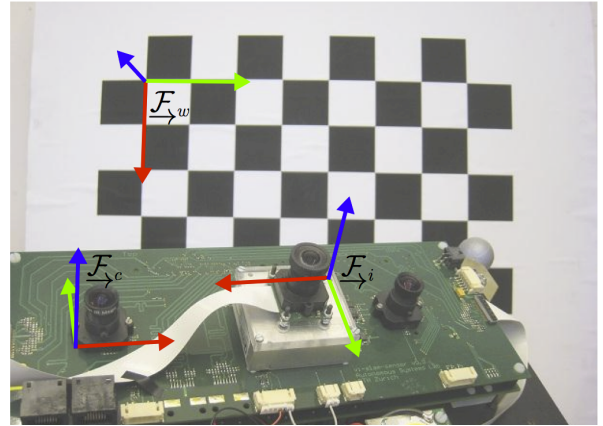


Fig. 1. This paper derives a unified framework for temporal and spatial calibration of multi-sensor systems such as this visual-inertial sensor used in the experiments. $\mathcal{F}_w$ marks the inertial frame attached to a static calibration pattern, while $\mathcal{F}_c$ and $\mathcal{F}_i$ show the camera and the IMU frame respectively. By waving the sensor in front of the calibration pattern, our framework is capable of estimating the 6-degrees-of-freedom transformation between $\mathcal{F}_c$ and $\mathcal{F}_i$ as well as time offset between the two devices.

request. In this case, the delay between sensor measurements is determined by communication delays and the internal sensor delays—introduced either by filters or logic—as well as stochastic delays resulting from task scheduling. After removing stochastic effects on these fixed rate measurements using the approach of Moon et al. [4], it is sometimes possible to infer the delay from sensor data. This delay is constant and can therefore be determined in an offline calibration procedure.

Conventional discrete-time estimation techniques generally require a state at each measurement time. This makes the estimation of temporal offsets difficult as measurement times shift when the offsets are updated. This has led to the development of specialized algorithms just for estimating time offsets [5], [6], that are applied prior to spatial calibration of the sensors.

In contrast, the continuous-time batch estimation algorithm proposed by Furgale et al. [7] makes it easy to fold time offsets directly into a principled maximum-likelihood estimator. Although we agree with the authors of [6] that jointly estimating uncorrelated quantities may potentially impair results, we believe that, given accurate measurement models, the optimality implications of maximum likelihood estimation extend to our approach. Consequently, we can achieve the highest accuracy by incorporating *all* available information into a unified estimate.

The results presented in section V support this assumption, suggesting that the gain in accuracy from information provided by additional sensors outweighs the drawbacks of

The authors are from the Autonomous Systems Lab at ETH Zurich, Switzerland. {paul.furgale, joern.rehder} at mavt.ethz.ch, rsiegwart at ethz.ch

potential interferences of uncorrelated parameters.

The contributions of this paper are as follows:

- 1) we propose a unified method of determining fixed time offsets between sensors using batch, continuous-time, maximum-likelihood estimation;
- 2) we derive an estimator for the calibration of a camera and inertial measurement unit (IMU) that simultaneously determines the transformation *and* the temporal offset between the camera and IMU;
- 3) we evaluate the estimator on simulated and real data (from the setup depicted in Figure 1) and show that it is sensitive enough to determine temporal offsets up to a fraction of the period of the highest-rate sensor, including differences due to camera exposure time; and
- 4) we demonstrate that the time delay estimation significantly benefits from the additional information comprised in acceleration measurements—information that was not exploited in previous approaches ([5], [6]).

II. RELATED WORK

Early efforts in determining the spatial relationship between an IMU and a camera were made by Alves et al. [8]. By mounting a visual-inertial sensor onto a pendulum equipped with a high-resolution encoder, the authors of this study were able to estimate the relative rotation between camera and IMU as well as scale factors and axis misalignments in the IMU. In [9], orientation estimation was facilitated using a set of still images and a gravity aligned visual calibration pattern. Furthermore, the study extended previous work to further calibrate for the relative displacement of camera and IMU in a separate procedure which involved mounting the device on a turntable and aligning it in a way that the IMU is not affected by accelerations other than gravity.

More recently, recursive approaches were employed to jointly estimate relative rotation and translation from measurements acquired by dynamically moving a camera IMU combination in front of a calibration pattern [10], [11]. Both studies further address the question of the observability of the calibration, deriving that it can be determined when sufficient rotational velocity is present in the dataset. Other approaches estimate the transformation in a batch optimization over a set of inertial measurements and calibration pattern observations [12], [13]. Among those, our algorithm is most similar to [13] in that it also employs B-splines to parameterize the motion of the device.

All these studies have in common that they assume that IMU and camera have been temporally calibrated in a separate, preceding step and that any offset in the timing of their measurements has been compensated for. In [5], a variation of the iterative closest point algorithm (ICP) was used to determine a fixed time offset by aligning orientation curves sensed by the camera and gyroscopes individually. However, the algorithm makes simplifying assumptions about the bias in inertial measurements and it remains unclear how this approach extends to other sensors than the ones presented on—especially as it does not make any use of the acceleration

measurements provided by the IMU. In [6], an estimate of the time offset is established either by temporally aligning the frame independent absolute rotational velocity or by determining the phase shift of common frequencies in frequency domain, thus avoiding the joint estimation of the time offset and the relative orientation of the sensors with respect to each other. Obviously, relative orientation and time offset are uncorrelated quantities, and by separating their estimation, the approach neatly avoids cases in which inaccuracies in one parameter affect the estimate of the other. However, the work demonstrates the separation only for gyroscope measurements, and it is not immediately clear how to achieve separation for other types of sensors. Like [5], this approach disregards the information comprised in the accelerometer readings. In contrast, our work suggests that the temporal calibration benefits more from additional measurements than from the rigorous separation of uncorrelated parameters in the estimation.

III. THEORY

In this section we consider the problem of determining the relative time offset between a pair of sensors. It is straightforward to extend the results to multiple sensors.

A. Estimating Time Offsets using Basis Functions

Throughout this section, we follow the basis function approach for batch continuous-time state estimation presented in Furgale et al. [7]. Time-varying states are represented as the weighted sum of a finite number of known analytical basis functions. For example, a D -dimensional state, $\mathbf{x}(t)$, may be written as

$$\Phi(t) := [\phi_1(t) \ \dots \ \phi_B(t)], \quad \mathbf{x}(t) := \Phi(t)\mathbf{c}, \quad (1)$$

where each $\phi_b(t)$ is a known $D \times 1$ analytical function of time and $\Phi(t)$ is a $D \times B$ stacked basis matrix. To estimate $\mathbf{x}(t)$, we simply estimate $\mathbf{c}$, the $B \times 1$ column of coefficients.

When estimating time offsets from measurement data, we will encounter error terms such as

$$\mathbf{e}_j := \mathbf{y}_j - \mathbf{h}(\mathbf{x}(t_j + d)), \quad (2)$$

where $\mathbf{y}_j$ is a measurement that arrived with timestamp t_j , $\mathbf{h}(\cdot)$ is a measurement model that produces a predicted measurement from $\mathbf{x}(\cdot)$, and d is the unknown time offset. Using basis functions, this becomes

$$\mathbf{e}_j = \mathbf{y}_j - \mathbf{h}(\Phi(t_j + d)\mathbf{c}), \quad (3)$$

which is easy to evaluate for different values of d as it changes during optimization. The analytical Jacobian of the error term, needed for nonlinear least squares estimation, is derived by linearizing (3) about a nominal value, $\bar{d}$, with respect to small changes, Δd . This results in the expression

$$\mathbf{e}_j \approx \mathbf{y}_j - \mathbf{h}(\Phi(t_j + \bar{d})\mathbf{c}) - \mathbf{H}\dot{\Phi}(t_j + \bar{d})\mathbf{c}\Delta d, \quad (4)$$

where the over dot represents a time derivative and

$$\mathbf{H} := \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\mathbf{x}(\Phi(t_j + \bar{d})\mathbf{c})}. \quad (5)$$

In (4), $\Phi(t)$ is a known function and we assume its time derivative, $\dot{\Phi}(t)$, is available analytically.

This approach has two clear benefits. Firstly, it allows us to treat the problem of estimating time offsets within the rigorous theoretical framework of maximum likelihood estimation. Secondly, it allows us to leave the problem in continuous time so that the delayed measurement equations and their Jacobians can be evaluated analytically.

In short, estimating the time offsets in a principled way becomes easy.

B. An Example: Camera/IMU Calibration

Rather than delving further into the general case, we will proceed with the specific example of calibrating a camera and IMU. The goal of calibration is to determine the relative rotation, translation, and time offset between the sensors.

To perform calibration, we collect a set of data over a short time interval, $T = [t_1, t_K]$ (typically 1–2 minutes), as the sensor head is waved in front of a static calibration pattern. Figure 1 shows the basic problem setup. Estimation is performed with respect to an inertial world coordinate frame, $\mathcal{F}_w$. The linear acceleration and angular velocity are measured in the IMU frame, $\mathcal{F}_i(t)$. The camera coordinate frame, $\mathcal{F}_c(t)$, is placed at the camera's optical center with the z -axis pointing down the optical axis.

1) *Quantities Estimated:* Our algorithm estimates time-invariant parameters for (i) the gravity direction, $\mathbf{g}_w$, expressed in $\mathcal{F}_w$, (ii) the transformation between the camera and the IMU, $\mathbf{T}_{c,i}$, and (iii) the offset between camera time and IMU time, d . It also estimates time-varying states: (iv) the pose of the IMU, $\mathbf{T}_{w,i}(t)$, and (v) the accelerometer ($\mathbf{b}_a(t)$) and gyroscope ($\mathbf{b}_\omega(t)$) biases.

2) *Parameterization of Time-Varying States:* Time-varying states are represented by B-spline functions. B-splines produce simple analytical functions of time with good representational power. Please see [14] for a thorough introduction.

The IMU pose is parameterized as a 6×1 spline, using three degrees of freedom for orientation and three for translation. The transformation that takes points from the IMU frame to the world frame at any time t can be built as

$$\mathbf{T}_{w,i}(t) := \begin{bmatrix} \mathbf{C}(\varphi(t)) & \mathbf{t}(t) \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (6)$$

where $\varphi(t) := \Phi_\varphi(t)\mathbf{c}_\varphi$ encodes the orientation parameters, $\mathbf{C}(\cdot)$ is a function that builds a rotation matrix from our parameters, and $\mathbf{t}(t) := \Phi_t(t)\mathbf{c}_t$ encodes the translation. The velocity, $\mathbf{v}(t)$, and acceleration, $\mathbf{a}(t)$, of the platform with respect to and expressed in the world frame are

$$\mathbf{v}(t) = \dot{\mathbf{t}}(t) = \dot{\Phi}_t(t)\mathbf{c}_t, \quad \mathbf{a}(t) = \ddot{\mathbf{t}}(t) = \ddot{\Phi}_t(t)\mathbf{c}_t. \quad (7)$$

For a given rotation parameterization, the relationship to angular velocity is of the form

$$\omega(t) = \mathbf{S}(\varphi(t))\dot{\varphi}(t) = \mathbf{S}(\Phi(t)\mathbf{c}_\varphi)\dot{\Phi}(t)\mathbf{c}_\varphi, \quad (8)$$

where $\mathbf{S}(\cdot)$ is the standard matrix relating parameter rates to angular velocity [15]. In this paper we used the axis/angle parameterization of rotation where $\varphi(t)$ represents rotation by the angle $\varphi(t) = \sqrt{\varphi(t)^T \varphi(t)}$ about the axis $\varphi(t)/\varphi(t)$.

3) *Measurement and Process Models:* Each accelerometer measurement, α_k , and gyroscope measurement, ϖ_k , is sampled at time t_k , where $k = 1 \dots K$. The pixel location of landmark, $\mathbf{p}_w^m$, seen at time $t_j + d$ is denoted $\mathbf{y}_{mj}$, where t_j is the image timestamp, d is the unknown time offset, and $j = 1 \dots J$ indexes the images. We use the standard, discrete-time IMU and camera measurement equations,

$$\alpha_k := \mathbf{C}(\varphi(t_k))^T (\mathbf{a}(t_k) - \mathbf{g}_w) + \mathbf{b}_a(t_k) + \mathbf{n}_{\alpha_k}, \quad (9a)$$

$$\varpi_k := \mathbf{C}(\varphi(t_k))^T \omega(t_k) + \mathbf{b}_\omega(t_k) + \mathbf{n}_{\omega_k}, \quad (9b)$$

$$\mathbf{y}_{mj} := \mathbf{h}(\mathbf{T}_{c,i}\mathbf{T}_{w,i}(t_j + d)^{-1}\mathbf{p}_w^m) + \mathbf{n}_{y_{mj}}, \quad (9c)$$

where each $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is assumed to be statistically independent of the others, $\mathbf{h}(\cdot)$ may be any nonlinear camera model, and there are M landmarks, $\{\mathbf{p}_w^m | m = 1 \dots M\}$. We have written these error terms as if the image measurements are delayed with respect to the IMU measurements. This assumption is only for convenience as it is easier to write out and implement delayed image error terms. Nothing is lost as d can be negative.

We model the IMU biases as driven by zero-mean white Gaussian processes:

$$\dot{\mathbf{b}}_a(t) = \mathbf{w}_a(t) \quad \mathbf{w}_a(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_a \delta(t - t')) \quad (10a)$$

$$\dot{\mathbf{b}}_\omega(t) = \mathbf{w}_\omega(t) \quad \mathbf{w}_\omega(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_\omega \delta(t - t')) \quad (10b)$$

We assume the processes are statistically independent so that $E[\mathbf{w}_a(t)\mathbf{w}_\omega(t')^T] = \mathbf{0}$ for all t, t' , where $E[\cdot]$ is the expectation operator.

4) *The Estimator:* We estimate the five quantities defined in Section III-B.1. Error terms associated with the measurements are constructed as the difference between the measurement and the predicted measurement given the current state estimate. The continuous-time process models for the IMU biases give rise to integral error terms (refer to [7] for more details). Altogether, our objective function is built from the following components:

$$\mathbf{e}_{y_{mj}} := \mathbf{y}_{mj} - \mathbf{h}(\mathbf{T}_{c,i}\mathbf{T}_{w,i}(t_j + d)^{-1}\mathbf{p}_w^m) \quad (11a)$$

$$J_y := \frac{1}{2} \sum_{j=1}^J \sum_{m=1}^M \mathbf{e}_{y_{mj}}^T \mathbf{R}_{y_{mj}}^{-1} \mathbf{e}_{y_{mj}} \quad (11b)$$

$$\mathbf{e}_{\alpha_k} := \alpha_k - \mathbf{C}(\varphi(t_k))^T (\mathbf{a}(t_k) - \mathbf{g}_w) + \mathbf{b}_a(t_k) \quad (11c)$$

$$J_\alpha := \frac{1}{2} \sum_{k=1}^K \mathbf{e}_{\alpha_k}^T \mathbf{R}_{\alpha_k}^{-1} \mathbf{e}_{\alpha_k} \quad (11d)$$

$$\mathbf{e}_{\omega_k} := \varpi_k - \mathbf{C}(\varphi(t_k))^T \omega(t_k) + \mathbf{b}_\omega(t_k) \quad (11e)$$

$$J_\omega := \frac{1}{2} \sum_{k=1}^K \mathbf{e}_{\omega_k}^T \mathbf{R}_{\omega_k}^{-1} \mathbf{e}_{\omega_k} \quad (11f)$$

$$\mathbf{e}_{b_a}(t) := \dot{\mathbf{b}}_a(t) \quad (11g)$$

$$J_{b_a} := \frac{1}{2} \int_{t_1}^{t_K} \mathbf{e}_{b_a}(\tau)^T \mathbf{Q}_a^{-1} \mathbf{e}_{b_a}(\tau) d\tau \quad (11h)$$

$$\mathbf{e}_{b_\omega}(t) := \dot{\mathbf{b}}_\omega(t) \quad (11i)$$

$$J_{b_\omega} := \frac{1}{2} \int_{t_1}^{t_K} \mathbf{e}_{b_\omega}(\tau)^T \mathbf{Q}_\omega^{-1} \mathbf{e}_{b_\omega}(\tau) d\tau \quad (11j)$$

The Levenberg-Marquardt (LM) algorithm [16] is used to minimize $J := J_y + J_\alpha + J_\omega + J_{b_a} + J_{b_w}$ to find the maximum likelihood estimate of all unknown parameters at once.

IV. IMPLEMENTATION

This section describes the implementation details of the estimator. We make the following assumptions:

- the camera intrinsic calibration is known;
- the IMU noise and bias models are known;
- we have a guess for the gravity in $\mathcal{F}_w$;
- we have a guess for the calibration matrix, $\mathbf{T}_{c,i}$;
- the geometry of the calibration pattern is known so that we can express the position of each point landmark, $\mathbf{p}_w^m$, in the world coordinate frame, $\mathcal{F}_w$; and
- we know the data association between an image point, $\mathbf{y}_{mj}$, and the corresponding point on the calibration pattern, $\mathbf{p}_w^m$.

We set the initial guess for the time offset to zero. For the position of the IMU, $\mathbf{T}_{w,i}(t)$, we produce an initial guess by first computing a rough estimate of the camera position, $\mathbf{T}_{w,c}(t_j)$, for each image using the perspective n-point algorithm from the Bouget camera calibration toolbox¹. This is transformed into an initial guess for the position of the IMU at this time, $\mathbf{T}_{w,i}(t_j) = \mathbf{T}_{w,c}(t_j)\mathbf{T}_{c,i}$. Finally, the pose spline is initialized using the linear solution of Schoenberg and Reinsch (Chapter XIV of [17]).

The IMU pose is encoded as a sixth-order B-spline (a piecewise fifth-degree polynomial). This high-order representation encodes acceleration as a cubic polynomial. We found this was necessary to accurately capture the dynamic motion of the sensor head during calibration. The biases are represented by cubic B-splines. Note that not only the order of the splines, but also the number of knots has to reflect the systems dynamics, requiring a greater number of knots for faster varying quantities. The algorithm allows for regularization terms, which are modeled as random walk processes (see [7]) that constrain the temporal evolution of estimates over periods with insufficiently constraining measurements. These terms correspond to physical constraints imposed onto the motion of the sensor system by its finite inertia and the limiting dynamics of the system actuating the sensors.

Due to the order of the B-spline used to represent the state and depending on the number of knots used, the system of equations that must be solved at each iteration of LM can be very large. However, the matrix is sparsely populated, primarily due to the compact support of B-spline basis functions. A sixth-order B-spline basis function is nonzero over exactly six intervals. The result is that the primary diagonal of the LM information matrix is block six diagonal (scalar $6 \times 6 = 24$) in the section associated with the pose spline, $\mathbf{T}_{w,i}(t)$. An example matrix for 0.1 seconds of data is shown in Figure 2. The CHOLMOD sparse matrix library is used to solve the resulting system at each iteration [18].

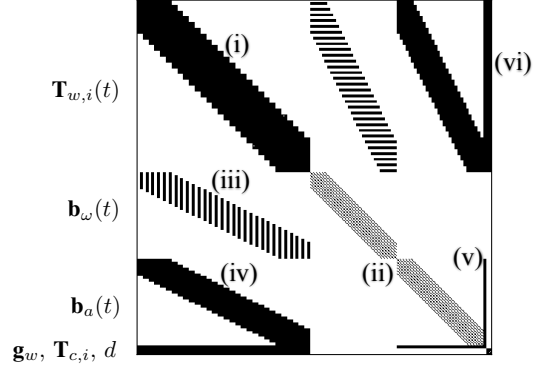


Fig. 2. A visualization of the sparsity pattern of the symmetric information matrix built during each iteration of Levenberg-Marquardt. To enhance the visibility of the matrix structure, only 0.1 seconds of data was used to generate this plot. The time-varying pose spline parameters produce a block diagonal (i) corresponding to $\mathbf{T}_{w,i}(t)$. The width of this diagonal depends on the spline order as well as the temporal padding. The diagonal blocks associated with the bias splines (ii) contain information from the bias motion model, (10). The parameters of the gyro bias spline, $\mathbf{b}_\omega(t)$, show striped correlation with the pose spline (iii) because they are only coupled with the IMU orientation through (11e). The parameters of the accelerometer bias, $\mathbf{b}_a(t)$ are correlated with orientation (iv), position (iv), and gravity (v) through (11c). Finally, $\mathbf{g}_w$, $\mathbf{T}_{c,i}$ and the time offset, d , are all correlated with the $\mathbf{T}_{w,i}(t)$ (vi) through (11a).

The optimization problem seems well-posed. Figure 3 shows the cost function evaluated in the neighborhood of the minimum on 40 real datasets from Section V-B. The figure clearly shows that the cost function in the neighborhood of the minimum is convex and steep with respect to changes in d .

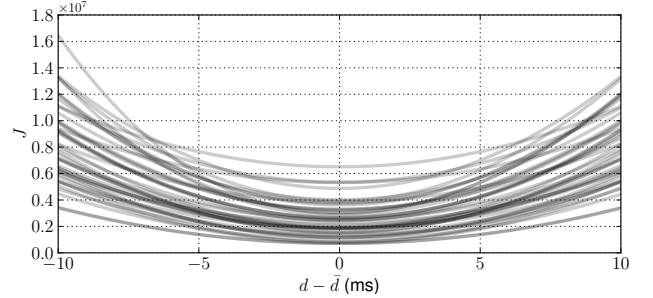


Fig. 3. The cost function evaluated for different values of the time offset, d , in the neighborhood of the minimum, $\bar{d}$, on 40 real datasets from Section V-B. In this neighborhood, the cost function is convex.

During optimization, error terms associated with the delayed image measurements may move across knot boundaries of the spline. This can cause the sparsity pattern of the linear system to change each iteration. It is significantly less computationally expensive to solve each iteration when the sparsity pattern does not change as we can cache a symbolic matrix factorization and preallocate our sparse matrix data structures. Consequently, the Jacobians of each image error term, (11a), are extended by additional columns to account for possible dependence on the parameters of neighboring b-spline coefficients. To this end, the code accepts a *temporal padding* value in seconds that limits the extent of this neighborhood and hence constrains the time delay to lie within the boundaries of its magnitude. Note that choosing this value to be too large results in increased processing time

¹Available at http://www.vision.caltech.edu/bouguetj/calib_doc/

as the number of potentially non-zero entries increases, while choosing it too small results in a failure of the optimization routine. For the experiments in this paper, we used a time padding of 0.04 seconds.

V. EXPERIMENTS

In this section, we will briefly demonstrate the accuracy and stability of our calibration framework by presenting results from 500 simulated and 40 real calibration datasets. For both simulated and real data, we fixed the inertial measurement rate to 200 Hz and the camera frame rate to 20 Hz. We determined the internal noise parameters of our IMU using Allan Variance analysis [19], [20]. The camera was calibrated intrinsically using the equidistant model and it was assumed that the landmark measurements seen in the images were subject to isotropic, zero-mean Gaussian distributed noise with a standard deviation of 0.5 pixels. The simulations were generated with the same sensor models as the real data.

In order to represent the pose and bias curves, 50 basis functions per second were used. This number was found to be necessary to enable the pose spline to accurately capture the dynamics of the platform. It remains future work to automatically select the minimal number of basis functions for a given motion.

The resulting sparse linear system of equations that must be solved at each iteration of LM is very large. A dataset of approximately 80 seconds corresponds to a system of over 12,400 vector-valued design variables and 144,000 error terms. The sparse matrix associated with the linear system is about $50,000 \times 50,000$ with 3,550,000 non-zero entries. Each of these iterations requires approximately 18 seconds to build the linear system and 0.2 seconds to solve it using CHOLMOD on a MacBook Pro with a 2.4 GHz Intel Core i7 and 8GB of RAM. Hence, the limiting factor is the number of error terms (which determines the amount of time it takes to build the linear system) and not the number of basis functions used (which determines how long it takes to solve the linear system). For the processed datasets, the algorithm converged to a solution in 3 to 15 iterations, which translates into a maximum time of 5 minutes for a single dataset.

A. Simulation

For the simulation, we used sums of sinusoidal functions of time for position and orientation in order to create a sensor trajectory of about 90 seconds. The average absolute angular velocity of 37° s^{-1} and average absolute acceleration of about 0.59 m/s^2 in the simulation roughly resembled values found for real calibration sequences. For each of five time offsets, spaced equally between -8ms and 8ms , we generated 100 realizations of this experiment by corrupting perfect inertial sensor measurements with random noise and an additive bias modelled as a random walk process. We simulated a camera rotated by 180° about the optical axis and displaced by $\mathbf{t} = [103 \ -15 \ -10]^T \text{ mm}$ with respect to the IMU. The initial guess for the relative orientation was accurate up to a few of degrees, while an initial displacement

of $\mathbf{t}_{\text{init}} = [0 \ 0 \ 0]^T \text{ mm}$ was provided. Figure 4 depicts a histogram of errors in time offset estimation overlaid with the marginal uncertainty returned by the estimator and plotted as a Gaussian probability density function (PDF). The plot shows that, given the correct noise models, the approach is capable of accurately estimating the time offset and returning a reasonable uncertainty of the estimate. The displacement of camera and IMU was estimated as $\bar{\mathbf{t}}_{\text{est}} = [103.73 \ -15.18 \ -9.98]^T \text{ mm}$ with standard deviations of $\sigma_t = [0.38 \ 0.98 \ 0.17]^T$. Yaw, pitch and roll were estimated as $\bar{\varphi}_{\text{est}} = [179.9999^\circ \ -0.0098^\circ \ 0.0003^\circ]^T$ with $\sigma_\varphi = [0.0032^\circ \ 0.0086^\circ \ 0.0072^\circ]^T$.

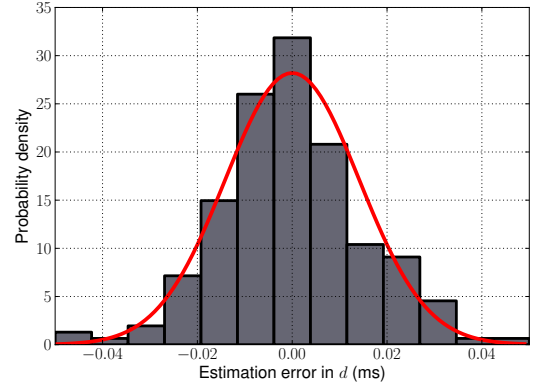


Fig. 4. A histogram of the error in the estimated time offset over 500 simulation trials with the time offset varying between -8ms and 8ms . The marginal uncertainty returned by the estimator is plotted as a Gaussian probability density function (solid red). The results clearly show that, if the correct noise models are known, this method is able to estimate the time offset between the two devices and return a reasonable uncertainty of the estimate.

B. Real Data

For this experiment, we used a custom-made sensor, consisting of multiple Aptina MT9V034 global shutter image sensors and a tactical grade IMU (Analog Devices ADIS16488). We routed all sensor data streams through an FPGA, recording the timestamps at the moment the image sensors were triggered and an IMU data request was initialized. Note that while using an FPGA for data acquisition helps with avoiding stochastic delays introduced by rescheduling tasks on a processor, it does not tackle the issue of logic and filter delays inside the sensors, and in our setup does not account for communication delays introduced when polling measurements.

For each of four fixed exposure times, we collected ten datasets by moving the sensor in front of the calibration pattern for about 90 seconds per dataset. To render all quantities of the calibration observable, we ensured sufficient rotational velocity was present in the motion [10], maintaining an average absolute angular velocity of about 55° s^{-1} and an average absolute acceleration of 1.1 m/s^2 over all datasets.

Figure 5 depicts the key results for the temporal calibration as a comparison between estimated time offsets and fixed exposure times. The middle of the exposure time constitutes the ideal point to timestamp an image [1] and Figure 6

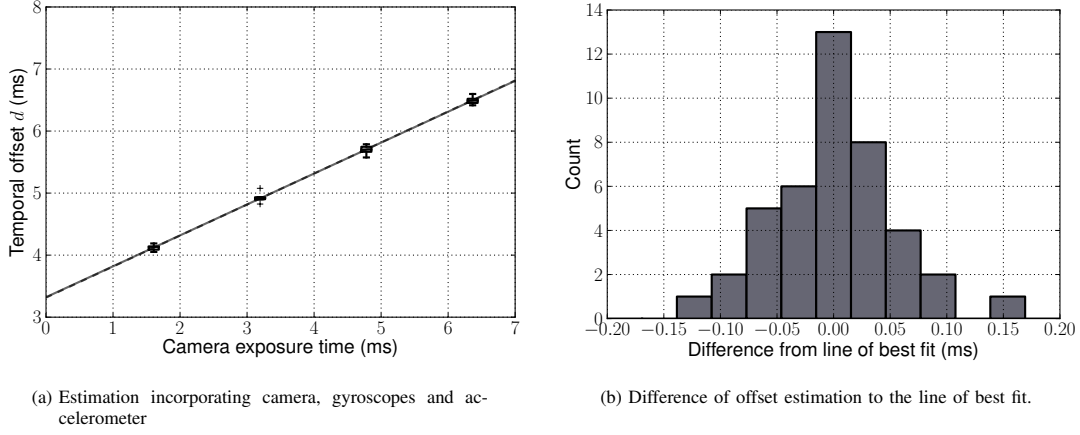


Fig. 5. The key results from our experiments. Figure 5(a) depicts the time offset estimated for four different, fixed exposure times and ten datasets per exposure setting. The estimation made use of all inertial sensors available in the IMU. The slope of the line of best fit (drawn as a solid gray line) is estimated as 0.498, which compares well with its theoretical value of 0.5 (marked by the dashed gray line). Figure 5(b) shows a histogram of the difference between the estimates and the line of best fit for all 40 experiments. For all datasets, the difference stays within a domain of ± 0.2 ms, which constitutes just 4% of the shortest measurement period in the experiment.

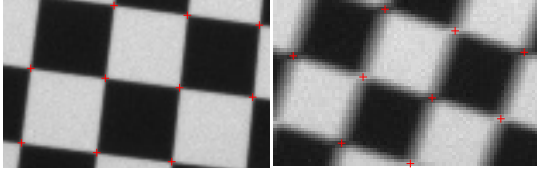


Fig. 6. Images of the checkerboard may be blurred due to the motion of the camera. This figure shows details from two images taken from one of the datasets. The corner finding algorithm used in this paper performs well for images taken with a static camera (left) as well as under motion blur (right), returning the location of the corner near the middle of the exposure time for the vast majority of motions.

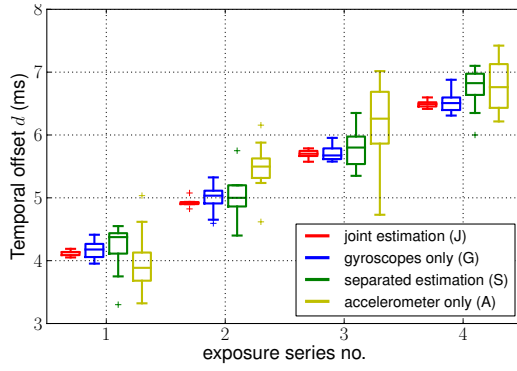
illustrates this; each corner point extracted from an image in the presence of motion blur resembles the position of the projection of the corresponding world point in the middle of the exposure time. In this experiment, we expected the time offset to account for fixed communication and filter delays *plus half the exposure time* as the images are timestamped at the start of the exposure time. Hence, we expect that a plot of temporal offset versus exposure time should show a linear relationship with slope 0.5. Note that by extrapolating the line for an exposure time of zero, one may estimate the communication and filter delay for the IMU. However, this only holds true under the assumption that the trigger pulse immediately initiates exposure. In case the internal logic of the image sensor adds additional delays to the trigger pulse, only the difference in delays can be observed. As the true delays are unavailable for our experiments, we use the deviation in slope of the line of best fit from the theoretical value as well as the RMS error with respect to a line of slope 0.5, fitted in a least squares sense to evaluate the results.

Figure 5(a) shows that our framework is capable of reproducing the inter-sensor time delay up to high accuracy, estimating a slope of 0.498. Figure 5(b) shows the differences of the estimates to the line of best fit, which are all below 0.2 ms. This suggests that the method is accurate to a fraction of the IMU sampling period of 5 ms. The spatial calibration between camera and IMU was determined to be $\mathbf{t}_{\text{est}} = [74.5374 \quad -8.6751 \quad 12.3919]^T$ mm with standard

deviations of $\sigma_t = [1.6081 \quad 0.9051 \quad 0.7609]^T$ for displacement and $\varphi_{\text{est}} = [180.7531^\circ \quad 0.1784^\circ \quad -0.1648^\circ]^T$ with $\sigma_\varphi = [0.0206^\circ \quad 0.0599^\circ \quad 0.0417^\circ]^T$ for yaw, pitch and roll.

Figure 7 visualizes a comparison of our method, using all available sensors in camera IMU calibration as well as only subsets with a reference implementation of the separated calibration based on the frame independent absolute angular velocity as proposed in [6]. The estimated slopes are 0.498 for the joint calibration using all sensors, 0.531 for the separate estimation of the time delay, and 0.493 and 0.553 for the approach using only angular velocities or accelerations in addition to images. The RMS errors are 0.054ms, 0.344ms, 0.165ms and 0.572ms respectively. The results suggest that incorporating measurements from all available sensors into a continuous-time batch optimization yields significantly more accurate and consistent results compared to calibrations that only make use of a subset of the measurements at hand. In our experiments, the gain from the additional information comprised in the accelerometer readings appears to outweigh possible drawbacks of jointly estimating parameters that could be separated otherwise.

Note that the temporal-spatial calibration solely based on camera and accelerometer readings, though less accurate, marks an extension in camera IMU calibration to a previously not considered combination of sensors. In this configuration and with constrained dynamics of the sensor, the spatial transformation between camera and IMU is fully observable. Due to the lack of frequent feedback about the sensor's orientation or the change of it respectively, a regularizing term on the orientation was introduced. Using a single dataset, we empirically identified the parameters of the random walk model employed in our approach and applied these settings to the evaluation of all datasets. The fact that our algorithm converged to reasonable values for time delay and inter-sensor transformation suggests that the significance of this approach is not limited to camera IMU calibration but presumably extends to other sensor setups as well.



	J	G	S	A
slope	0.498	0.493	0.531	0.553
RMS	0.054ms	0.165ms	0.344ms	0.572ms

Fig. 7. Comparison of approaches for determining the time delay. The joint estimation incorporating all sensor information available results in significantly reduced variance in the estimates and the most consistent results. Using a subset of all sensors results—either only the gyroscopes or only the accelerometer in addition to the camera—yields less accurate estimates. In our experiments, a separation of temporal and spatial calibration as proposed in [6] resulted in less accurate estimates, suggesting that the calibration may benefit more from additional measurements than from the separation of uncorrelated parameters.

VI. CONCLUSION AND FUTURE WORK

In this study, we presented a novel approach to jointly calibrate for temporal offsets and spatial transformations between multiple sensors. Using a continuous state representation allows us to treat the problem of estimating temporal offsets within the rigorous theoretical framework of maximum likelihood estimation.

For the case of camera IMU calibration, we showed that it was beneficial to calibrate for time offsets and inter-sensor transformations in a single estimator, rather than determining these quantities in separate procedures. We believe that this holds true for all multi-sensor systems where the estimated quantities are well observable.

However, further questions need to be addressed in future work. For sensor setups without a common clock, the time offset may be different from start up to start up and subject to drift over longer periods of time, hence requiring continuous estimation of the offset in operation and in the absence of a known visual calibration target.

Furthermore, we strongly believe that the application domain of this framework extends to sensor combinations other than camera and IMU and future work will include experiments with a broader spectrum of sensors and setups.

ACKNOWLEDGMENT

The authors would like to thank Elmar Mair for helpful discussions and valuable contributions on the issue of temporal calibration, as well as Janosch Nikolic and Michael Burri for providing the visual-inertial sensor that enabled the experiments, and Faraz Mirzaei, Stefan Weiss, Stefan Leutenegger, Margarita Chli, and Jonathan Kelly for their inputs on cameras and inertial sensors. This work was supported by the EU project V-Charge (FP7-269916). The

development of the visual-inertial sensor was supported by CTI project 13394.1 PFFLE-NM.

REFERENCES

- [1] D. F. Maune, Ed., *Digital elevation model technologies and applications : the DEM user manual*, 2nd. edition. Bethesda, Maryland, USA: American Society for Photogrammetry and Remote Sensing, 2007.
- [2] D. Mills, "Network time protocol version 4: Protocol and algorithms specification," RFC 5905, Fremont, California, USA, 2010. [Online]. Available: <http://www.ietf.org/rfc/rfc5905.txt>
- [3] A. Harrison and P. Newman, "TICSynC: Knowing when things happened," in *Proc. IEEE International Conference on Robotics and Automation (ICRA2011)*, Shanghai, China, May 2011, 05.
- [4] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 227–234.
- [5] J. Kelly and G. S. Sukhatme, "A general framework for temporal calibration of multiple proprioceptive and exteroceptive sensors," in *12th International Symposium on Experimental Robotics, 2010*, Delhi, India, Dec 2010.
- [6] E. Mair, M. Fleps, M. Suppa, and D. Burschka, "Spatio-temporal initialization for IMU to camera registration," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*. IEEE, 2011, pp. 557–564.
- [7] P. T. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MN, 14–18 May 2012, pp. 2088–2095.
- [8] J. Alves, J. Lobo, and J. Dias, "Camera-inertial sensor modelling and alignment for visual navigation," in *Int. Conf. on Advanced Robotics*, 2003, pp. 1693–1698.
- [9] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007.
- [10] F. Mirzaei and S. Roumeliotis, "A kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1143–1156, 2008.
- [11] J. Kelly and G. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.
- [12] F. M. Mirzaei and S. I. Roumeliotis, "IMU-camera calibration: Bundle adjustment implementation," Department of Computer Science and Engineering, University of Minnesota, Tech. Rep., August 2007.
- [13] M. Fleps, E. Mair, O. Ruepp, M. Suppa, and D. Burschka, "Optimization based IMU camera calibration," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3297–3304.
- [14] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Los Altos, California, USA: Morgan Kaufmann Publishers Inc., 1987.
- [15] P. C. Hughes, *Spacecraft Attitude Dynamics*. New York: John Wiley & Sons, 1986.
- [16] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.
- [17] C. de Boor, *A practical guide to splines*. New York, USA: Springer Verlag, 2001.
- [18] Y. Chen, T. Davis, W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 3, p. 22, 2008.
- [19] IEEE Aerospace and Electronic Systems Society. Gyro and Accelerometer Panel and Institute of Electrical and Electronics Engineers, *IEEE Standard Specification Format Guide and Test Procedure for Single-axis Interferometric Fiber Optic Gyros*, ser. IEEE (std.). IEEE, 1998.
- [20] —, *IEEE Standard Specification Format Guide and Test Procedure for Linear, Single-axis, Nongyroscopic Accelerometers*, ser. IEEE (std.). IEEE, 1999.

H Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems

Authors: Barfoot, Furgale

Link to [Table of Contents](#).

Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems

Timothy D. Barfoot, *Member, IEEE*, and Paul T. Furgale, *Member, IEEE*

Abstract—In this paper, we provide specific and practical approaches to associate uncertainty with 4×4 transformation matrices, which is a common representation for pose variables in 3-D space. We show constraint-sensitive means of perturbing transformation matrices using their associated exponential-map generators and demonstrate these tools on three simple-yet-important estimation problems: 1) propagating uncertainty through a compound pose change, 2) fusing multiple measurements of a pose (e.g., for use in pose-graph relaxation), and 3) propagating uncertainty on poses (and landmarks) through a nonlinear camera model. The contribution of the paper is the presentation of the theoretical tools, which can be applied in the analysis of many problems involving 3-D pose and point variables.

Index Terms—Exponential maps, homogeneous points, matrix Lie groups, pose uncertainty, transformation matrices.

I. INTRODUCTION

THE main contribution of this paper is to provide simple and practical techniques to associate uncertainty with 4×4 transformation matrices (commonly used to represent pose variables) for use in robotic estimation problems. The challenge in choosing a representation of pose and its associated uncertainty is that representations either have singularities or constraints; this results from the fact that rotation variables are not *vectors* but rather members of a *noncommutative group*. The advantage of the technique that we will present is that it is free of singularities but avoids the need to enforce constraints when solving optimal estimation problems. This is accomplished by storing the mean as a (singularity-free) 4×4 transformation matrix and using a 6×1 (constraint-sensitive) perturbation of the pose (with an associated 6×6 covariance matrix). As we will see, our approach amounts to getting the best of both worlds as it facilitates rather straightforward implementations of common estimation problems such as bundle adjustment [1].

Transformation matrices are by no means new, having risen to popularity through manipulator robotics [2] and computer vision [3]. We will use standard definitions for the *special orthogonal*

group that represents rotation

$$SO(3) := \{ \mathbf{C} \in \mathbb{R}^{3 \times 3} \mid \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det \mathbf{C} = 1 \}$$

where $\mathbf{1}$ is the identity matrix, and the *special Euclidean group* that represents rotation and translation

$$SE(3) := \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 4} \mid \{ \mathbf{C}, \mathbf{r} \} \in SO(3) \times \mathbb{R}^3 \right\}. \quad (1)$$

Both are examples of *matrix Lie groups*, for which Stillwell [4] provides an accessible introduction. We will avoid rehashing the basics of group theory here but stress that we cannot apply the usual approach of additive uncertainty for such quantities as they are not members of a *vector space*. In other words

$$\mathbf{x} = \bar{\mathbf{x}} + \delta \quad (2)$$

where $\mathbf{x} \in \mathbb{R}^n$ is a random variable, $\bar{\mathbf{x}}$ is a ‘large,’ noise-free value, and δ is a ‘small,’ uncertain perturbation (i.e., zero-mean noise), which does not directly apply to members of $SO(3)$ and $SE(3)$. This paper investigates one means to overcome this problem.

In robotics, it has been well established that estimating uncertain spatial relationships is a fundamentally important problem. For example, the early works of Brooks [5], Smith and Cheeseman [6], Durrant-Whyte [7], and Smith *et al.* [8] spurred the popular subfield of *simultaneous localization and mapping* [9]. Our approach is similar to the work of Chirikjian [10] and [11], who provide a rigorous treatment of representing and propagating uncertainty on Lie groups. It is also related to and extends the ideas presented by Su and Lee [12], [13] and Smith *et al.* [14]. Finally, in a recent paper, Hertzberg *et al.* [15] take a similar approach to associate uncertainty with general manifolds, while we provide more details on handling $SE(3)$ specifically. One could view our paper as providing an accessible and specific implementation of a means of associating and manipulating uncertainties for $SE(3)$ [and, thus, collaterally with $SO(3)$]. The differences between our approach and the previously published works are subtle but important. Rather than list them here in detail, we will take care to point out specific differences throughout the paper, then summarize them in the conclusion.

We also demonstrate the usefulness of our method on three key problems: 1) propagating uncertainty through a compounding of two transformation matrices, 2) fusing multiple uncertain measurements of a pose into a single estimate, and 3) propagating pose (and landmark) uncertainty through a nonlinear (stereo) camera model. In our calculations, we will retain as many terms in the noise variables as possible, thereby demonstrating the power of our approach.

Manuscript received August 8, 2013; revised November 17, 2013; accepted December 20, 2013. This paper was recommended for publication by Associate Editor R. Eustice and Editor D. Fox upon evaluation of the reviewers’ comments. This work was supported by the Natural Sciences and Engineering Research Council of Canada and the EU FP7 2007–2013 Programme, Challenge 2, Cognitive Systems, Interaction, Robotics, under Grant 269916, V-Charge.

T. D. Barfoot is with the Autonomous Space Robotics Lab, University of Toronto, ON M5S 2J7, Canada (e-mail: tim.barfoot@utoronto.ca).

P. T. Furgale is with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland (e-mail: paul.furgale@mavt.ethz.ch).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TRO.2014.2298059

This paper is organized as follows. Section II presents the mathematical machinery that we will use throughout the paper. Section III discusses how to compound two poses and their uncertainties, Section IV shows how to fuse multiple estimates of pose together, and Section V shows how to propagate pose and landmark uncertainty through a nonlinear measurement model. A conclusion rounds out the paper.

II. MATHEMATICAL PRELIMINARIES

A. Random Variables and Probability Density Functions

In contrast with (2), we will define random variables for $SE(3)$ according to

$$\mathbf{T} := \exp(\xi^\wedge) \bar{\mathbf{T}} \quad (3)$$

where $\bar{\mathbf{T}} \in SE(3)$ is a ‘large,’ noise-free value, and $\xi \in \mathbb{R}^6$ is a ‘small,’ noisy perturbation. The $\wedge$ operator¹ turns ξ into a 4×4 member of the *Lie algebra* $\mathfrak{se}(3)$ according to

$$\xi^\wedge := \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\wedge = \begin{bmatrix} \phi^\wedge & \rho \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (4)$$

where $\rho, \phi \in \mathbb{R}^3$ are 3×1 , and $\wedge$ also turns ϕ into a member of the *Lie algebra* $\mathfrak{so}(3)$:

$$\phi^\wedge := \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \quad (5)$$

We will use $\vee$ as the inverse operation of $\wedge$. This perturbation approach ensures that $\mathbf{T} \in SE(3)$ and is sometimes referred to as an *injection* of noise onto the group $SE(3)$ [20]. Additional notation and a closed-form expression for the matrix exponential in (3) can be found in the Appendix.

In this paper, we will define the small perturbation variable to be zero-mean Gaussian $p(\xi) = \mathcal{N}(\mathbf{0}, \Sigma)$, where Σ is a 6×6 covariance matrix. We stress that we are directly defining this probability density function (PDF) in the vectorspace, $\mathbb{R}^6$. This choice in turn induces a PDF over $SE(3)$, $p(\mathbf{T})$. To see what this PDF is, we note that by our definition

$$\int_{\mathbb{R}^6} p(\xi) d\xi = \int_{\mathbb{R}^6} \eta \exp\left(-\frac{1}{2}\xi^T \Sigma^{-1} \xi\right) d\xi = 1 \quad (6)$$

where $\eta = \frac{1}{\sqrt{(2\pi)^6 \det(\Sigma)}}$ is the (constant) normalization factor.

To switch variables and write a PDF over $\mathbf{T}$, we must define what we mean by an infinitesimal volume of $SE(3)$, which is written as $d\mathbf{T}$. Along the lines of Chirikjian [11], we will relate perturbations in ξ to perturbations in $\mathbf{T}$ according to

$$\delta\mathbf{T} := \ln(\mathbf{T}'\mathbf{T}^{-1})^\vee \quad (7)$$

¹The $\wedge$ operator is ‘overloaded’ in the sense that it can be applied to both 6×1 and 3×1 columns [2] [11] [16]. For the 6×1 case, $\wedge$ is similar to the $\boxplus$ operator defined by [17] but without the negative sign in front of the ρ component. For the 3×1 case, $\phi^\wedge$ is sometimes written as $\phi^\times$ [18] or $[[\phi]]$ [19]; sometimes, the $\wedge$ is written above the symbol, as in $\hat{\phi}^\wedge$; however, we shall stick to the original definition in Murray *et al.* [2, pp. 26 and 41], which has it off to the right, since we often apply it to expressions involving multiple variables.

where

$$\mathbf{T}' := \exp((\xi + \delta\xi)^\wedge) \bar{\mathbf{T}}, \quad \mathbf{T} := \exp(\xi^\wedge) \bar{\mathbf{T}}. \quad (8)$$

It can then be shown that

$$\delta\mathbf{T} = \mathcal{J} \delta\xi \quad (9)$$

where

$$\mathcal{J} := \int_0^1 \exp(\xi^\wedge)^\alpha d\alpha \quad (10)$$

is referred to as the *Jacobian* for $SE(3)$ [21]; we will discuss this in more detail in the next section. We define the λ operator² to be

$$\xi^\lambda = \begin{bmatrix} \rho \\ \phi \end{bmatrix}^\lambda = \begin{bmatrix} \phi^\wedge & \rho^\wedge \\ \mathbf{0} & \phi^\wedge \end{bmatrix} \quad (12)$$

which is 6×6 . We will also use γ as the inverse operation of λ , for convenience.

The infinitesimal volume elements can then be related [21] by

$$d\mathbf{T} = |\det(\mathcal{J})| d\xi. \quad (13)$$

Using $\xi = \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^\vee$ and (13) in (6), we may find the PDF over $\mathbf{T}$ by a straightforward change of coordinates³

$$\begin{aligned} 1 &= \int_{\mathbb{R}^6} \eta \exp\left(-\frac{1}{2}\xi^T \Sigma^{-1} \xi\right) d\xi \\ &= \int_{SE(3)} \underbrace{\beta \exp\left(-\frac{1}{2} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^\vee{}^T \Sigma^{-1} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^\vee\right)}_{p(\mathbf{T})} d\mathbf{T} \\ &= \int_{SE(3)} p(\mathbf{T}) d\mathbf{T} \end{aligned} \quad (14)$$

where

$$\beta = \frac{\eta}{|\det(\mathcal{J})|} = \frac{1}{\sqrt{(2\pi)^6 \det(\mathcal{J}\Sigma\mathcal{J}^T)}}. \quad (15)$$

Importantly, $p(\mathbf{T})$ is not Gaussian because of the fact that β depends on $\mathbf{T}$ via $\mathcal{J}$. Again, we stress that our $p(\mathbf{T})$ is only indirectly defined via $p(\xi)$.

Our approach to define uncertainty on $SE(3)$ via ξ is very similar to that introduced by Su and Lee [12] and [13], although we apply the perturbation in (3) on the left rather than the right (both are valid). However, our paper works out the consequences

²Our λ operator is similarly defined as $\times$ by D’Eleuterio [22] and [23] for use in manipulator kinematics/dynamics. In relation to the other notation in the literature [11], we have

$$\text{Ad}(\exp(\xi^\wedge)) = \exp(\xi^\lambda) = \exp(\text{ad}(\xi^\wedge)) \quad (11)$$

where $\text{Ad}(\cdot)$ and $\text{ad}(\cdot)$ are the *adjoints* for $SE(3)$ and $\mathfrak{se}(3)$, respectively. Thus, by introducing λ , we have a means to compute both adjoints.

³Technically speaking, to carry out this change of coordinates rigorously, we would like the integration limits on the ϕ part of ξ to be such that $\|\phi\| < \pi$. However, a Gaussian has infinite tails, and therefore, we are effectively allowing the tails to ‘fold’ around by integrating over all of $\mathbb{R}^6$. In most practical situations, the Gaussian is not too spread out, and there is very little probability mass beyond the π limit. When a global representation of uncertainty on $SO(3)$ is required, we may turn to the approach of Lee *et al.* [24].

of this parameterization further (and to higher accuracy) than Su and Lee, particularly the compounding and fusing of uncertain transformations.

Our approach is also similar but subtly different from the approach of Chirikjian [11], Wang and Chirikjian [25], [26], and Wolfe *et al.* [27]. These works start by defining $p(\mathbf{T})$ directly and switch to parameterizing $\mathbf{T}$ via exponential coordinates to carry out computations. In terms of our notation, they define a Gaussian PDF over $\mathbf{T}$ as

$$p'(\mathbf{T}) = \beta' \exp \left(-\frac{1}{2} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^{\vee T} \Sigma^{-1} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^{\vee} \right) \quad (16)$$

where β' is a (constant) normalization factor chosen to make $p'(\mathbf{T})$ a valid PDF [27, eq. 14]. By contrast, we define $p(\xi)$ first and induce $p(\mathbf{T})$ through (3). The difference between these works and ours is essentially between β' (a constant) and β (dependent on $\mathbf{T}$); both are in fact valid PDFs, but we believe our approach has merit in terms of its relative ease of computation.

In practice, we simply work with $p(\xi)$ and never need to compute $p(\mathbf{T})$. For example, if we want to compute the expected value $E[\cdot]$ of some nonlinear function of a transformation $f(\mathbf{T})$, we can simply use $p(\xi)$ as follows:

$$\begin{aligned} E[f(\mathbf{T})] &= \int_{SE(3)} f(\mathbf{T}) p(\mathbf{T}) d\mathbf{T} \\ &= \int_{SE(3)} f(\mathbf{T}) \underbrace{\beta \exp \left(-\frac{1}{2} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^{\vee T} \Sigma^{-1} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^{\vee} \right)}_{p(\mathbf{T})} d\mathbf{T} \\ &= \int_{\mathbb{R}^6} f(\exp(\xi^\wedge) \bar{\mathbf{T}}) \underbrace{\beta |\det(\mathcal{J})| \exp \left(-\frac{1}{2} \xi^T \Sigma^{-1} \xi \right)}_{p(\xi)} d\xi \\ &= \int_{\mathbb{R}^6} f(\exp(\xi^\wedge) \bar{\mathbf{T}}) p(\xi) d\xi. \end{aligned} \quad (17)$$

We underscore that we do not require any approximation to write down this last expression, which does not contain $|\det(\mathcal{J})|$, and this is because of our choice to define $p(\xi)$ directly. However, $f(\cdot)$ is still a nonlinearity, and therefore, by expanding the matrix exponential, and performing a Taylor-series expansion in the ξ noise variables, we can easily compute approximations of the expectation to as many terms as we like.

In terms of the statistical moments of $p(\mathbf{T})$, our approach would seem to be the more natural choice, as we now show. One common (left) definition of the *mean* of $p(\mathbf{T})$ [25] is the unique $\mathbf{M} \in SE(3)$ that satisfies the following equation:

$$\int_{SE(3)} \ln(\mathbf{T}\mathbf{M}^{-1})^{\vee} p(\mathbf{T}) d\mathbf{T} = \mathbf{0}. \quad (18)$$

It is easy to see, under our version of $p(\mathbf{T})$, that in fact, $\mathbf{M} = \bar{\mathbf{T}}$ since

$$\int_{SE(3)} \ln(\mathbf{T}\bar{\mathbf{T}}^{-1})^{\vee} p(\mathbf{T}) d\mathbf{T} = \int_{\mathbb{R}^6} \xi p(\xi) d\xi = \mathbf{0}. \quad (19)$$

As pointed out by Wolfe *et al.* [27], using the other PDF definition, $p'(\mathbf{T})$, it is not necessarily true that $\mathbf{M}' = \bar{\mathbf{T}}$, where

$$\int_{SE(3)} \ln(\mathbf{T}\mathbf{M}'^{-1})^{\vee} p'(\mathbf{T}) d\mathbf{T} = \mathbf{0}. \quad (20)$$

Likewise, we see that our Σ matches one common (left) definition of *covariance* for $SE(3)$ [25]

$$\begin{aligned} \Sigma &:= \int_{SE(3)} \ln(\mathbf{T}\mathbf{M}^{-1})^{\vee} \ln(\mathbf{T}\mathbf{M}^{-1})^{\vee T} p(\mathbf{T}) d\mathbf{T} \\ &= \int_{\mathbb{R}^6} \xi \xi^T p(\xi) d\xi. \end{aligned} \quad (21)$$

Using the other PDF definition, $p'(\mathbf{T})$, we let

$$\Sigma' := \int_{SE(3)} \ln(\mathbf{T}\mathbf{M}'^{-1})^{\vee} \ln(\mathbf{T}\mathbf{M}'^{-1})^{\vee T} p'(\mathbf{T}) d\mathbf{T} \quad (22)$$

and then it is not necessarily true that $\Sigma' = \Sigma$. As discussed by Wolfe *et al.* [27], as Σ gets very small, we have that

$$\mathbf{M}' \approx \mathbf{M} = \bar{\mathbf{T}}, \quad \Sigma' \approx \Sigma \quad (23)$$

because in this case, $\mathcal{J} \approx \mathbf{1}$, and therefore, $\beta' \approx \beta$.

There is another reason why our approach to inject uncertainty into $SE(3)$ may be the more natural choice for robotics applications. Consider the equation for the continuous-time kinematics of a rigid body

$$\dot{\mathbf{T}} = \varpi^\wedge \mathbf{T} \quad (24)$$

where $\varpi := [\begin{smallmatrix} \nu \\ \omega \end{smallmatrix}]$ is the combined linear and angular velocity. If we assume that ϖ is constant from t_1 to t_2 , then the discrete-time update is

$$\mathbf{T}(t_2) = \exp(t\varpi^\wedge) \mathbf{T}(t_1) \quad (25)$$

where $t := t_2 - t_1$. If we are dead-reckoning from the measured velocity, then in recursive estimation schemes such as the extended Kalman filter, we typically assume that ϖ is corrupted by Gaussian noise. This is precisely what our perturbation scheme in (3) assumes.

Finally, a deterministic change of coordinates, via transformation $\mathbf{T}_d \in SE(3)$, is straightforward under our parameterization [12], since

$$\mathbf{T}_d \mathbf{T} = \mathbf{T}_d \exp(\xi^\wedge) \bar{\mathbf{T}} = \exp((\mathcal{T}_d \xi)^\wedge) \mathbf{T}_d \bar{\mathbf{T}}, \quad (26)$$

where $\mathcal{T}_d = \text{Ad}(\mathbf{T}_d)$. The new mean is therefore $\mathbf{T}_d \bar{\mathbf{T}}$ and the new covariance is $\mathcal{T}_d \Sigma \mathcal{T}_d^T$. Thus, we do not require any approximation to apply a deterministic change of coordinates to our chosen PDF. We will further discuss the compounding of two uncertain transformations in Section III.

B. Baker–Campbell–Hausdorff Formula

One of the main tools that we will use to manipulate uncertainties on $SE(3)$ is the *Baker–Campbell–Hausdorff* (BCH) formula, which can be used to compound two matrix

exponentials

$$\begin{aligned} \ln(\exp(\mathbf{A}) \exp(\mathbf{B})) &= \mathbf{A} + \mathbf{B} + \frac{1}{2} [\mathbf{A}, \mathbf{B}] + \frac{1}{12} [\mathbf{A}, [\mathbf{A}, \mathbf{B}]] \\ &+ \frac{1}{12} [\mathbf{B}, [\mathbf{B}, \mathbf{A}]] - \frac{1}{24} [\mathbf{B}, [\mathbf{A}, [\mathbf{A}, \mathbf{B}]]] + \dots \end{aligned} \quad (27)$$

where the *Lie bracket* is given by

$$[\mathbf{A}, \mathbf{B}] := \mathbf{AB} - \mathbf{BA}. \quad (28)$$

Note, the BCH formula is an infinite series. If we keep only terms linear in $\mathbf{A}$, it becomes [28]

$$\ln(\exp(\mathbf{A}) \exp(\mathbf{B})) \approx \mathbf{B} + \sum_{n=0}^{\infty} \frac{B_n}{n!} \underbrace{[\mathbf{B}, [\mathbf{B}, \dots [\mathbf{B}, \mathbf{A}] \dots]]}_n. \quad (29)$$

The B_n are the *Bernoulli numbers*

$$B_0 = 1, B_1 = -\frac{1}{2}, B_2 = \frac{1}{6}, B_3 = 0, B_4 = -\frac{1}{30}, B_5 = 0, \dots, \quad (30)$$

which appear frequently in number theory.

In the particular case of $SE(3)$, we can use the fact that

$$[\xi_a^\wedge, \xi_b^\wedge] = \xi_a^\wedge \xi_b^\wedge - \xi_b^\wedge \xi_a^\wedge = (\xi_a^\wedge \xi_b)^\wedge \quad (31)$$

to show that

$$\begin{aligned} \ln(\mathbf{T}_a \mathbf{T}_b)^\vee &= \ln(\exp(\xi_a^\wedge) \exp(\xi_b^\wedge))^\vee = \xi_a + \xi_b + \frac{1}{2} \xi_a^\wedge \xi_b \\ &+ \frac{1}{12} \xi_a^\wedge \xi_a^\wedge \xi_b + \frac{1}{12} \xi_b^\wedge \xi_b^\wedge \xi_a - \frac{1}{24} \xi_b^\wedge \xi_a^\wedge \xi_a^\wedge \xi_b + \dots \end{aligned} \quad (32)$$

Alternatively, if we assume that ξ_a is small, then using the approximate formula stated previously, we can show that

$$\ln(\mathbf{T}_a \mathbf{T}_b)^\vee = \ln(\exp(\xi_a^\wedge) \exp(\xi_b^\wedge))^\vee \approx \xi_b + \mathcal{J}_b^{-1} \xi_a \quad (33)$$

where

$$\mathcal{J}_b^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (\xi_b^\wedge)^n. \quad (34)$$

Inverting, we find

$$\mathcal{J}_b = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\xi_b^\wedge)^n = \int_0^1 \exp(\xi_b^\wedge \alpha) d\alpha. \quad (35)$$

In Lie group theory, $\mathcal{J}_b$ is referred to as the left⁴ *Jacobian* of $SE(3)$. We saw this quantity in action in the previous section already; however, it is useful to understand that it comes directly from the BCH formula under the assumption that ξ_a is small. The appendix contains a closed-form expression to compute $\mathcal{J}_b$ from ξ_b .

III. COMPOUNDING POSES

In this section, we investigate the problem of compounding two poses, each with associated uncertainty, as depicted in Fig. 1.

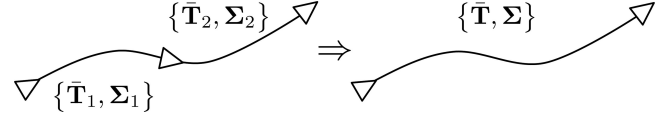


Fig. 1. Combining a chain of two poses into a single compound pose.

A. Theory

Consider two noisy poses $\mathbf{T}_1$ and $\mathbf{T}_2$; we keep track of their nominal values and associated uncertainties

$$\{\bar{\mathbf{T}}_1, \Sigma_1\}, \quad \{\bar{\mathbf{T}}_2, \Sigma_2\}.$$

Now, suppose, we let

$$\mathbf{T} = \mathbf{T}_1 \mathbf{T}_2$$

as depicted in Fig. 1. What is $\{\bar{\mathbf{T}}, \Sigma\}$? Under our perturbation scheme, we have

$$\exp(\xi^\wedge) \bar{\mathbf{T}} = \exp(\xi_1^\wedge) \bar{\mathbf{T}}_1 \exp(\xi_2^\wedge) \bar{\mathbf{T}}_2. \quad (36)$$

Moving all the uncertain factors to the left side, we have

$$\exp(\xi^\wedge) \bar{\mathbf{T}} = \exp(\xi_1^\wedge) \exp((\bar{\mathbf{T}}_1 \xi_2)^\wedge) \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2 \quad (37)$$

where $\bar{\mathbf{T}}_1 = \text{Ad}(\bar{\mathbf{T}}_1)$. If we let

$$\bar{\mathbf{T}} = \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2 \quad (38)$$

we are left with

$$\exp(\xi^\wedge) = \exp(\xi_1^\wedge) \exp((\bar{\mathbf{T}}_1 \xi_2)^\wedge). \quad (39)$$

Letting $\xi'_2 := \bar{\mathbf{T}}_1 \xi_2$, we can apply the BCH formula to find

$$\begin{aligned} \xi &= \xi_1 + \xi'_2 + \frac{1}{2} \xi_1^\wedge \xi'_2 + \frac{1}{12} \xi_1^\wedge \xi_1^\wedge \xi'_2 + \frac{1}{12} \xi'_2 \xi_1^\wedge \xi_1^\wedge \\ &- \frac{1}{24} \xi_2^\wedge \xi_1^\wedge \xi_1^\wedge \xi'_2 + \dots \end{aligned} \quad (40)$$

For our approach to hold, we require that $E[\xi] = \mathbf{0}$. Assuming that $\xi_1 \sim \mathcal{N}(\mathbf{0}, \Sigma_1)$ and $\xi'_2 \sim \mathcal{N}(\mathbf{0}, \Sigma'_2)$ are uncorrelated with one another, we have

$$E[\xi] = -\frac{1}{24} E[\xi_2'^\wedge \xi_1^\wedge \xi_1^\wedge \xi'_2] + O(\xi^6) \quad (41)$$

since everything except the fourth-order term has zero mean. Thus, to third order, we can safely assume that $E[\xi] = \mathbf{0}$, and thus, (38) seems to be a reasonable way to compound the mean transformations.

It is also possible to show that the fourth-order term has zero mean, $E[\xi_2'^\wedge \xi_1^\wedge \xi_1^\wedge \xi'_2] = \mathbf{0}$, if Σ_1 is of the special form

$$\Sigma_1 = E[\xi_1 \xi_1^T] = \begin{bmatrix} \Sigma_{1,\rho\rho} & \mathbf{0} \\ \mathbf{0} & \sigma_{1,\phi\phi}^2 \mathbf{1} \end{bmatrix} \quad (42)$$

where $\xi_1 = [\rho_1]$. This is a common situation for Σ_1 when we are, for example, propagating uncertainty on velocity, as in (25). In this case, we are justified to assume $E[\xi] = \mathbf{0}$ all the way out to fifth order (and possibly further).

⁴There is also a right Jacobian if we instead assume ξ_b is small.

The next task is to compute $\Sigma = E[\xi\xi^T]$. Multiplying out to fourth order, we have

$$\begin{aligned} E[\xi\xi^T] \approx E & \left[\xi_1\xi_1^T + \xi_2'\xi_2'^T + \frac{1}{12} \left((\xi_1^\wedge \xi_1^\wedge) (\xi_2'\xi_2'^T) \right. \right. \\ & + (\xi_2'\xi_2'^T) (\xi_1^\wedge \xi_1^\wedge)^T + (\xi_2^\wedge \xi_2^\wedge) (\xi_1\xi_1^T) \\ & \left. \left. + (\xi_2^\wedge \xi_2^\wedge) (\xi_1\xi_1^T) \right) + \frac{1}{4} \xi_1^\wedge (\xi_2'\xi_2'^T) \xi_1^{\wedge T} \right] \quad (43) \end{aligned}$$

where we have omitted showing any terms that have an odd power in either ξ_1 or ξ_2' since these will by definition have expectation zero. This expression may look daunting, but we can take it term by term. To save space, we define and make use of the following two linear operators:

$$\langle\langle \mathbf{A} \rangle\rangle := -\text{tr}(\mathbf{A}) \mathbf{1} + \mathbf{A} \quad (44)$$

$$\langle\langle \mathbf{A}, \mathbf{B} \rangle\rangle := \langle\langle \mathbf{A} \rangle\rangle \langle\langle \mathbf{B} \rangle\rangle + \langle\langle \mathbf{B}\mathbf{A} \rangle\rangle \quad (45)$$

with $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$. These provide the useful identity

$$-\mathbf{u}^\wedge \mathbf{A} \mathbf{v}^\wedge \equiv \langle\langle \mathbf{v}\mathbf{u}^T, \mathbf{A}^T \rangle\rangle \quad (46)$$

where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$ and $\mathbf{A} \in \mathbb{R}^{3 \times 3}$. Making use of this repeatedly, we have out to fourth order

$$E[\xi_1\xi_1^T] = \Sigma_1 = \begin{bmatrix} \Sigma_{1,\rho\rho} & \Sigma_{1,\rho\phi} \\ \Sigma_{1,\rho\phi}^T & \Sigma_{1,\phi\phi} \end{bmatrix} \quad (47)$$

$$E[\xi_2'\xi_2'^T] = \Sigma_2' = \begin{bmatrix} \Sigma_{2,\rho\rho}' & \Sigma_{2,\rho\phi}' \\ \Sigma_{2,\rho\phi}'^T & \Sigma_{2,\phi\phi}' \end{bmatrix} = \bar{\mathbf{T}}_1 \Sigma_2 \bar{\mathbf{T}}_1^T \quad (48)$$

$$E[\xi_1^\wedge \xi_1^\wedge] = \mathcal{A}_1 = \begin{bmatrix} \langle\langle \Sigma_{1,\phi\phi} \rangle\rangle & \langle\langle \Sigma_{1,\rho\phi} + \Sigma_{1,\rho\phi}^T \rangle\rangle \\ \mathbf{0} & \langle\langle \Sigma_{1,\phi\phi} \rangle\rangle \end{bmatrix} \quad (49)$$

$$E[\xi_2^\wedge \xi_2^\wedge] = \mathcal{A}_2' = \begin{bmatrix} \langle\langle \Sigma_{2,\phi\phi}' \rangle\rangle & \langle\langle \Sigma_{2,\rho\phi}' + \Sigma_{2,\rho\phi}'^T \rangle\rangle \\ \mathbf{0} & \langle\langle \Sigma_{2,\phi\phi}' \rangle\rangle \end{bmatrix} \quad (50)$$

$$E[\xi_1^\wedge (\xi_2'\xi_2'^T) \xi_1^{\wedge T}] = \mathcal{B} = \begin{bmatrix} \mathbf{B}_{\rho\rho} & \mathbf{B}_{\rho\phi} \\ \mathbf{B}_{\rho\phi}^T & \mathbf{B}_{\phi\phi} \end{bmatrix} \quad (51)$$

where

$$\begin{aligned} \mathbf{B}_{\rho\rho} &= \langle\langle \Sigma_{1,\phi\phi}, \Sigma_{2,\rho\rho}' \rangle\rangle + \langle\langle \Sigma_{1,\rho\phi}^T, \Sigma_{2,\rho\phi}' \rangle\rangle \\ &+ \langle\langle \Sigma_{1,\rho\phi}, \Sigma_{2,\rho\phi}'^T \rangle\rangle + \langle\langle \Sigma_{1,\rho\rho}, \Sigma_{2,\phi\phi}' \rangle\rangle \end{aligned} \quad (52)$$

$$\mathbf{B}_{\rho\phi} = \langle\langle \Sigma_{1,\phi\phi}, \Sigma_{2,\rho\phi}'^T \rangle\rangle + \langle\langle \Sigma_{1,\rho\phi}^T, \Sigma_{2,\phi\phi}' \rangle\rangle \quad (53)$$

$$\mathbf{B}_{\phi\phi} = \langle\langle \Sigma_{1,\phi\phi}, \Sigma_{2,\phi\phi}' \rangle\rangle. \quad (54)$$

The resulting covariance is then

$$\begin{aligned} \Sigma_{4\text{th}} &\approx \underbrace{\Sigma_1 + \Sigma_2'}_{\Sigma_{2\text{nd}}} \\ &+ \underbrace{\frac{1}{12} (\mathcal{A}_1 \Sigma_2' + \Sigma_2' \mathcal{A}_1^T + \mathcal{A}_2' \Sigma_1 + \Sigma_1 \mathcal{A}_2'^T)}_{\text{additional fourth-order terms}} + \frac{1}{4} \mathcal{B} \end{aligned} \quad (55)$$

correct to fourth order.⁵ This result is essentially the same as that of Wang and Chirikjian [26] but worked out for our slightly

⁵The sixth-order terms require a lot more work, but it is possible to compute them using Isserlis' theorem.

different PDF, as discussed earlier; it is important to note that, while our method is fourth order in the perturbation variables, it is only second order in the covariance, which is the same as [26]. In summary, to compound two poses, we propagate the mean using (38) and the covariance using (55).

B. Sigmoid Method

We can also make use of the popular *sigmapoint transformation* [29] to pass uncertainty through the compound pose change. In this section, we tailor this to our specific type of $SE(3)$ perturbation. Our approach to handling sigmapoints is quite similar to that taken by Hertzberg *et al.* [15] as well as Brookshire and Teller [30]. In our case, we begin by approximating the joint input Gaussian using a finite number of samples, $\{\mathbf{T}_{1,\ell}, \mathbf{T}_{2,\ell}\}$

$$\mathbf{SS}^T := \text{diag}\{\Sigma_1, \Sigma_2\}, \quad (\text{Cholesky decomposition})$$

$$\psi_\ell := \sqrt{\lambda} \text{col}_\ell \mathbf{S}, \quad \ell = 1 \dots L$$

$$\psi_{\ell+L} := -\sqrt{\lambda} \text{col}_\ell \mathbf{S}, \quad \ell = 1 \dots L$$

$$\begin{bmatrix} \xi_{1,\ell} \\ \xi_{2,\ell} \end{bmatrix} := \psi_\ell, \quad \ell = 1 \dots 2L$$

$$\mathbf{T}_{1,\ell} := \exp(\xi_{1,\ell}^\wedge) \bar{\mathbf{T}}_1, \quad \ell = 1 \dots 2L$$

$$\mathbf{T}_{2,\ell} := \exp(\xi_{2,\ell}^\wedge) \bar{\mathbf{T}}_2, \quad \ell = 1 \dots 2L$$

where λ is a user-definable scaling constant,⁶ and $L = 12$. We then pass each of these samples through the compound pose change and compute the difference from the mean

$$\xi_\ell = \ln(\mathbf{T}_{1,\ell} \mathbf{T}_{2,\ell} \bar{\mathbf{T}}^{-1})^\vee, \quad \ell = 1 \dots 2L. \quad (56)$$

These are combined to create the output covariance according to

$$\Sigma_{\text{sp}} = \frac{1}{2\lambda} \sum_{\ell=1}^{2L} \xi_\ell \xi_\ell^T. \quad (57)$$

Note that we have assumed that the output sigmapoint samples have zero mean in this formula, to be consistent with our mean propagation. Interestingly, this turns out to be algebraically equivalent to the second-order method (from the previous section) for this particular nonlinearity, since the noise sources on $\mathbf{T}_1$ and $\mathbf{T}_2$ are assumed to be uncorrelated.

C. Simple Compound Example

In this section, we present a simple qualitative example of pose compounding and in Section III-D, we carry out a more quantitative study on a different setup. To see the qualitative difference between the second- and fourth-order methods, let us consider the case of compounding transformations many times in a row

$$\exp(\xi_K^\wedge) \bar{\mathbf{T}}_K = \left(\prod_{k=1}^K \exp(\xi_k^\wedge) \bar{\mathbf{T}} \right) \exp(\xi_0^\wedge) \bar{\mathbf{T}}_0. \quad (58)$$

⁶For all experiments in this paper, we used $\lambda = 1$; we need to ensure that the sigmapoints associated with the rotational degrees of freedom have length less than π to avoid numerical problems.

As discussed earlier, this can be viewed as a discrete-time integration of the $SE(3)$ kinematic equations, as in (25). To keep things simple, we make the following assumptions:

$$\bar{\mathbf{T}}_0 = \mathbf{1}, \quad \xi_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{0}) \quad (59)$$

$$\bar{\mathbf{T}} = \begin{bmatrix} \bar{\mathbf{C}} & \bar{\mathbf{r}} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \xi \sim \mathcal{N}(\mathbf{0}, \Sigma) \quad (60)$$

$$\bar{\mathbf{C}} = \mathbf{1}, \quad \bar{\mathbf{r}} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}, \quad \Sigma = \text{diag}(0, 0, 0, 0, 0, \sigma^2). \quad (61)$$

Although this example uses our 3-D tools, it is confined to a plane for the purpose of illustration and ease of plotting; it corresponds to a rigid body moving along the x -axis but with some uncertainty only on the rotational velocity about the z -axis. This could model a unicycle robot driving in the plane with a constant translational speed and slightly uncertain rotational speed (centered about zero). We are interested in how the covariance matrix fills in over time.

According to the second-order scheme, we have

$$\bar{\mathbf{T}}_K = \begin{bmatrix} 1 & 0 & 0 & Kr \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (62)$$

$$\Sigma_K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{K(K-1)(2K-1)}{6} r^2 \sigma^2 & 0 & 0 & 0 & -\frac{K(K-1)}{2} r \sigma^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{K(K-1)}{2} r \sigma^2 & 0 & 0 & 0 & K \sigma^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (63)$$

where we see that the top-left entry of Σ_K , corresponding to uncertainty in the x -direction, does not have any growth of uncertainty. However, in the fourth-order scheme, the fill-in pattern is such that the top-left entry is nonzero. This happens for several reasons, but mainly through the $\mathbf{B}_{\rho\rho}$ submatrix of $\mathbf{B}$. This leaking of uncertainty into an additional degree of freedom cannot be captured by keeping only the second-order terms. Fig. 2 provides a numerical example of this effect. It shows that both the second- and fourth-order schemes do a good job of representing the ‘banana’-like distribution over poses, as discussed by Long *et al.* [31]. However, the fourth-order scheme has some finite uncertainty in the straight-ahead direction (as do the sampled trajectories), while the second-order scheme does not.

D. Compound Experiment

To quantitatively evaluate the pose-compounding techniques, we ran a second numerical experiment in which we compounded two poses including their associated covariance matrices

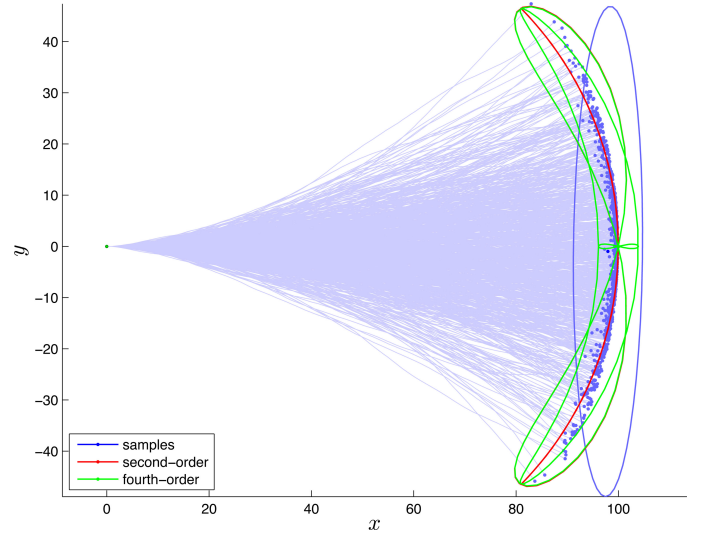


Fig. 2. Example of compounding $K = 100$ uncertain transformations (see Section III-C). The light blue lines and blue dots show 1000 individual sampled trajectories starting from $(0, 0)$ and moving nominally to the right at constant translational speed but with some uncertainty on the rotational velocity. The blue 1-sigma covariance ellipse is simply fitted to the blue samples to show what keeping xy -covariance relative to the start looks like. The red (second order) and green (fourth order) lines are the principal great circles of the 1-sigma covariance ellipsoid, given by Σ_K , mapped to the xy plane. Looking at the area $(95, 0)$, corresponding to straight ahead, the fourth-order scheme has some nonzero uncertainty (as do the samples), whereas the second-order scheme does not. We used $r = 1$ and $\sigma = 0.03$.

$$\bar{\mathbf{T}}_1 := \exp(\bar{\xi}_1^\wedge), \quad \bar{\xi}_1 := [0 \quad 2 \quad 0 \quad \pi/6 \quad 0 \quad 0]^T$$

$$\Sigma_1 := \alpha \times \text{diag} \left\{ 10, 5, 5, \frac{1}{2}, 1, \frac{1}{2} \right\}$$

$$\bar{\mathbf{T}}_2 := \exp(\bar{\xi}_2^\wedge), \quad \bar{\xi}_2 := [0 \quad 0 \quad 1 \quad 0 \quad \pi/4 \quad 0]^T$$

$$\Sigma_2 := \alpha \times \text{diag} \left\{ 5, 10, 5, \frac{1}{2}, \frac{1}{2}, 1 \right\}$$

where $\alpha \in [0, 1]$ is a scaling parameter that increases the magnitude of the input covariances parametrically.

We compounded these two poses according to (36), which results in a mean of $\bar{\mathbf{T}} = \bar{\mathbf{T}}_1 \bar{\mathbf{T}}_2$. The covariance Σ was computed using four methods.

- 1) *Monte Carlo*: We drew a large number, $M = 1\,000\,000$, of random samples $(\xi_{m_1}$ and $\xi_{m_2})$ from the input covariance matrices, compounded the resulting transformations, and computed the covariance as $\Sigma_{mc} = \frac{1}{M} \sum_{m=1}^M \xi_m \xi_m^T$ with $\mathbf{T}_m = \exp(\xi_{m_1}^\wedge) \bar{\mathbf{T}}_1 \exp(\xi_{m_2}^\wedge) \bar{\mathbf{T}}_2$, and $\xi_m = \ln(\mathbf{T}_m \bar{\mathbf{T}}^{-1})^\vee$. This slow-but-accurate approach served as our benchmark to which the other three much faster methods were compared.
- 2) *Second Order*: We used the second-order method described previously to compute Σ_{2nd} .
- 3) *Fourth Order*: We used the fourth-order method described previously to compute Σ_{4th} .
- 4) *Sigmapoint*: We used the *sigmapoint* transformation described previously [29] to compute Σ_{sp} .

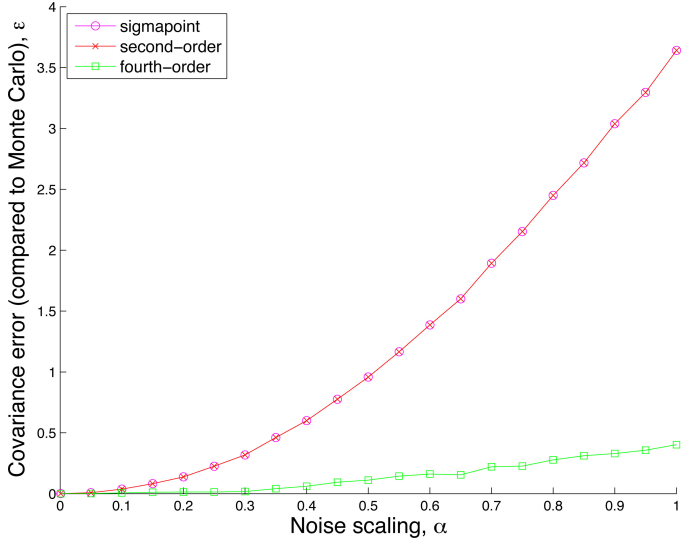


Fig. 3. Results from compound experiment (see Section III-D): Error ϵ in computing covariance associated with compounding two poses using three methods, as compared with Monte Carlo. The sigmapoint and second-order methods are algebraically equivalent for this problem and, thus, appear the same on the plot. The input covariances were gradually scaled up via the parameter α , highlighting the improved performance of the fourth-order method.

We compared each of the last three covariance matrices with the Monte Carlo one, using the *Frobenius norm*

$$\epsilon := \sqrt{\text{tr} \left((\Sigma - \Sigma_{\text{mc}})^T (\Sigma - \Sigma_{\text{mc}}) \right)}.$$

Fig. 3 shows that for small input covariance matrices (i.e., α small), there is very little difference between the various methods and the errors are all low compared with our benchmark. However, as we increase the magnitude of the input covariances, all the methods get worse, with the fourth-order method faring the best by about a factor of seven based on our error metric. Note, since α is scaling the covariance, the applied noise is increasing quadratically.

The second-order method and the sigmapoint method have indistinguishable performance, as they are algebraically equivalent. The fourth-order method goes beyond both of these by considering higher-order terms in the input covariance matrices. We did not compare the computational costs of the various methods as they are all extremely efficient, as compared with Monte Carlo.

It is also worth noting that our ability to correctly keep track of uncertainties on $SE(3)$ decreases with increasing uncertainty. This can be seen directly in Fig. 3, as error increases with increasing uncertainty. This suggests that it may be wise to use only relative pose variables in order to keep uncertainties small; this idea was suggested early on by Brooks [5] and demonstrated recently by Sibley *et al.* [32]. If uncertainty must be tracked globally on $SE(3)$, it may be necessary to move to an approach more along the lines of Lee *et al.* [24].

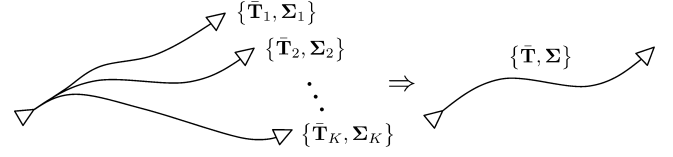


Fig. 4. Combining K pose estimates into a single fused estimate.

IV. FUSING POSES

This section will investigate a different type of nonlinearity, i.e., the fusing of several pose estimates, as depicted in Fig. 4.

A. Theory

Suppose that we have K estimates of a pose and associated uncertainties

$$\{\bar{\mathbf{T}}_1, \Sigma_1\}, \{\bar{\mathbf{T}}_2, \Sigma_2\}, \dots, \{\bar{\mathbf{T}}_K, \Sigma_K\}. \quad (64)$$

If we think of these as uncertain measurements of the true pose, $\mathbf{T}_{\text{true}}$, how can we optimally combine these into a single estimate $\{\bar{\mathbf{T}}, \Sigma\}$?

The vectorspace solution to fusion is straightforward and can be found exactly in closed form

$$\bar{\mathbf{x}} = \Sigma \sum_{k=1}^K \Sigma_k^{-1} \bar{\mathbf{x}}_k, \quad \Sigma = \left(\sum_{k=1}^K \Sigma_k^{-1} \right)^{-1}. \quad (65)$$

The situation is somewhat more complicated when dealing with $SE(3)$, and we shall resort to an iterative scheme.

We define the error (that we will seek to minimize) as $\epsilon_k \sim \mathcal{N}(\mathbf{0}, \Sigma_k)$, which occurs between the individual estimate and the optimal estimate $\bar{\mathbf{T}}^*$ so that

$$\begin{aligned} \epsilon_k &:= \ln(\bar{\mathbf{T}}^* \bar{\mathbf{T}}_k^{-1})^\vee = \ln(\exp(\xi^\wedge) \underbrace{\bar{\mathbf{T}} \bar{\mathbf{T}}_k^{-1}}_{\text{small}})^\vee \\ &= \ln(\exp(\xi^\wedge) \exp(\xi_k^\wedge))^\vee \end{aligned} \quad (66)$$

where $\bar{\mathbf{T}}$ is our best guess so far, ξ is a small (unknown but optimal) perturbation between our best guess and the optimum, and

$$\xi_k := \ln(\bar{\mathbf{T}} \bar{\mathbf{T}}_k^{-1})^\vee \quad (67)$$

is the difference between our best guess so far and each individual estimate.

Applying the version of the BCH formula in (29), we have

$$\epsilon_k \approx \xi_k + \mathcal{J}_k^{-1} \xi \quad (68)$$

correct to first order in ξ with

$$\mathcal{J}_k^{-1} := \sum_{n=0}^{\infty} \frac{B_n}{n!} (\xi_k^\wedge)^n. \quad (69)$$

Since ξ_k is fairly small, this series will converge rapidly and we can get away with keeping just a few terms. With our iterative scheme, ξ will (hopefully) converge to zero, and hence, we are justified to keep only terms linear in this quantity. Note that it is also possible to keep all the terms in the previous expansion by

utilizing the expression in (100) to compute $\mathcal{J}_k$ analytically and then invert. The ξ_k will be the residual errors after convergence.

We define the cost function that we want to minimize as

$$\begin{aligned} V &:= \frac{1}{2} \sum_{k=1}^K \epsilon_k^T \Sigma_k^{-1} \epsilon_k \\ &\approx \frac{1}{2} \sum_{k=1}^K (\xi_k + \mathcal{J}_k^{-1} \epsilon_k)^T \Sigma_k^{-1} (\xi_k + \mathcal{J}_k^{-1} \epsilon_k) \end{aligned}$$

which is already quadratic in ξ . It is in fact a *Mahalanobis distance* [33], since we have chosen the weighting matrices to be the inverse covariance matrices; thus, minimizing V with respect to ξ is equivalent to maximizing the joint likelihood of the individual estimates. It is worth noting that because we are using a constraint-sensitive perturbation scheme, we do not need to worry about enforcing any constraints on our state variables during the optimization procedure. Taking the derivative with respect to ξ and setting to zero results in the following system of linear equations for the optimal value of ξ :

$$\left(\sum_{k=1}^K \mathcal{J}_k^{-T} \Sigma_k^{-1} \mathcal{J}_k^{-1} \right) \xi = - \sum_{k=1}^K \mathcal{J}_k^{-T} \Sigma_k^{-1} \epsilon_k.$$

While this may look strange compared with (65), the Jacobian terms appear because our choice of error definition is in fact nonlinear owing to presence of the matrix exponentials. We then apply this optimal perturbation to our best guess so far

$$\bar{\mathbf{T}} \leftarrow \exp(\xi^\wedge) \bar{\mathbf{T}}$$

which ensures $\bar{\mathbf{T}}$ remains in $SE(3)$, and iterate to convergence. This optimization strategy was also used by Strasdat *et al.* [34], who argued that it avoids singularities in the representation of the mean, $\bar{\mathbf{T}}$, but retains a minimal parameterization during the optimization step ξ . At the last iteration, we take

$$\Sigma = \left(\sum_{k=1}^K \mathcal{J}_k^{-T} \Sigma_k^{-1} \mathcal{J}_k^{-1} \right)^{-1}$$

for the covariance matrix. This approach has the form of a Gauss–Newton method.

This fusion problem is similar to the one investigated by Smith *et al.* [14], but they only discuss the $K = 2$ case. Our study is closer to that of Long *et al.* [31], who discuss the $N = 2$ case and derive closed-form expressions for the fused mean and covariance for an arbitrary number of individual measurements, K ; however, they do not iterate their solution and they are tracking a slightly different PDF, as discussed earlier. Wolfe *et al.* [27] also discuss fusion at length, albeit again using a slightly different PDF than us. They discuss noniterative methods of fusion for arbitrary K and show numerical results for $K = 2$. We believe our approach generalizes all of these previous works by 1) allowing the number of individual estimates K to be arbitrary, 2) keeping an arbitrary number of terms in the approximation of the inverse Jacobian N , and 3) iterating to convergence via a Gauss–Newton style optimization method. Our approach may also be simpler to implement than some of these previous methods.

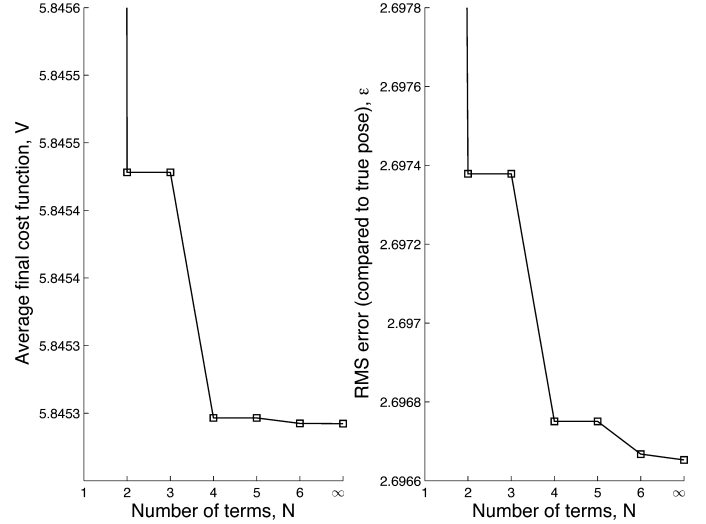


Fig. 5. Results from the fusion experiment (see Section IV-B). (Left) Average final cost function V as a function of the number of terms N , kept in $\mathcal{J}_k^{-1}$. (Right) Same for the root-mean-squared pose error with respect to the true pose. Both plots show there is benefit in keeping more than one term in $\mathcal{J}_k^{-1}$. The datapoint that is denoted by ‘ ∞ ’ uses the analytical expression in (100) to keep all the terms in the expansion.

B. Fusion Experiment

To validate the pose fusion method from the previous section, we used a true pose given by

$$\mathbf{T}_{\text{true}} := \exp(\xi_{\text{true}}^\wedge), \quad \xi_{\text{true}} := [1 \ 0 \ 0 \ 0 \ 0 \ \pi/6]^T$$

and then generated three random pose measurements

$$\begin{aligned} \bar{\mathbf{T}}_1 &:= \exp(\xi_1^\wedge) \mathbf{T}_{\text{true}}, & \bar{\mathbf{T}}_2 &:= \exp(\xi_2^\wedge) \mathbf{T}_{\text{true}} \\ \bar{\mathbf{T}}_3 &:= \exp(\xi_3^\wedge) \mathbf{T}_{\text{true}} \end{aligned} \quad (70)$$

where

$$\begin{aligned} \xi_1 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{10, 5, 5, \frac{1}{2}, 1, \frac{1}{2}\right\}\right) \\ \xi_2 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{5, 15, 5, \frac{1}{2}, \frac{1}{2}, 1\right\}\right) \\ \xi_3 &\sim \mathcal{N}\left(\mathbf{0}, \text{diag}\left\{5, 5, 25, 1, \frac{1}{2}, \frac{1}{2}\right\}\right). \end{aligned} \quad (71)$$

We then solved for the pose using our Gauss–Newton technique (iterating until convergence), using the initial condition $\bar{\mathbf{T}} = \mathbf{1}$. We repeated this for $N = 1 \dots 6$, which is the number of terms kept in $\mathcal{J}_k^{-1}$ in (103), shown later. We also used the expression given in (100), shown later, to compute $\mathcal{J}_k$ analytically (and then inverted numerically) and this is denoted by ‘ $N = \infty$ ’.

Fig. 5 plots two performance metrics. First, it plots the final converged value of the cost function, V_m , averaged over $M = 1000$ random trials $V := \frac{1}{M} \sum_{m=1}^M V_m$. Second, it plots the root-mean-squared pose error (with respect to the true pose) of our estimate $\bar{\mathbf{T}}_m$, again averaged over the same M random trials

$$\varepsilon := \sqrt{\frac{1}{M} \sum_{m=1}^M \varepsilon_m^T \varepsilon_m}, \quad \varepsilon_m := \ln(\mathbf{T}_{\text{true}} \bar{\mathbf{T}}_m^{-1})^\vee.$$

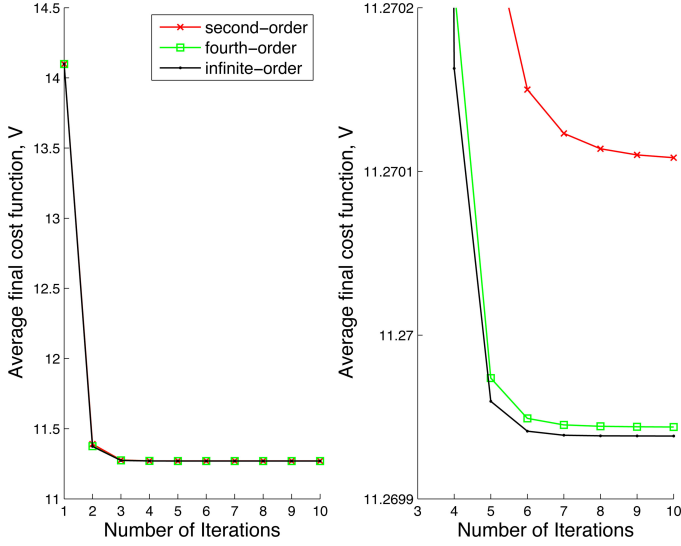


Fig. 6. Results from the fusion experiment. (Left) Convergence of the cost function V with successive Gauss-Newton iterations. This is for just one of the M trials that is used to generate Fig. 5. (Right) Same as left but zoomed in to show that the $N = 2, 4, \infty$ solutions converge to progressively lower costs.

The plots show that both measures of error are monotonically reduced with increasing N . Moreover, we see that for this example, almost all of the benefit is gained with just four terms (or possibly even two). The results for $N = 2, 3$ are identical as are those for $N = 4, 5$. This is because in the Bernoulli number sequence, $B_3 = 0$ and $B_5 = 0$; therefore, these terms make no additional contribution to $\mathcal{J}_k^{-1}$ in (103). It is also worth stating that if we make the rotational part of the covariances in (71) any bigger, we end up with a lot of samples that have rotated by more than angle π , and this can be problematic for the performance metrics we are using.

Fig. 6 shows the convergence history of the cost, V , for a single random trial. The left side shows the strong benefit of iterating over the solution, while the right side shows that the cost converges to a lower value by keeping more terms in the approximation of $\mathcal{J}_k^{-1}$ (cases $N = 2, 4, \infty$ shown). It would seem that taking $N = 4$ for about seven iterations gains most of the benefit, for this example.

V. PROPAGATING UNCERTAINTY THROUGH A NONLINEAR CAMERA MODEL

In estimation problems, we are often faced with passing uncertain quantities through nonlinear measurement models to produce expected measurements. Typically, this is carried out via linearization [35]. Sibley [36] shows how to carry out a second-order propagation for a stereo camera model accounting for landmark uncertainty but not pose uncertainty. Here, we derive the full second-order expression for the mean (and covariance) and compare this with Monte Carlo, the sigmapoint transformation, and linearization. We begin by discussing our representation of points and then present the Taylor-series expansion of the measurement (camera) model followed by an experiment.

A. Homogeneous Points

Points in $\mathbb{R}^3$ can be represented using 4×1 homogeneous coordinates [3] as follows:

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ sz \\ s \end{bmatrix} = \begin{bmatrix} \epsilon \\ \eta \end{bmatrix}$$

where s is some real, non-negative scalar, $\epsilon \in \mathbb{R}^3$, and η is scalar. When s is zero, it is not possible to convert back to $\mathbb{R}^3$, as this case represents points that are infinitely far away. Thus, homogeneous coordinates can be used to describe near and distant landmarks with no singularities or scaling issues [37]. They are also a natural representation in that points may then be transformed from one frame to another very easily using transformation matrices (e.g., $\mathbf{p}_2 = \mathbf{T}_{21} \mathbf{p}_1$).

We will also make use of the following two operators⁷ to manipulate 4×1 columns

$$\begin{bmatrix} \epsilon \\ \eta \end{bmatrix}^{\odot} := \begin{bmatrix} \eta \mathbf{1} & -\epsilon^{\wedge} \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}, \quad \begin{bmatrix} \epsilon \\ \eta \end{bmatrix}^{\ominus} := \begin{bmatrix} \mathbf{0} & \epsilon \\ -\epsilon^{\wedge} & \mathbf{0} \end{bmatrix} \quad (72)$$

which result in a 4×6 and 6×4 , respectively. With these definitions, we have the following useful identities:

$$\xi^{\wedge} \mathbf{p} \equiv \mathbf{p}^{\odot} \xi, \quad \mathbf{p}^T \xi^{\wedge} \equiv \xi^T \mathbf{p}^{\ominus} \quad (73)$$

where $\xi \in \mathbb{R}^6$ and $\mathbf{p} \in \mathbb{R}^4$, which will prove useful when manipulating expressions involving points and poses together. We also have the identity

$$(\mathbf{T}\mathbf{p})^{\odot} \equiv \mathbf{T}\mathbf{p}^{\odot} \mathbf{T}^{-1} \quad (74)$$

which is similar to (108) and (109), shown later. To the best of our knowledge, these operators and identities have not previously appeared in the literature.

To perturb points in homogeneous coordinates, we will operate directly on the xyz components by writing

$$\mathbf{p} = \bar{\mathbf{p}} + \mathbf{D} \zeta \quad (75)$$

where $\zeta \in \mathbb{R}^3$ is the perturbation and $\mathbf{D}$ is a dilation matrix given by

$$\mathbf{D} := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \quad (76)$$

We thus have that $E[\mathbf{p}] = \bar{\mathbf{p}}$ and that

$$E[(\mathbf{p} - \bar{\mathbf{p}})(\mathbf{p} - \bar{\mathbf{p}})^T] = \mathbf{D} E[\zeta \zeta^T] \mathbf{D}^T \quad (77)$$

with no approximation.

B. Taylor-Series Expansion of a Camera Model

It is common to linearize a nonlinear observation model for use in pose estimation. In this section, we show how to do a

⁷The $\odot$ operator for 4×1 columns is similar to the $\boxtimes$ operator defined by Furgale [17], which did not have the negative sign.

more general Taylor-series expansion of such a model and work out the second-order case in detail. Our camera model will be

$$\mathbf{y} = \mathbf{f}(\mathbf{T}\mathbf{p}) \quad (78)$$

where $\mathbf{T}$ is the pose of the camera, and $\mathbf{p}$ is the position of a landmark (as a homogeneous point). The nonlinear function $\mathbf{f}(\cdot)$ maps homogeneous points in the camera frame $\mathbf{T}\mathbf{p}$ to measurements $\mathbf{y}$. Our task will be to pass a Gaussian representation of the pose and landmark, given by $\{\mathbf{T}, \mathbf{p}, \Xi\}$, where Ξ is a 9×9 covariance for both quantities, through the camera model to produce a mean and covariance for the measurement $\{\mathbf{y}, \mathbf{R}\}$.

We can think of this as the composition of two nonlinearities, one to transfer the landmark into the camera frame and one to produce the observations. We will treat each one in turn. If we change the pose of the camera and/or the position of the landmark a little bit, we have

$$\begin{aligned} \mathbf{T}\mathbf{p} &= \exp(\xi^\wedge) \bar{\mathbf{T}} (\bar{\mathbf{p}} + \mathbf{D}\zeta) \\ &\approx \left(\mathbf{1} + \xi^\wedge + \frac{1}{2} \xi^\wedge \xi^\wedge \right) \bar{\mathbf{T}} (\bar{\mathbf{p}} + \mathbf{D}\zeta) \end{aligned} \quad (79)$$

where we have kept the first two terms in the Taylor series for the pose perturbation. If we multiply out and continue to keep only those terms that are second order or lower in ξ and ζ , we have

$$\mathbf{T}\mathbf{p} \approx \mathbf{g} + \mathbf{G}\theta + \frac{1}{2} \sum_{i=1}^4 \underbrace{\theta^T \mathcal{G}_i \theta}_{\text{scalar}} \mathbf{1}_i \quad (80)$$

where $\mathbf{1}_i$ is the i th column of the 4×4 identity matrix, and

$$\mathbf{g} := \bar{\mathbf{T}}\bar{\mathbf{p}} \quad (81)$$

$$\mathbf{G} := [(\bar{\mathbf{T}}\bar{\mathbf{p}})^\odot \quad \bar{\mathbf{T}}\mathbf{D}] \quad (82)$$

$$\mathcal{G}_i := \begin{bmatrix} \mathbf{1}_i^\odot (\bar{\mathbf{T}}\bar{\mathbf{p}})^\odot & \mathbf{1}_i^\odot \bar{\mathbf{T}}\mathbf{D} \\ (\mathbf{1}_i^\odot \bar{\mathbf{T}}\mathbf{D})^T & \mathbf{0} \end{bmatrix} \quad (83)$$

$$\theta := \begin{bmatrix} \xi \\ \zeta \end{bmatrix}. \quad (84)$$

Arriving at these expressions requires repeated application of the identities in the previous section.

To then apply the nonlinear camera model, we use the chain rule (for first and second derivatives) so that

$$\mathbf{f}(\mathbf{T}\mathbf{p}) \approx \mathbf{h} + \mathbf{H}\theta + \frac{1}{2} \sum_{j=1}^J \underbrace{\theta^T \mathcal{H}_j \theta}_{\text{scalar}} \mathbf{1}_j \quad (85)$$

correct to second order in θ , where

$$\mathbf{h} := \mathbf{f}(\mathbf{g}) \quad (86)$$

$$\mathbf{H} := \mathbf{F}\mathbf{G}, \quad \mathbf{F} := \left. \frac{\partial \mathbf{f}}{\partial \mathbf{g}} \right|_{\mathbf{g}} \quad (87)$$

$$\mathcal{H}_j := \mathbf{G}^T \mathcal{F}_j \mathbf{G} + \sum_{i=1}^4 \underbrace{\mathbf{1}_j^T \mathbf{F} \mathbf{1}_i}_{\text{scalar}} \mathcal{G}_i \quad (88)$$

$$\mathcal{F}_j := \left. \frac{\partial^2 f_j}{\partial \mathbf{g} \partial \mathbf{g}^T} \right|_{\mathbf{g}} \quad (89)$$

j is an index over the J rows of $\mathbf{f}(\cdot)$, and $\mathbf{1}_j$ is the j th column of the $J \times J$ identity matrix. The Jacobian of $\mathbf{f}(\cdot)$ is $\mathbf{F}$, and the Hessian of the j th row $f_j(\cdot)$ is $\mathcal{F}_j$.

If we only care about the first-order perturbation, we simply have

$$\mathbf{f}(\mathbf{T}\mathbf{p}) = \mathbf{h} + \mathbf{H}\theta \quad (90)$$

where $\mathbf{h}$ and $\mathbf{H}$ are unchanged.

These perturbed measurement equations can then be used within any estimation scheme we like; in the next section, we will use these with a stereo camera model to show the benefit of the second-order terms.

C. Propagating Gaussian Uncertainty Through the Camera

Suppose that the input uncertainties, embodied by θ , are zero-mean, Gaussian

$$\theta \sim \mathcal{N}(\mathbf{0}, \Xi) \quad (91)$$

where we note that in general there could be correlations between the pose $\mathbf{T}$ and the landmark $\mathbf{p}$.

Then, to first order, our measurement is given by

$$\mathbf{y}_{1st} = \mathbf{h} + \mathbf{H}\theta \quad (92)$$

and $\bar{\mathbf{y}}_{1st} = E[\mathbf{y}_{1st}] = \mathbf{h}$, since $E[\theta] = \mathbf{0}$ by assumption. The (second-order) covariance associated with the first-order camera model is given by

$$\mathbf{R}_{2nd} = E[(\mathbf{y}_{1st} - \bar{\mathbf{y}}_{1st})(\mathbf{y}_{1st} - \bar{\mathbf{y}}_{1st})^T] = \mathbf{H}\Xi\mathbf{H}^T. \quad (93)$$

For the second-order camera model, we have

$$\mathbf{y}_{2nd} = \mathbf{h} + \mathbf{H}\theta + \frac{1}{2} \sum_{j=1}^J \theta^T \mathcal{H}_j \theta \mathbf{1}_j \quad (94)$$

and consequently

$$\bar{\mathbf{y}}_{2nd} = E[\mathbf{y}_{2nd}] = \mathbf{h} + \frac{1}{2} \sum_{j=1}^J \text{tr}(\mathcal{H}_j \Xi) \mathbf{1}_j \quad (95)$$

which has an extra nonzero term as compared with the first-order camera model. The larger the input covariance Ξ is, the larger this term can become, depending on the nonlinearity. For a linear camera model, $\mathcal{H}_j = \mathbf{0}$ and the second- and first-order camera model means are identical.

We will also compute a (fourth-order) covariance, but with just second-order terms in the camera model expansion. To do this properly, we should expand the camera model to third order as there is an additional fourth-order covariance term involving the product of first- and third-order camera-model terms; however, this would involve a complicated expression employing the third derivative of the camera model. As such, the approximate fourth-order covariance we will use is given by

$$\begin{aligned} \mathbf{R}_{4th} &\approx E[(\mathbf{y}_{2nd} - \bar{\mathbf{y}}_{2nd})(\mathbf{y}_{2nd} - \bar{\mathbf{y}}_{2nd})^T] \\ &= \mathbf{H}\Xi\mathbf{H}^T - \frac{1}{4} \left(\sum_{i=1}^J \text{tr}(\mathcal{H}_i \Xi) \mathbf{1}_i \right) \left(\sum_{j=1}^J \text{tr}(\mathcal{H}_j \Xi) \mathbf{1}_j \right)^T \end{aligned}$$

$$+ \frac{1}{4} \sum_{i,j=1}^J \sum_{k,\ell,m,n=1}^9 \mathcal{H}_{ik\ell} \mathcal{H}_{jmn} (\Xi_{k\ell} \Xi_{mn} + \Xi_{km} \Xi_{\ell n} + \Xi_{kn} \Xi_{\ell m})$$

where $\mathcal{H}_{ik\ell}$ is the (k, ℓ) th element of $\mathcal{H}_i$, and $\Xi_{k\ell}$ is the (k, ℓ) th element of Ξ . The first- and third-order terms in the covariance expansion are identically zero owing to the symmetry of the Gaussian density. The last term in the above makes use of *Isserlis' theorem* for Gaussian variables

$$E[\xi_k \xi_\ell \xi_m \xi_n] = E[\xi_k \xi_\ell] E[\xi_m \xi_n] + E[\xi_k \xi_m] E[\xi_\ell \xi_n] + E[\xi_k \xi_n] E[\xi_\ell \xi_m].$$

D. Sigmapoint Method

Finally, we can also make use of the popular *sigmapoint transformation* [29] to pass uncertainty through the nonlinear camera model. As in the pose compounding problem, we tailor this to our specific type of $SE(3)$ perturbation. We begin by approximating the input Gaussian using a finite number of samples $\{\mathbf{T}_\ell, \mathbf{p}_\ell\}$

$$\mathbf{S}\mathbf{S}^T := \Xi, \quad (\text{Cholesky decomposition})$$

$$\boldsymbol{\theta}_\ell := \mathbf{0}$$

$$\boldsymbol{\theta}_\ell := \sqrt{L + \kappa} \text{col}_\ell \mathbf{S}, \quad \ell = 1 \dots L$$

$$\boldsymbol{\theta}_{\ell+L} := -\sqrt{L + \kappa} \text{col}_\ell \mathbf{S}, \quad \ell = 1 \dots L$$

$$\begin{bmatrix} \boldsymbol{\xi}_\ell \\ \boldsymbol{\zeta}_\ell \end{bmatrix} := \boldsymbol{\theta}_\ell$$

$$\mathbf{T}_\ell := \exp(\boldsymbol{\xi}_\ell^\wedge) \bar{\mathbf{T}}$$

$$\mathbf{p}_\ell := \bar{\mathbf{p}} + \mathbf{D} \boldsymbol{\zeta}_\ell$$

where κ is a user-definable constant,⁸ and $L = 9$. We then pass each of these samples through the nonlinear camera model

$$\mathbf{y}_\ell = \mathbf{f}(\mathbf{T}_\ell \mathbf{p}_\ell), \quad \ell = 0 \dots 2L.$$

These are combined to create the output mean and covariance according to

$$\begin{aligned} \bar{\mathbf{y}}_{\text{sp}} &= \frac{1}{L + \kappa} \left(\kappa \mathbf{y}_0 + \frac{1}{2} \sum_{\ell=1}^{2L} \mathbf{y}_\ell \right) \\ \mathbf{R}_{\text{sp}} &= \frac{1}{L + \kappa} \left(\kappa (\mathbf{y}_0 - \bar{\mathbf{y}}_{\text{sp}})(\mathbf{y}_0 - \bar{\mathbf{y}}_{\text{sp}})^T + \frac{1}{2} \sum_{\ell=1}^{2L} (\mathbf{y}_\ell - \bar{\mathbf{y}}_{\text{sp}})(\mathbf{y}_\ell - \bar{\mathbf{y}}_{\text{sp}})^T \right). \end{aligned}$$

The next section will provide the details for a specific nonlinear camera model $\mathbf{f}(\cdot)$ representing a stereo camera.

E. Stereo Camera Model

To demonstrate the propagation of uncertainty through a nonlinear measurement model, $\mathbf{f}(\cdot)$, we will employ a stereo camera

given by

$$\mathbf{f}(\mathbf{p}) = \mathbf{M} \frac{1}{p_3} \mathbf{p}$$

where

$$\mathbf{f} := \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}, \quad \mathbf{p} := \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}, \quad \mathbf{M} := \begin{bmatrix} f_u & 0 & c_u & f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u \frac{b}{2} \\ 0 & f_v & c_v & 0 \end{bmatrix}$$

and f_u, f_v are the horizontal, vertical focal lengths (in pixels), (c_u, c_v) is the optical center of the images (in pixels), and b is the separation between the cameras (in meters). The optical axis of the camera is along the z -, or p_3 -, direction.

The Jacobian of this measurement model is given by

$$\frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \mathbf{M} \frac{1}{p_3} \begin{bmatrix} 1 & 0 & -\frac{p_1}{p_3} & 0 \\ 0 & 1 & -\frac{p_2}{p_3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{p_4}{p_3} & 1 \end{bmatrix}$$

and the Hessian is given by

$$\begin{aligned} \frac{\partial^2 f_1}{\partial \mathbf{p} \partial \mathbf{p}^T} &= \frac{f_u}{p_3^2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & \frac{2p_1 + bp_4}{p_3} & -\frac{b}{2} \\ 0 & 0 & -\frac{b}{2} & 0 \end{bmatrix} \\ \frac{\partial^2 f_2}{\partial \mathbf{p} \partial \mathbf{p}^T} &= \frac{\partial^2 f_4}{\partial \mathbf{p} \partial \mathbf{p}^T} = \frac{f_v}{p_3^2} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & \frac{2p_2}{p_3} & 0 \\ 0 & 0 & \frac{p_3}{2} & 0 \end{bmatrix} \\ \frac{\partial^2 f_3}{\partial \mathbf{p} \partial \mathbf{p}^T} &= \frac{f_u}{p_3^2} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & \frac{2p_1 - bp_4}{p_3} & \frac{b}{2} \\ 0 & 0 & \frac{b}{2} & 0 \end{bmatrix} \end{aligned}$$

where we have shown each component separately.

F. Camera Experiment

We used the following methods to pass a Gaussian uncertainty on camera pose and landmark position through the nonlinear stereo camera model.

- 1) *Monte Carlo*: We drew a large number $M = 1000\,000$ of random samples from the input distribution, passed these through the camera model, and then computed the mean $\bar{\mathbf{y}}_{\text{mc}}$ and covariance $\mathbf{R}_{\text{mc}}$. This slow-but-accurate approach served as our benchmark to which the other three much faster methods were compared.
- 2) *First/Second Order*: We used the first-order camera model to compute $\bar{\mathbf{y}}_{1\text{st}}$ and $\mathbf{R}_{2\text{nd}}$, as described previously.

⁸For all experiments in this paper, we used $\kappa = 0$.

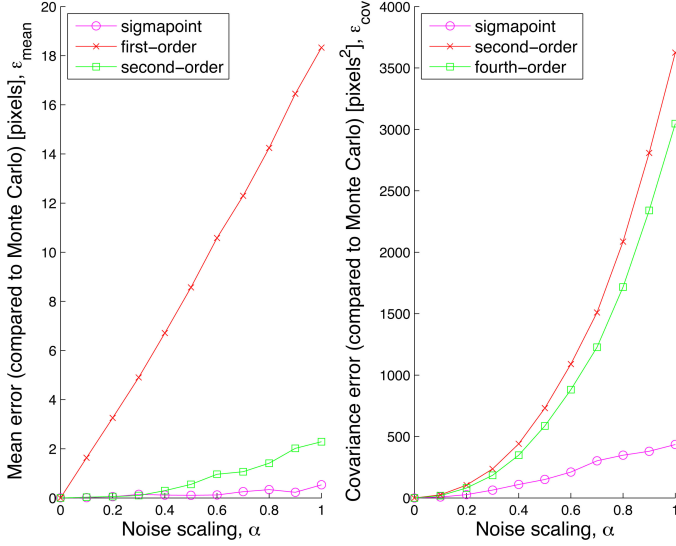


Fig. 7. Results from the stereo camera experiment (see Section V-F). (Left) Mean and (right) covariance errors $\varepsilon_{\text{mean}}$ and ε_{cov} for three methods of passing a Gaussian uncertainty through a nonlinear stereo camera model, as compared with Monte Carlo. The parameter α scales the magnitude of the input covariance matrix.

- 3) *Second/Fourth Order*: We used the second-order camera model to compute $\bar{\mathbf{y}}_{2\text{nd}}$ and $\mathbf{R}_{4\text{th}}$, as described previously.
- 4) *Sigmapoint*: We used the sigmapoint method that has been described previously to compute $\bar{\mathbf{y}}_{\text{sp}}$ and $\mathbf{R}_{\text{sp}}$.

The camera parameters were

$$b = 0.25 \text{ m}, f_u = f_v = 200 \text{ pixels}, c_u = c_v = 0 \text{ pixels}.$$

We used the camera pose $\mathbf{T} := \mathbf{1}$ and let the landmark be located at $\mathbf{p} := [10 \ 10 \ 10 \ 1]^T$. For the combined pose/landmark uncertainty, we used an input covariance of

$$\Xi := \alpha \times \text{diag} \left\{ \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{100}, \frac{1}{100}, \frac{1}{100}, 1, 1, 1 \right\}$$

where $\alpha \in [0, 1]$ is a scaling parameter that allowed us to parametrically increase the magnitude of the uncertainty.

To gauge the performance, we evaluated both the mean and covariance of each method by comparing the results to those of the Monte Carlo simulation according to the following metrics:

$$\varepsilon_{\text{mean}} := \sqrt{(\bar{\mathbf{y}} - \bar{\mathbf{y}}_{\text{mc}})^T (\bar{\mathbf{y}} - \bar{\mathbf{y}}_{\text{mc}})}$$

$$\varepsilon_{\text{cov}} := \sqrt{\text{tr}((\mathbf{R} - \mathbf{R}_{\text{mc}})^T (\mathbf{R} - \mathbf{R}_{\text{mc}}))}$$

where the latter is the *Frobenius norm*.

Fig. 7 shows the two performance metrics, $\varepsilon_{\text{mean}}$ and ε_{cov} , for each of the three techniques over a wide range of noise scalings α . We see that the sigmapoint technique does the best on both mean and covariance. The second-order technique does reasonably well on the mean, but the corresponding fourth-order technique does poorly on the covariance (because of our inability to compute a fully fourth-order-accurate covariance, as explained earlier).

Fig. 8 provides a snapshot of a portion of the left image of the stereo camera with the mean and one-standard-deviation

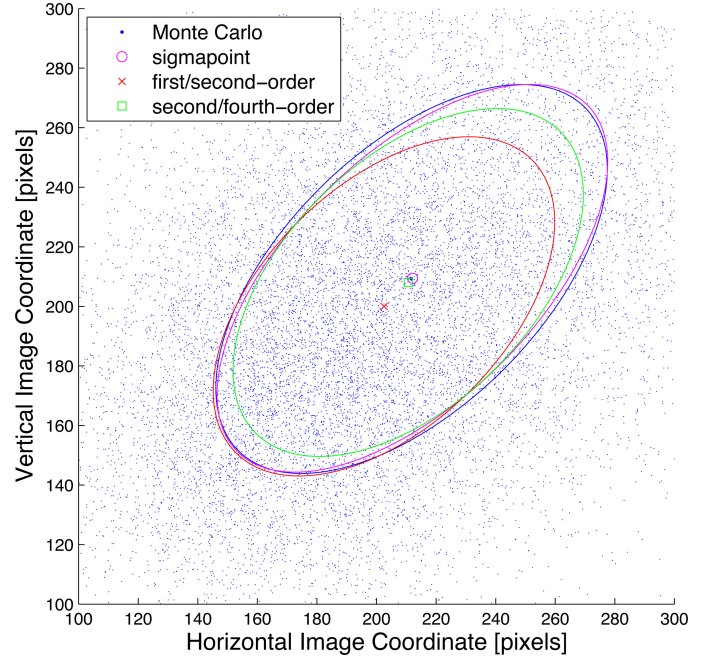


Fig. 8. Results from the stereo camera experiment (see Section V-F). A portion of the left image of a stereo camera showing the mean and covariance (as a one-standard-deviation ellipse) for four methods of imaging a landmark with Gaussian uncertainty on the camera's pose and the landmark's position. This case corresponds to the $\alpha = 1$ datapoint in Fig. 7.

covariance ellipses shown for all techniques. We see that the sigmapoint technique does an excellent job on both the mean and the covariance, while the others do not fare as well.

We do not believe there has been another comparison of these methods of propagating uncertainty through a nonlinear camera model with uncertainty on the camera pose and landmark position, the closest being the work of Sibley [36].

VI. CONCLUSION

We have presented some generic techniques (and provided accompanying MATLAB scripts⁹) to associate uncertainty with 3-D pose (and landmark) variables and shown how to use these in three worked examples:

- 1) compounding two poses with associated uncertainties (to fourth order in our noise variables or second order in the associated covariances);
- 2) fusing multiple uncertain estimates of a pose (iteratively, to arbitrary order);
- 3) propagating uncertainty through a nonlinear camera model (to fourth order in our noise variables or second order in the associated covariances).

The general outcome is that depending on the application, different methods have advantages, with the higher-order methods always doing better than the lower ones. The choice of whether to use a sigmapoint method to propagate uncertainty depends on

⁹An accompanying set of scripts to implement many of the key equations and regenerate all of the plots in the paper is downloadable from <http://asrl.utias.utoronto.ca/code/>

the application: *no* for pose compounding but *yes* to propagate uncertainty through a nonlinear camera model.

The contributions of this paper are both the specific examples as well as the detailed notation and identities to manipulate poses and associated uncertainties. Our use of the BCH formula in these manipulations, particularly our use of (29) and the tangible connection to the $SE(3)$ Jacobian, is novel; this is important because it allows us to avoid unnecessarily introducing the concept of *Lie derivatives*, and instead, we can simply manipulate perturbations representing noise variables. We believe our comparison of the sigmapoint transformation to the Taylor-expansion methods to be new as well. Finally, the appendix contains a few previously unpublished tidbits including a closed-form expression for the $SE(3)$ Jacobian in terms of all of its constituent blocks.

We have been using these methods in several bundle-adjustment and pose-graph optimization problems for the past several years with good success. Looking forward, we believe these techniques could find application in other problems requiring detailed bookkeeping of pose uncertainties and, thus, hope others will find them useful.

APPENDIX

This Appendix contains additional notation and useful expressions for implementation of key expressions in this paper.

A. Useful Closed-Form Expressions

The matrix exponential to build a transformation can be evaluated as

$$\mathbf{T} = \exp(\xi^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\xi^\wedge)^n = \begin{bmatrix} \mathbf{C} & \mathbf{J}\rho \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (96)$$

where the rightmost expression can be computed in closed form. To do so, we use the components of $\xi = [\frac{\rho}{\phi}]$ to compute the rotation matrix $\mathbf{C} \in SO(3)$

$$\begin{aligned} \mathbf{C} &:= \exp(\phi^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\phi^\wedge)^n \\ &= \cos \phi \mathbf{1} + (1 - \cos \phi) \mathbf{a}\mathbf{a}^T + \sin \phi \mathbf{a}^\wedge \end{aligned} \quad (97)$$

where $\phi = \|\phi\|$ is the angle of rotation, and $\mathbf{a} = \phi/\phi$ is the axis of rotation. We also compute

$$\begin{aligned} \mathbf{J} &:= \int_0^1 \mathbf{C}^\alpha d\alpha = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\phi^\wedge)^n \\ &= \frac{\sin \phi}{\phi} \mathbf{1} + \left(1 - \frac{\sin \phi}{\phi}\right) \mathbf{a}\mathbf{a}^T + \frac{1 - \cos \phi}{\phi} \mathbf{a}^\wedge \end{aligned} \quad (98)$$

which is the (left) *Jacobian* of $SO(3)$. The inverse of $\mathbf{C}$ is simply $\mathbf{C}^{-1} = \mathbf{C}^T$, and the inverse of $\mathbf{J}$ is given by

$$\begin{aligned} \mathbf{J}^{-1} &= \sum_{n=0}^{\infty} \frac{B_n}{n!} (\phi^\wedge)^n \\ &= \frac{\phi}{2} \cot \frac{\phi}{2} \mathbf{1} + \left(1 - \frac{\phi}{2} \cot \frac{\phi}{2}\right) \mathbf{a}\mathbf{a}^T - \frac{\phi}{2} \mathbf{a}^\wedge. \end{aligned} \quad (99)$$

Similar closed-form expressions can be found in [21] and [38].

We can also compute the Jacobian for $SE(3)$ according to

$$\mathcal{J} := \int_0^1 \mathcal{T}^\alpha d\alpha \equiv \sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\xi^\wedge)^n \equiv \begin{bmatrix} \mathbf{J} & \mathbf{Q} \\ \mathbf{0} & \mathbf{J} \end{bmatrix} \quad (100)$$

where

$$\mathcal{T} = \text{Ad}(\mathbf{T}) = \exp(\text{ad}(\xi^\wedge)) = \exp(\xi^\wedge) = \begin{bmatrix} \mathbf{C} & (\mathbf{J}\rho)^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix}. \quad (101)$$

The closed-form expression on the right can be populated using $\mathbf{J}$ above and

$$\begin{aligned} \mathbf{Q} &= \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{(n+m+2)!} (\phi^\wedge)^n \rho^\wedge (\phi^\wedge)^m \\ &= \frac{1}{2} \rho^\wedge + \frac{\phi - \sin \phi}{\phi^3} (\phi^\wedge \rho^\wedge + \rho^\wedge \phi^\wedge + \phi^\wedge \rho^\wedge \phi^\wedge) \\ &\quad - \frac{1 - \frac{\phi^2}{2} - \cos \phi}{\phi^4} (\phi^\wedge \phi^\wedge \rho^\wedge + \rho^\wedge \phi^\wedge \phi^\wedge - 3\phi^\wedge \rho^\wedge \phi^\wedge) \\ &\quad - \frac{1}{2} \left(\frac{1 - \frac{\phi^2}{2} - \cos \phi}{\phi^4} - 3 \frac{\phi - \sin \phi - \frac{\phi^3}{6}}{\phi^5} \right) \\ &\quad \times (\phi^\wedge \rho^\wedge \phi^\wedge \phi^\wedge + \phi^\wedge \phi^\wedge \rho^\wedge \phi^\wedge). \end{aligned} \quad (102)$$

To the best of our knowledge, this last expression does not appear in the previous literature. The inverse of $\mathcal{J}$ is simply

$$\mathcal{J}^{-1} = \sum_{n=0}^{\infty} \frac{B_n}{n!} (\xi^\wedge)^n \equiv \begin{bmatrix} \mathbf{J}^{-1} & -\mathbf{J}^{-1} \mathbf{Q} \mathbf{J}^{-1} \\ \mathbf{0} & \mathbf{J}^{-1} \end{bmatrix}. \quad (103)$$

Bullo and Murray [39, eq. (2.21)] also discuss $\mathcal{J}^{-1}$, pointing out that the two diagonal blocks are indeed $\mathbf{J}^{-1}$ but do not work out the details of top-right block, which amounts to knowing the details of $\mathbf{Q}$.

B. Converting Between ξ and $\mathbf{T}$

The detailed steps to convert (in closed form) between a $\xi \in \mathbb{R}^6$ and a $\mathbf{T} \in SE(3)$ are provided as follows.

1) From ξ to $\mathbf{T}$:

- Deconstruct $\xi = [\frac{\rho}{\phi}]$ into ρ and ϕ .
- Using ϕ , compute $\mathbf{C}$ and $\mathbf{J}$ according to (97) and (98).
- Using $\mathbf{C}$, $\mathbf{J}$, and ρ , construct $\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{J}\rho \\ \mathbf{0}^T & 1 \end{bmatrix}$.

We have exactly that $\mathbf{T} = \exp(\xi^\wedge)$.

2) From $\mathbf{T}$ to ξ :

- Deconstruct $\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix}$ into $\mathbf{C}$ and $\mathbf{r}$.
- Using $\mathbf{C}$, compute ϕ . This can be done by
 - computing the rotation axis $\mathbf{a}$ as the eigenvector corresponding to a unit eigenvalue of $\mathbf{C}$ (since $\mathbf{C}\mathbf{a} = \mathbf{a}$);

- ii) solving for the rotation angle ϕ in $\text{tr}(\mathbf{C}) = 2 \cos \phi + 1$ [check for the correct sign by plugging ϕ and $\mathbf{a}$ back into (97)];
 - iii) letting $\phi = \phi_a$.
 - c) Using ϕ , compute $\mathbf{J}^{-1}$ according to (99).
 - d) Using $\mathbf{J}^{-1}$ and $\mathbf{r}$, compute $\boldsymbol{\rho} = \mathbf{J}^{-1} \mathbf{r}$.
 - e) Using $\boldsymbol{\rho}$ and ϕ , construct $\boldsymbol{\xi} = [\frac{\boldsymbol{\rho}}{\phi}]$.
- We have exactly that $\boldsymbol{\xi} = \ln(\mathbf{T})^\vee$.

C. Proof that $\text{Ad}(\exp(\boldsymbol{\xi}^\wedge)) \equiv \exp(\text{ad}(\boldsymbol{\xi}^\wedge))$

We were unable to find this direct proof in the literature and therefore provide it. Starting from the right-hand side, we have

$$\begin{aligned} \exp(\text{ad}(\boldsymbol{\xi}^\wedge)) &= \exp(\boldsymbol{\xi}^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} \begin{bmatrix} \phi^\wedge & \boldsymbol{\rho}^\wedge \\ \mathbf{0} & \phi^\wedge \end{bmatrix}^n = \begin{bmatrix} \mathbf{C} & \mathbf{K} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \end{aligned} \quad (104)$$

where $\mathbf{C}$ is the usual expression in (97), and

$$\mathbf{K} := \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{(n+m+1)!} (\phi^\wedge)^n \boldsymbol{\rho}^\wedge (\phi^\wedge)^m$$

which can be found through careful manipulation. Starting from the left-hand side, we have

$$\text{Ad}(\exp(\boldsymbol{\xi}^\wedge)) = \text{Ad}\left(\begin{bmatrix} \mathbf{C} & \mathbf{J}\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}\right) = \begin{bmatrix} \mathbf{C} & (\mathbf{J}\boldsymbol{\rho})^\wedge \mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \quad (105)$$

where $\mathbf{J}$ is given in (98). Comparing (104) and (105), what remains to be shown is that $\mathbf{K} = (\mathbf{J}\boldsymbol{\rho})^\wedge \mathbf{C}$. To see this, we use the following sequence of manipulations:

$$\begin{aligned} (\mathbf{J}\boldsymbol{\rho})^\wedge \mathbf{C} &= \left(\int_0^1 \mathbf{C}^\alpha d\alpha \boldsymbol{\rho} \right)^\wedge \mathbf{C} = \int_0^1 (\mathbf{C}^\alpha \boldsymbol{\rho})^\wedge \mathbf{C} d\alpha \\ &= \int_0^1 \mathbf{C}^\alpha \boldsymbol{\rho}^\wedge \mathbf{C}^{1-\alpha} d\alpha = \int_0^1 \exp(\alpha \phi^\wedge) \boldsymbol{\rho}^\wedge \exp((1-\alpha)\phi^\wedge) d\alpha \\ &= \int_0^1 \left(\sum_{n=0}^{\infty} \frac{1}{n!} (\alpha \phi^\wedge)^n \right) \boldsymbol{\rho}^\wedge \left(\sum_{m=0}^{\infty} \frac{1}{m!} ((1-\alpha)\phi^\wedge)^m \right) d\alpha \\ &= \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{1}{n! m!} \left(\int_0^1 \alpha^n (1-\alpha)^m d\alpha \right) (\phi^\wedge)^n \boldsymbol{\rho}^\wedge (\phi^\wedge)^m \end{aligned}$$

where we have used that $\wedge$ is linear and that $(\mathbf{C}\mathbf{u})^\wedge = \mathbf{C}\mathbf{u}^\wedge \mathbf{C}^T$. We can show that $\int_0^1 \alpha^n (1-\alpha)^m d\alpha = \frac{n! m!}{(n+m+1)!}$ (after several integrations by parts), and therefore, $\mathbf{K} \equiv (\mathbf{J}\boldsymbol{\rho})^\wedge \mathbf{C}$, concluding the proof.

D. Additional Useful Identities

Some more useful identities for $SE(3)$ and $se(3)$ are the following:

$$\boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_a \equiv \mathbf{0} \quad (106)$$

$$\boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_b \equiv -\boldsymbol{\xi}_b^\wedge \boldsymbol{\xi}_a \quad (107)$$

$$(\mathcal{T}_a \boldsymbol{\xi}_b)^\wedge \equiv \mathbf{T}_a \boldsymbol{\xi}_b^\wedge \mathbf{T}_a^{-1} \quad (108)$$

$$(\mathcal{T}_a \boldsymbol{\xi}_b)^\wedge \equiv \mathcal{T}_a \boldsymbol{\xi}_b^\wedge \mathcal{T}_a^{-1} \quad (109)$$

$$\exp((\mathcal{T}_a \boldsymbol{\xi}_b)^\wedge) \equiv \mathbf{T}_a \exp(\boldsymbol{\xi}_b^\wedge) \mathbf{T}_a^{-1} \quad (110)$$

$$\exp((\mathcal{T}_a \boldsymbol{\xi}_b)^\wedge) \equiv \mathcal{T}_a \exp(\boldsymbol{\xi}_b^\wedge) \mathcal{T}_a^{-1} \quad (111)$$

$$\text{Ad}(\boldsymbol{\xi}_b^\wedge \mathbf{T}_a) \equiv \boldsymbol{\xi}_b^\wedge \mathcal{T}_a \quad (112)$$

$$[\boldsymbol{\xi}_a^\wedge, \boldsymbol{\xi}_b^\wedge] = \boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_b^\wedge - \boldsymbol{\xi}_b^\wedge \boldsymbol{\xi}_a^\wedge \equiv (\boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_b)^\wedge \quad (113)$$

$$[\boldsymbol{\xi}_a^\wedge, \boldsymbol{\xi}_b^\wedge] = \boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_b^\wedge - \boldsymbol{\xi}_b^\wedge \boldsymbol{\xi}_a^\wedge \equiv (\boldsymbol{\xi}_a^\wedge \boldsymbol{\xi}_b)^\wedge \quad (114)$$

where $\boldsymbol{\xi}_a, \boldsymbol{\xi}_b \in \mathbb{R}^6$ and $\mathbf{T}_a = \exp(\boldsymbol{\xi}_a^\wedge)$, $\mathcal{T}_a = \exp(\boldsymbol{\xi}_a^\wedge) \in SE(3)$.

ACKNOWLEDGMENT

The authors would like to thank Prof. F. Dellaert at Georgia Tech and H. Sommer at ETH for many useful discussions during the preparation of this paper.

REFERENCES

- [1] D. C. Brown, "A solution to the general problem of multiple station analytical stereotriangulation," Patrick Air Force Base, Florida, RCA-MTP Data Reduction Tech. Rep. 43 (or AFMTC TR 58-8), 1958.
- [2] R. M. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC, 1994.
- [3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
- [4] J. Stillwell, *Naive Lie Theory*. New York, NY, USA: Springer-Verlag, 2008.
- [5] R. Brooks, "Visual map making for a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1985, vol. 2, pp. 824–829.
- [6] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robot. Res.*, vol. 5, no. 4, pp. 56–68, 1986.
- [7] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE J. Robot. Autom.*, vol. 4, no. 1, pp. 23–31, Feb. 1988.
- [8] R. C. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY, USA: Springer-Verlag, 1990, pp. 167–193.
- [9] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *IEEE Robot. Autom. Mag.*, vol. 11, no. 3, pp. 99–110, Jun. 2006.
- [10] G. S. Chirikjian, *Stochastic Models, Information Theory, and Li Groups: Classical Results and Geometric Methods*. vol. 1, New York, NY, USA: Birkhauser, 2009.
- [11] G. S. Chirikjian, *Stochastic Models, Information Theory, and Li Groups: Analytic Methods and Modern Applications*. vol. 2, New York, NY, USA: Birkhauser, 2009.
- [12] S. F. Su and C. S. G. Lee, "Uncertainty manipulation and propagation and verification of applicability of actions in assembly tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1991, vol. 3, pp. 2471–2476.
- [13] S. F. Su and C. S. G. Lee, "Manipulation and propagation of uncertainty and verification of applicability of actions in assembly tasks," *IEEE Trans. Syst., Man Cybern.*, vol. 22, no. 6, pp. 1376–1389, Nov./Dec. 1992.
- [14] P. Smith, T. Drummond, and K. Roussopoulos, "Computing MAP trajectories by representing, propagating, and combining PDFs over groups," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 1275–1282.
- [15] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inf. Fusion*, vol. 14, no. 1, pp. 57–77, 2013.
- [16] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. New York, NY, USA: Springer-Verlag, 1999.
- [17] P. T. Furgale, "Extensions to the visual odometry pipeline for the exploration of planetary surfaces" Ph.D. dissertation, Dept. Aersp. Eng., Univ. Toronto, Toronto, ON, Canada, 2011.
- [18] P. C. Hughes, *Spacecraft Attitude Dynamics*. New York, NY, USA: Wiley, 1986.
- [19] M. D. Shuster, "A survey of attitude representations," *J. Astronaut. Sci.*, vol. 41, no. 4, pp. 439–517, 1993.

- [20] H. McKean, *Stochastic Integrals*. New York, NY, USA: Academic, 1969.
- [21] F. C. Park, "The optimal kinematic design of mechanisms" Ph.D. dissertation, Harvard Univ., Cambridge, MA, USA, 1991.
- [22] G. M. T. D'Eleuterio, "Multibody dynamics for space station manipulators: Recursive dynamics of topological chains," Dynacon Enterprises Ltd., Mississauga, ON, Canada, Tech. Rep. SS-3, Jun. 1985.
- [23] G. M. T. D'Eleuterio, "Dynamics of an elastic multibody chain: Part C—Recursive dynamics," *Dyn. Stabil. Syst.*, vol. 7, no. 2, pp. 61–89, 1992.
- [24] T. Lee, M. Leok, and N. H. McClamroch, "Global symplectic uncertainty propagation on $SO(3)$," in *Proc. 47th IEEE Conf. Decision Control*, 2008, pp. 61–66.
- [25] Y. Wang and G. S. Chirikjian, "Error propagation on the Euclidean group with applications to manipulator kinematics," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 591–602, Aug. 2006.
- [26] Y. Wang and G. S. Chirikjian, "Nonparametric second-order theory of error propagation on motion groups," *Int. J. Robot. Res.*, vol. 27, no. 11, pp. 1258–1273, 2008.
- [27] K. Wolfe, M. Mashner, and G. Chirikjian, "Bayesian fusion on Lie groups," *J. Algebr. Statist.*, vol. 2, no. 1, pp. 75–97, 2011.
- [28] S. Klarsfeld and J. A. Oteo, "The Baker-Campbell-Hausdorff formula and the convergence of the Magnus expansion," *J. Phys. A: Math. Gen.*, vol. 22, pp. 4565–4572, 1989.
- [29] S. Julier and J. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Robotics Res. Group, Univ. Oxford, Oxford, U.K., Tech. Rep., 1996.
- [30] J. Brookshire and S. Teller, "Extrinsic calibration from per-sensor ego-motion," presented at the Robot.: Sci. Syst. Conf., Sydney, Australia, Jul. 2012.
- [31] A. W. Long, K. C. Wolfe, M. J. Mashner, and G. S. Chirikjian, "The banana distribution is Gaussian: A localization study with exponential coordinates," presented at the Robot.: Sci. Syst. Conf., Sydney, Australia, 2012.
- [32] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast scale outdoor navigation using adaptive relative bundle adjustment," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 958–980, Jul. 2010.
- [33] P. Mahalanobis, "On the generalized distance in statistics," in *Proc. Nat. Inst. Sci.*, 1936, vol. 2, pp. 49–55.
- [34] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular SLAM," presented at the Robot.: Sci. Syst. Conf., Zaragoza, Spain, Jun. 2010.
- [35] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Autom.*, vol. RA-3, no. 3, pp. 239–248, Jun. 1987.
- [36] G. Sibley, "Long range stereo data-fusion from moving platforms" Ph.D. dissertation, Dept. Comput. Sci., Univ. Southern California, Los Angeles, CA, USA, 2007.
- [37] W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment: A modern synthesis," in *Vision Algorithms: Theory and Practice* (LNCS series), W. Triggs, A. Zisserman, and R. Szeliski, Eds. New York, NY, USA: Springer-Verlag, 2000, pp. 298–375.
- [38] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A Lie group formulation of robot dynamics," *Int. J. Robot. Res.*, vol. 14, pp. 609–618, 1995.
- [39] F. Bullo and R. M. Murray, "Proportional derivative control on the Euclidean group," in *Proc. Eur. Control Conf.*, 1995, pp. 1091–1097.



Timothy D. Barfoot (M'10) received the Ph.D. degree from University of Toronto, Toronto, ON, Canada, in 2002.

He currently leads the Autonomous Space Robotics Lab, University of Toronto, which develops methods to allow mobile robots to operate in large-scale, unstructured, 3-D environments, using rich on-board sensing (e.g., cameras and laser rangefinders) and computation. He is also an Associate Professor with the University of Toronto Institute for Aerospace Studies, where he holds the Canada Research Chair

(Tier II) in Autonomous Space Robotics and works in the area of guidance, navigation, and control of mobile robots for space and terrestrial applications. Prior to his current position, he spent four years at MDA Space Missions, where he developed autonomous vehicle navigation technologies for both planetary rovers and terrestrial applications such as underground mining.

Dr. Barfoot is an Ontario Early Researcher Award holder and a licensed Professional Engineer in the Province of Ontario.



Paul T. Furgale (M'11) received the B.A.Sc. degree in computer science from the University of Manitoba, Winnipeg, MB, Canada, in 2006 and the Ph.D. degree from the University of Toronto Institute for Aerospace Studies, Toronto, ON, Canada, 2011.

He is currently the Deputy Director with the Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland, where he is the scientific coordinator of the European Project V-Charge. His research interests include state estimation, sensor fusion, and calibration for complicated real-world

robotic systems.