Information and Communication Technologies
Collaborative Project

"V-Charge"

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



V-CHARGE

Deliverable Number	2.4
Deliverable Name	Final Integration and evaluation of algorithms and semantic mapping concept
WP Number	2
Date of Preparation	22.07.2015
Date of Submission	31.07.2015
Editor/Beneficiary	Paz/Oxford, Sattler/ETH Zürich

Contents

1 Introduction **3**

1.1 Task Description . . . 3

1.2 Overview of the contributions . . . 3

2 Semantic Mapping **4**

2.1 The Road Graph . . . 6

2.2 The Parking Labels . . . 6

2.3 The Speed Map . . . 7

3 Evaluation of the Semantic Map Layer **7**

3.1 Lane Evaluation . . . 9

3.2 Evaluation of labels: Parking spots and charging bays . . . 13

3.3 Evaluation of the speed map . . . 13

4 Constructing Metric Maps **16**

5 Semantic Localisation Concepts **16**

6 Summary of Milestones **18**

6.1 Milestone 6, Task 2.1 . . . 18

6.2 Milestone 6, Task 2.2 . . . 18

6.3 Milestone 6, Task 2.3 . . . 19

6.4 Milestone 6, Task 2.4 . . . 19

7 Conclusion **20**

A Integrating Metric and Semantic Maps for Vision-Only Automated Parking **22**

B Introspective Classification for Robot Perception **31**

C Summary Maps for Lifelong Visual Localization **52**

D Leveraging Image-Based Localization for Infrastructure-Based Calibration of a Multi-Camera Rig **90**

1 Introduction

1.1 Task Description

The overall objective of this WP is to develop and verify novel semantic mapping and localization concepts using visual perception. In particular, WP2 has the specific goals:

- Creation of dense geometric maps
- Creation of wide baseline stable visual features for localisation
- Creation of Semantic Maps
- Creation of maps which aid motion planning

1.2 Overview of the contributions

Our efforts on this deliverable have focused on the following aspects: increasing the robustness of image-based classification and detection of semantic labels (contribution 2) and integration of two layers of map which are often required for fully automated operation: metric and semantic.

In order to address the V-Charge goals, in this deliverable we report evaluation results on different datasets from experiments conducted in three different testing scenarios with two different platforms available: The BOSCH parking House (Stuttgart, GE, Grobi platform), The VW Mobile Life Campus (MLC) parking lot (Wolfsburg, GE, Grobi platform) and the ASL car park (ETH, Zurich, CH, Kermit platform) . By following a similar idea of A, B, the classification back-end employed for this V-Charge task relies on the concept of introspection. Due to the lack of road marks in the place, our results are focused on the classification of parking spots. Nonetheless, our approach can extrapolate to most general and richer scenarios with multiple labels. In the following subsections we provide details of the semantic map pipeline shown in Fig. 1.

In addition, this deliverable also discusses the overall metric mapping pipeline, including camera calibration (appending D), the dependencies between metric maps and semantic maps as well as the use of maps captured under different viewing conditions for localization (appendix C).

Work on all tasks in WP2 has continued since deliverable D2.3. More details about the following contributions can be found in the appendix.

	Contributions	From	Link
1	Integrating Metric and Semantic Maps for Vision-Only Automated Parking, Hugo Grimmett, Mathias Buerki, Lina Maria Paz, Pedro Pinies, Paul Furgale, Ingmar Posner and Paul Newman, ICRA 2015	UOXF.DF	A
2	Introspective Classification for Robot Perception, H. Grimmett, R. Triebel, R. Paul, and I. Posner, International Journal of Robotics Research (IJRR), 2015	UOXF.DF	B
3	Summary Maps for Lifelong Visual Localization, P. Mühlfellner, M. Buerki, M. Bosse, W. Derendarz, R. Philippsen, P. Furgale, Journal of Field Robotics, 2015	ETHZ-ASL, VW	C
4	Leveraging Image-Based Localization for Infrastructure-Based Calibration of a Multi-Camera Rig, L. Heng, P. Furgale, M. Pollefeys, Journal of Field Robotics, 2015	ETHZ-CVG	D

The remainder of this report is structured as follows: Section 2 describes the semantic mapping concept, which is then evaluated in section 3. Section 4 discusses the pipeline used for creating metric and dense geometric maps. Section 5 briefly describes how localization is made robust by merging metric maps captured under different illumination conditions.

2 Semantic Mapping

The creation and maintenance procedures for a semantic map of a car park were the principle contribution of A, presented at ICRA 2015. In this paper we present a system for creating the map, split into two distinct layers: the metric and the semantic, and iteratively improving it as the car collects new data upon subsequent revisits, as shown in Figure 1.

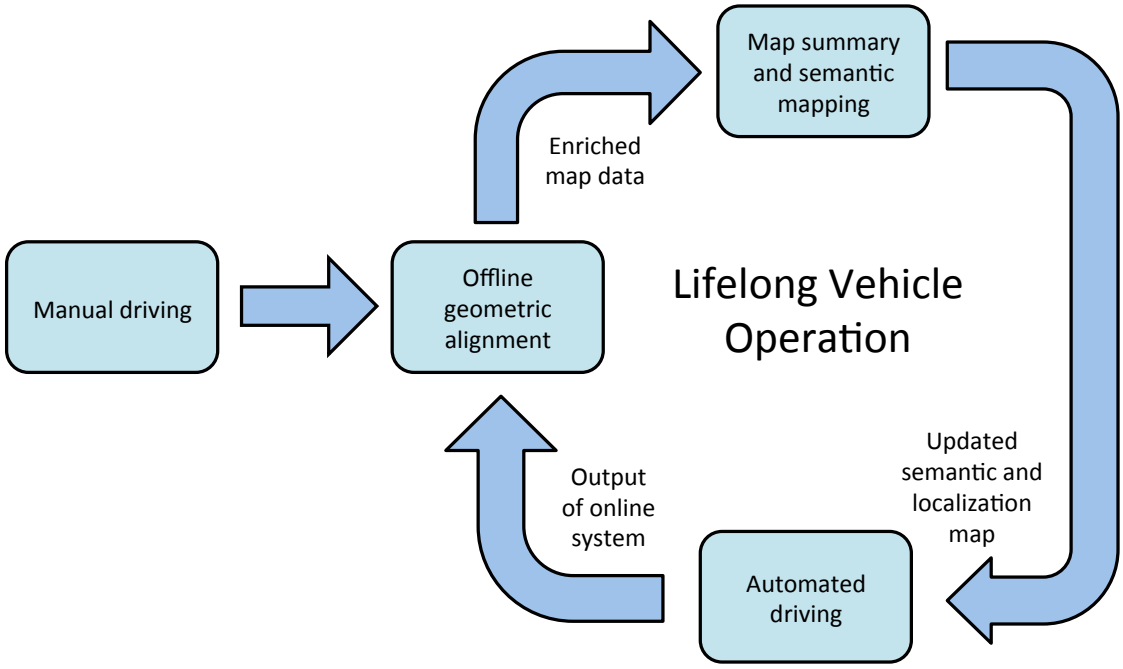
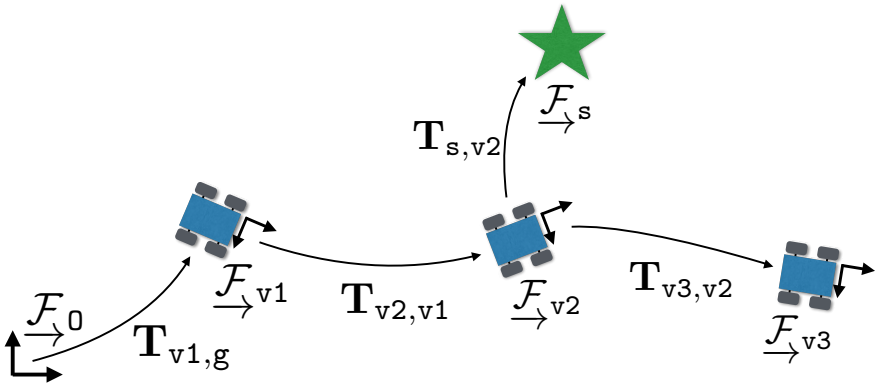
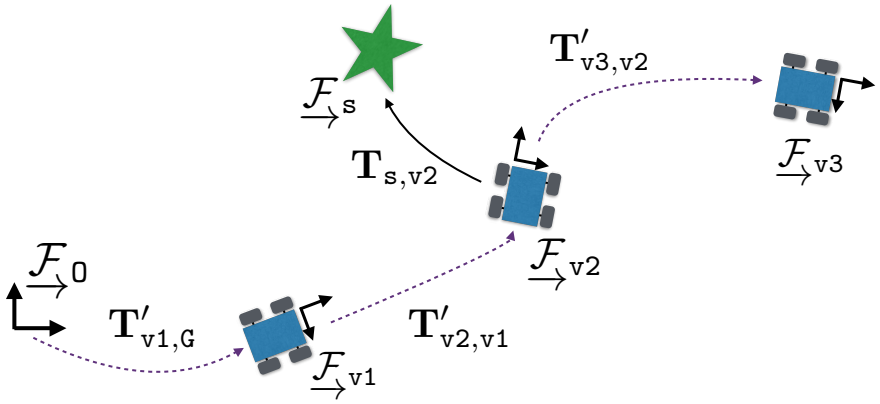


Figure 1: Here we show the cycle of improvement to the map as the vehicle revisits the car park. First, the car is driven manually around all the lanes, and then the metric and semantic layers are generated offline. Then the car is ready for automated operation, during which more data are collected, iteratively improving both layers of the map.

We describe the latest mapping procedure, whereby the metric layer is initially created by a full bundle adjustment over the landmarks, the observation constraints over those landmarks, and the wheel odometry constraints between successive vehicle poses. This part of the process is unsupervised. Over this metric layer lies the semantic layer, comprising three distinct parts: the road graph, detailing the positions of the lanes and intersections; the positions of the parking spaces; and the speed profile at which it is sensible for the vehicle to move through the car park. The most expensive step in the map creation is the labelling of the parking spaces, which requires a human in the loop. Because we anchor the semantic labels to poses in the metric layer, any updates to the metric layer allow the labels to shift and warp with the updates without having to be recreated manually, as shown in Figure 2a and Figure 2b. This allows us to update the map without having to duplicate expensive labelling procedures. Next we detail the creation of the three components of the semantic layer.



(a)



(b)

Figure 2: a) The initial visit (see caption of Figure 3 for more explanation). b) Revisiting the same place. While the vehicle positions [v1, v2, v3] may be recomputed between the initial visit and the revisit, the position of the semantic object s relative to v2 remains unchanged. This allows us to update the metric map without having to recreate the semantic labels. $\overset{F}{\rightarrow}_0$ represents the global origin coordinate frame, and $\mathtt{T}_{\mathrm{a,b}}$ represents the 6 d.o.f. transformation from b to a.

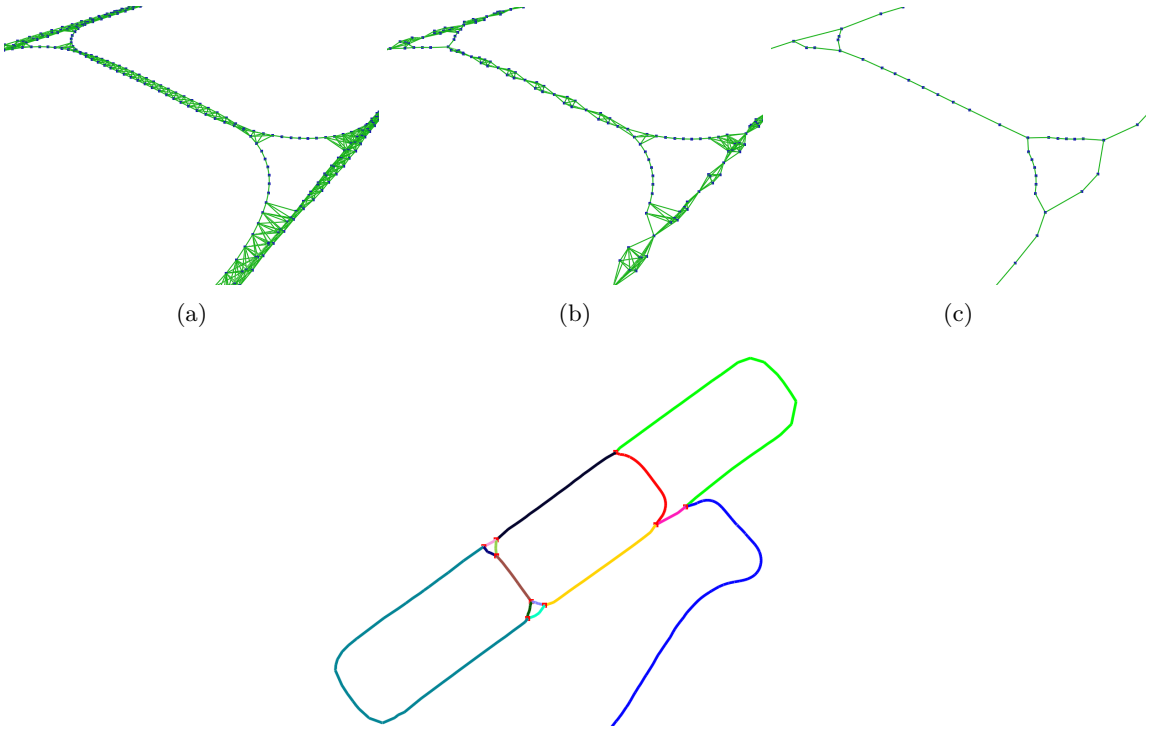


Figure 3: a) The fully connected graph. Vehicle poses become nodes in the graph, and edges are created between both nearby and subsequent nodes. b) The graph after one round of pruning. The largest cliques are replaced by the nearest pose to their centre-point. c) The finished graph after the iterative pruning procedure. d) The lanes and intersections at the end of the algorithm.

2.1 The Road Graph

The road graph requires only the vehicle poses from the metric layer, under the assumption that the car has driven every lane in the car park at least once. This sequence of poses gives a picture of the layout of the lanes in the car park, but the trajectories will overlap imperfectly as the car passes over certain lanes more than once, as shown in Figure 3a. In order to yield the underlying skeleton structure of the lanes, we impose a graph over the poses, considering each pose as a node, and creating edges between both subsequent and nearby poses, as shown in Figure 3a. By identifying the cliques in the graph and iteratively replacing each clique (starting with the largest) with its centre-point (or the pose nearest to its centre-point, to avoid drift over many iterations), we remove the clusters of poses and replace them by the underlying structure, as shown in Figure 3b. What results is a very clean graph as shown in Figure 3c. From this graph we can easily extract the intersections, and infer the driving directions of each lane from the original pose chain, as shown in Figure 3d. The algorithm is described in more detail in the paper.

2.2 The Parking Labels

Using the vehicle poses in the metric map, together with the fish-eye images from the four cameras, we create a synthetic overhead image as shown in Figure 4. Using this image we can use active learning classification

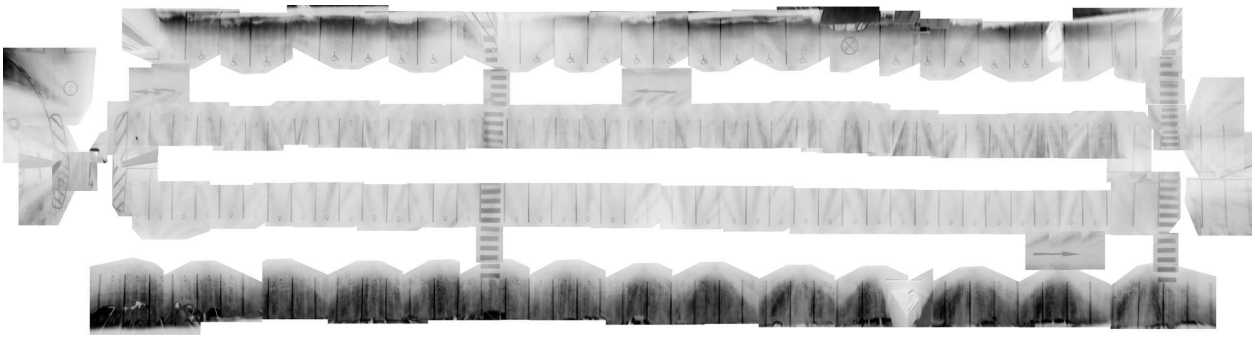


Figure 4: The synthetic overhead image for the Stuttgart car park.

with a human in the loop to quickly identify the parking spaces. Perhaps unexpectedly, parking spaces look very different from car park to car park; each of the four car parks for which we have created a semantic map has contained parking spaces with greatly differing visual characteristics. As a result of this, it has been necessary to train a new classifier per car park. In order to make this process as efficient as possible, that is, requiring the least amount of human involvement, we have used active learning. It has been shown in previous V-Charge publications [1,2] that introspective classifiers make active learning more efficient, and thus we use the Bayesian Logistic Regression (BLR), a form of the linear Gaussian process classifier, in order to do our classification. Within a few rounds of learning we have labelled all the parking spaces to the required precision, thus we can always achieve the requisite 90% as outlined in Task 2.3, as shown in Figure 5. The resulting locations of the parking spaces are shown in Figure 6.

2.3 The Speed Map

This speed map tells the vehicle how fast it should drive as it moves through the car park. In order to make the automated operation as safe as possible for pedestrians and other road users, the vehicle's speed should be lower in areas where visibility might be worse, or where people generally tend to walk. Pedestrian crossings can go some distance to inform the speed, but nothing is more effective than directly observing pedestrians move across the car park. Firstly, we create a probabilistic graphical model using a prior based on the positions of the lanes and the parking spaces, as shown in Figure 7. Using the images collected during the visits to the car park, we detect pedestrians (initially using the same active learning framework as for the parking spaces, only this time using the fish-eye images, as shown in Figure 7a) and project their locations into the prior map frame. These locations mark positive observations in the Markov Random Field imposed over the prior map, as shown in Figure 7b. A smoothing procedure over the map allows us to update the prior map to include the detected pedestrians and slow the vehicle down as it approaches heavily pedestrianised areas, as shown in Figure 8. This speed map can be continually updated with new detections as the vehicle revisits the car park.

3 Evaluation of the Semantic Map Layer

In the paper presented at ICRA 2015, we use a checkerboard to determine the position of the car for five parking manoeuvres in the base map and three subsequent improvements of the metric layer. We show that those parking manoeuvres were consistently close to the centre of the actual parking space despite the fact that the metric and hence semantic layers were moving around, as shown in Figure 9.

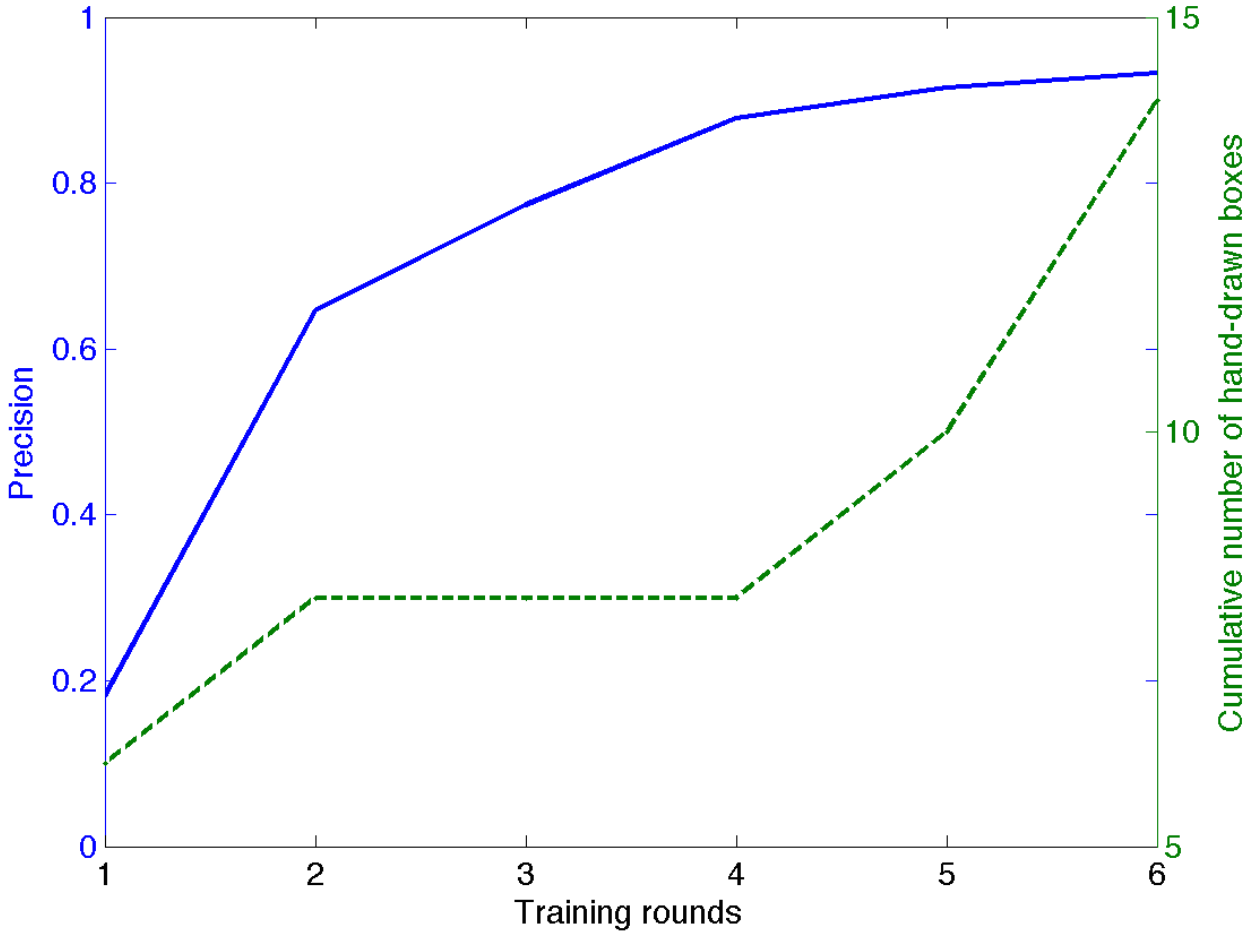


Figure 5: The increase in precision as the active learning rounds progress. We also show the number of hand-drawn boxes on the right-hand axis.

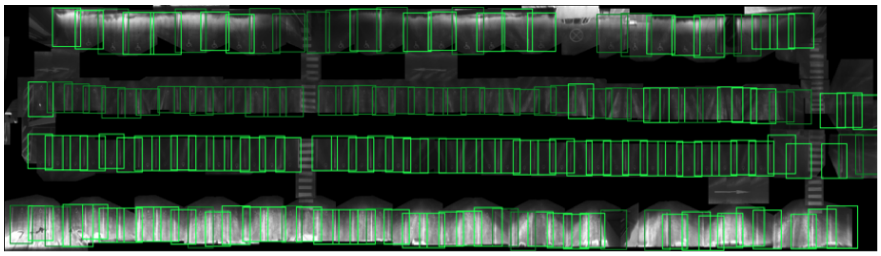
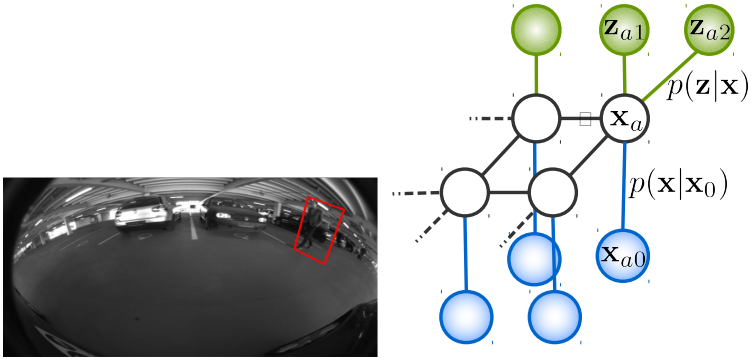


Figure 6: The resulting detections of parking spaces in the synthetic overhead image.



(a) (b)

Figure 7: The conditional random field imposed over the semantic layer to calculate the likelihood that a pedestrian will be in a particular place. The prior, in blue, is informed by the positions of the lanes and parking spaces. The green nodes are observation in from the fish-eye images. The white nodes represent the likelihood of the pedestrian in that place.

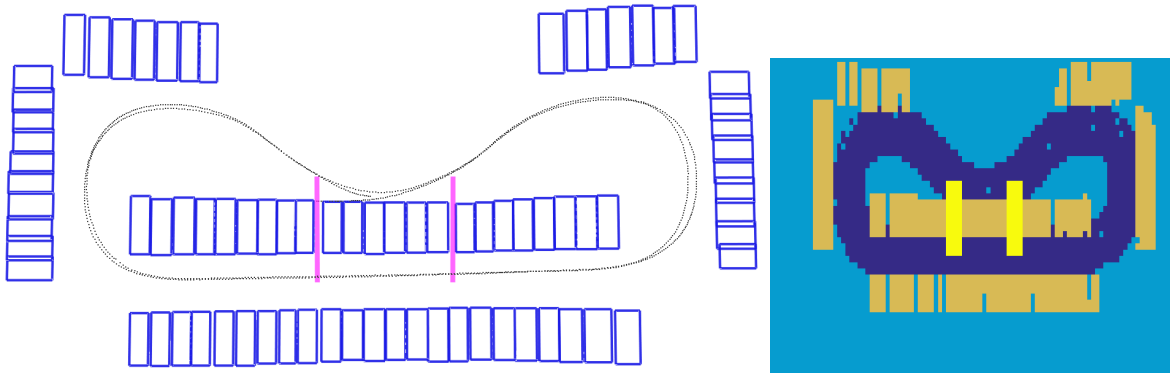
However, these accuracies incorporate many sources of error. Errors in the semantic layer of course, but also in the local planner, the visual localisation system, the camera calibration, and the evaluation with the chequerboard. We also need to evaluate the positions of the lanes and parking spaces with a system that allows us to isolate where those errors might originate.

In the following, we present an extension of the semantic map evaluation for the experiments conducted at the MLC site.

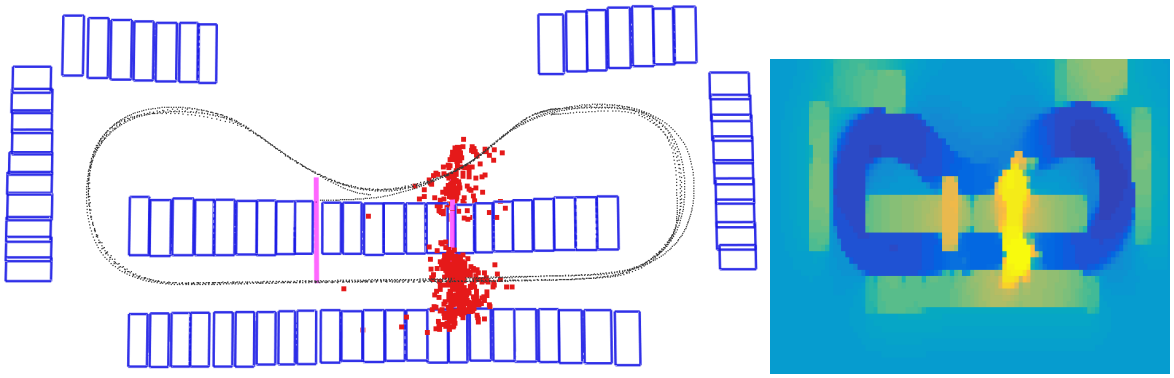
3.1 Lane Evaluation

In order to calculate the error in the road graph for the MLC site, we manually drove around all the lanes in the road graph, making sure to drive as perfectly as possible – that is, in the centre of each lane and taking each turn as smoothly as possible. For ground-truth localisation we used a differential GPS/INS system. We compare the ground-truth INS poses with the nodes in the lanes and calculate the displacements between them, as shown in Figure 10. There are three places in which the errors appear to be largest.

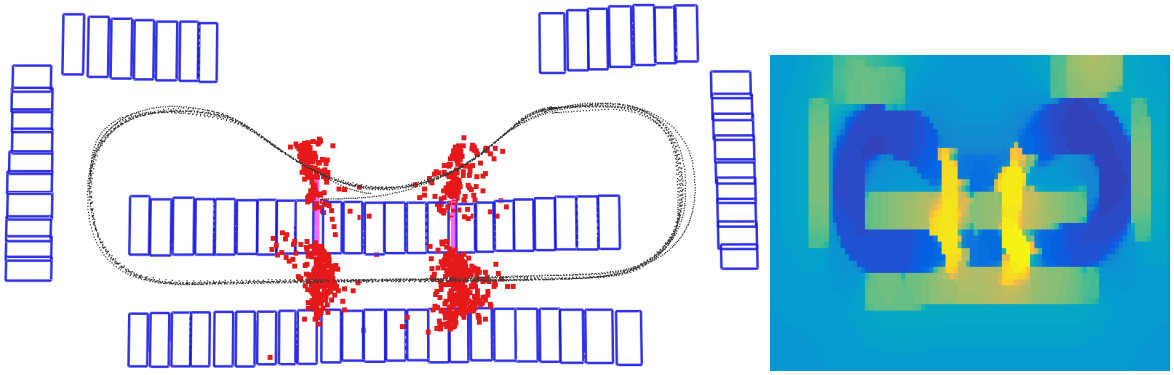
The first is in the South East of the map, and is enlarged and shown with aerial photography in Figure 11. We suggest that the driven lane (in red) passes on the wrong side of the road and too close to the parked car, and thus the road graph (in black) is a preferable trajectory, thus excusing this perceived error. The second are of error is in the South West of the map, again enlarged in Figure 12. This are is a free-driving zone, and so the consistent displacement does not represent a true deviation from a correct trajectory. The driver chose a slightly different path for the ground truth run, but both as correct as each other, again excusing this perceived error. The third area with largest error occurs in the North East corner, as shown in Figure 13. The ground-truth trajectory passes a little further from the cars parked on the right, and takes the turn into the lane on the right a little tighter than is recommended in the map. It is possible that with oncoming traffic in the lane, the road-graph lane is more appropriate, but we accept the error in the turn exists in the map itself.



(a) Initial drive.



(b) The maps from data up to and including the 3$^{\text{rd}}$ revisit.



(c) The maps from data up to and including the 6$^{\text{th}}$ revisit.

Figure 8: On the left, the red dots represent projections of the detected pedestrians into the static map. On the right, the evolution of the dynamic map as more pedestrians are detected. In the dynamic map, blue represents lower danger (higher speed) and yellow represents more danger (higher speed). The top dynamic map is calculated using only the prior over pedestrians given the static map.

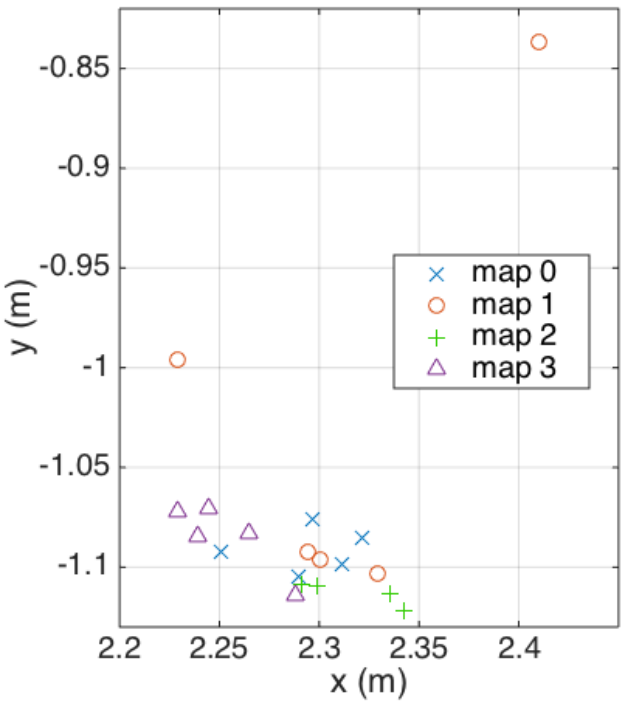


Figure 9: The consistency of autonomous parking manoeuvres in Zurich. Map 0 is the base map, and 1 2 3 are subsequent improvements to the metric map following revisits.

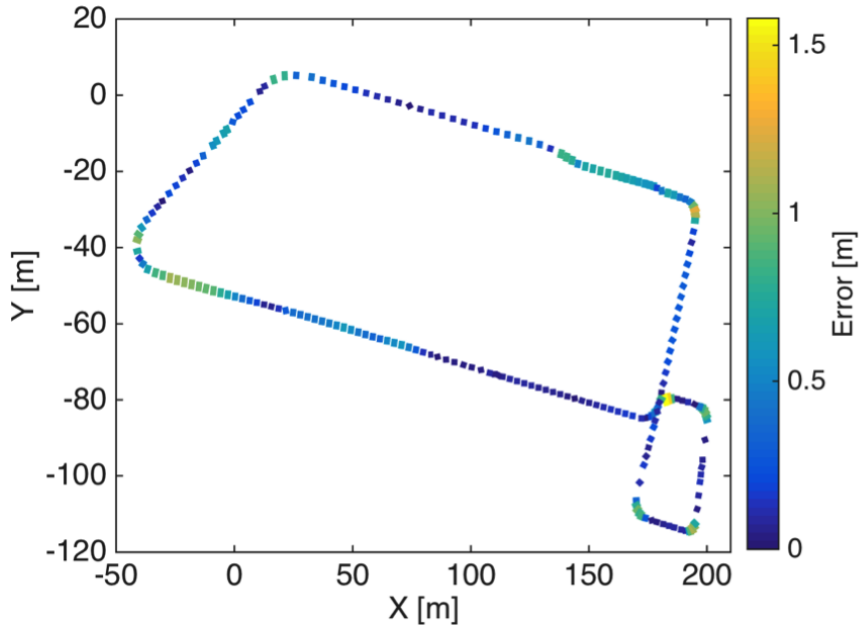


Figure 10: The displacement between the INS ground-truth manual drive, and the nodes in the road graph.

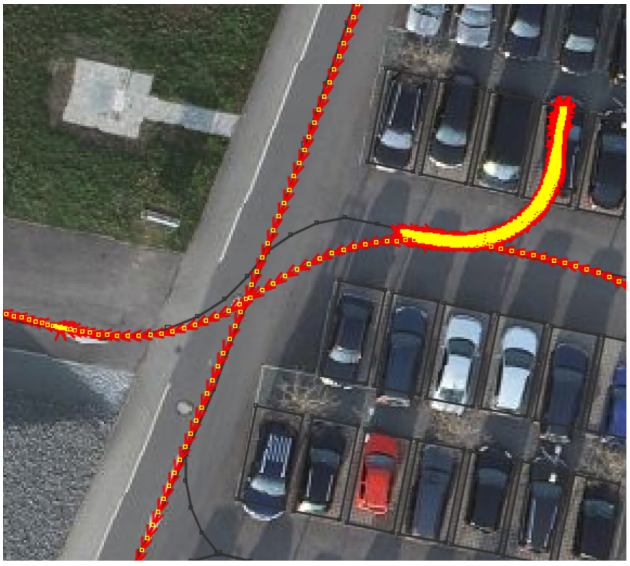


Figure 11: The source of error in South East hot-spot of the lane error map. In red we see the driven path, and in black the lane in the road graph. We suggest that in fact the driven path passes too close to the parked car and on the wrong side of the road, and thus the road graph path is preferable.

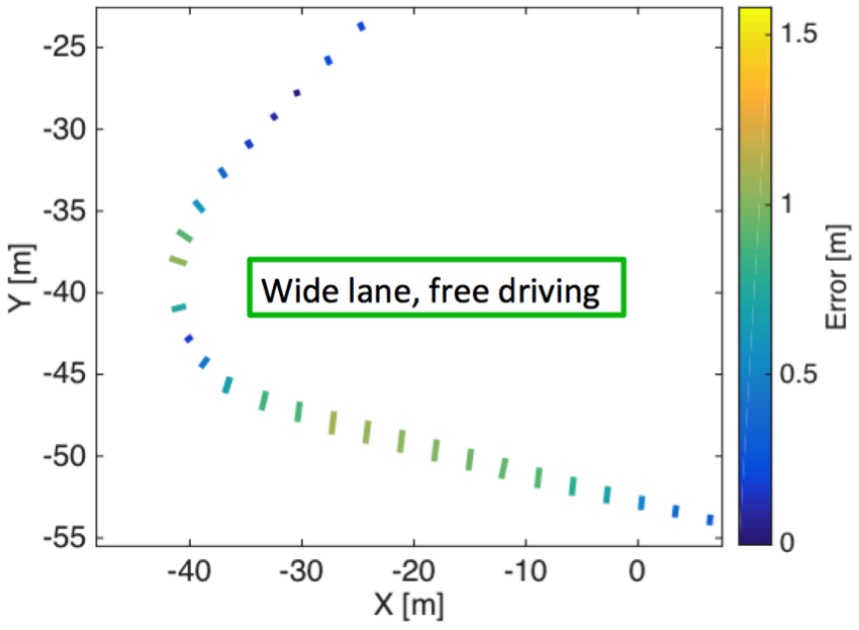


Figure 12: A second hot-spot in the South West of the map. Here we see a consistent error as the vehicle drives around a corner in a free driving area. There is no delineated lane, and thus the vehicle is free to choose its path. This does not represent a real error in the map.

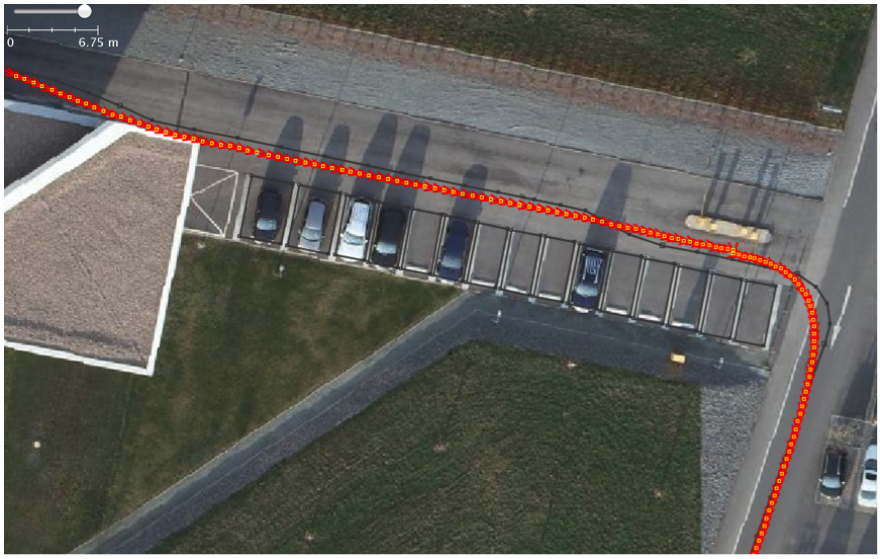


Figure 13: Here the ground-truth driver has chosen to leave a little more space between himself and the cars parked on the right. He also takes the turn onto the lane on the right a little tighter than the recommended path in the map.

3.2 Evaluation of labels: Parking spots and charging bays

In order to test the accuracy of the positions of the parking spaces and charging bays in the semantic layer, we performed ten manual parking manoeuvres in a parking space, and ten more in a charging bay. Unfortunately there was a lot of INS drift in the charging bay manoeuvres, and thus the results are limited by this drift, as shown in Figure 14a and Figure 14b. The parking space manoeuvres, however, are shown to be extremely accurate, down to the maximum resolution of the INS itself, 4cm, as shown in Figure 14a and Figure 14b. We show the overall error bars in Figure 15. We conclude that the estimation of the error in the parking manoeuvres is limited to the accuracy of the ground-truth system we have used, but that even when it is working well (for the parking spot), the accuracy of the map is excellent.

3.3 Evaluation of the speed map

Following the same approach in section 2.3, we estimated an speed map for the MLC site which is shown in Figure 16. The posterior map includes the places where pedestrians tend to walk, which are two main places: the junction to the North East of the building, and the pedestrian walkway along the South of the car park.

This feature of the semantic map is available. The integration into the local motion planning framework is trivial as explained in D4.4 and will without complications lead to the desired reduction in speed. However due to restrictive code freeze policies in the run-ups to the final demonstration, the speed map has not been integrated into the local motion planning framework yet.

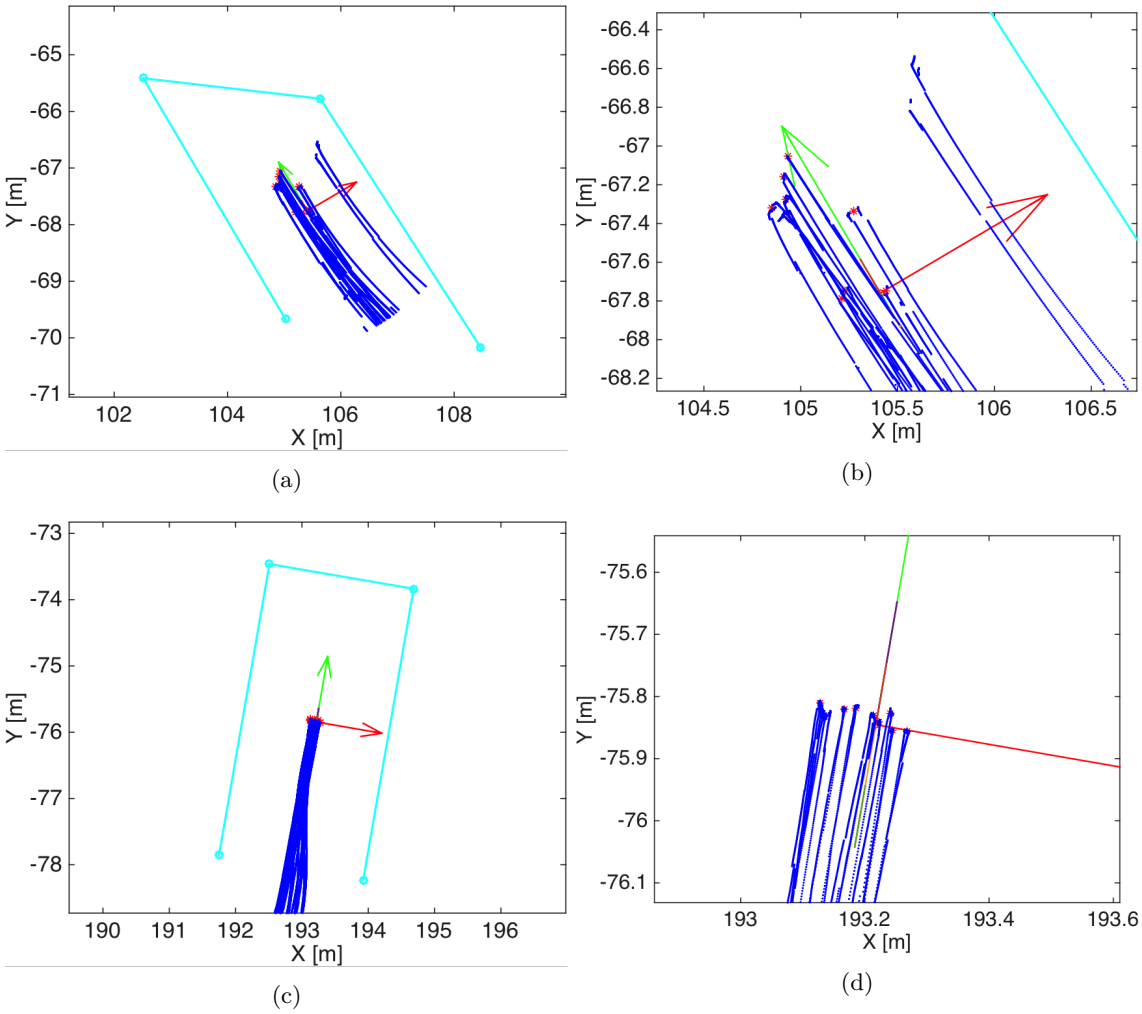


Figure 14: a) The vehicle trajectories for ten manual parking manoeuvres in a charging bay. The INS drift here is significant. b) Zooming in on those ten manual parking manoeuvres in the charging bay. The tears in the trajectories show the drift in the INS ground-truth system. c) The trajectories of the ten manual parking manoeuvres in a parking space. d) Zooming in on those parking trajectories to see the displacement between the car centres according to INS and the centre of the parking space according to the semantic layer.

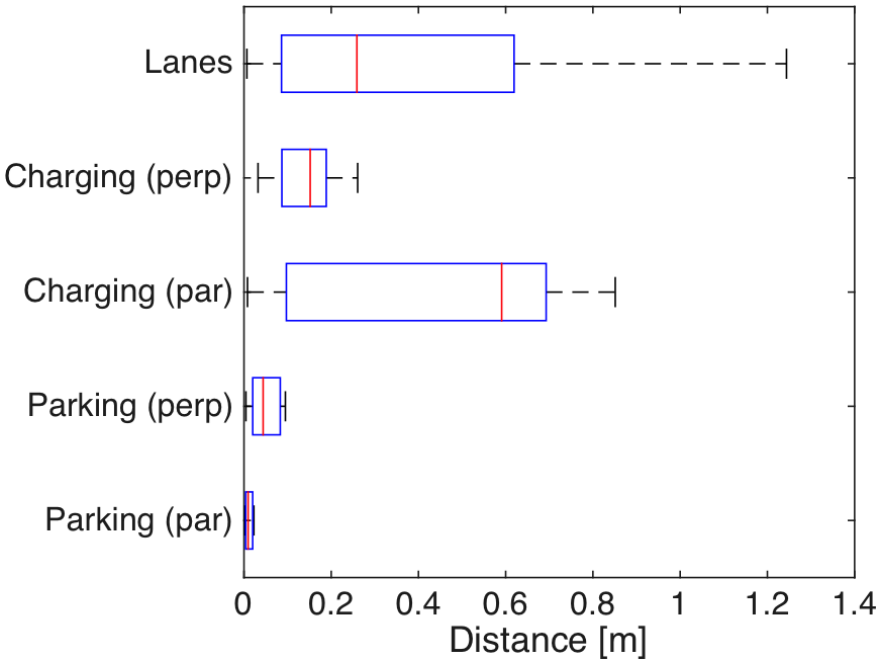
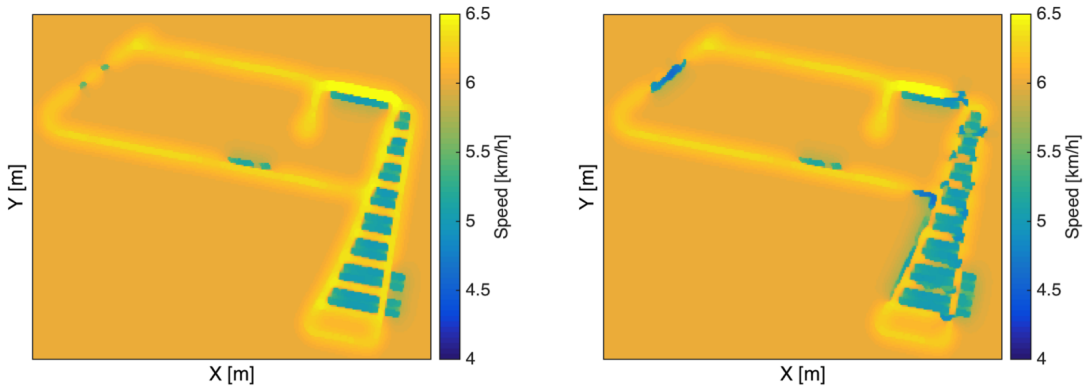


Figure 15: The overall errors for the various components of the semantic layer. We separate the errors for parking into two components, (a) the parallel error, which is along the direction of the parking bay, and (b) the perpendicular error, that is the sideways displacement of the vehicle in the bay.



(a) (b)

Figure 16: a) The speed prior. This is the recommended driving speed, given the positions of the lanes and the parking spaces. b) The speed posterior. This is the recommended driving speed of the prior, augmented with information about where pedestrians tend to walk. In this case, in the North East junction and the South along the left-hand side of the car park.

4 Constructing Metric Maps

One of the core tasks of this work package is the construction of metric maps that can be used for localization and navigation (work package 3) and for path planning (work package 4), as well as for generating semantic maps. The mapping process consists of three stages. In the first stage, the cameras of the car are calibrated with respect to the odometry frame. In the second stage, a consistent metric map is build from measurements taken by all four cameras using visual odometry. The resulting map and the camera images can then be used to create a dense geometric map using multi-view stereo methods. In the following, camera calibration and dense mapping are discussed. Creating metric maps for localization is described in Section 5.

Given the intrinsic calibration of the cameras, the extrinsic calibration of the four fisheye cameras in the car with respect to the wheel odometry is determined from image data captured while manually driving the car. For each individual camera, visual odometry is used to determine the trajectory of the camera. The trajectories are then used to synchronize the camera movement with the data recorded by the wheel odometry. Finally, loop-closures between different cameras are detected and used to refine the calibration: A 3D map is build using structure-from-motion, which is subsequently refined to yield more accurate camera poses and thus a more accurate extrinsic calibration. The whole calibration process can easily take up multiple hours.

Due to vibrations during driving or expansion and contraction of the car frame between seasons, the cameras of the car should be re-calibrated from time to time. Obviously, the calibration pipeline described above is not very practicable for this task due to its long run-time. For the use cases considered in the V-Charge project, it is safe to assume that there often already exists a metric map of the environment. This map can be used to significantly simplify the calibration process: Each image taken by a camera is individually localized against the map by estimating the camera pose from 2D-3D matches between features found in the camera image and 3D points in the map, yielding the pose of the cameras at different points in time. These poses are then used to infer the positions of the cameras relative to each other, followed by a non-linear optimization of these calibrations. In a final step, the pose of the resulting camera rig is determined relative to the wheel odometry. In contrast to the pipeline described in the previous paragraph, this structure-based calibration approach does not need to build a map of the environment. As a result, the camera calibration can be computed within a few minutes. Both calibration pipelines are described in detail in appendix D, including the experimental evaluation of their accuracy.

Given the camera poses and the corresponding images, we can create dense geometric maps using our multi-view stereo pipeline [3,4]. Figure 17 shows a top-down view of a dense map created for the ETH parking lot. The height of the different parts of the scene is color coded, with elements on the ground being colored in dark blue and the tallest elements, e.g., points on the two ramps leading up from the parking lot, colored in dark red. Unobserved space, e.g., space behind walls, is marked in black. In order to measure the accuracy of the map, we manually measured distances in the parking space (marked in the figure) and compared these ground truth measurements with the distances measured in our dense map. The results of the comparison can be found in Table 1 and show that the map is accurate up until less than 20cm.

5 Semantic Localisation Concepts

Metric maps built for a place will become outdated as the scene changes: For example, trees and vegetation in a outdoor parking lot will be changing between spring and winter. In a parking space such as the Bosch Parking Space in Stuttgart that is partially open to the outside, the illumination conditions can change drastically between morning and afternoon due to different shadows caused by the movement of the sun. As a result of such changes, the keypoint detector that is used to detect salient points in the images fires at

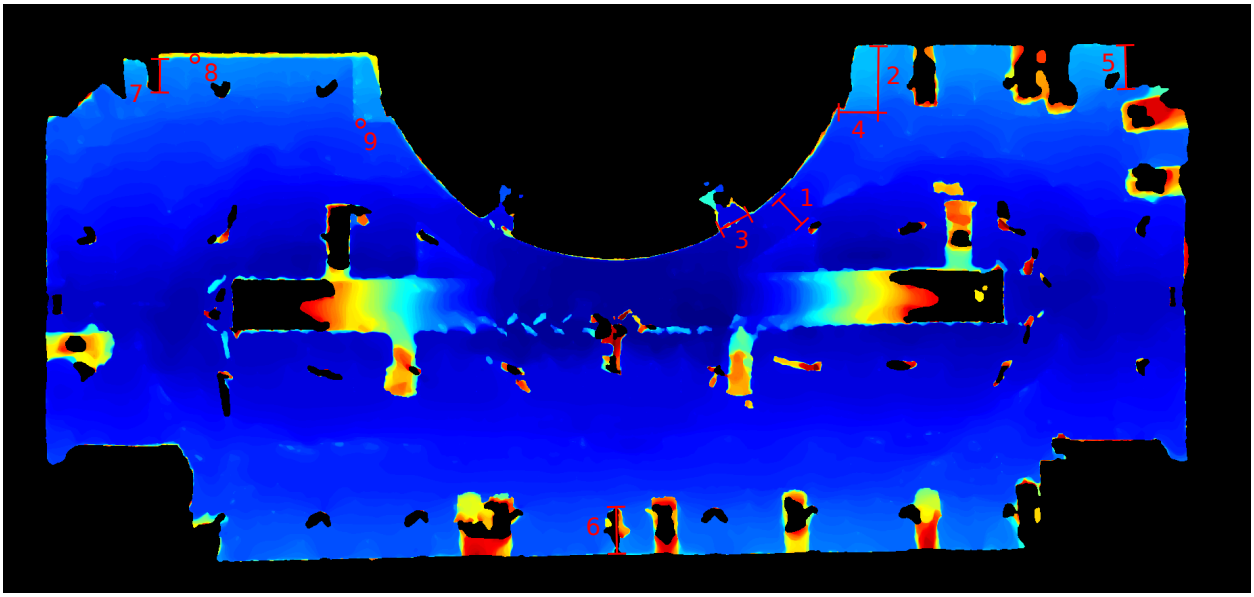


Figure 17: Color coded height map (dark blue lowest, dark red highest) with the positions of the 9 measurements indicated in red. Measurements 1 to 7 are length measurements and measurements 8 and 9 measure the height difference that occurs in the indicated circle.

Nr.	Ground truth	Map	Error
1	2.725m	2.650m	7.5cm
2	4.895m	4.800m	9.5cm
3	2.570m	2.485m	8.5cm
4	2.740m	2.900m	16.0cm
5	3.140m	3.100m	4.0cm
6	3.450m	3.400m	5.0cm
7	2.505m	2.450m	5.5cm
8	0.850m	0.800m	5.0cm
9	0.125m	0.100m	2.5cm

Table 1: Quantitative evaluation for the 9 indicated measurements in Fig. 17. The mean error over all the 9 measurements is 7.06cm.

different parts of the scene. Thus, it becomes significantly harder to establish matches between 3D points in the maps and features extracted in the camera images of the car because the feature detector does not detect keypoints at the image positions actually corresponding to the 3D points. Consequently, localization against the map becomes less stable and can even become impossible under larger changes, e.g., in illumination. Thus, the metric maps need to be updated and adapted over time such that they contain 3D points that correspond to features detected in the car's images (c.f. Figure 1). This is done by capturing data under different viewing conditions and fusing the resulting metric maps into a single *summary map*. In order to avoid excessive memory consumption, the individual maps are compressed by selecting only a subset of all available landmarks. Since these maps are created for localization, only landmarks useful for determining the position of the car are selected. Details on the mapping and localization concept can be found in appendix C.

The previous deliverable 2.3 had discussed further semantic localization concepts that use semantic labels extracted from each image to guide the localization process. While the research performed for the deliverable showed that such approaches can improve upon localization methods that do not use class labels, it was shown that they are not practically applicable due to increased run-time costs. The discussed approaches have thus not been integrated into the V-Charge project.

6 Summary of Milestones

6.1 Milestone 6, Task 2.1

The results presented in Figure 17 and Table 1 clearly show that the dense local maps created from the onboard sensors, more specifically using motion stereo on the fisheye images, are accurate within <20cm. In our experiments, moving obstacles presented no problem to the dense mapping pipeline. Either, motion stereo did not find enough matches on them or spurious 3D measurements resulting from moving objects were filtered out by free space measurements.

The dense mapping process generates a multi-layer height map of the scene. In a first pass, evidence for free space above a ground layer is accumulated, which is subsequently used during optimization to determine which areas are free and which are occupied. The second pass computes exact heights for places where the ground is visible. As a result, the dense maps can be used to determine which areas of the scene can be traversed by the car. In practice, the path planner uses the road graph provided by the semantic mapping process.

We consider this task of the milestone to be achieved.

6.2 Milestone 6, Task 2.2

The goals for this task is to "Extract features from image and 3D data (onboard sensors) and use them for position estimation, Repeatability of features and matching performance > standard SIFT". As for any other feature detector and descriptor combination, SIFT can only successfully match features if its detector fires at corresponding image regions, which is often not the case under larger changes in the scene (cf. Section 5). The experimental evaluation in appendix C clearly shows the benefit of using multiple maps compared to using a single map, both in terms of successful localisations and the localisation accuracy. Thus, we consider this task of the milestone to be achieved.

6.3 Milestone 6, Task 2.3

After the first manual drive around the car park, the metric layer and components of the semantic layer are processed offline. Now that a full semantic map is available, the vehicle makes automated operation available, and collects more data regarding the surroundings. These new data can provide more landmarks, used to update the bundle adjustment, improve the metric layer. Because the semantic labels are created in relation to the poses in the metric layer, the improvement of the metric poses allows the semantic labels to shift and warp around them without having to be recreated. As a result of this, the semantic components that are expensive to create (namely the locations of the parking spots, which require a human in the loop) only have to be created once, even though the metric layer is always changing. The speed map is the only component that is actively changing upon revisits to the car park: more pedestrian detections allow the updating of the speed map, improving the speed profile for the vehicle's future operations.

In all the car parks for which we have created semantic maps, there have been very few traffic flow markings, stop signs and other road markings. Therefore it has not made sense to detect these, but merely to hand-label them during the initial mapping procedure, or in the case of the traffic flow to deduce them automatically from the vehicle trajectory. We focused mainly in the detection of parking spots as they are the most common labels encountered in the testing scenarios. In order to make this process as efficient as possible, we train our classifiers in an active learning fashion achieving a performance of more than 90% precision at 90% recall. All in all, our designed pipeline pipeline demonstrates the creation and maintenance of a semantic map of a car park. Therefore, we consider the goal of Milestone 6, Task 2.3 to be fulfilled.

6.4 Milestone 6, Task 2.4

The efficacy of semantic maps in pose estimation and localisation procedures: The semantic layer is critical to the pose estimation of the vehicle in the car park. The visual localization system works by comparing the landmarks visible in the cameras to those stored in the metric map, and solving for the pose of the vehicle. The quality of this pose solution is heavily tied to the number of visible landmarks and to their distribution all around the vehicle. If there were too few or none in a particular direction, then the estimate would suffer. Therefore, the car must never deviate too far from the trajectories present in the available map, because only in those places are the landmarks plentiful and well distributed. The mechanism by which the car is tethered to these existing trajectories is its connection to the road graph and parking space components in the semantic layer. That road graph is itself generated from the landmark-heavy trajectories. The global planner plots a trajectory from start point to goal (the goal can be a position in the road graph or a parking or charging bay) through the road graph, thus ensuring that it never deviates too far from the original trajectories, and that a good localisation estimate is always available.

Demonstrate that semantics aid in planning procedures: The planning procedures are almost entirely built around the semantic layer. As it is described in deliverable D4.4, the global planner requires the road graph to generate a trajectory from start point to the goal, and while driving the vehicle uses the width of the lanes in order to know how much it can deviate around dynamic and static obstacles such as badly parked cars or pedestrians. The intersections inform the car that it must be careful about oncoming vehicles from the sides, and thus it will stop at the entrance of an intersection if it is on a low-priority lane. However, if the car is in a high-priority lane, it knows that it has priority over merging traffic and does not wait for them to go.

Show that a plan made in the absence of the semantic map will break the rules of the road or increase the risk of pedestrian injury: Here are five situations in which these are shown:

1. The speed map allows the vehicle to proceed more slowly across pedestrian crossings or where pedestri-

ans have been observed to frequent. Without this, the car would move inappropriately fast and could cause pedestrian injury.

2. While the vehicle is avoiding badly parked cars or pedestrians it can deviate from the centre of the lane, but not so much as to leave the confines of the lane or to stray into the oncoming lane, as shown in Figure 18a .

3. The appropriate behaviour at junctions (proceeding smoothly through if the car is on a high-priority road as shown in Figure 18b, or waiting before merging if approaching from a low-priority road as shown in Figure 18c) makes the vehicle's movement predictable to both pedestrians and other road users. This predictable and sensible behaviour is extremely important so that it does not confuse other uses and encourage them to behave in an erratic manner, thus increasing the likelihood of a collision or pedestrian injury.

4. Using the knowledge of the road network, the vehicle can predict the directions from which other vehicles might approach, which allows the vehicle to pause and make sure that the planned trajectory is collision-free, as shown in Figure 19a.

5. Without the knowledge of intersections and the road graph, the vehicle cannot make any predictions about the movement of other road users. It would wait for a passing car to leave a large radius before allowing itself to proceed, as shown in Figure 19b. However, with the knowledge of the road graph, it has a better idea of what the passing vehicle is doing and can allow itself to plan a following course before the passing vehicle leaves the large radius, as shown in Figure 19b.

7 Conclusion

In this deliverable we have focused in the scientific contributions regarding the integration, maintenance and evaluation of the main two layers: the metric map and the semantic map. We first describe how the two layers interact into the automation loop designed for lifelong vehicle operation in the context of the V-Charge project. We show how the components of the semantic map can be updated and maintained as the metric layer changes over time. We show also an extended evaluation of each of the components: the road graph, the labels and the speed map.

In addition, this deliverable discussed the metric mapping pipeline and the integration of the semantic localization concepts developed in this work package.

As can be seen from the reports attached to this document, Work Package 2 of the V-Charge project is well on track with its scientific, engineering, integration and evaluation goals at the time of writing.

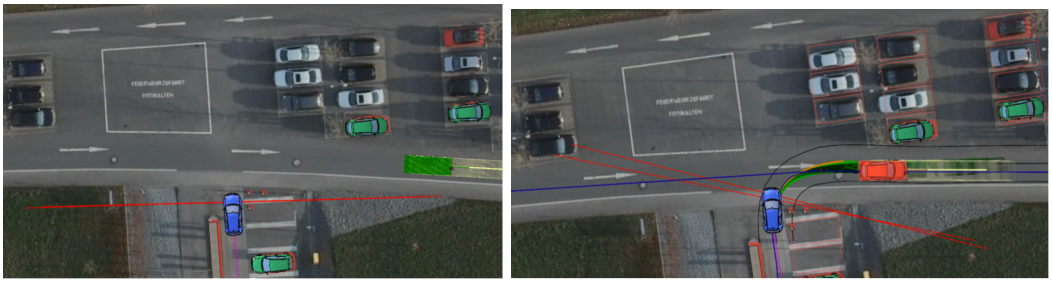


(a) (b)



(c)

Figure 18: a) The car deviates from the green lane in order to avoid the yellow dynamic obstacles. However, the deviation is limited by the width of the lane in the semantic layer. b) Being on a high-priority lane, the car knows not to stop at this intersection. c) Being in a low priority lane merging with a high priority lane, the car knows to stop here and wait for any traffic coming from the right.



(a) (b)

Figure 19: a) Without the road graph, the car waits for the passing car (green, right) to leave the large radius (the red bar) before it will plan an ongoing trajectory. b) With the road graph, the car allows itself to plan a safe following trajectory before the passing car (red, right) has left the large radius (red bar).

A Integrating Metric and Semantic Maps for Vision-Only Automated Parking

Authors: Hugo Grimmett, Mathias Buerki, Lina Maria Paz, Pedro Pinies, Paul Furgale, Ingmar Posner and Paul Newman

Link to Table of Contents.

Integrating Metric and Semantic Maps for Vision-Only Automated Parking

Hugo Grimmett†, Mathias Buerki‡, Lina Paz†,
Pedro Pinies†, Paul Furgale‡, Ingmar Posner†, Paul Newman†
†Mobile Robotics Group, University of Oxford, UK
{hugo, lina, ppinies, ingmar, pnewman}@robots.ox.ac.uk
‡Autonomous Systems Lab (ASL), ETH Zurich, Switzerland
{mathias.buerki, paul.furgale}@mavt.ethz.ch

***Abstract*— We present a framework for integrating two layers of map which are often required for fully automated operation: metric and semantic. Metric maps are likely to improve with subsequent visitations to the same place, while semantic maps can comprise both permanent and fluctuating features of the environment. However, it is not clear from the state of the art how to update the semantic layer as the metric map evolves.**

The strengths of our method are threefold: the framework allows for the unsupervised evolution of both maps as the environment is revisited by the robot; it uses vision-only sensors, making it appropriate for production cars; and the human labelling effort is minimised as far as possible while maintaining high fidelity. We evaluate this on two different car parks with a fully automated car, performing repeated automated parking manoeuvres to demonstrate the robustness of the system.

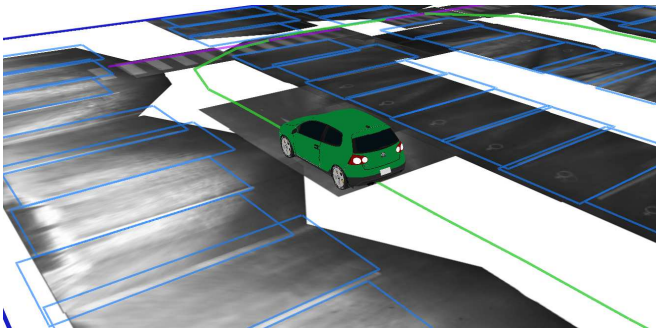


Fig. 1: The semantic information placed relative to the metric map. Shown are the driving lanes in green, and the parking spaces in blue.

I. Introduction

The environments in which our robots operate are often very complex, sharing aspects which are continually evolving along with some much more permanent features. Alongside these varying scales of change, certain tasks required for autonomous operation can be carried out in an unsupervised manner, making them cheap, while others require significant human involvement due to either the difficulty of the task or the need for accuracy guarantees. In this paper we propose a framework which strives for the best of both worlds: we manage the reprocessing of tasks based on how often they require updating, and we streamline tasks which require human involvement while maintaining the accuracies required for safety-critical automated driving.

The particular tasks we consider are those of creating metric maps of our robot's environment, required for path planning, and creating semantic maps which are crucial for the robot's ability to reason about and interact appropriately with their surroundings. An example of such a map is shown in Fig. 1.

We envisage a system by which our metric and semantic maps improve as our robots revisit previously-explored areas, as shown in Fig. 2. In this work we employ vision-only sensors for both mapping and localisation, and so the accuracy of any metric map relies upon the information gathered from the cameras during any particular run. However, the data collected during a revisit can be used to refine the metric map from the previous visit, and that in turn can be used to refine aspects of the semantics.

In safety-critical situations such as the operation where automated vehicles and humans coexist, semantic maps are often created (or at least maintained) by hand to ensure their high quality. Here we distinguish between two types of semantic labels: static semantics which represent more permanent features of the environment such as fixed obstacles or points of interaction, and dynamic semantics such as the characteristics of transient objects within an environment. By definition, static semantics are unlikely to change between revisits, and as such represent a significant but necessary cost in terms of human labelling effort, which ideally should not need repeating upon subsequent visits. We use dynamic maps to estimate the likelihood of dynamic obstacles being in any particular location of the robot's environment, in our case pedestrians, which allows the robot to drive at an appropriate speed. Dynamic semantics can be recomputed at each revisit in an unsupervised manner without requiring further human labelling effort.

Once initial semantic and metric maps have been created for a place, subsequent revisits to that place should allow the improvement of those maps without further labelling taking place. More loop closures improve the metric map, and those changes should propagate through to the positions of the semantic labels. We do this by performing the initial labelling in a frame of reference local to the sensor in which the object is visible. This is such that when the metric map is recomputed, the new position of the semantic label is an unmodified local transformation of the newly computed vehicle frame. Fig. 3 demonstrates this principle.

As the promise of robots for the general public is increas-

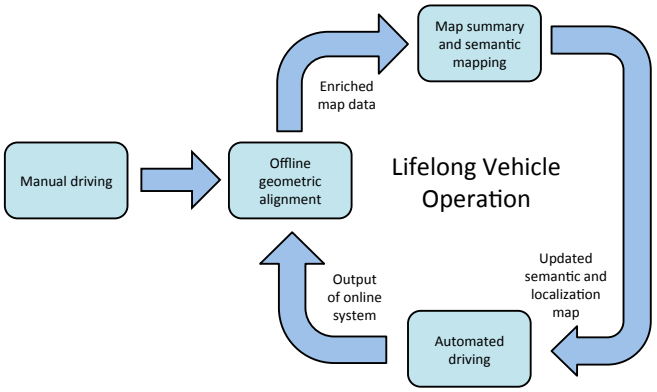
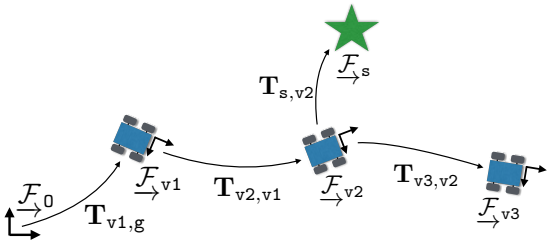
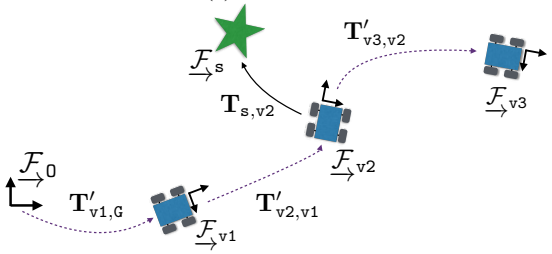


Fig. 2: The cycle of improvement for both metric and semantic maps as a vehicle autonomously revisits a place.



(a) Initial visit



(b) Revisiting the same place

Fig. 3: While the vehicle positions $[\mathtt{v1}, \mathtt{v2}, \mathtt{v3}]$ may be recomputed between (a) the initial visit and (b) the revisit, the position of the semantic object $\mathtt{s}$ relative to $\mathtt{v2}$ remains unchanged. This allows us to update the metric map without having to recreate the semantic labels. $\mathcal{F}_0$ represents the global origin coordinate frame, and $\boldsymbol{T}_{\mathtt{a},\mathtt{b}}$ represents a 6 d.o.f. transformation from $\mathtt{b}$ to $\mathtt{a}$.

ing, parking is a clear early target for widespread automation. The ubiquity of the requirement to park vehicles in car parks motivates the V-Charge project, whose goal is to provide a repeatable, robust, automated valet-parking system which requires neither expensive sensors nor specialised infrastructure in the car parks themselves.

The main contributions of this paper are:

- The implementation and evaluation of a life-long metric and semantic mapping system on a fully automated car,
- The proposal of a lifelong mapping cycle that seamlessly integrates metric map updates with both static and dynamic semantic labels,
- An algorithm for automatically generating road network graphs, and
- The use of introspective active learning for both parking space detection and pedestrian detection.

In Sec. IV we explain the localisation and mapping pipeline. In Sec. V we describe the process of creating the static maps and updating the dynamic maps. In Sec. VI we evaluate these systems in a real car park, showing the behaviour following revisits.

II. Related Works

The field of automated driving has a rich history, gradually making its way into production cars. This push to make robotics relevant for the wider public has led to significant progress in terms of localisation robustness using vision-only systems [1], [2].

Semantic mapping is also a rapidly developing field, but creating reliable maps with vision-only is difficult. Paper [3] fuses laser and camera data to map urban environments. The labelling of parking lots in particular has been approached using overhead images from online photographic repositories (such as Google Maps) [4], but this method makes assumptions about the layouts of parking lots. We assume no particular structure, and label in synthetic overhead maps made by projecting the camera views into simulated ground surfaces, which is useful for mapping underground car parks where satellite imagery is not available. The authors of [5] use a generative model of the geometry of urban scenes to label the lanes and a dynamic map. We overlay this style of map with an extra layer of semantics related to parking and the likely motion of pedestrians.

The matter of storing and subsequently making use of semantic labels was tackled during the Urban Challenge, with the road network being hand-labelled and then distributed to the competitors in the RNDF format [6]. The competitors then had to perform automated parking manoeuvres based on these maps [7].

In this work we leverage the adaptability of relative maps. Hybrid topological-metric maps have emerged to be a dominant representation in the field of autonomous driving: examples include sub-mapping [8], to manifold mapping [9], to completely relative representations [10].

We draw from all these influences, but present a robust and well-integrated end-to-end system which generates constantly-improving, rich semantic information and uses it to perform repeated parking manoeuvres with high accuracy.

III. The Autonomous Car

For the V-Charge project, two VW Golf series vehicles have been modified for drive-by-wire capability, with an integrated camera system, IMU, and computer systems.

The test vehicle is equipped with four 180° field-of-view fish-eye cameras, one facing each cardinal direction of the car (see Fig. 4). Images are recorded simultaneously at 12.5Hz in gray-scale format with a resolution of 1280 x 960. Subsequent downsampling to half the original size and extraction of BRISK features allows for fast, CPU-based image processing. In addition, wheel odometry running at 50Hz is used as a motion sensor yielding constraints between adjacent image acquisition poses. The intrinsic and extrinsic parameters of the camera system are calibrated using the CamOdoCal library [11].

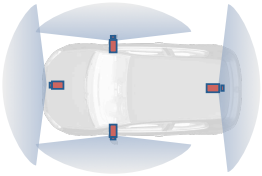


Fig. 4: The configuration of the four 180° field-of-view fish-eye cameras used for mapping, localisation, and dynamic object detection.

IV. Metric Mapping

In this section, we describe the metric mapping and localisation system which is required for the semantic labelling pipeline, and for autonomous navigation around the car park. For more detail the reader is referred to [12] and [13].

A. Map Representation

Let $\mathcal{F}_{\rightarrow \mathtt{A}}$ denote a coordinate frame A, and let further ${}_{\mathtt{A}}\mathbf{p}$ denote a point in space expressed with respect to $\mathcal{F}_{\rightarrow \mathtt{A}}$. In addition to that, let $\boldsymbol{T}_{\mathtt{A},\mathtt{B}}$ denote a 6DoF transformation, relating two coordinate frames such that ${}_{\mathtt{A}}\mathbf{p} = \boldsymbol{T}_{\mathtt{A},\mathtt{B}} \cdot {}_{\mathtt{B}}\mathbf{p}$. We further define $\mathcal{F}_{\rightarrow 0}$ to be an arbitrary, spatially fixed global origin frame. Our map used for localisation can then be thought of as a collection of landmarks ${}_{\mathtt{M_i}}\mathbf{p}$ expressed with respect to individual map frames $\mathcal{F}_{\rightarrow \mathtt{M_i}}$ as well as a set of transformations $\boldsymbol{T}_{0,\mathtt{M_i}}$ associated with each map frame. Landmarks here refer to 3D points in Euclidean space, associated with a BRISK feature descriptor. As will become clear later, these map frames are fixed in space at all times and lie along the car path of the initial dataset used to form the base-map. The landmarks are originally observed and inferred from the cameras mounted on the vehicle, for which the exact car poses with respect to the global coordinate frame for every captured image is inherently unknown and subject to change as more information becomes available in subsequent mapping sessions. Therefore, the map also contains transformations between vehicle poses and associated mapping frames. During the course of the map refinement, both the location of the landmarks as well as the vehicle poses are updated, whereas the mapping frames stay stationary.

B. The Base Map

Since the four cameras record images simultaneously, we will refer to an image frame as the collection of all four images recorded at that time. In a first stage, the initial image stream is subsampled such that consecutive image frames are evenly spaced along the car trajectory, separated by a 20cm baseline. A vehicle pose is associated with each of the remaining image frames, forming a pose-graph. Subsequently, BRISK features are extracted for each camera image, followed by a loop-closing engine looking for matching images, which then form relative transformation constraints based on a RANSAC estimate of the camera poses of the images involved. Together with the constraints between successive car poses from integrated wheel odometry, a pose-graph relaxation problem is formulated whose solution consists of a geometrically consistent graph of car poses. In the next stage, the BRISK features are tracked along subsequent images - individually and independently for each camera. The guess of the car poses together with the camera extrinsic and intrinsic calibration allow to infer the landmark positions in the global frame for all feature tracks. Finally, a full bundle adjustment optimisation problem is solved, which involves all landmark observations (reprojection constraints between the inferred 3D landmark position and the observed 2D feature in the images) and all wheel odometry constraints between successive car poses. This yields a refined estimate of the car poses along the pose-graph and all landmarks observed in the base-map. To complete the initial base-map, a mapping frame $\mathcal{F}_{\rightarrow \mathtt{M_i}}$ is created for each car pose.

C. Localisation and Multi-Session Mapping

With the base-map described in the previous section, the car is able to navigate autonomously within the mapped area. This is achieved by repeatedly estimating the transformation $\boldsymbol{T}_{\mathtt{M_i},\mathtt{v}}$ between the car and a nearby map frame. Let $\widehat{\boldsymbol{T}_{\mathtt{M_i},\mathtt{v}}}$ be a guess of the car's current pose based on a previous localisation estimate and wheel odometry. We now retrieve the landmarks from the map in the vicinity of our guess and project them into the camera frames, allowing to match them against the extracted image features by taking both the image space and descriptor distance into account when forming the set of 2D-3D associations. From these associations, constraints on the vehicle pose are formed in the context of a non-linear least squares optimisation problem. The vehicle pose is estimated by minimising the reprojection errors based on these associations. This estimate is then used in the proximate iteration as a new starting point.

Since between recording the initial dataset for the base-map and a later attempt to localise in it, the environment may change, not all image features will be successfully matched against landmarks from the base-map during localisation. These unmatched features can be used in an offline processing step to track new landmarks which are then added to the base-map, enriching the latter with new information representing the change in the environment. For the reasons mentioned above, we also add the new vehicle poses to the map. Both the new landmarks as well as the new vehicle poses are associated with a nearby map frame. This associations, corresponding to transformations between the new vehicle poses and the fix map frames from the base-map, stem from the localisation estimate based on successfully matching a large enough subset of features in the current image frames to landmarks already present in the map. Consequently, the image frames from the dataset used to localise against the map (subsequently referred to as the "localisation dataset") contain associations both between landmarks from the base-map as well as the new landmarks and hence allow formulating constraints between the vehicle poses from the base-map and the ones from the localisation dataset. This fact is exploited in a full bundle adjustment step. The resulting map is referred to as a multi-session map now consisting of two datasets. The procedure can naturally be extended to an arbitrary number of datasets. This multi-session mapping allows the map to be refined as more datasets are processed resulting in more accurate map geometry and increased localisation performance. Please note

that the multi-session mapping described in this form does not explicitly deal with landmarks that disappear or shift their position in changing environments over longer time spans. Outdated landmarks remain in the map, but will fail to match against current image observations. This is acceptable as long as a large enough set of valid landmarks can be matched against current images at all times. Also note that, as described above, image features failing to match against the map are tracked over subsequent images and form new landmarks that may account for an outdated landmark that shifted its position or changed its appearance. Techniques to curate maps and filter out outdated or unhelpful are studied in detail in [13].

V. Semantic Mapping

In most parking environments there is a wealth of permanent semantic information which can, if correctly leveraged, make a complex manoeuvre such as driving along a lane and parking a much more straightforward task. Typically the creation of such a map involves hand-labelling features such as parking spaces and lanes, but this is an extremely expensive and tedious task for a human. Therefore, we make use of machine learning to reduce the effort on the labeller. In one-shot machine learning, it is necessary to have a large amount of labelled data to train a classifier to detect pedestrians or parking spaces with sufficient accuracy to make it useful, which may not generalise well over car parks. To combat this and also maximise flexibility, we use active-learning algorithms in which a human works in the loop with the machine to answer the most challenging classifications as chosen by the algorithm, thus reducing the labelling quantity for a given desired performance level. In order to further reduce the labelling effort, we make use of introspective active learning algorithms [14].

In the car parks we have encountered during the V-Charge project, we have included the following classes in the semantic map: lane structures, parking spaces, pedestrian crossings, and a recommended driving speed for any point in the environment. The recommended driving speed comes from the *dynamic map*, which brings together permanent aspects of the environment (such as lanes) and dynamic objects such as pedestrians or vehicles. As the automated vehicle drives through the car park, it detects these dynamic objects and incorporates these new observations into the existing map, which better informs the vehicle how to safely navigate the area during future visits.

A. Static Map

We use active learning on a synthetic overhead image to accurately locate all the parking spaces in the car park. The pedestrian crossings, being so few in number, are labelled by hand.

1) Synthetic overhead image: The synthetic overhead image is made by first rectifying the fisheye images, then using the vehicle pose from the metric map to project the image points onto a virtual ground plane. When projections from multiple images lie on the same pixel in the ground plane, the running mean of the values is used. The overhead images created of the car parks used in the experiments section are shown in Figs. 6 and 11.

2) Classification: From the synthetic overhead image, we train a linear Gaussian process classifier to detect parking spaces. Gaussian process classifiers have been shown to perform well in active learning [15]. Initially the classifier has no training data, and so the user draws rectangles around a few parking spaces. Negative examples are randomly extracted from the image at various scales. Training occurs using the HOG feature [16] representations of these data, and the classifier returns a pre-defined number of hypotheses of where further parking spaces may lie in the image. The hypotheses come from a search over the image at several scales calculated relative to the average size of the user-supplied positive examples, over which non-maximal suppression is applied to increase their positional accuracy. The user then has the option to accept the hypotheses, or mark some of them as positive and negative examples, and retraining the classifier. This cycle can continue until all the parking spaces are detected to a sufficiently high level as set by the user.

3) Relative association: Now that we have the positions of the parking spaces and pedestrian crossings in the synthetic overhead image, for each label we find the fisheye image in which they are visible and most central. We then associate that label with that camera pose, and calculate the transformation between the label and the vehicle pose ($\mathbf{T}_{\mathrm{s,v2}}$ in Fig. 3). This is what allows us to avoid relabelling the static semantic labels whenever the metric map is updated.

B. Road Network

We have developed an algorithm for automatically generating a road network, whose only requirement is for the vehicle to have been driven through each lane at least once. The algorithm uses only the three-dimensional positions of the vehicle at regular time intervals, and detects both lanes and intersections. The method is presented in Algorithms 1 and 2. In summary, we consider the vehicle positions as nodes in a graph, and connect them to nearby nodes in order to simplify lanes which are driven several times with slight displacement each time (see Fig. 8a). We then repeatedly prune this graph by replacing maximal cliques by their centre point (or rather, an existing node nearest to their centre point, see Fig. 8b) until we have a skeleton graph (see Fig. 8c). The final step involves iteratively exploring the pruned graph to find the distinct lanes and intersection points, as shown in Fig. 8d.

C. Dynamic Map

We use a probabilistic representation of dynamic obstacles to create a recommended speed map we call the *dynamic map*. First of all, we project the pixel locations of the static labels into the global frame, obtaining a metrically consistent semantic map. In order to use the semantic map for motion aid we estimate the likelihood of any place in the map $\mathbf{x}$ being occupied by a dynamic object, given the prior knowledge of the detected labels at $\mathbf{x}_0$ and partial observations of pedestrians $\mathbf{z}$. This posterior distribution is then used to create a recommended speed map for safer future traversals of the car park. Note that the pedestrian detections come from detections in the fisheye images. This

Algorithm 1: Creating and simplifying the adjacency matrix required for Algorithm 2.

Data: vehicle poses $\mathcal{X} \in \mathcal{R}^3$, connecting distance d, number of pruning loops P
Result: Symmetric adjacency matrix A

// Connect all nodes within a radius d of each other
forall the *pairs of poses* $(x_i, x_j) \in \mathcal{X}$ **do**
  if $\|x_i - x_j\| \leq d$ **then** $A_{ij} \leftarrow 1$
  else
    $A_{ij} \leftarrow 0$; $A_{ji} \leftarrow 0$ // Symmetric A

// Disconnect temporally subsequent nodes within d
forall the *poses* $x_i \in \mathcal{X}$ **do**
  $n \leftarrow 1$
  while $\|x_i - x_{i+n}\| \leq d$ // Exit if assertion fails
  do
    $A_{i,i+n} \leftarrow 0$; $A_{i+n,i} \leftarrow 0$
    $n \leftarrow n + 1$

// Connect nodes in the order they were driven
forall the *poses* $x_i \in \mathcal{X}$ **do**
  $A_{i,i+1} \leftarrow 1$; $A_{i+1,i} \leftarrow 1$

// Prune cliques larger than 2, replacing them by the node nearest to the clique centre
for $p \leftarrow 1$ **to** P **do**
  // Produce an ordered list of maximal cliques
  $C \leftarrow$ `findMaximalCliques`(A)
  $C \leftarrow$ `sortDescending`(C)
  $R = \emptyset$ // The set of nodes marked for removal
  forall the *cliques* $c \in C$ *such that* $|c| > 2$ **do**
    $x_\mu \leftarrow$ `mean`(x_c)
    $x_b \leftarrow$ `returnNearestNode`$(x_\mu, \mathcal{X})$
    $N \leftarrow$ `returnNeighbours`(c, A) // N includes indices $i \notin c$ but adjacent to at least one node in c
    // Skip cliques containing nodes marked for removal
    if $N \cap R = \emptyset$ **then**
      $A_{b,N} \leftarrow 1$; $A_{N,b} \leftarrow 1$
      // Find the dead-end nodes
      $D \leftarrow \{i \in N$ for which `degree`$_A(i) = 1\}$
      // Mark these nodes for removal
      $R \leftarrow R \cup c \cup D \setminus \{b\}$
  $A \leftarrow$ `removeRowsAndCols`(R, A) // remove the rows and columns of A which refer to the nodes in R

Algorithm 2: Iteratively exploring the adjacency matrix from Algorithm 1 to discover the road network and its intersections.

Data: Symmetric adjacency matrix A with M rows and columns
Result: Road network R, intersections I^*

$v \leftarrow \emptyset$ // Current node
$T \leftarrow \emptyset$ // Termination nodes
$R \leftarrow \emptyset$ // Road network
$I^* \leftarrow \{i \in \{1, \ldots, M\}$ for which `degree`$_A(i) > 2\}$
while *A nonempty* **do**
  // Intersection nodes
  $I \leftarrow \{i \in \{1, \ldots, M\}$ for which `degree`$_A(i) > 2\}$
  // Dead-end nodes
  $D \leftarrow \{i \in \{1, \ldots, M\}$ for which `degree`$_A(i) = 1\}$
  if $D \neq \emptyset$ **then**
    $v \leftarrow$ some $d \in D$ // Start at a dead-end node
    $T \leftarrow I$ // Terminate at an intersection node
  else
    if $I \neq \emptyset$ **then**
      $v \leftarrow$ some $i \in I$
      $T \leftarrow I \setminus \{v\}$
    else
      // Remaining road network is only one loop
      $v \leftarrow 1$ // Start anywhere
      $T \leftarrow \emptyset$
  $L \leftarrow v$ // Lane
  $O \leftarrow$ `returnNeighbours`(v, A) // Options
  while $(i \notin T) \wedge (O \neq \emptyset)$ **do**
    $v \leftarrow$ some $o \in O$
    $L \leftarrow (L, v)$ // add v to the vector L
    $O \leftarrow$ `returnNeighbours`$(v, A) \setminus L$ // with L as a set
  $R \leftarrow R \cup \{L\}$ // R is a set of vectors
  if $L_1 \in I$ **then**
    $L \leftarrow L \setminus L_1$ // don't include start node
  if $L_{|L|} \in I$ **then**
    $L \leftarrow L \setminus L_{|L|}$ // don't include terminal node
  $A \leftarrow$ `removeRowsAndCols`(L, A)

problem can be represented by a graphical model $G(V, E)$ whose nodes $x_i \in V$ are the set of discrete map locations. Fig. 5 shows an example of the probabilistic graphical model used in this method. A prior probability $p(\mathbf{x}_0)$ on the nodes is assigned by considering the various types of space encoded in the semantic map (i.e., pedestrian crossings, parking spaces and driving lanes). The maximum a posteriori estimate is calculated by solving:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \big(-\log \overbrace{p(\mathbf{z}|\mathbf{x})}^{\text{likelihood}}\ p(\mathbf{x}|\mathbf{x}_0) \underbrace{p(\mathbf{x}_0)}_{\text{prior}} \big), \tag{1}$$

where the prior distribution and pedestrian likelihood are modelled as normal distributions over the map, making the optimisation linear. To account for the effect of a node x_i on its neighbours $N(i)$, we impose regularisation by adding linear binary constraints (the second sum in Eq. 2).

$$p(\mathbf{x}|\mathbf{x}_0) \propto \sum_i \| x_i - x_{i0} \|_2^2 + \sum_i \sum_{j \in N(i)} \| x_i - x_j \|_2^2 \tag{2}$$

$$p(\mathbf{z}|\mathbf{x}) \propto \sum_i \| z_i - x_i \|_2^2 . \tag{3}$$

Due to the linear nature of the problem, an exact solution is achieved after a single batch iteration.

The final step is to convert the posterior probability of a pedestrian over the car park into a recommended speed profile. We do this by defining minimum and maximum driving speeds, and use the probability to interpolate linearly between the two. A high likelihood of a pedestrian lowers the recommended speed at that location. This produces a map such as those on the right hand side of Fig. 10.

VI. Experimental Results

In this section we describe the experiments carried out in two different car parks to validate our metric and semantic mapping algorithms, and the manner in which they evolve together. The evaluation is split into two subsections, one for the Stuttgart car park (Sec. VI-A), and the latter for the Zurich car park (Sec. VI-B).

A. The Stuttgart Car Park

We have run the metric and semantic mapping pipelines described in Sections IV and V using data from a car park in Stuttgart. As a proof of concept, we drove the car manually around the car park several times. This car park contains 164 parking spaces. The lighting conditions during the data collection were consistent in the middle of the car park,

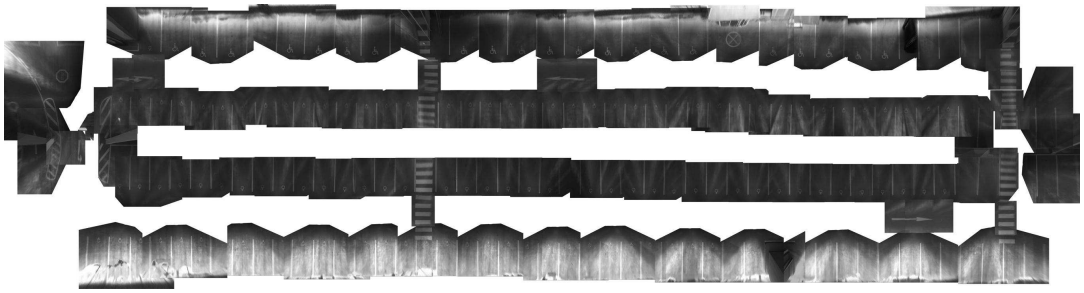


Fig. 6: The synthetic overhead image from the Stuttgart car park. The car park is larger and the image much cleaner than that of the Zurich car park, and so is better suited to active learning.

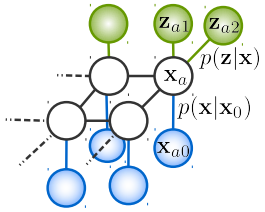


Fig. 5: The graphical model used to represent the probability distribution of pedestrians. White nodes $\mathbf{x}$ correspond to the set of discrete map locations. Blue nodes $\mathbf{x}_0$ represent values after applying the initial prior probability on a subset of nodes according to their classification as parking spots, pedestrian crossings, lanes, or other. Green nodes $\mathbf{z}$ correspond to the observations of dynamic objects in a particular map location. Note that a dynamic object could be observed multiple times (z_{a1}, z_{a2}) at the same location a, thus a node x_a would updated with several observations.

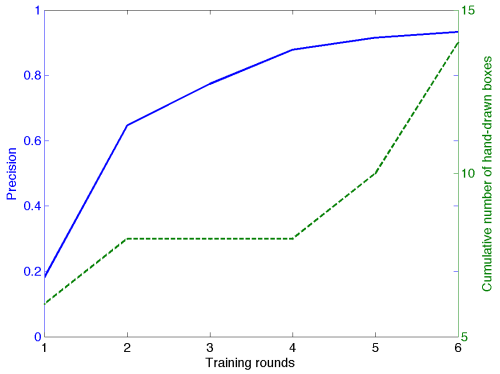


Fig. 7: The precision of the parking space detections (blue) across active learning loops. The cumulative number of hand-drawn labels is shown as the dashed green line. Note that at each loop, the user specifies some of the classifier hypotheses as correct and incorrect, and these are not shown as they are much quicker to input than drawing new labels from scratch.

becoming brighter closer to the long edges (which receive more natural light). The synthetic overhead image does not have too many undesirable artefacts (see Fig. 6).

We then use active learning to detect the parking spots in the synthetic overhead image. In total, the user only had to manually label 14 parking spaces over 6 rounds of active learning to achieve a precision of over 0.93 (see Fig. 7), which substantially reduces the human effort in such a labelling task. In addition, if we were to map further floors of this car park, the classifier learned from this floor could be applied to those to make the process very easy for the labeller. Note that the number of times the user accepts or rejects the hypotheses returned by the classifier at each round of training is not shown. This feedback is very quick and is a negligible contributor to the labelling effort. A further note is that the only negative examples in the training data were rejected hypotheses from the classifier, thus at epoch 1 there is only one negative: the vector 0.

In Fig. 8 we show some of the intermediary steps and the final result of the road network generation algorithm on the Stuttgart car park. In a totally unsupervised manner (aside from choosing the parameters r and p for Algorithm 1) the entire road structure is extracted from the poses. This road structure together with the parking spots could now be used for autonomous driving. We also detect pedestrians and process the dynamic map for this car park, but this is not shown. For a demonstration of this, see Sec. VI-B.

B. The Zurich Car Park

For the evaluation of the improvement cycle and automated parking we use an underground car park in Zurich.

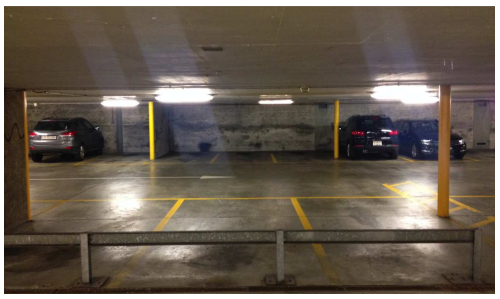
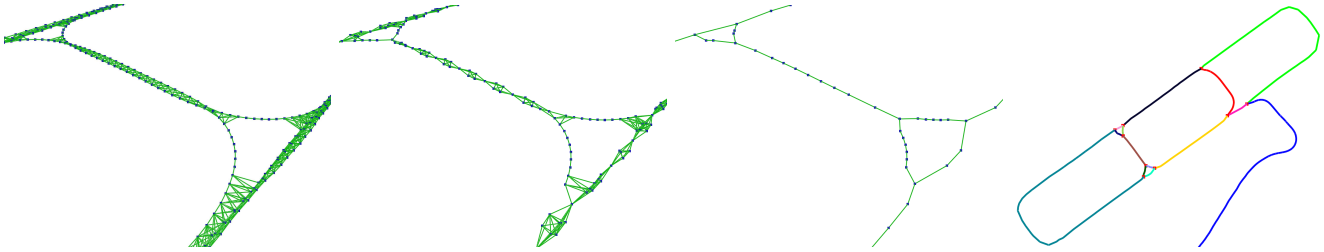


Fig. 9: The car park in Zurich. Note the difficult lighting conditions, resulting in glare off the ground.

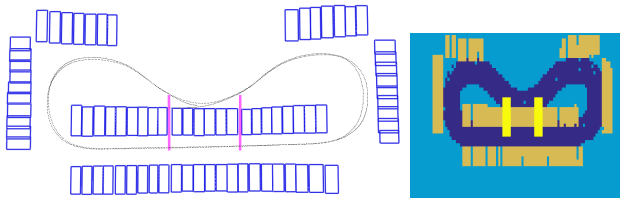
There are 80 parking spaces which vary in size depending on their position relative to the edges and corners. The parking spaces are delineated by yellow markers as shown in Fig. 9. The car park is not planar; there is a drain running along the centre, and the ground is hinged at this point all the way to the edges. Some of the yellow lines on the floor are omitted in favour of yellow columns, as pictured. There are two pedestrian crossings, one is shown in the figure as yellow hatching on the right.

The data are organised as follows: the car was driven twice around the car park as an initial visit, and then there were six subsequent 'revisit' loops. During each revisit loop, pedestrians walk between the pedestrian crossings; in revisit loops 1 to 3, they walk along the right-hand pedestrian crossing, and then in loops 4 to 6 they walk along the left-hand pedestrian crossings, as shown in the left hand side of

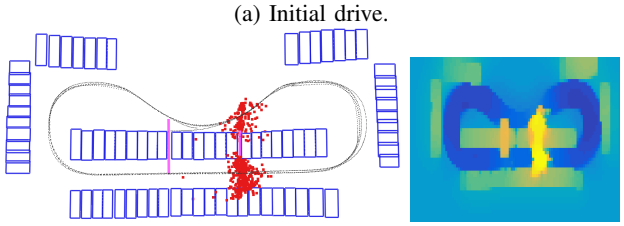


(a) The fully connected graph (Algorithm 1, lines 1 to 12). (b) The graph after one pruning loop (Alg. 1, lines 13 to 25). (c) The finished graph at the end of Alg. 1. (d) The lanes as calculated by Alg. 2.

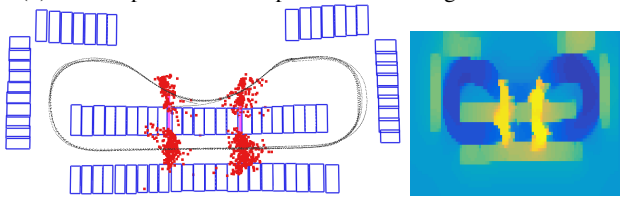
Fig. 8: The process of calculating the road network using the method from Sec. V-B applied to the Stuttgart car park.



(a) Initial drive.



(b) The maps from data up to and including the 3rd revisit.

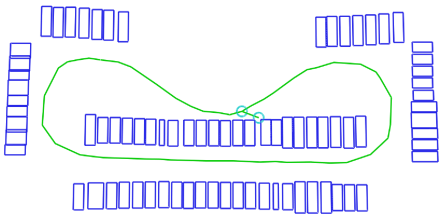


(c) The maps from data up to and including the 6th revisit.

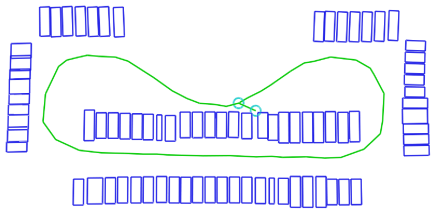
Fig. 10: On the left, the red dots represent projections of the detected pedestrians into the static map. On the right, the evolution of the dynamic map as more pedestrians are detected. In the dynamic map, blue represents lower danger (higher speed) and yellow represents more danger (higher speed). The top dynamic map is calculated using only the prior over pedestrians given the static map.



Fig. 11: The synthetic overhead image of the Zurich car park.



(a) Initial visit



(b) First revisit

Fig. 12: The evolution of the semantic map through the original drive, and the first revisit. The parking spaces are the orange rectangles, lanes are in blue and the intersections are purple circles. Notice that the top and middle rows of parking spaces shift visibly from (a) to (b) as the changes in the metric map propagate through to the positions of the semantic labels.

Fig. 10.

We then use these loops to emulate the process in Fig. 2: the first map (comprising both metric and semantic components) is created using the initial run, then as we go around the evolution cycle, the next map comprises the initial loop plus the 1st revisit. The third map comprises the initial run, the 1st and 2nd revisits, and so on. In total we have seven maps, each more informed than the previous one.

Because we want to minimise the human labelling effort, we only label the parking spaces and pedestrian crossings once, in the base map. The synthetic overhead image used for creating those labels is shown in Fig. 11. However, because the labels are associated with the vehicle positions from which they were visible in the images, the changes in the metric map shifts the global positions of those semantic labels as the maps evolve. This evolution process is shown in Fig. 12. Note that the overhead image for the Zurich car park is much more distorted and blurry than the one for the Stuttgart car park. This reduces the effectiveness of active learning, and generally the human will have to do more manual labelling if this is the case. Secondly, the positions of the semantic labels do visibly change between the first and second maps in Fig. 12.

Next we use the first four maps to test the repeatability of the autonomous parking system. We performed the following actions:

1) The car is driven to a point 20m away from the desired

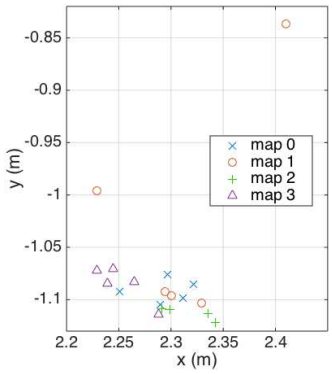


Fig. 13: The positions of the five autonomous parking manoeuvres of the vehicle in map for maps 0 to 3. Map 0 comprises the initial drive, then map 1 includes both the initial drive and the 1st revisit, then map 2 includes those for map 1 plus the 2nd revisit, and so on. The displacement between the parking centres and the position of the parking spot (at the origin) is due to the difference between the vehicle centre and the centre of the parking spot.

parking location, then

2) The car localises itself in the metric map and uses the semantic map to drive autonomously along the lane and park in the parking space.
3) Next, the car is manually driven out of the parking space to the same starting location 20m away, and the process is repeated for a total of five times per map.

Calculating the ground truth of localisation systems for mobile platforms is an open problem, so we have done this by estimating the position of the vehicle relative to a chequerboard at a fixed position in the car park which is visible in the camera images. The parking positions are shown in Fig. 13. This shows that despite the fact that the metric maps are recalculated in an unsupervised manner and the positions of the semantic labels shift acoordingly, the parking accuracy is very consistent, with a significant cluster of points within 0.13m and the few outliers never varying by more than 0.3m. We attribute these outliers to errors in the odometry, and the fact that the parking planner plotted a bad course for the vehicle during one of the parking manoeuvres during the first revisit.

Evolution of the speed map: During the initial drive there are no pedestrians in the car park, but during each revisit there are pedestrians walking along the pedestrian crossings. Using the same active learning framework explained in Sec. V-A.2 applied to the raw camera images, we detect these pedestrians and then using the method detailed in Sec. V-C we update the dynamic map. This dynamic map represents the recommended driving speed for subsequent visits to the car park. In Fig. 10 we show the positions of the pedestrians as red points on the left, and the evolution of the dynamic map on the right. It is also convenient that as we see more pedestrians, the classifier becomes increasingly good at detecting them, thus reducing the human labelling effort as time goes on.

VII. Conclusions

We have presented a functioning end-to-end system which provides a solution to the question of how metric and semantic maps should be fused together. This approach has been framed within with principle that maps should be continuously improving as an autonomous vehicle revisits previously explored locations. We have shown that parking accuracy using these maps is consistently good as these revisits and map refinements take place. The whole framework is streamlined in terms of reducing the human supervision effort, while maintaining the high degree of accuracy required for autonomous operation. We have demonstrated the stability of the system on two car parks as part of the V-Charge project.

VIII. Acknowledgements

This work is funded under the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916 and by the UK EPSRC Grant Number EP/J012017/1.

References

[1] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Muhlfellner, S. Wonneberger, J. Timpner, S. Rottmann, B. Li, *et al.*, "Toward automated driving in cities using close-to-market sensors: An overview of the V-Charge Project," in *IEEE Intell. Vehicles Symposium (IV)*, 2013.

[2] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, "Making Bertha See," *IEEE ICCV Workshop Computer Vision for Autonomous Vehicles*, 2013.

[3] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte, "Classification and Semantic Mapping of Urban Environments," *The Intern. Journal of Robotics Research (IJRR)*, 2011.

[4] Y.-W. Seo and C. Urmson, "A Hierarchical Image Analysis for Extracting Parking Lot Structures from Aerial Images," Tech. Rep., 2009.

[5] A. Geiger, M. Lauer, and R. Urtasun, "A Generative Model for 3D Urban Scene Understanding from Movable Platforms," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[6] D. U. Challenge, "Route Network Definition File (RNDF) and Mission Data File (MDF) formats," Tech. Rep., Defense Advanced Research Projects Agency, Tech. Rep., 2007.

[7] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The Stanford Entry in the Urban Challenge," *Journal of Field Robotics*, 2008.

[8] J. Guivant and E. Nebot, "Improving Computational and Memory Requirements of Simultaneous Localization and Map Building Algorithms," in *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 2002.

[9] A. Howard, G. S. Sukhatme, and M. J. Matarić, "Multi-Robot Mapping using Manifold Representations," *IEEE - Special Issue on Multi-robot Systems*, 2006.

[10] G. Sibley, C. Mei, I. Reid, and P. Newman, "Planes, Trains and Automobiles Autonomy for the Modern Robot," in *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 2010.

[11] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic Intrinsic and Extrinsic Calibration of a Rig with Multiple Generic Cameras and Odometry," in *IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[12] P. Muehlfellner, P. T. Furgale, W. Derendarz, and R. Philippsen, "Evaluation of Fisheye-Camera Based Visual Multi-Session Localization in a Real-World Scenario," in *IEEE Intell. Vehicles Symposium (IV)*, 2013.

[13] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, "Summary Maps for Lifelong Visual Localization," 2014, manuscript submitted for publication.

[14] H. Grimmett, R. Paul, R. Triebel, and I. Posner, "Knowing When We Dont Know: Introspective Classification for Mission-Critical Decision Making," in *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 2013.

[15] R. Triebel, H. Grimmett, R. Paul, and I. Posner, "Driven Learning for Driving: How Introspection Improves Semantic Mapping," in *Intern. Symposium on Robotics Research (ISRR)*, Singapore, 2013.

[16] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

B Introspective Classification for Robot Perception

Authors: H. Grimmett, R. Triebel, R. Paul, and I. Posner

Link to Table of Contents.

Article

The International Journal of
Robotics Research
1–20
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364915587924
ijr.sagepub.com

Introspective classification for robot perception

Hugo Grimmett[1], Rudolph Triebel[2], Rohan Paul[1] and Ingmar Posner[1]

Abstract

In robotics, the use of a classification framework which produces scores with inappropriate confidences will ultimately lead to the robot making dangerous decisions. In order to select a framework which will make the best decisions, we should pay careful attention to the ways in which it generates scores. Precision and recall have been widely adopted as canonical metrics to quantify the performance of learning algorithms, but for robotics applications involving mission-critical decision making, good performance in relation to these metrics is insufficient. We introduce and motivate the importance of a classifier's introspective capacity: the ability to associate an appropriate assessment of confidence with any test case. We propose that a key ingredient for introspection is a framework's potential to increase its uncertainty with the distance between a test datum its training data. We compare the introspective capacities of a number of commonly used classification frameworks in both classification and detection tasks, and show that better introspection leads to improved decision making in the context of tasks such as autonomous driving or semantic map generation.

Keywords

Robotics, perception, introspection, classification, uncertainty, decisions

1. Introduction

In robotics, the outputs of our classification frameworks are intended to be used to make decisions. We want the output of a classifier to help the robot decide whether to stop at a traffic light, whether to slow down in front of a pedestrian, or how to populate a semantic map. The processes by which we go from data to decision must be very carefully examined not least when our robots' behavior can impact the safety of humans sharing their workspace.

A common way to improve a robot's interactions is to give it prior information in the form of a semantic map, informing the robot about how its environment behaves and how it can interact with it. In almost all safety-critical applications these maps are hand-made, because the current state-of-the-art solutions to automatic mapping are not robust enough to ensure the high quality of maps required for safe, autonomous robot operation.

Following classical decision theory, in situations where a poor choice of action can incur a large cost (e.g. driving forwards into another vehicle, or incorrectly placing a particular semantic label in a map), a robot will only choose that action if its classifier is supremely certain. In practice, we see that commonly used classification frameworks can assign extremely high certainty or confidence to classifications which turn out to be incorrect. In order to avoid these large costs, it follows that when our classifiers make mistakes they should do so only with high uncertainty.

For example, a classification error can occur when the classifier is presented with a test datum which is unlike anything it saw during training. This could be as a result of the training set not containing the true class of this test datum, or because it is a new viewpoint of an existing class. In the context of autonomous driving, the same parked car can appear very differently given changing weather or time of day. These new and unusual test data are common in practice since the training data can never be fully representative of the continually evolving and complex environments in which our robots operate. In this situation, we argue that the appropriate response is for our classifier to respond with high uncertainty.

There is a tendency to choose one particular classification framework over another based on the standard metrics

[1]Mobile Robotics Group, Oxford University, Oxford, UK
[2]Technical University of München, München, Germany

Corresponding author:
Hugo Grimmett, Mobile Robotics Group, Oxford University, 23 Banbury Road, Oxford OX2 6NN, UK.
Email: hugo@robots.ox.ac.uk

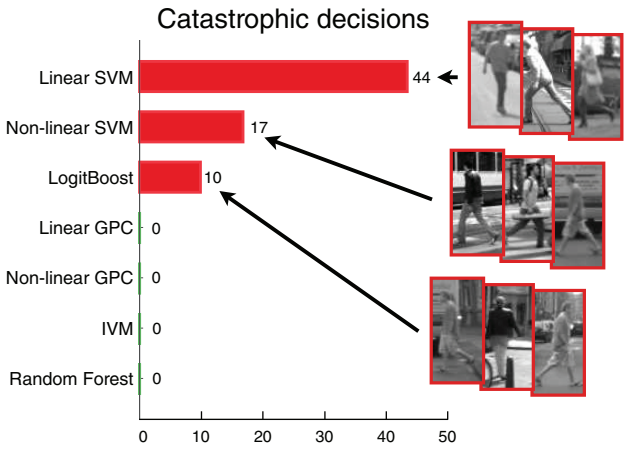


Fig. 1. False-negative classifications lead to decisions which can have terrible consequences. The histogram shows the number of catastrophic decisions made by various classifiers across three data sets. Here, false negatives are instances where a classifier *incorrectly* asserts that $p(\text{obstacle}) < 10^{-5}$. The commonly used linear and non-linear Support Vector Machines (SVM) make 44 and 17 catastrophic decisions respectively out of a total of 11,000, which may seem low, but these represent potential losses of life. The last four classifiers comprise three based on Gaussian Process Classifiers (GPC), and Random Forests, which all made *zero* catastrophic decisions: a perfect score. To the right of the histogram are examples of when the other classifiers elected to drive forward when they mistakenly believed the way was clear (pedestrians comprised one of the three data sets, the others being cars and traffic lights).

of classification: precision and recall. Here we show that these are insufficient to characterize whether a classifier will provide appropriate uncertainties. Without these appropriate uncertainties, our robots are doomed to make the costly and potentially catastrophic decisions shown in Figure 1.

Therefore, rather than using a classifier which makes correct and incorrect decisions with similarly high confidence, it is preferable to use a classifier which makes mistakes only with high uncertainty, and correct classifications with high certainty. Hence, we seek classifiers with the capacity to adjust the confidence of a particular classification on the basis of how 'qualified' they are given their own prior knowledge, embodied by their training data. If a classification framework leads to an overconfident estimate of the class label, then the entire decision-making process may be ineffective. Our work investigates this *introspective* capacity in a number of classification frameworks commonly used in robotics: the Support Vector Machine (SVM), LogitBoost, the Random Forest, the Gaussian Process Classifier (GPC), and the Informative Vector Machine (IVM). We use the term 'introspective' to describe a classifier that gives appropriate probabilities, thus making true classifications with confidence and mistakes only with high uncertainty.

We carefully examine how these classification frameworks use the *distance* between training and test data to moderate the confidence in a classification. Intuitively, a test datum which is far away in feature space from the training data is more likely to be misclassified than one which lies in the middle of a dense cluster from one class, and thus the classification should be made with greater uncertainty. Most classification frameworks make use of a model, such that instead of calculating a distance from the test datum to all of the training data, they calculate a distance between the test datum and the model (which can be further affected by the use of a kernel, effectively warping the feature space). Therefore, the choice of model and the measure of distance greatly affect the uncertainty with which classifications are made. Some frameworks consider one single discriminant to separate the classes, while others average over a variety of possible discriminants. The results we present indicate that the latter tends to be characteristic of classifiers with a better sense of introspection, as a result of their ability to predict the variance from the responses of the individual discriminants for a test datum.

The key contributions of this work are:

- the concept of a classifier's introspective quality, regarding how it expresses the relevance of its prior information when making detections;
- a comparison of how commonly used classification frameworks generate probabilities, and insights regarding whether they are likely to display introspective qualities from a theoretic standpoint;
- results to show the introspective behavior of those classification frameworks when applied to tasks commonly tackled in robotics, such as classification and detection; and
- the further application of those classifiers to decision-making problems, and results which indicate that introspective classification leads to better decision making; this motivates the opinion that considering the introspective quality of classification frameworks is critical in robotics.

All of the above are evaluated using publicly available data sets which are relevant to mobile robotics. Some of this work appeared in Grimmett et al. (2013). Here we present a more detailed treatment of the concepts and substantial evaluation on two additional publicly available data sets, along with the important implications of introspection in terms of decision making.

We start by offering a theoretical insight into why some classification frameworks may exhibit greater introspective qualities than others. We do so by examining the methods by which commonly used algorithms generate probabilities (Section 3), and specifically detailing the key methods for the classifiers we are comparing (Section 4). Then we demonstrate the various behaviors of those classifiers in several scenarios related to autonomous driving. We consider the similar but nuanced cases of *classification*

(estimating the likelihood that an image contains one particular class of object over another, Section 5.3) and *detection* (estimating the likelihood that an image contains one privileged class of object over a background class comprising all other classes, Section 5.4). Finally, we demonstrate the behavior of the classifiers in terms of decision making (in Section 5.5). As we vary the relative costs of false-positive and false-negative errors, and we show that the GPC-based classifiers and Random Forests act most appropriately and consistently as the costs become increasingly imbalanced, in a way that is faithful to the spirit of the cost function.

2. Related work

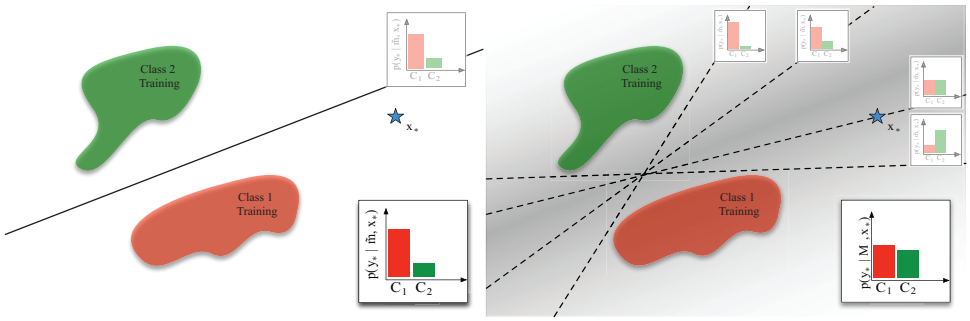
For a number of years now robots have routinely consumed higher-order abstractions from raw sensor data. Successful applications are as diverse as the detection of ground traversability (e.g. Thrun et al., 2006), the detection of lanes for autonomous driving (e.g. Huang and Teller, 2010), the consideration of classifier output to guide trajectory planning and exploration (e.g. Meger et al., 2008; Velez et al., 2011) or the active disambiguation of human–robot dialog (Tellex et al., 2012). These works commonly exploit classification output on a model-trust basis; systems are optimized with respect to precision and recall, and egregious misclassifications (including vastly overconfident marginal distributions obtained from some classification frameworks) are accepted as par for the course. However, the suitability of the classification framework employed with respect to its introspective capacity has not previously been considered in robotics. Thus, we consider motivating, defining, and investigating introspection in a robotics context to be the primary contribution of our work.

The concept of introspection as introduced here is closely related to considerations in active learning, where uncertainty estimates and model selection steps are often employed to guide data selection and gathering for an incremental learning algorithm. Kapoor et al. (2010), for example, present an active learning framework for object categorization using a GPC where classifications with large uncertainty (as judged by posterior variance) lead to a query for a ground-truth label and are subsequently used to improve classification performance. Joshi et al. (2009) address multiclass image classification using SVMs and propose criteria based on entropy and best-versus-second-best measures (see Section 3.2) for disambiguating uncertain classifications. Holub et al. (2008) propose an information-theoretic criterion that maximizes expected information gain with respect to the entire pool of unlabeled data available. Hospedales et al. (2013) discuss optimizing rare class discovery and classification using a combination of generative and discriminative classifiers. In the related field of reinforcement learning, Li et al. (2008) present a general framework which determines whether enough labeled data have been provided to constrain certain problems. If the learners space of solutions is insufficiently constrained such that its output cannot be guaranteed to be within ϵ of the true solution with probability $1 - \delta$, it asks for more labeled data. This accuracy guarantee is the same for both false-positive and false-negative errors, and thus the framework is not appropriate for situations in which costs associated with those errors are imbalanced. In the context of autonomous systems, the costs are commonly imbalanced.

Our treatment of introspection is further informed by an ongoing discussion in the machine learning community regarding how best to account for variance in the space of feasible classifier models when training on, essentially, an incomplete set of data. For example, Tong and Koller (2002) present an incremental algorithm for text classification using SVMs and the notion of a *version space*, the set of consistent hyperplanes separating the data in a feature space induced by the kernel function. Zhang et al. (2012) introduce a max-margin classifier achieving better generalization to unseen test data given a limited training corpus. Here, distinctiveness of training instances is assessed using the local classification uncertainty. A global classifier then incorporates these uncertainties as margin constraints, yielding a classifier that places less confusing instances farther away from the global decision boundary. We share the intuition that accounting for variance in version space when selecting a model leads to an increased introspective capacity. As a secondary contribution, therefore, our results serve to further corroborate this intuition.

The semantic mapping of a robot's workspace has become a popular line of research in recent years. A rich body of work now exists in which semantic labels are generated based on a variety of sensor modalities and classification frameworks (e.g. Anguelov et al., 2005; Douillard et al., 2008; Martínez-Mozos et al., 2007; Paul et al., 2012; Posner et al., 2009; Pronobis and Jensfelt, 2012; Sengupta et al., 2012). We consider introspection to be paramount to reducing the human effort required to automatically generate semantic maps which we can then use for autonomous operation.

Niculescu-Mizil and Caruana (2005) recognize that the question of whether the probabilities produced by various classification frameworks are appropriate is important, a sentiment we clearly share. They conclude that poorly calibrated frameworks (in a probabilistic sense) can be effectively corrected using an additional learned calibration using either Platt's method or isotonic regression. They find Random Forests to perform well pre-calibration (although they did exhibit a tendency to be underconfident, consistent with our findings), and that SVMs perform well after post-calibration. They associate the need for further calibration specific to the classifiers using max-margin optimization, rather than the treatment of distances in feature space and the distribution of models over version space, as we do. They also do not explore the effects of making decisions using these probabilities.



(a) (b)

Fig. 2. An illustration of the two types of classification frameworks considered: (a) during training a *single* model is selected to classify an unknown datum $\mathbf{x}_*$ some way removed from the training data; (b) training leads to a distribution over models which is considered entirely to arrive at the final prediction. This illustration is for the family of linear models (indicated by solid (a) and dashed (b) lines). Each predictor is further annotated with its individual prediction. The overall predictive distribution is shown in the bottom right of each subplot. The shading in part (b) indicates the probability weights associated with individual models. Darker regions contain more weight. Note that the overall predictive distribution in (a) stems from the single model used and is, in this case, inappropriately confident. In part (b), however, the overall predictive distribution is moderated by computing the expectation over all models. This gives rise to a much more appropriate uncertainty estimate: the introspective quality we seek.

Berczi et al. (2015) have confirmed the introspective power of GPCs over SVMs, employing them to avoid areas of terrain for which the height may be misclassified.

3. Introspection, uncertainty, and decision making

In this section we first describe a crucial property we expect classification frameworks to require in order to be introspective: marginalization over possible models (Section 3.1). Then we describe some measures of uncertainty, motivating the use of normalized entropy as the most appropriate measure (Section 3.2). We finish by describing a manner in which to obtain decisions from probabilistic classification results, and motivate the practice of choosing outcome costs directly rather than adjusting thresholds to modify a robot's behavior (Section 3.3).

3.1. Introspective capacity

Introspection concerns not the final class decision but rather the confidence with which this decision is made. The concept is motivated by the desire to take appropriate action when a classifier indicates high uncertainty. Our approach to introspection is grounded in the fact that the often-cited assumption of independent and identically distributed (*iid*) training and test data is routinely violated in robotics; in the limit of continuous operation in the real world, one-shot classifier training is rarely performed on a complete (or even fully representative) set of data.

Let a classifier map an input $\mathbf{x} \in \mathbb{R}^d$ to one of a set of classes $\mathcal{C} = \{C_1, \ldots, C_c\}$ via an associated label $y \in \{1, \ldots, c\}$, where c is the number of classes. Prior to training, domain-specific knowledge is often used to constrain the family of classification models employed (for example in the form of a kernel or a type of base classifier). Classifier training then involves learning a set of (hyper)parameters given a training data set $\{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denotes the set of N feature vectors and $\mathcal{Y}$ denotes the set of corresponding class labels. The training data implicitly give rise to a probability distribution over the set of all possible models (or *hypotheses*) within the chosen family, $\mathcal{M}$, such that

$$\{\mathcal{X}, \mathcal{Y}\} \rightarrow p(m|\mathcal{X}, \mathcal{Y}), \quad m \in \mathcal{M} \tag{1}$$

With a slight abuse of notation, m here denotes any member of the family of possible models, $\mathcal{M}$. In the following we make this relationship explicit by conditioning on both a model (or family of models) as well as on a test datum $\mathbf{x}_*$. Typically, training leads to the selection of a *single* model, $\tilde{m}$ from $\mathcal{M}$ such that a prediction y_* for a new, unseen feature vector $\mathbf{x}_*$ is obtained by approximating

$$p(y_*|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_*|\tilde{m}, \mathbf{x}_*), \quad \tilde{m} \in \mathcal{M} \tag{2}$$

This is illustrated in Figure 2(a). Common examples of this type of classification framework include SVMs and Boosting classifiers, where an optimization is performed to select the best model given the training data (see Section 4). The *iid* assumption here is inherent since it is assumed that $\tilde{m}$ remains the best model for all predictions of unseen data. Breaking this assumption therefore often renders the chosen model suboptimal.

An alternative to the single model approach is a classification framework which takes into account the *entire set* of possible models in the specified family conditioned on the training data, such that

$$p(y_*|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_*|\mathcal{M}, \mathbf{x}_*) \tag{3}$$

This case is illustrated in Figure 2(b). Here the shading indicates the distribution $p(m|\mathcal{X}, \mathcal{Y})$ with darker shades indicating increased probability. To aid intuition,

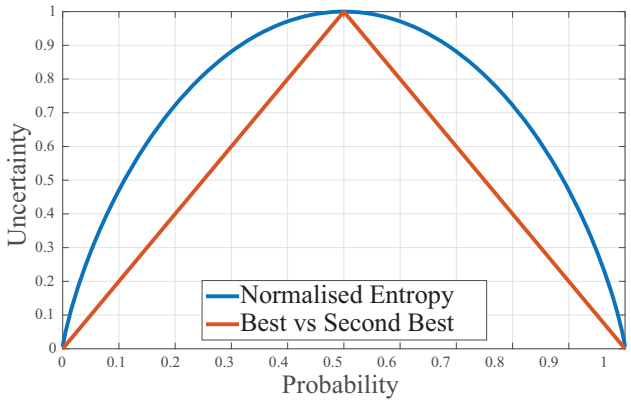


Fig. 3. Normalized entropy and best-versus-second-best as measures of uncertainty in the binary classification case.

predictions of four randomly selected members of $\mathcal{M}$ are also illustrated. Final predictions are made by taking into account opinions from all members of $\mathcal{M}$, often via the computation of an expectation such as for a GPC (see Section 4). Crucially, when considering an expectation over all of $\mathcal{M}$, the increased variance in feasible (and therefore likely) models at a distance from the training data leads to a moderation of the class predictions.

Between the two extremes lies the Random Forest, which chooses a number of differing samples from $\mathcal{M}$, and averages over the responses from these.

We believe that this marginalization over plausible models in version space is a key component of an introspective classifier.

3.2. Characterizing uncertainty

In order to characterize the introspective capacity of a classification framework, a well-tempered measure of the inherent uncertainty in the classification output is required. For this purpose, we use an information-theoretic quantity known as normalized entropy, H_{N}, defined as

$$H_{\mathrm{N}} = -\sum_{i=1}^{c} p(y = C_i \mid \mathbf{x}) \log_c [p(y = C_i \mid \mathbf{x})] \qquad (4)$$

This is equivalent to the Shannon entropy measure normalized by its maximum, which is the entropy of the c-dimensional uniform distribution, $log(c)$. The result is a measure ranging between 0 and 1 where a *higher* value indicates *greater* uncertainty in the classifier's belief, as shown by the blue curve in Figure 3.

An alternative uncertainty measure proposed in the active learning literature is the best-versus-second-best (BvSB) heuristic (Joshi et al., 2009) which equals one minus the difference between the largest and the second largest class likelihood estimates, as shown by the orange curve in Figure 3. This measure attempts to characterize the reliability of the maximum likelihood estimate rather than encoding the shape of the full distribution over class labels. The BvSB and normalized entropy measures are closely related in the binary classification setting, which is that of this paper. We use normalized entropy throughout the remainder of this work for two reasons: first, it is formed from an information-theoretic point of view, compared to BvSB which is an ad hoc heuristic; second, in multiclass settings it considers the entire distribution over classes, rather than BvSB which only takes into account the two classes with the highest probability.

3.3. Decision making

Autonomous robots typically have at their disposal a set of actions, each of which is appropriate to particular situations. The difficulty lies in choosing which action to perform when there is uncertainty about the state of the world. Following standard decision theory (LaValle, 2006), we calculate the expected loss of performing a particular action when we have a set of likelihoods for each state of the world $(p(C_1), p(C_2), \ldots, p(C_{|\mathcal{C}|}))$, defined as

$$\bar{L}(a) = \sum_{i=1}^{|\mathcal{C}|} L(a, C_i) p(C_i) \qquad (5)$$

where $L(a, C_i)$ is the cost or loss associated with each potential outcome. We then choose to perform the action a which minimizes this expected loss.

For our decision-making experiments, we later consider a simple scenario in which there are two states the world can be in: either there is an object in the way (C_2, e.g. a pedestrian, car, or traffic light), or there is not (C_1). There are also two available actions $a \in \{\text{stop, go}\}$. We wish our robot to *stop* if there is an object in its path, or *go* if the way is clear. The losses will vary from application to application, but in the case of autonomous driving it is sensible to associate a very high cost to performing the *go* action when there is in fact an object in the way (C_2), resulting in a collision, and a lesser cost to performing the action *stop* when the path is clear (C_1), resulting in an unnecessary delay. While inefficient, this false-positive error is more desirable than running a red light or colliding with another vehicle. Interestingly, in the case of driver assistance systems (e.g. automatic emergency braking) the costs are reversed: the loss associated with a false positive (an unnecessary emergency stop) is very large, and a false negative (a missed opportunity to perform an emergency stop) is a less undesirable outcome. In Figure 4(a) we show the expected losses of the two actions when there is equal cost associated with each type of error. We can see that the intersection between the two lines occurs at $p(C_1) = p(C_2) = 0.5$. As we increase the cost of a false negative (performing the *go* action when there is a person, C_2), the range of detection probabilities $p(C_2)$ which result in a *go* action reduces, as seen in Figure 4(b). Since an introspective classifier is uncertain when it makes mistakes, the errors will be close to $p(C_2) = 0.5$ as the teal distribution in Figure 4(c), and so those errors will

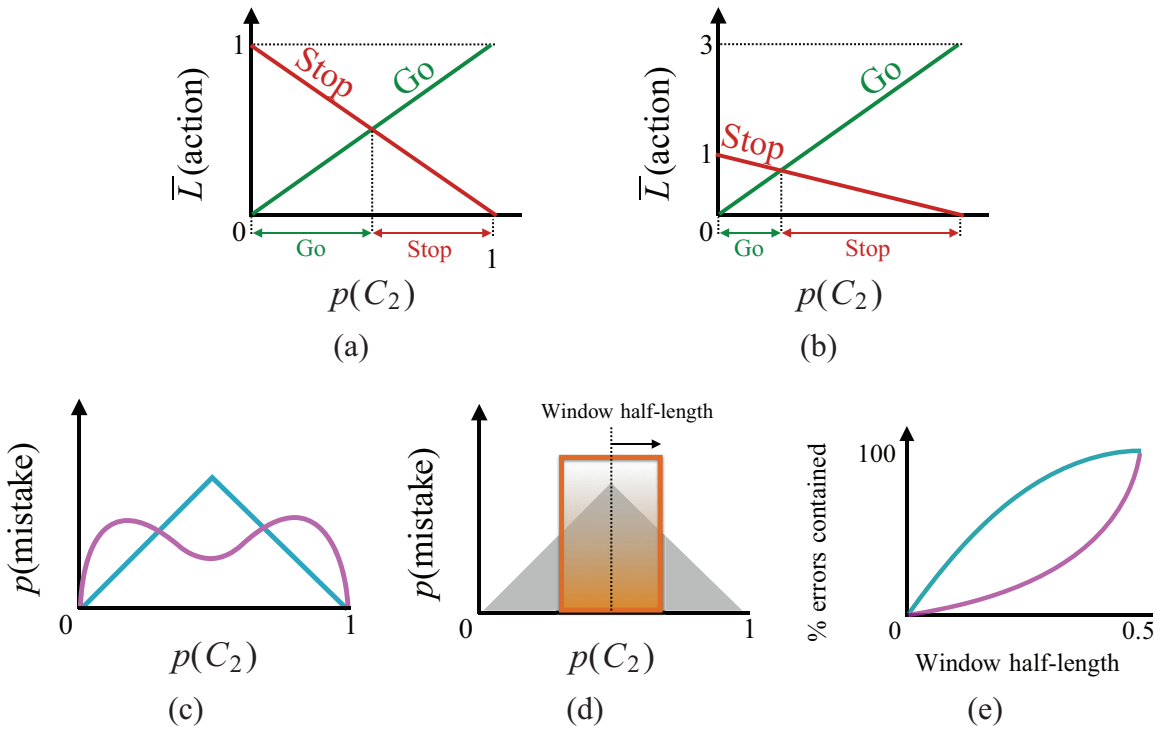


Fig. 4. (a) We have set equal cost to a false positive (take the *stop* action when there is, in fact, no person: C_1) and a false negative (take the *go* action when there is a person: C_2). The expected losses from the two actions meet at $p(C_1) = p(C_2) = 0.5$, and choose the action which minimizes the expected loss. (b) We have made the cost of a false negative three times the cost of a false positive, which reduces the probability region for which we choose the *go* action. By increasing the cost of accidentally hitting a pedestrian, we are trying to create a more cautious system, which will take the *stop* action more of the time. (c) A more introspective classifier (teal) will make most of its mistakes with high uncertainty, when $p(C_2)$ is near 0.5. Less introspective classifiers (purple) will make mistakes with low uncertainty. (d) As we grow the orange box outward from the center, we can calculate how many errors are contained for a particular distribution in (c). (e) We show the result of plotting the number of errors contained as we grow the orange box for the two idealized classifiers in (c). The teal (more introspective) classifier catches more errors when the box is small than for the purple (less introspective) classifier. It also reaches a steady state because there are very few errors around $p(C_2) = 0$ and $p(C_2) = 1$, when the classifier is confident.

largely be subsumed by the *stop* action. A less introspective classifier will make more mistakes near the extremes of the class marginal spectrum (purple in Figure 4(c)) and so more of those errors will occur in the *go* region, resulting in a greater prevalence of very expensive errors.

Thus, ideally, as we make the cost of a false negative much greater than that of a false positive, our classifiers become more and more cautious, employing safer actions and incurring less overall cost. Crucially, this relies upon the assumption that most of a classifier's mistakes lie in the middle of the probability spectrum. Another way to characterize the desirable introspective property is to consider the proportion of the errors contained in some window around $p(C_2) = 0.5$, represented by the orange box in Figure 4(d). As we increase the half-length of the box from 0 to 0.5, we would like the proportion of errors to increase quickly, and then stabilize as we encompass the regions of high confidence, represented by the teal curve in Figure 4(e). A less introspective classifier would have errors near $p(C_2) = 0$ or $p(C_2) = 1$, and so the contained errors would resemble the purple curve in Figure 4(e). We will present curves resembling these for each classifier from real data in Section 5.5.

Often, the temptation is to tune the costs to steer the robot towards the 'desired' behavior. Instead, we ought to focus on whether the costs are correct (because it is usually easier to quantify these than a probability threshold) and allow the decision theory to choose the behavior which is true to the spirit of the cost function. This is only possible if the probabilities and uncertainties supplied by the classification frameworks are sensible.

4. Classification frameworks

We now present a brief overview of the specific classification frameworks considered in this work: SVMs, LogitBoost, Random Forest, GPCs, and the IVM. The implementations of these are all off-the-shelf, using popular libraries detailed in each subsection. The goal is not to find the most accurate classifier, but rather to examine the consistency of the confidences with which certain decisions are made. We believe that this consistency in choosing the appropriate decision given the potential losses is an often-ignored and paramount characteristic of classification frameworks, and that in the context of safety-critical tasks

it could be worth accepting a decrease in accuracy if it results in an increased introspective consistency. In the following descriptions of the frameworks, we focus on properties pertinent to introspection, specifically how the use of distance between data affects the classification confidence, and what type of models they use. For simplicity, but without loss of generality, this work considers predominantly binary classification such that $\mathcal{C} = \{C_1, C_2\}$. For consistency we adhere to notation commonly found in the literature where a discriminant function is often denoted as $f(\cdot)$. We note that this is equivalent to a particular model m as described in the previous section.

4.1. Support vector classification

SVM classification is well established in robotics so that here we provide only an overview. For a detailed account the reader is referred to e.g. Burges (1998). SVMs are based on a linear discriminant framework which aims to maximize the margin between two classes. The model parameters are found by solving a convex optimization problem, thereby guaranteeing the final classifier to be the best feasible discriminant given the training data. Once training is complete, predictions on future observations are made based on the signed distance of the observed feature vector from the optimal hyperplane, defined by the weight vector $\mathbf{w}$ and bias w_0, such that

$$f(\mathbf{x}_*) = \mathbf{w}^\top \phi(\mathbf{x}_*) + w_0 = \sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_*) + w_0 \qquad (6)$$

where N is the size of the training set, α_i refers to a Lagrange multiplier associated with datum i, w_0 denotes a bias parameter, ϕ refers to the feature map, and $k(\mathbf{x}_i, \mathbf{x}_j)$ denotes the kernel function.

The parameters α_i and w_0 characterizing the discriminant function are obtained by an optimization procedure, and α_i is then non-zero only for *support vectors* $\mathbf{x}_i$. The SVM algorithm selects a particular weight vector (as defined by the *support vectors*), which gives rise to a *maximum* margin separator.

The kernel function amounts to a scalar product between two data, which have been transformed from d-dimensional feature space into some higher dimensional space. The nature of this mapping between spaces is inherent in the choice of kernel and need not be specified explicitly (the kernel trick). The regularization and kernel parameters are learned using 10-fold cross-validation. We discuss our choices of kernel function in Section 4.6.

In its original form, the SVM classifier output is an uncalibrated real value. A common means of obtaining a probabilistic interpretation is by using Platt's method (Platt, 1999). This algorithm was later improved by Lin et al. (2007) and is implemented in the library we use for all SVM training, calibration, and testing, LIBSVM (Chang and Lin, 2011). Here, using a hold-out set not used for classifier training, a parametric sigmoid model is fit directly to the class posterior $p(y_* = C_2|f(\mathbf{x}_*))$, such that

$$p(y_* = C_2|f(\mathbf{x}_*)) = \frac{1}{1 + \exp(Af(\mathbf{x}_*) + B)} \qquad (7)$$

The sigmoid parameters A and B are determined using Newton's method with backtracking line search. Note that class likelihoods are derived here using only a *single* estimate of the discriminative boundary obtained from the classifier training procedure. No other feasible solutions are considered. Hence, the predictive variance of the discriminant $f(\mathbf{x})$ is not taken into account while determining probabilistic output (Rasmussen and Williams, 2006). Although there is no guarantee that the method converges, in general it works very well and finds the global optimum owing to the convexity of the objective function.

4.2. LogitBoosting classifiers

Boosting is a widely used classification framework which involves training an ensemble of weak learners in sequence. The error function used to train a particular weak learner depends on the performance of the previous models (Bishop, 2006). Each weak learner $h(\mathbf{x})$ is trained using a weighted form of the data set in which the weights depend on the performance of the previous classifiers. Predictions from a boosted classifier are obtained using a weighted combination of the individual weak learner outputs such that

$$\text{sign}(f(\mathbf{x}_*)) = \text{sign}\left(\sum_{i=1}^{M} w_i h_i(\mathbf{x}_*)\right) \qquad (8)$$

where M is the number of weak learners used.

LogitBoost (Friedman et al., 1998) is a popular choice for a Boosting-based classifier as it natively outputs class probability estimates following a calibration via a sigmoid. Weak learners are often chosen to be decision trees and training is conducted by fitting additive logistic regression models by stage-wise optimization (using Newton steps) of the Bernoulli log-likelihood. The algorithm works in the logistic framework and yields a predictor function $f(\mathbf{x})$ learned from iterative hypothesis training. Cross-validation is used to set parameters like the learning rate, tree depth, and the number of Boosting rounds. The class-conditional probabilities are obtained from the predictor function via

$$p(y_* = C_1 \mid \mathbf{x}_*) = \frac{\exp(f(\mathbf{x}_*))}{\exp(f(\mathbf{x}_*)) + \exp(-f(\mathbf{x}_*))} \qquad (9)$$

which is the same sigmoid used in the SVM in Section 4.1 with parameters $A = -2$ and $B = 0$. The procedure possesses asymptotic optimality as a maximum likelihood predictor (Friedman et al., 1998; Hastie and Tibshirani, 1986). However, the method of converting the output of the predictor function to class-conditional probabilities is not fully

probabilistic and does not account for variance in the underlying predictor function. In our experiments we use 500 learners for training. Throughout this work we use the MATLAB implementation of LogitBoost for classifier training and testing.

Because the LogitBoost classifier ultimately settles on a single decision boundary across the input space, we expect that it will suffer from the same introspective issues as the SVM.

4.3. Gaussian process classification

Binary classification using a Gaussian process (GP) (Rasmussen and Williams, 2006; Williams and Barber, 1998) is formulated by first introducing a *latent* function $f(\mathbf{x})$ and then applying a sigmoid function Φ (similar to the sigmoid described in Section 4.1, except that the predictive variance of the GP is used as well as the predictive mean) to obtain the prediction $p(y_* = C_1|\mathbf{x}_*) = \Phi(f(\mathbf{x}_*))$. A GP prior is placed on the latent function $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ characterized by a *mean* function $\mu(\mathbf{x})$ and a *covariance* (or kernel) function $k(\mathbf{x}, \mathbf{x}')$. GPC training establishes values for the hyperparameters specifying the kernel function k by maximizing the log marginal likelihood of the training data.

Probabilistic predictions for a test point are obtained in two steps. First, the distribution over the latent variable corresponding to the test input is obtained using

$$p(f_*|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int p(f_*|\mathcal{X}, \mathbf{x}_*, f)p(f|\mathcal{X}, \mathcal{Y})\mathrm{d}f \qquad (10)$$

where $p(f|\mathcal{X}, \mathcal{Y}) = p(\mathcal{Y}|f)p(f|\mathcal{X})/p(\mathcal{Y}|\mathcal{X})$ is the posterior distribution over latent variables. This is followed by *marginalizing* over the latent f_* to yield the class likelihood $p(y_* = C_1|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$ as

$$p(y_* = C_1|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int \sigma(f_*)p(f_*|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*)\mathrm{d}f_* \qquad (11)$$

Exact inference is analytically intractable due to the sigmoid likelihood function. Instead, we leverage Expectation Propagation (EP) (Minka, 2001), a method widely used for this purpose.

The GPC framework offers two key benefits over the other approaches discussed here (Rasmussen and Williams, 2006). First, the classification output has a clear probabilistic interpretation as it directly results in the class likelihood. In contrast, neither the SVM nor the Boosting framework provides an inherently probabilistic output in the Bayesian sense, but instead estimates a suitable calibration. Secondly, and crucially, the GP formulation addresses uncertainty or *predictive variance* in the latent function $f(\mathbf{x})$ via *marginalization* (or averaging) over all models induced by the training set resulting in the estimate $p(y_* = C_1|\mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$ from equation (11). This process also gives rise to the well-known property of increased variance while far away from the data in GP regression. Again, this is in contrast to the SVM or boosting estimate $p(y = C_i|\hat{f}, \mathbf{x}_*)$ that relies on a single-discriminant estimate $\hat{f} : \mathcal{X} \to \mathcal{Y}$ learned via minimization. In the context of introspection, the ability to account for predictive variance is a key advantage of Bayesian classification approaches. Throughout this work we use the Gaussian Processes for Machine Learning (GPML) toolbox (Rasmussen and Nickisch, 2010) for GPC training and testing.

4.4. The informative vector machine

A key drawback of a GPC is its significant computational demand in terms of memory and run time during training and testing, more than any of the other frameworks considered here. This is due to the fact that the GP maintains a mean $\boldsymbol{\mu}$, as well as a covariance matrix Σ, which is computed from a kernel function and is of size $N \times N$. A number of sparsification methods have been proposed in order to mitigate this computational burden. For efficiency, in this work we adopt one such sparsification method: the Informative Vector Machine (IVM) (Lawrence et al., 2003). The main idea of this algorithm is to only use a subset of the training points denoted the *active set*, $\mathcal{I}$, from which an approximation $q(f|X, \mathbf{y}) = \mathcal{N}(f|\mu, \Sigma)$ of the posterior distribution $p(f|X, \mathbf{y})$ is computed. The IVM algorithm computes $\boldsymbol{\mu}$ and Σ incrementally, and at every iteration j selects the training point $(\mathbf{x}_k, y_k)$ which maximizes the entropy difference ΔH_{jk} between q_{j-1} and q_j for inclusion in the active set. As q is Gaussian, ΔH_{jk} can be computed by

$$\Delta H_{jk} = -\frac{1}{2}\log|\Sigma_{jk}| + \frac{1}{2}\log|\Sigma_{j-1}| \qquad (12)$$

We use an efficient form of this, the details of which can be found in Lawrence et al. (2005). The algorithm stops when the active set has reached a desired size. We choose this size to be a fixed fraction q of the training set, which we set to be 0.4. Throughout this work we use the IVM MATLAB toolbox (`http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/mltools/`) for both training and testing.

To find the kernel hyperparameters $\boldsymbol{\theta}$ of an IVM, two steps are processed in a loop for a given number of times: estimation of $\mathcal{I}$ from $\boldsymbol{\theta}$ and minimizing the marginal likelihood $q(\mathbf{y}|X)$, thereby keeping $\mathcal{I}$ fixed. Although there are no convergence guarantees, in practice a small number of iterations is sufficient to find good kernel hyperparameters.

Importantly for our work, since inference with the IVM is similar to that with a GPC, the IVM retains the model averaging described in equation (11). We argue, therefore, that the IVM provides a significant and well-established improvement in processing speed over a GPC while maintaining its introspective properties (see Section 5 for details).

4.5. Random Forests

Random Forests (Breiman, 2001) are made up of an ensemble of decision trees generated via bagging. Bagging (a portmanteau of 'bootstrap aggregating') involves creating multiple classifiers using different subsets of some aspect of the training data, in this case two aspects are bagged simultaneously: the training data, and the feature dimensions. During testing, the output $p(C_2)$ is the fraction of the individual trees which classified the datum as being from that class.

The trees contain multiple binary nodes or branches, each of which thresholds on a particular feature dimension of the data, learning the threshold which helps split the training data into the two classes. We have set each tree to use a number of feature dimensions equal to the square root of the total number, as recommended by the literature, with a total of 500 trees. Throughout this work we use the Bagged Decision Tree functions in the MATLAB statistics toolbox (which is an implementation of Random Forests) for both training and testing.

This combination of many differing decision boundaries (one boundary per tree) represents a sampling and then averaging over the version space, similar to the marginalization over version space which takes place in the GPC. A crucial difference is that in the GPC, each possible model is weighted by its likelihood, and in Random Forests each tree is weighted equally. However, these trees are carefully selected to separate the chosen subset of training data, so this biasing is in a sense a $\{0, 1\}$ weighting. This could be thought of as sampling 500 decision boundaries from the shaded region in Figure 2(b) and taking an expectation over their responses. This suggests that they should behave in a more introspective manner than the other single-discriminant frameworks like LogitBoost or SVMs, but perhaps a more sensitively weighted combination of the trees could perform better.

4.6. Kernel types

Evaluation of the discriminant function for an SVM and the covariance matrix for GPC inference requires the specification of a kernel function, $k(\cdot, \cdot)$. A rich body of literature exists on different choices of kernels for both frameworks. However, since our focus here is on a like-for-like comparison of different classification frameworks we choose two representative kernels rather than performing exhaustive model selection to optimize performance for a particular application. Firstly, as an example of the simplest kernel function available, we consider the linear kernel defined as

$$k_{\mathrm{LIN}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + r \tag{13}$$

where r is a constant real number. The linear kernel is an apt choice where a linear separation of the data in feature space leads to adequate performance, or where computational efficiency is of the essence. Often, however, a more sophisticated, non-linear kernel is required. In this category we use the Squared Exponential (SE) function as a canonical representative. The SE kernel with length scale parameter l is defined as

$$k_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2l^2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{14}$$

In the context of an SVM, the SE function is more commonly known as a Radial Basis Function (RBF).

5. Experimental results

Our experiments investigate the introspective capacity of the classifiers introduced in Section 4 in settings relating to autonomous driving. Specifically, we focus on two tasks: the *classification* of cropped images of road signs, and the *detection* of a salient class against a broad background class. For the detection case, we repeat our experiments across the three data sets detailed in Section 5.1, which together contain traffic lights, cars, and pedestrians. In investigating both classification and detection we aim to address the full spectrum of applications commonly encountered in robotics. Classification addresses the case where a decision is made between two, well-defined classes (e.g. two types of traffic signs). We investigate classifier performance when a third, previously unseen class is presented. The detection case is arguably more commonplace, where a single class is separated from a broad (in terms of intra-class variation) *background* class (which is relevant in semantic mapping or detection). Here, the concept of a previously unseen class does not exist explicitly: the inherent assumption is that the data representing the background class capture any non-class object likely to be encountered. In practice this is rarely true, leading to a significant number of novel instances which often result in misclassification. While it could be argued that this issue is ameliorated somewhat by expanding the data set used for training, we propose that the complexity of the feature space encountered during persistent, long-term autonomy will keep perplexing even the most expansively trained classifiers.

We then apply a decision-making process to the classifiers trained for detection, and show how the quality of each classifier's decisions change depending on the values chosen for the cost function.

We finish by examining the uncertainty with which each classifier makes errors, and compare the idealized drawings from Figure 4 to the real curves generated from each of the three data sets.

5.1. Data sets

In order to demonstrate the consistency of the introspective capacities of the various frameworks, we evaluate our experiments on several commonly used data sets which encompass several domains of robotics, namely the detection of various key classes on the road.

Table 1. The seven classes of the German Traffic Sign Recognition Benchmark (GTSRB) data set considered in our work. The numbers in brackets indicate the number of data available per class.

						
roadworks ahead (1500)	*right ahead* (688)	*stop* (780)	*keep left* (298)	*lorries prohibited* (420)	*speed limit* (1980)	*yield* (2159)

Table 2. The parameters for the features for the TLR and GTSRB (using Torralba features) and the DP and KITTI data sets (using HOG features).

Parameter	TLR	GTSRB	DP	KITTI
Cropped image height	30	32	96	26
Cropped image width	12	32	48	32
HOG cell size	n/a	n/a	10	10
No. of orientations	n/a	n/a	5	6
Final feature dimension	200	200	950	198

5.1.1. Traffic Lights Recognition. The Traffic Lights Recognition (TLR) data set (`http://www.lara.prd.fr/benchmarks/trafficlightsrecognition`) is a sequence of color images taken by a monocular camera from a car driving through central Paris. The TLR data set comprises just over 11,000 frames, in which most of the traffic lights have been labeled with bounding boxes and further metadata such as the status of the signal or whether a particular label is ambiguous (e.g. the image suffers from motion blur, the scale is inappropriate, or a traffic light is facing the wrong way). A few traffic lights have been omitted altogether. As recommended by the authors, we exclude from our experiments any labels of class *ambiguous* or *yellow signal* and any instances which are partially occluded. We split the data set into two parts (at frame 7200 of 11,178), with an approximately equal number of remaining labels in each part and with no physical traffic lights in common. Positive data are extracted as labeled. Negative *background* data are extracted by sampling patches of random size and position from scenes in the data set while ensuring that the patches do not overlap with positive instances.

5.1.2. German Traffic Sign Recognition Benchmark. The German Traffic Sign Recognition Benchmark (GTSRB) data set (Stallkamp et al., 2012) comprises over 50,000 loosely cropped images of 42 classes of road signs, with associated bounding boxes and class labels. From this data set we specifically focus on the seven classes shown in Table 1. The images are resized according to the parameters in Table 2, and then we use the Torralba features from Section 5.2 for classification.

5.1.3. Daimler Pedestrian. The examples we use come from the Daimler multicue-occluded Pedestrian data set (DP) (Enzweiler et al., 2010), and we use the non-occluded monocular intensity images. There are over 52,000 positive and 32,000 negative examples split into training and test sets. The images are resized according to the parameters in Table 2, and then we use the HOG features from Section 5.2 for classification.

5.1.4. KITTI. The KITTI data set (Geiger et al., 2012) comprises over 7400 non-sequential color images from a camera pointing out from the front of a car driving through a German city. The images come with ground-truth information for vehicles, with up to 15 in each frame. The images are cropped and resized according to the parameters in Table 2, and then we use the Histogram of Oriented Gradients (HOG) features from Section 5.2 for classification.

5.2. Features

A rich body of work on the detection and classification of road signs and traffic lights has established a successful track record of template-based features for this purpose. Specifically, we leverage the approach proposed by Torralba et al. (2007) in which a dictionary of partial templates is constructed, against which test instances are matched. A single feature consists of an image patch (ranging in size from 8×8 to 14×14 pixels) and its location within the object as indicated by a binary mask ($h \times w$ pixels according to Table 2). For any given test instance, the normalized cross-correlation is computed for each feature

Table 3. The classification performance when separating *stop* signs from the *lorries prohibited* signs from the GTSRB data set. Note that different class combinations were found to yield classifiers of similar quality.

Classifier	Precision	Recall	F_1
IVM	1.000	1.000	1.000
Non-linear GPC	1.000	1.000	1.000
Linear GPC	1.000	1.000	1.000
Non-linear SVM	1.000	1.000	1.000
Linear SVM	1.000	1.000	1.000
LogitBoost	1.000	1.000	1.000
Random Forest	1.000	1.000	1.000

in the dictionary. Therefore, per instance (or window, in the detection case) a feature vector of length d is obtained, where d is the size of the dictionary. We found empirically that $d > 200$ leads to a negligible performance increase in classification. Throughout our experiments we therefore set $d = 200$.

For the Daimler Pedestrian and KITTI data sets, we have chosen to use HOG (Dalal and Triggs, 2005) features because the classes in question (pedestrians and cars, respectively) have much greater variation than traffic lights, and so a gradient-based feature method performs better than a template-matching-based method, which is more appropriate for classes with consistent appearance. We use the implementation in *vlfeat* (Vedaldi and Fulkerson, 2010) and use the parameters detailed in Table 2.

5.3. Introspection in classification

This section investigates classification output when the classifiers are trained on two classes, and then a third, previously unseen class is presented to the classifier. This is an important experiment because classifiers deployed in real-world applications will encounter images which do not closely resemble the data used to train them, and an introspective classifier will respond to these with high uncertainty. As examples of classes typically encountered in autonomous driving applications we use a subset of the GTSRB data set (see Section 5.1.2). We arbitrarily select two classes for training: *stop* and *lorries prohibited*. To investigate the efficacy of the features used and training procedures employed, classifiers were trained separating these two classes using a balanced training set of 400 data (200 per class). Classifier performance was evaluated using standard metrics on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The results are shown in Table 3, and show that classification performance by the commonly used metrics (precision, recall, and F_1 measure) is commensurate across all classifiers. The corresponding Precision–Recall (PR) curve confirms the perfect separation of the classes and has been omitted here as it is otherwise uninformative.

The classifiers are then tested on 200 instances of the previously unseen classes *roadworks ahead, keep left, 70 kph, yield*, and *right ahead*. The normalized entropy histograms for both the seen and the unseen test classes are shown in Figure 5.

All classifiers are confident when tested on classes which were present in the training set, which is what we would expect. For the unseen test classes, the mean normalized entropies for the GPC-based classifiers (IVM, non-linear GPC, and linear GPC) and the Random Forests are more consistently high than those of the other classification frameworks, indicating that they reliably exhibit greater uncertainty in their judgment. Conversely, the LogitBoost classifier is extremely confident in all of its classifications with a very narrow distribution, and the non-linear and linear SVMs have inconsistent levels of uncertainty. These are effects consistently observed throughout our experiments, which we attribute to the manner in which the probabilities are estimated (as detailed in Section 4). The unseen sign for which the classifiers respond with the lowest uncertainty (greatest confidence) is the 70 kph sign. We propose that this is due to its similarity with one of the training classes, namely the 'lorries prohibited' sign.

In order to mitigate any influence of the specific training and test data selected we repeated the above experiment across a number of random dictionaries, data samples, and unseen classes. Specifically, for each of 5 different unseen classes, we perform 40 iterations of classifier training and testing with a random dictionary and training and test data sets resampled for each run. The results, presented in Table 4, are consistent with those in Figure 5 in that the GPCs and Random Forest tend to be more consistently uncertain for the unseen test classes, while SVM and LogitBoost are more confident with an often significantly narrower distribution of normalized entropy values. The results in Table 4 indicate that the gap in uncertainty between the different frameworks is more pronounced for some unseen classes than for others. We attribute this to the varying degree of similarity in feature space between some unseen class and the classes in the training set.

We draw the conclusion that when faced with test data that are not represented by the training data, the GPC-based classifiers and Random Forest are more consistently uncertain than the other classifiers, which is the introspective behavior we seek.

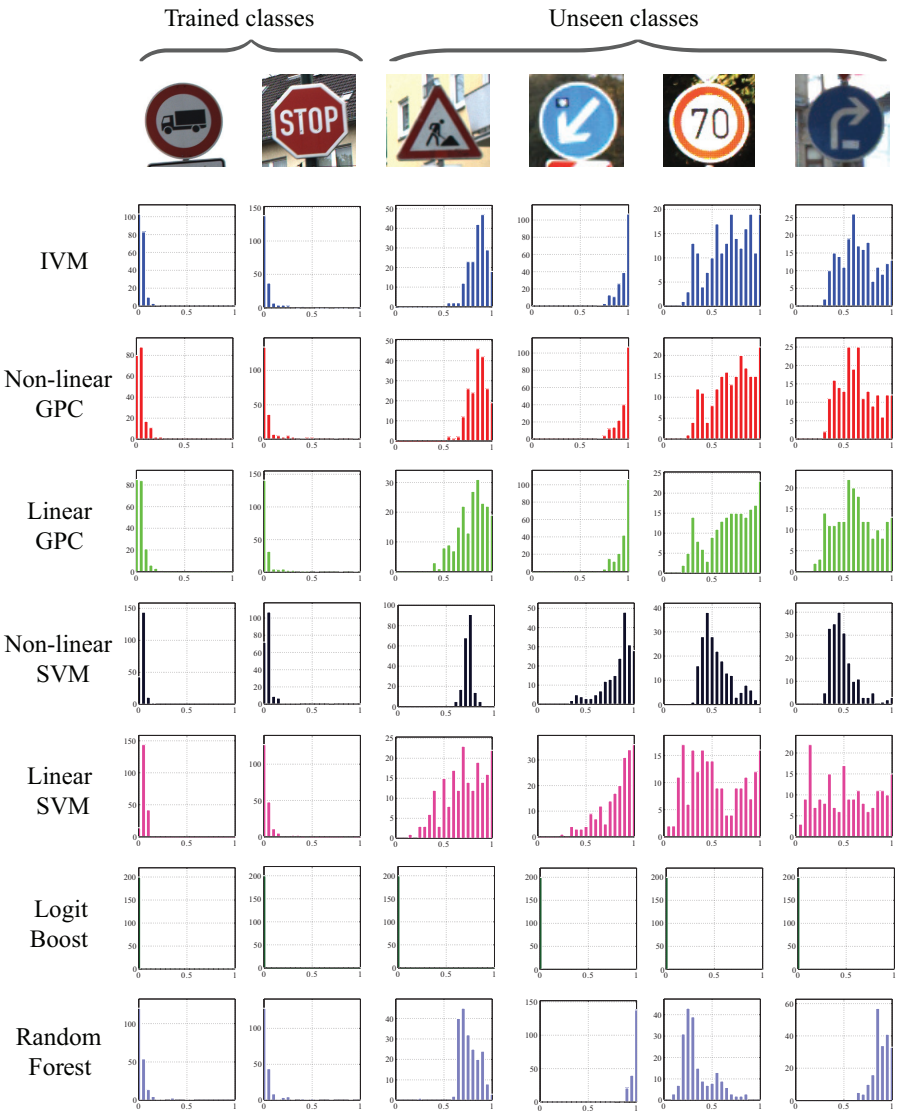


Fig. 5. Normalized entropy histograms (frequency versus NE) of the marginal probabilities for a variety of classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on not only the training classes, but also classes which do not appear in the training set (*roadworks ahead, keep left, 70 kph*, and *right ahead*). Higher values of normalized entropy imply more uncertainty in the classifier output, so we expect the more introspective classifiers to be certain (low NE, left-hand end of the x-axis) on the trained classes and uncertain (high NE, right-hand end of the x-axis) for the unseen classes.

5.4. Introspection in detection

We investigate the same classification frameworks as before on various detection tasks, which each have a salient positive class and a broad background class. We evaluate the classifiers on three data sets: TLR (traffic lights), Daimler Pedestrian, and KITTI (cars), as detailed in Section 5.1.

As with the classification task, we first verify the efficacy of the features selected and the training procedures employed. Table 5 shows the classification performance for classifiers trained using the quantity of data shown in Table 6. We have chosen these values for two reasons. First, we are trying to highlight low-probability catastrophic events, which will be few in number for the size of the test sets we are considering here, but over the life-long autonomy we envisage for our robots will occur in non-negligible numbers; larger training sets reduce the prevalence of these low-probability events, but will never be able to eliminate them. Second, we are using off-the-shelf implementations of commonly used classification frameworks to keep the comparisons fair, and some of these (namely the GPCs) struggle to train successfully when using many (approximately over 800) data with the numbers of feature dimensions we are considering here. We note that in autonomous driving scenarios we typically see more negative examples than positive examples, and so have kept the training and test sets roughly to the same 1:2 ratio of positives to negatives. While scanning an entire urban scene for pedestrians is likely to yield many thousands more negatives than

Table 4. Mean and standard deviation of normalized entropies (including standard errors) from 40 iterations of classifier training and testing, each with a randomly created dictionary and both training and test data sets resampled. Results are presented for classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on five different unseen classes as shown.

Test Class	Classifier	Normalized entropy	
		$\mu \pm$ std. err.	$\sigma \pm$ std. err.
	IVM	**0.776 ± 0.081**	0.145 ± 0.030
	Non-linear GPC	0.751 ± 0.087	0.152 ± 0.029
	Linear GPC	**0.776 ± 0.108**	0.150 ± 0.041
	Non-linear SVM	0.476 ± 0.101	0.089 ± 0.056
	Linear SVM	0.664 ± 0.122	0.250 ± 0.041
	LogitBoost	0.019 ± 0.025	0.041 ± 0.073
	Random Forest	0.756 ± 0.137	0.149 ± 0.053
	IVM	**0.794 ± 0.117**	0.106 ± 0.026
	Non-linear GPC	0.779 ± 0.124	0.107 ± 0.024
	Linear GPC	0.777 ± 0.202	0.124 ± 0.058
	Non-linear SVM	0.537 ± 0.126	0.028 ± 0.036
	Linear SVM	0.494 ± 0.239	0.222 ± 0.049
	LogitBoost	0.016 ± 0.022	0.031 ± 0.059
	Random Forest	0.736 ± 0.166	0.078 ± 0.027
	IVM	0.539 ± 0.140	0.173 ± 0.023
	Non-linear GPC	0.546 ± 0.144	0.168 ± 0.023
	Linear GPC	**0.569 ± 0.166**	0.177 ± 0.026
	Non-linear SVM	0.407 ± 0.077	0.076 ± 0.053
	Linear SVM	0.315 ± 0.195	0.197 ± 0.058
	LogitBoost	0.008 ± 0.004	0.012 ± 0.026
	Random Forest	0.394 ± 0.121	0.138 ± 0.029
	IVM	0.579 ± 0.133	0.137 ± 0.020
	Non-linear GPC	0.577 ± 0.130	0.136 ± 0.019
	Linear GPC	0.585 ± 0.188	0.151 ± 0.029
	Non-linear SVM	0.488 ± 0.111	0.039 ± 0.034
	Linear SVM	0.177 ± 0.127	0.155 ± 0.056
	LogitBoost	0.014 ± 0.019	0.030 ± 0.056
	Random Forest	**0.668 ± 0.161**	0.113 ± 0.027
	IVM	**0.931 ± 0.025**	0.080 ± 0.026
	Non-linear GPC	**0.934 ± 0.021**	0.079 ± 0.023
	Linear GPC	0.925 ± 0.031	0.085 ± 0.027
	Non-linear SVM	0.641 ± 0.168	0.100 ± 0.047
	Linear SVM	0.705 ± 0.142	0.212 ± 0.049
	LogitBoost	0.059 ± 0.103	0.077 ± 0.127
	Random Forest	0.904 ± 0.089	0.088 ± 0.043

positives, it is common (Enzweiler et al., 2012; Fairfield and Urmson, 2011) to use 3D information or prior maps to greatly reduce the portion of each image that needs to be scanned, and thus making the ratio of positive to negative windows much more even.

Figure 6 shows the corresponding precision–recall curves for the classifiers across the data sets. The detection task, having a varied background class and greater variation within the positive class, is more challenging than the classification task. Classification performance according to the conventional metrics is commensurate across all frameworks. The Random Forest performs best for the TLR data set, and the non-linear SVM and IVM perform consistently highly in the Daimler Pedestrian and KITTI data sets. The GPC-based classifiers all have commensurate performance in terms of precision and recall.

In Figures 7, 8, and 9 we demonstrate how the lack of introspection can impact classification performance when accept/reject decisions are determined by classification confidence, with one figure per data set. Specifically, we show the *cumulative* effect of accepting classifications below a given uncertainty threshold. First we note that when classifications are accepted at any level of uncertainty (i.e. up to and including unity normalized entropy) we get values which correspond to those in Table 5. It is desirable for a classifier to be close to the top left-hand corner of the graphs pertaining to *true* classifications (top row) and close to the bottom right of the graphs pertaining to *false* classifications (bottom row). This would correspond to making true classifications with low uncertainty (high confidence) and making incorrect decisions with high uncertainty.

Although the SVMs and LogitBoost classifiers generally make true-positive and true-negative classifications with higher certainty (i.e. low normalized entropy) than for the GPC variants, they are also more confident when making mistakes. This balance is discussed in more detail in Section 6, but in summary we consider the avoidance of

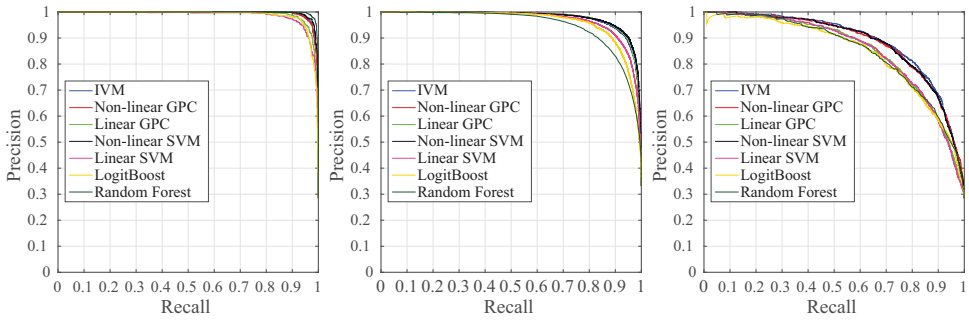


Fig. 6. Precision–recall curves for the (a) TLR, (b) Daimler Pedestrian and (c) KITTI data sets. Note the increasing difficulty of the data sets, and the consistency and commensurate nature of the classifiers in terms of these metrics. (Best viewed in color.)

Table 5. The classifiers' performances for the detection tasks across data sets according to conventional metrics. Precision, recall, and F_1 measure are calculated by thresholding the classifiers' probabilities at 0.5. The SVMs and GPCs give very similar results across the data sets, with the LogitBoost and Random Forest performing slightly worse than the others with the more difficult data sets.

	TLR			Daimler Pedestrian			KITTI		
Classifier	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
IVM	0.995	0.916	0.954	0.953	0.872	0.911	0.868	0.725	0.790
Non-linear GPC	0.992	0.912	0.950	0.956	0.874	0.913	0.853	0.735	0.790
Linear GPC	0.988	0.899	0.941	0.956	0.875	0.914	0.816	0.708	0.758
Non-linear SVM	0.996	0.920	0.956	0.959	0.869	0.912	0.836	0.749	0.790
Linear SVM	0.967	0.910	0.938	0.932	0.876	0.903	0.813	0.709	0.757
LogitBoost	0.978	0.908	0.942	0.961	0.794	0.869	0.826	0.681	0.747
Random Forest	1.000	0.897	0.946	0.984	0.598	0.744	0.894	0.551	0.682

Table 6. The number of training and test data of each class used for the detection experiments. The quantities of data from the GTSRB data set for the classification experiments are detailed in Section 5.3.

	Training data		Test data	
Data set	Positives	Negatives	Positives	Negatives
TLR	250	500	1000	2500
DP	250	500	8000	16,000
KITTI	200	500	2000	5000

high-confidence errors to be of primary importance, and after that, an increase in classifications which are both confident and true, results in a more useful classifier.

The GPC-based classifiers (IVM, non-linear and linear GPCs) behave very similarly to each other, particularly in the TLR and Daimler Pedestrian data sets, and perform very well in terms of making mistakes with very high uncertainty. The price paid for this more realistic assessment of the classification confidence is a reduction in correct classifications above the normalized entropy threshold. Note that this does not mean that subsequent samples are misclassified. It only implies that some other remedial action might be taken, for example obtaining label confirmation from a human, or gathering otherwise additional data to aid disambiguation.

The Random Forest is consistently uncertain in terms of all four decision outcomes across all data sets. This is because the probabilities it outputs are rarely far from $p(C_2) = 0.5$. The fact that it performs rather well in terms of accuracy, precision and recall indicates that it is underconfident.

The difficulties of the data sets clearly vary from the PR curves and the confidences of the true detections, with TLR being the easiest, followed by the Daimler Pedestrian data set, and then KITTI being the most challenging. This is likely to be a result of the variation within the positive class paired with the low number of positive exemplars in the training set (see Table 6).

5.5. Decision making

In Section 3.3 we discussed the importance of the loss function $L(a, C_i)$ and how it shapes the decision of which action a to choose, given some estimates of the state of the

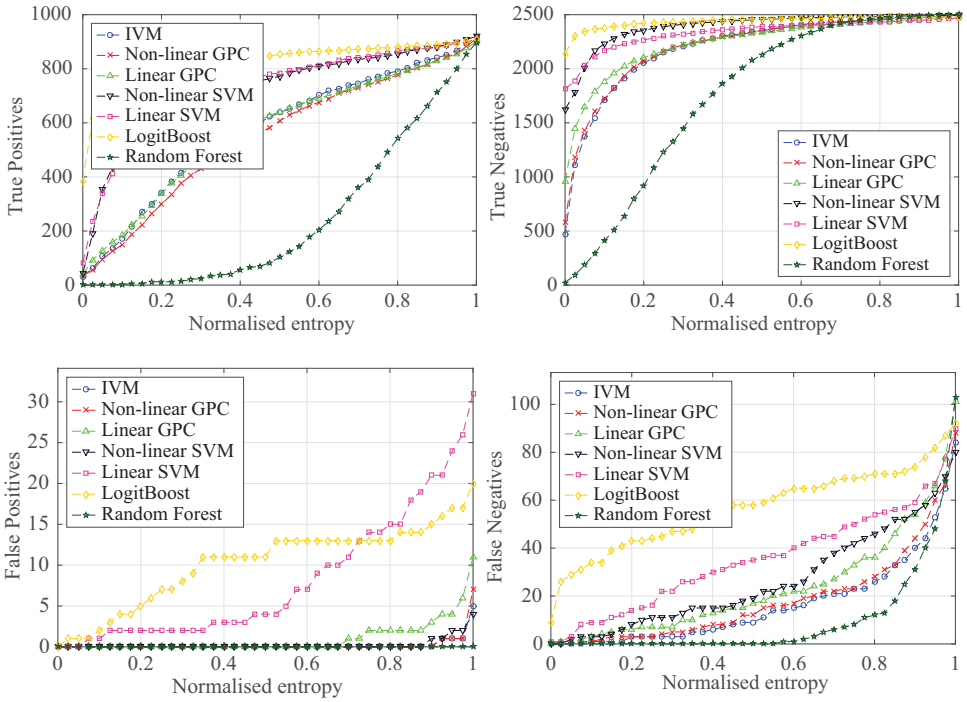


Fig. 7. Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalized entropy. The classifiers have been trained on 250 traffic lights against 500 background patches, and tested on 1000 instances of traffic lights and 2500 background patches. Note that lower normalized entropy implies more certainty in classification. A more introspective classifier is one that simultaneously exhibits higher uncertainty (as witnessed by larger normalized entropy in its output) when processing difficult instances and is more confident when it is correct. Consequently, class decisions above a given normalized entropy threshold are deferred since the output is deemed ambiguous. This is desirable since a single bad decision can have disastrous consequences.

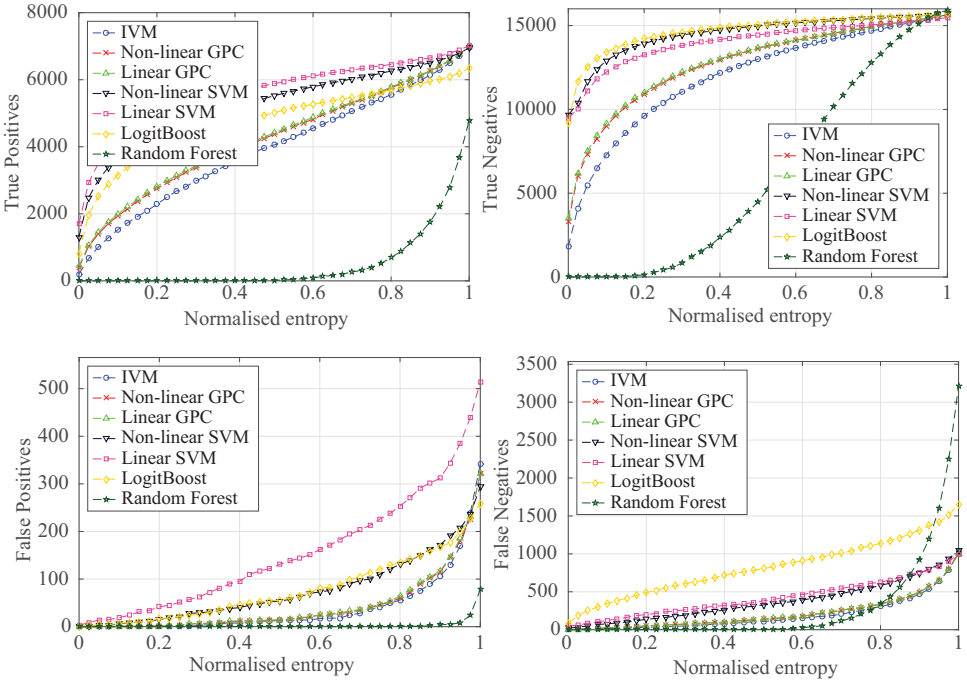


Fig. 8. Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalized entropy (uncertainty), using the Daimler Pedestrian data set. The classifiers are trained on 250 and 500 instances of pedestrians and background respectively, and are tested on 8000 and 16,000 of those classes. See the caption to Figure 7 for more detail.

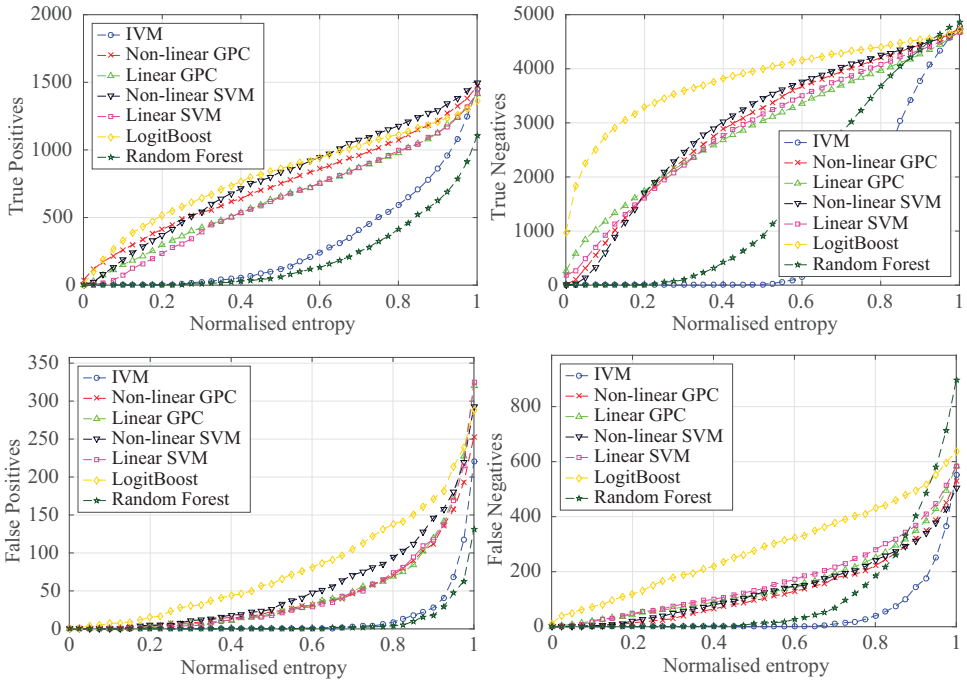


Fig. 9. Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalized entropy (uncertainty), using the KITTI data set. The classifiers are trained on 200 and 500 instances of pedestrians and background respectively, and are tested on 2000 and 5000 of those classes. See the caption to Figure 7 for more detail.

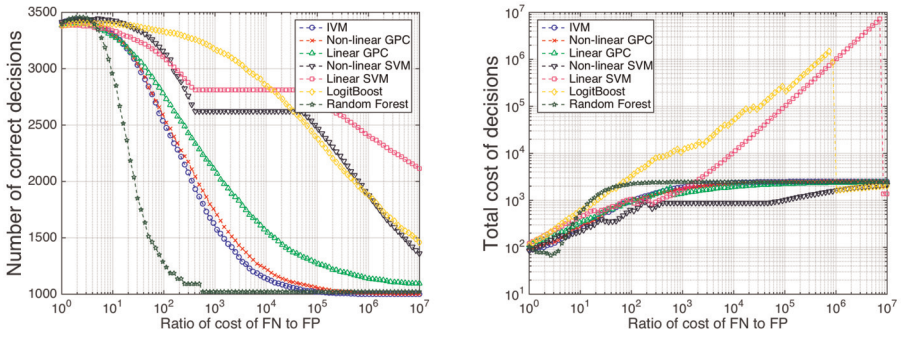


Fig. 10. Decision making for the TLR data set, on a test set of 1000 positive and 2500 negative examples. The x-axes represent the cost of a false-negative (FN, in this case missing a traffic light) error, while the cost of a false-positive (FP) error is held at 1. On the left we show the number of correct decisions made (positive and negative) as we vary the cost of a false negative, and on the right we show the total cost of all decisions (correct decisions carry 0 cost). Ideal behavior on the left is to smoothly become more conservative (which requires making fewer correct decisions) as the cost increases, and on the right the ideal is to incur minimal total cost at every point on the x-axis.

environment $\{p(C_1), \ldots, p(C_{|\mathcal{C}|})\}$. In robotics, we seek classification frameworks which allow our robots to make decisions which are faithful to the values instilled by the losses incurred for particular outcomes. For instance, if we make the cost associated with a particular outcome very large, then the actions which can lead to that outcome should be chosen more infrequently, or at least only when the classifier gives a very confident estimate of the state of the environment. The behavior we seek is for classifiers to behave appropriately given any relative costs associated with the possible outcomes. We characterize this 'appropriateness' by comparing the total cost incurred when using each classifier as part of the decision-making pipeline, and we do so while varying the ratio of the costs of false-positive and false-negative outcomes. Note that we cannot mitigate the dangerous tendencies of less introspective classifiers by adjusting the costs; each decision is made by weighting the probabilities produced by a particular framework, and thus if the probabilities are a poor indicator of the truth of the classification, wrong decisions will be made regardless of the costs set.

For each of the three data sets, we use the probabilistic output of the classifiers to drive the decision-making pipeline, and evaluate the decisions made. We set the costs of

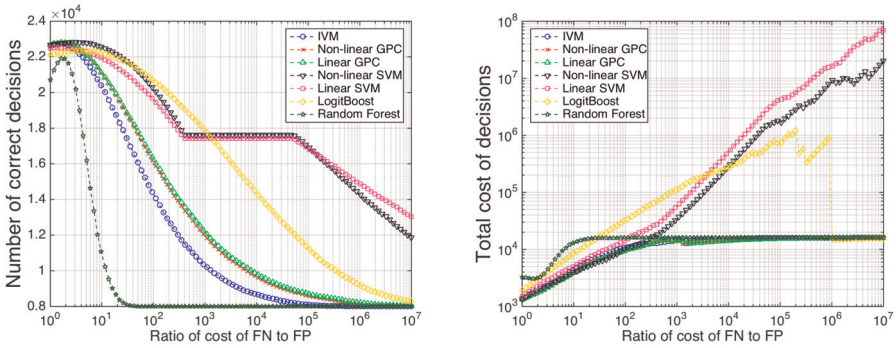


Fig. 11. Decision making for the Daimler Pedestrian data set, on a test set of 8000 positive and 16,000 negative examples. See the caption to Figure 10 for more information.

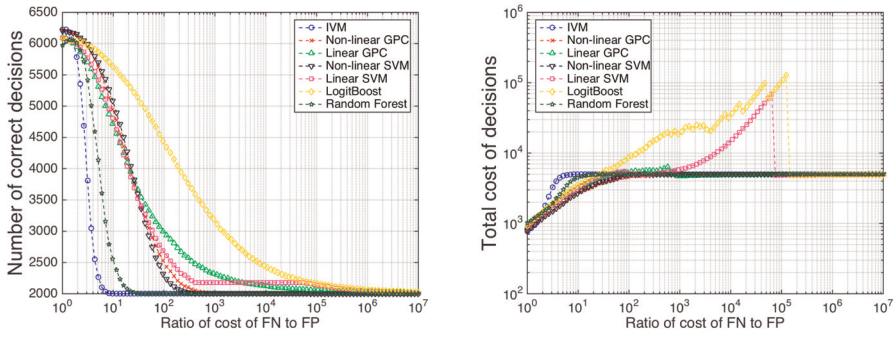


Fig. 12. Decision making for the KITTI data set, on a test set of 2000 positive and 5000 negative examples. See the caption to Figure 10 for more information.

true-positive and true-negative outcomes as 0, and the cost of a false-positive outcome as 1. The value for the last outcome, the false negative or missed pedestrian, is varied from 1 to 10^7. This cost of the false-negative error appears on the *x*-axes of Figures 10, 11, and 12. The *y*-axes of the *left-hand* figures in each pair denote the number of true outcomes (both positive and negative together), and the *y*-axes of the *right-hand* figures denote the total cost of all the decisions made.

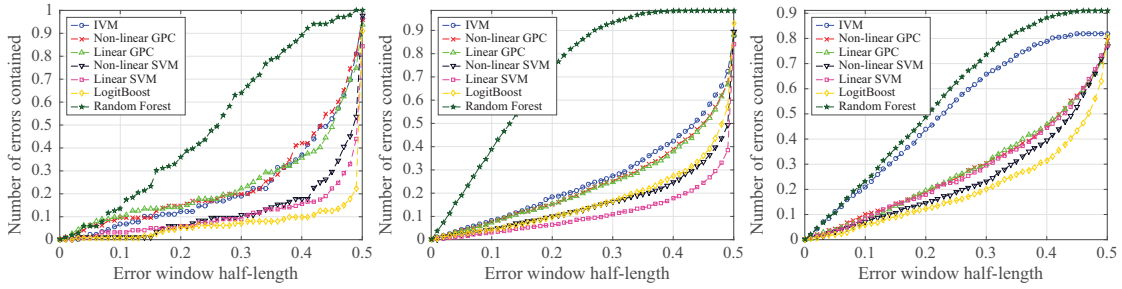
These pairs of graphs demonstrate the trade-off between classifiers which avoid catastrophic decisions, and those which might be so cautious that they never take the higher risk action. The *left-hand* graphs demonstrate the rate at which the classifiers' decisions become more and more cautious as the cost of a false negative increases. We do not consider one to be superior to another in terms of introspection, albeit it may tell you about the usefulness of that classifier. On the right-hand graphs, the ideal is for a curve to be as low as possible (close to the *x*-axis). This would represent a classifier which makes good decisions given any particular cost ratio.

Comparing the classifiers across the data sets, we make the following observations. The GPC-based classifiers and the Random Forests have consistently low total cost in the right-hand graphs. This is very desirable because they do not commit any false-negative errors when the costs begin to increase. This indicates that they have considerable introspective qualities. The LogitBoost and linear SVM classifiers consistently make catastrophic mistakes, resulting in very high costs across all three data sets, while the non-linear SVM exhibits this inconsistent, risky behavior in one of the three data sets, making an error with such extreme confidence that it persists even when the cost is 10^7. These jagged curves characterize the unpredictable behavior of those single-discriminant classifiers, and demonstrate how they are much less desirable in the context of mission-critical decision making, despite the fact that they may perform better in terms of F_1 measure.

The steady-state values in all these graphs are a result of the decision system always choosing the *stop* action, thus incurring a cost of 1 per false positive for all the negatives in the test sets. It is natural for the curves to reach this steady state, because by setting the ratio of the costs to be so large we are effectively imposing such a high cost on the *go* action that it will never be chosen in favor of the safer *stop* action (unless $p(C_2) = 0$), and thus they incur a cost equal to the total number of negatives.

The flat sections which occur for the SVMs in the graphs on the left of Figures 10, 11, and 12 are a result of there being no test data yielding probability values in the region which the decision boundary (the value of $p(C_2)$ where *stop* meets *go* in Figure 4(b)) sweeps as the cost ratio moves from 300 to 40,000.

In Section 3.3 we described another way to characterize the introspective tendencies of a classifier: by examining



(a) TLR (b) DP (c) KITTI

Fig. 13. Following Figure 4, we show the errors contained about $p(C_2) = 0.5$ as we increase the size of the orange box. To generate these curves, we randomly sampled 1000 positive and 1000 negative test data and count the number of errors. Note that a classifier which is uncertain when it makes mistakes will be closer to the top left of each plot. This, together with the confidence of correct decisions show the two sides of the introspective coin.

the distribution of errors across $p(C_2)$. In Figure 4 we showed that the number of cumulative errors as you increase the half-length of the orange box (from Figure 4(d)) from 0 to 0.5 can take various shapes, and we showed the shape induced by a more introspective classifier in teal (in Figure 4(e)). In Figure 13 we show those curves for our classifiers across the three data sets. Comparing it to Figure 4(e), we see that for all three data sets, the curves for the classifiers which consider multiple discriminants (Random Forests and the GPC-based classifiers) are closer to the desirable teal curve than the single-discriminant classifiers (the SVMs and LogitBoost). The Random Forest very strongly resembles the teal curve across all data sets, and in the case of the KITTI data set the IVM does also. This is a further, strong indication of the introspective power of the Random Forests and GPC-based classifiers over the others.

6. Discussion

To what degree is introspection a property of a single classifier, or of a classification framework? Can one SVM be more introspective than another SVM? Are GPCs invariably more introspective than SVMs? The results in this paper indicate a consistent behavior of particular classification frameworks across particular tasks (such as classification or detection), which we attribute to the manner in which the classifiers are designed (see Section 4). Thus we expect that introspection quality is inherent to classification *frameworks* rather than individual classifiers.

That said, varying the choice of kernel (and its parameters) does produce very different behaviors, and it may be possible to instill an improved introspective sense with an appropriate choice of kernel. The reason for this is that in every framework there is a link between classification confidence and distance in kernel space. In the GPC, a test datum which lies far away from the training data (in kernel space) yields a more uncertain classification. As we have illustrated, single-discriminant models like the SVM function similarly, but their principal failing comes from the fact that distance is only measured orthogonal to the decision plane, and that distance along the decision plane is not taken into account. But how can we guarantee that new, unseen classes will be far away from our training data? In truth, we cannot. We can only hope that the kernel has found a warping of the feature space which adequately separates the two classes of training data, and that new, unseen classes will be sufficiently disparate from the training data in kernel space to yield an uncertain classification. Owing to the opaque nature of the kernel function, we cannot assume that points which are close together in feature space will also be close in kernel space. This brings into question the sanity of using distance in kernel space as a metric for uncertainty in classification.

It should be noted that the sparse nature of the IVM could be expected to reduce its introspective capabilities at the expense of computational efficiency, when in fact it seems to outperform the non-linear GPC in many cases. We attribute this to the fact that the two implementations are from different libraries and it is likely that the optimization procedures in the IVM are superior to those in the GPC, resulting in a better choice of hyperparameters and thus a more effective sense of how distance should relate to uncertainty.

In this paper we have investigated a variety of applications, feature types, class types and quantities, as well as the nuances between classification and detection. This is because it is not always possible to determine the introspective quality of a classification framework based on a single classifier and test set. Can we say whether a classifier which is uncertain about *all* decisions like the Random Forest is introspective or not? We suggest that it is perhaps introspective, but certainly not as useful as one which can also make correct classifications confidently. A very introspective classifier will have a strong correlation between confidence and correctness. Some situations, such as when a classifier is uncertain about everything, do not yield enough information to determine a classifier's introspective capacity. Another such situation is where a classifier gives perfect classification results. Without errors, we cannot

judge how a classifier deals with the unexpected (unless we evaluate it on truly alien data, such as in the unseen class experiment of Section 5.3).

We have motivated the desire for a classifier to make mistakes with high uncertainty, and we have shown that if this is not the case, it will make unpredictable and expensive errors. In addition, for the classifier to be useful, we would also like its true classifications to be made with high confidence. All the classifiers we have investigated in this work have fallen short of both targets, but to varying degrees. If we cannot have both, is there a trade-off to be struck, and are some combinations more desirable than others? One difficulty lies in formally defining how a 'perfectly introspective' classifier should behave. We have approached this in terms of increased distance between training and test data leading to uncertainty, and that the degree of uncertainty should indicate the likelihood of a mistake being made. More rigidly, we propose that classifications with an uncertainty of 0.9 should be incorrect by 1 mistake in 10, on average. If this were true, we could make very well-informed decisions.

Although we have found that the GPC-based classifiers exhibit a greater introspective quality than the other classifiers tested in this work, it must be said that they are still far from perfectly introspective, regardless of our choice of the definition of 'perfect'. One obvious point of improvement for the Random Forest is that it is very underconfident when it makes true classifications, which is also a feature of the GPC-based classifiers, although to a much lesser extent. It is interesting to note that IVMs with a successively higher active set fraction q get more and more confident, both in terms of true and false classifications, so it may be possible to improve them by simply training them on more data. If an introspective classifier's predictions all have high uncertainty, this could be a useful sign that the problem is too complex and more training data are required.

7. Conclusions

This work demonstrates how performance metrics traditionally used in machine learning for classifier training and evaluation may be insufficient to characterize system performance in a robotics context, where a single misjudgment can have disastrous consequences. To remedy this shortcoming, we propose the concept of *introspection*: the ability to mitigate potentially overconfident classifications by an appropriate assessment of predictive variance. Our experimental results imply that, despite commensurate performance as measured by more conventional metrics, GPC-based classifiers possess a more pronounced introspective capacity than other classification frameworks commonly employed in robotics, maintaining a useful balance between being confident when they are correct, and uncertain when they are making mistakes. We attribute this to their consideration of distance between data, and accounting for predictive variance over the space of feasible classification models. This is in contrast to other commonly employed classification frameworks which often only consider a one-shot (maximum likelihood, or maximum a posteriori) solution. As a result of this, model-averaging classifiers make better decisions than single-discriminant classifiers like SVMs, and thus will cause fewer catastrophic accidents despite appearing worse in terms of F_1 measure.

Funding

This work was supported by the European Community's Seventh Framework Programme (grant number FP7/2007-2013, 269916 (V-CHARGE)) and by the UK EPSRC (grant number EP/J012017/1).

References

Anguelov D, Taskar B, Chatalbashev V, Koller D, Gupta D, Heitz G and Ng A (2005) Discriminative learning of Markov random fields for segmentation of 3D scan data. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, San Diego, USA, 20–26 June, pp. 169–176. IEEE.

Berczi LP, Posner I and Barfoot TD (2015) Learning to assess terrain from human demonstration using an introspective Gaussian process classifier. In: *IEEE international conference on robotics and automation (ICRA)*, Washington, USA, 26–30 May.

Bishop CM (2006) *Pattern Recognition and Machine Learning*. New York: Springer, pp. 658–659.

Breiman L (2001) Random forests. *Machine Learning* 45(1): 5–32.

Burges CJC (1998) A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2): 121–167.

Chang CC and Lin CJ (2011) LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2(3): 27.

Dalal N and Triggs B (2005) Histograms of oriented gradients for human detection. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, San Diego, USA, 20–26 June, pp. 886–893. IEEE.

Douillard B, Fox D and Ramos F (2008) Laser and vision based outdoor object mapping. In: *Robotics: science and systems IV*, Zurich, Switzerland, 25–28 June, pp. 9–16.

Enzweiler M, Eigenstetter A, Schiele B and Gavrila DM (2010) Multi-cue pedestrian classification with partial occlusion handling. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, San Francisco, USA, 13–18 June. pp. 990–997. IEEE.

Enzweiler M, Hummel M, Pfeiffer D and Franke U (2012) Efficient stixel-based object recognition. In: *IEEE intelligent vehicles symposium (IV)*, Alcala de Henares, Spain, 3–7 June, pp. 1066–1071. IEEE.

Fairfield N and Urmson C (2011) Traffic light mapping and detection. In: *IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9–13 May, pp. 5421–5426. IEEE.

Friedman J, Hastie T and Tibshirani R (1998) Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28(2): 337–407.

Geiger A, Lenz P and Urtasun R (2012) Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, USA, 16–21 June, pp 3354–3361.

Grimmett H, Paul R, Triebel R and Posner I (2013) Knowing when we don't know: Introspective classification for mission-critical decision making. In: *IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May, pp. 4531–4538.

Hastie TJ and Tibshirani RJ (1986) *Generalized Additive Models*. *Statistical Science* 1(3): 297–310.

Holub A, Perona P and Burl MC (2008) Entropy-based active learning for object recognition. In: *IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, Anchorage, USA, 24–26 June, pp. 1–8. IEEE.

Hospedales T, Gong S and Xiang T (2013) Finding rare classes: Active learning with generative and discriminative models. *IEEE Transactions on Knowledge and Data Engineering* 25(2): 374–386.

Huang A and Teller S (2010) Probabilistic lane estimation using basis curves. In: *Robotics: science and systems*, Zaragoza, Spain, 27–30 June, pp. 269–283.

Joshi AJ, Porikli F and Papanikolopoulos N (2009) Multi-class active learning for image classification. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Miami, USA, 20–26 June, pp. 2372–2379.

Kapoor A, Grauman K, Urtasun R and Darrell T (2010) Gaussian processes for object categorization. *International Journal of Computer Vision* 88(2): 169–188.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

Lawrence ND, Platt JC and Jordan MI (2005) Extensions of the informative vector machine. In: *1st international conference on deterministic and statistical methods in machine learning*, pp. 56–87. Springer-Verlag.

Lawrence ND, Seeger M and Herbrich R (2003) Fast sparse Gaussian process methods: The informative vector machine. In: *Proceedings of the 16th annual conference on neural information processing systems*, pp. 609–616.

Li L, Littman ML and Walsh TJ (2008) Knows what it knows: A framework for self-aware learning. In: *25th international conference on machine learning (ICML)*, Helsinki, Finland, 5–9 July, pp. 568–575. New York: ACM.

Lin HT, Lin CJ and Weng RC (2007) A note on Platt's probabilistic outputs for support vector machines. *Journal of Machine Learning* 68(3): 267–276.

Martínez-Mozos O, Triebel R, Jensfelt P, Rottmann A and Burgard W (2007) Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems* 55(5): 391–402.

Meger D, Forssén PE, Lai K, Helmer S, McCann S, Southey T, Baumann M, Little JJ and Lowe DG (2008) Curious George: An attentive semantic robot. *Robotics and Autonomous Systems* 56(6): 503–511.

Minka TP (2001) Expectation propagation for approximate Bayesian inference. In: *17th conference on uncertainty in artificial intelligence*, Seattle, USA, 2–5 August, pp. 362–369. Morgan Kaufmann Publishers Inc.

Niculescu-Mizil A and Caruana R (2005) Predicting good probabilities with supervised learning. In: *22nd international conference on machine learning*, Bonn, Germany, 7–11 August, pp. 625–632. New York: ACM.

Paul R, Triebel R, Rus D and Newman P (2012) Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In: *International conference on intelligent robots and systems (IROS)*, Vilamoura, Portugal, 7–11 October, pp. 2404–2410.

Platt JC (1999) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola AJ, Bartlett P, Scholkopf B, et al. (eds) *Advances in Large Margin Classifiers*. Cambridge: MIT Press, pp. 61–74.

Posner I, Cummins M and Newman P (2009) A generative framework for fast urban labeling using spatial and temporal context. *Autonomous Robots* 26(2–3): 153–170.

Pronobis A and Jensfelt P (2012) Large-scale semantic mapping and reasoning with heterogeneous modalities. In: *IEEE international conference on robotics and automation (ICRA)*, St. Paul, USA, 14–18 May, pp. 3515–3522.

Rasmussen CE and Nickisch H (2010) Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research* 11: 3011–3015.

Rasmussen CE and Williams CKI (2006) *Gaussian Processes for Machine Learning*. Cambridge: MIT Press.

Sengupta S, Sturgess P, Ladick L and Torr P (2012) Automatic dense visual semantic mapping from street-level imagery. In: *IEEE international conference on robotics and autonomous systems*, St. Paul, USA, 14–18 May, pp. 857–862.

Stallkamp J, Schlipsing M, Salmen J and Igel C (2012) Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32: 323–332.

Tellex S, Thaker P, Deits R, Kollar T and Roy N (2012) Toward information theoretic human–robot dialog. In: *Robotics: science and systems*, Sydney, Australia, 9–13 July, p. 409.

Thrun S, Montemerlo M and Dahlkamp H, et al. (2006) Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics* 23(9): 661–692.

Tong S and Koller D (2002) Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2: 45–66.

Torralba A, Murphy KP and Freeman WT (2007) Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machince Intelligence* 29(5): 854–869.

Vedaldi A and Fulkerson B (2010) VLFeat: An open and portable library of computer vision algorithms. In: *ACM international conference on multimedia*, Firenze, Italy, 25–29 October, pp. 1469–1472.

Velez J, Hemann G, Huang AS, Posner I and Roy N (2011) Planning to perceive: Exploiting mobility for robust object detection. In: *International conference on automated planning and scheduling (ICAPS)*, Freiburg, Germany, 11–16 June pp. 266–273.

Williams CKI and Barber D (1998) Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(12): 1342–1351.

Zhang W, Yu SX and Teng S-H (2012) Power SVM: Generalization with exemplar classification uncertainty. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Providence, USA, 16–21 June, pp. 2144–2151. IEEE Press.

C Summary Maps for Lifelong Visual Localization

Authors: P. Mühlfellner, M. Buerki, M. Bosse, W. Derendarz, R. Philippsen, P. Furgale

Link to Table of Contents.

Summary Maps for Lifelong Visual Localization

Peter Mühlfellner
Dept. for Driver Assistance and Integrated Safety
Volkswagen AG, Halmstad University
Letter Box 011/1777, Wolfsburg, Germany
peter.muehlfellner@volkswagen.de

Mathias Buerki
Autonomous Systems Lab, ETH Zürich
Leonhardstrasse 21, Zürich, Switzerland
mathias.buerki@mavt.ethz.ch

Mike Bosse
Autonomous Systems Lab, ETH Zürich
Leonhardstrasse 21, Zürich, Switzerland
mike.bosse@mavt.ethz.ch

Wojciech Derendarz
Dept. for Driver Assistance and Integrated Safety
Volkswagen AG
Letter Box 011/1777, Wolfsburg, Germany
wojciech.derendarz@volkswagen.de

Roland Philippsen
Intelligent Systems Lab, Halmstad University
Kristian IV's väg 3, Halmstad, Sweden
roland.philippsen@hh.se

Paul Furgale
Autonomous Systems Lab, ETH Zürich
Leonhardstrasse 21, Zürich, Switzerland
paul.furgale@mavt.ethz.ch

Abstract

Robots that use vision for localization need to handle environments which are subject to seasonal and structural change, and operate under changing lighting and weather conditions. We present a framework for lifelong localization and mapping designed to provide robust and metrically accurate online localization in these kinds of changing environments. Our system iterates between offline map building, map summary, and online localization. The offline mapping fuses data from multiple visually varied datasets, thus dealing with changing environments by incorporating new information. Before passing this data to the online localization system, the map is summarized, selecting only the landmarks that are deemed useful for localization. This *Summary Map* enables online localization that is accurate and robust to the variation of visual information in natural environments while still being computationally efficient.

We present a number of summary policies for selecting useful features for localization from the multi-session map and explore the tradeoff between localization performance and computational complexity. The system is evaluated on 77 recordings, with a total length of 30 kilometers, collected outdoors over sixteen months. These datasets cover all seasons, various times of day, and changing weather such as sunshine, rain, fog, and snow. We show that it is possible to build consistent maps that span data collected over an entire year, and cover day-to-night transitions. Simple statistics computed on landmark observations are enough to produce a Summary Map that enables robust and accurate localization over a wide range of seasonal, lighting, and weather conditions.

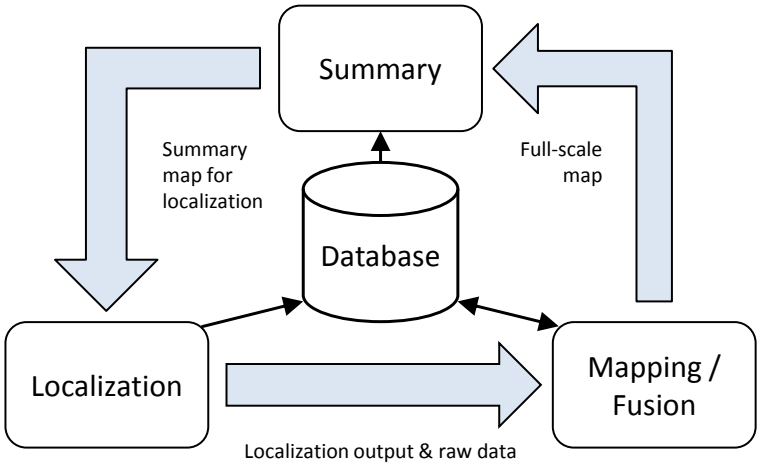


Figure 1: *The life cycle for a map. A database functions as the repository of all information necessary for localization, mapping, and summary of the map.*

This is a preprint of an article accepted for publication in Journal of Field Robotics,
Copyright 2011 Wiley-Blackwell.

1 Introduction

Map-based robotic navigation is one of the dominant paradigms for implementing driverless cars (Furgale et al., 2013, Lategahn and Stiller, 2012, Ziegler et al., 2014b). Under this strategy, the environment is mapped prior to any automated run, and a pre-built map that contains information relevant to tasks such as localization and navigation (i.e. visual landmarks and the road network) is provided to the vehicle. With map-based navigation, *localization* is a core competency for online operation. The difficult problems of map management and fusion with all previously seen data can be deferred to a server and processed offline, in batch. We see this as the natural progression as we transition to a world with tens, hundreds, or thousands of service robots deployed in a single area. For example, automated cars might not manage their own world maps based on their own data. Instead, they could rely on centralized services that produce maps for a given task based on data collected from all the robots over extended periods.

This paper proposes and evaluates a general strategy for producing visual maps for robots to localize as they revisit an area multiple times. Vision-based localization and mapping has been shown to produce excellent results for large-scale mapping (Sibley et al., 2010) and long-range navigation (Furgale and Barfoot, 2010). In order to support planning and control such as relevant for driverless cars, the proposed localization system provides *locally accurate metric pose estimation*. Beside localization itself, a crucial aspect for real-world deployment is *lifelong mapping*, particularly the ability to deal with changes in the world. In the context of visual localization for driverless vehicles, these changes are for example caused by diverse weather conditions, changing billboards and parked cars, seasonal variation in trees, and the apparent motion of the sun. While some of these changes can be handled using image processing techniques, more profound changes can only be dealt with by incorporating new information in the map. Our goal is to provide metric pose estimation that employs all pertinent information available about a scene. This goal is distinct from long-term, condition-invariant, *place recognition* (e.g. (Milford and Wyeth, 2012)) which does not provide pose estimates. Furthermore, we aim to handle structural- and not just lighting-dependent appearance-changes, in contrast to systems such as (Lategahn et al., 2013, McManus et al., 2014a). Our goals for accuracy are also much tighter than those of other metric estimation systems, notably McManus et al. (2014b), as we aim for an accuracy on the order of decimeters.

To the best of our knowledge, the problem of lifelong updating of visual maps remains unsolved, and this

paper makes significant contributions to this research area. Some of the phenomena that cause visual change are relatively abrupt while others occur more or less slowly. We therefore model the changing world as a *continuum of visual features*, allowing features from different visits to the same place to be observed together at a later point in time. Environment dynamics can contain a broad spectrum of time scales, and in this work we have focused on a setting which is particularly challenging in this respect. The environment we consider is subject to seasonal change and various weather conditions, and our driverless vehicle operates at different times of day, on a backdrop of trees, shrubs, mostly static buildings, and with parked cars that change significantly every morning and evening on working days.

Another important issue is computational cost, in general and also for on-board operation. Resource limits preclude simply growing the map whenever new perceptual information arises. In some environments, the number of possible visual configurations can be practically unbounded and change daily. Furthermore, a significant portion of this information may stem from ephemeral parts of the environment, and would only bloat the map and disturb or slow down localization. Thus, the knowledge gained about an environment as new data arrives should be used not only to add, but also to *strategically discard* parts of the previously collected information. The resulting reduced map should contain only those parts that are sufficiently relevant. We refer to this process as *summarizing* a map, and propose a general scheme for the lifelong management of maps which incorporates this aspect. We demonstrate the effectiveness of our approach using an in-depth analysis of long-term datasets, and provide several suggestions for map-summary strategies. We also demonstrate the generality of our method by taking two state-of-the-art algorithms for selecting useful information from long-term maps (Dayoub et al., 2011, Konolige and Bowman, 2009) and expressing them within the formalism of our framework. As an additional benefit, our approach allows us to directly compare these algorithms with each other as well as with our own summarizing strategies.

A central question of mapping is to decide what should determine the quality of a map. This of course depends on the intended use of the map. The evaluation of a mapping approach and the particular maps it produces must also be done in a way which reflects the identified requirements. For the navigation of driverless cars, we believe that robust and locally accurate pose estimation is of crucial importance. The map is the repository of geometric and semantic information – such as road networks, parking spaces, and traffic light positions – needed to achieve complex tasks. The positions of all semantic objects can naturally be linked to the on-board perception system when they are expressed with respect to the local geometry. The usefulness of the map thus stems from the ability of connecting its localization estimates and the positions of nearby semantic objects. This suggests that the system should be evaluated based on its ability to produce a metric estimate of the robot's pose within the map, with computational complexity suitable for online operation. In contrast, we do not believe it to be very informative when maps are evaluated based on global 3D reconstructions or some global accuracy measures when compared to a separate survey of the environment.

In this paper, we use three evaluation criteria (see sections 4.3): *availability* measures the percentage of time the robot is able to localize within the map, local *accuracy* measures the metric error of the robot's six degree of freedom (6-DOF) pose estimate, and *computational complexity* measures the computational cost of localizing within the map.

The contributions of this paper are as follows:

1. We outline a general scheme for solving the problem of vision-based lifelong localization. Our approach is based on the concept of online localization and offline map maintenance, extending the concepts described by Biber and Duckett (2005). The proposed map life cycle is depicted in Figure 1.
2. We propose a method to produce a *Summary Map* for localization based on *selection* of subset of the landmarks available within the global map.
3. We present an implementation of a complete long-term visual mapping and localization framework.
4. We evaluate the trade-offs between availability, accuracy, and computational complexity for a set of summary schemes, on a total of 77 recordings collected outdoors at two seperate locations, over one and a half years. These schemes include adaptations from related work as well as novel methods

based on our approach.

5. We show that it is possible to build consistent maps using data from an entire year, as well as from recordings that cover day- and nighttime settings. The resulting Summary Maps enable robust and accurate localization, covering a wide range of seasonal, lighting as well as weather conditions.

2 Related Work

Achieving long-term autonomy is one of the big open challenges in robotics. The problem of long-term mapping, also known as lifelong or multi-session mapping, has thus received a considerable amount of attention. While the goal of building a map that is useful over a long time frame is common to many published works, the exact problem that is targeted with the resulting map can vary considerably. We target the localization of a robot using on-board cameras, and focus primarily on reviewing work that likewise targets camera-based systems.

Many of the most successful long-term systems are concerned with the coarse matching of an image to another image, or place, in the map. This long-term *place recognition* approach is exemplified by SeqSLAM (Milford and Wyeth, 2012), which matches sequences of images acquired under extremely varied conditions, matching from day to night, rain to sunshine. SeqSLAM has been extended to include better handling of different vehicle speeds (Pepperell et al., 2014) and match-verification (Milford et al., 2014). Similarly, Naseer et al. (2014) robustly select matches under severe seasonal changes using the data-association graph for full datasets. Valgren and Lilienthal (2010) have investigated the long-term performance of visual features for place recognition across seasons; they point out the importance of using image geometry (in the form of epipolar constraints) to validate matches, and show that the Upright SURF descriptor performs well. Neubert et al. (2013) predict the visual words present in an image recorded during winter, based on an image of the same place recorded during summer, and vice-versa. Krajnik et al. (2014) represent the periodic reoccurrence of features using a spectral approach. For a vehicle travelling on a road network, it is also possible to perform place recognition without visual information in the map, as Brubaker et al. (2013) show — they match short paths recorded via Visual Odometry (VO) to a street map. These systems solve the difficult problem of going from a completely unknown pose to a rough localization (in the form of a matching place). While this form of localization is useful for detecting loop-closures, and for helping recover from a lost state, it does not on its own provide enough information for planning and control of a robot. Furthermore, the above approaches – with the exception of Krajnik et al. (2014) – do not incorporate new information about an environment as it becomes available, but rather match to a static representation. In contrast, our approach relies on a rough estimate of the robot's position within the map during initialization, and then tracks the robot's progress through the map with high metric accuracy. Because of these differing goals, we see these two lines of research as complementary. Place recognition is necessary for initialization and fault recovery, but localization is needed to support accurate local planning and control. Consequently, we do not compare our work directly to any of the algorithms above.

Krajník et al. (2010) propose a simple method for estimating the bearing of a robot, which allows for basic navigation along a path composed of straight segments. Their approach can provide accurate estimates using only two maps, one for winter and one for summer, in an environment that is subject to strong seasonal changes (Krajnik et al., 2013). In order to perform tasks with an automated vehicle, such as obstacle avoidance or parking into a spot that is off the recorded path, we sometimes need to deviate from the mapped path. Bearing-only or purely appearance based systems do not offer enough flexibility in order to perform these important tasks.

In contrast to the above, a different approach to the long-term mapping problem is to maintain a model of the environment that includes some form of geometric information for pose estimation. The poses typically have six degrees of freedom (DOF). Many approaches in literature are primarily concerned with providing high accuracy for the parameters of this model. In this context, a pose-graph based representation of the environment is often used. Nodes of the pose-graph usually contain views of the environment, whereas

edges are (estimated) relative transformations between views. As places are revisited, the graph grows and eventually is pruned, discarding some of the previously observed information. Johannsson et al. (2012) partition the mapped space into a regular grid over 2D-position and heading, using only the most recent view to represent appearance. Maddern et al. (2013) use the information content of a view relative to its neighbours to select views for deletion and replace them with relative constraints. In all of these approaches, the primary goal is to produce a consistent map – the question of what is best for localization is not addressed. We do not tackle the problem of sparsifying a pose graph under long-term operation as more and more data is added to a map. Rather, we aim to provide the best possible localization for an automated vehicle in a given map. In other words, we are not as much concerned with maintaining an *accurate* model of the world, but rather with maintaining a *useful* model. In this context, the major challenge is handling dynamic scenes, that are observed under varying lighting conditions and that change in structure between visits to the environment.

A recent research direction focuses specifically on the effects of varying lighting conditions. This is done either at the level of individual pixels – transforming raw image data – or by learning sparse image features that are resistant to lighting changes. Approaches that use illumination-invariant image transformations have shown very promising results (Corke et al., 2013, McManus et al., 2014a). They employ the spectral characteristics of a single light-source in order to remove the effect of different illumination angles and cast-shadows, at a pixel level. Maddern et al. (2014) have combined the illumination-invariant image transformation proposed by McManus et al. (2014a) with the prior structure recorded from a pushbroom Laser (Stewart and Newman, 2012), to provide accurate metric localization that is invariant to lighting variations. Lategahn et al. (2013) propose learning an illumination invariant descriptor, using a set of basic building blocks, from a representative set of recordings taken at different times of day. While they employ this learned descriptor mainly for place recognition, they have also shown excellent results in robust localization for a driverless vehicle in urban environments (Lategahn and Stiller, 2014). Training neural networks to extract descriptors that are robust to illumination changes has been proposed by Carlevaris-Bianco and Eustice (2014). Techniques such as illumination-invariant image transformations provide an excellent way of dealing with light and shadow variations, but are limited to scenarios with a well-known light source, e.g. outdoors under direct sunlight. While these methods bring robustness to changing lighting, more general structural change in scenes is not modelled by them. In comparison, we take a unified approach that naturally copes with lighting changes (both natural and artificial), transitions smoothly from indoors to outdoors, and updates based on scene changes, all with a single map-management strategy.

To handle structural changes in scenes, Biber and Duckett (2005) propose a dynamic map that represents multiple timescales at once. Their map is composed of discretized 360 degree range scans, called local maps. There are multiple local maps associated with each place in a global coordinate frame, each updated by sampling new measurements at a given rate; the sampled measurements replace an equal number of existing range-values. Higher update rates allow incorporating changes into the map quickly, whereas lower update rates favour more stable long-term structures. For localization, the local map that best explains the current observations, among the maps for the closest global place, is used for Laser scan matching. In the scheme described by Biber and Duckett, a short-term map with a very high update rate is updated continually during online operation, while all the measurements are kept around until a later offline phase, where the various long-term timescales are updated. While we propose a similar scheme for offline map-management, we aim to build a map that provides a unified representation for all timescales.

With regards to solving the problem of localizing within a dynamic environment, the approach of Churchill and Newman (2013) is one of the most successful. They propose a system where the robot localizes in multiple experiences simultaneously, whenever it re-visits a previously mapped location. If localization against the known experiences fails, the map is augmented with a new experience based on Visual Odometry (VO). The new experience then becomes available for localization on the next visit. It is unclear how semantic information, such as road-definitions for automated navigation, would be handled in such a map as experiences have only topological connections, and no metric relationship. In contrast, our system fuses data from multiple visits to the same place, by adding locally resolved metric connections between experiences. Semantic data is then attached to a pose graph that remains the same over all visits.

Another successful visual multi-session mapping system has been presented by Dayoub et al. (2011). Their map-representation consists of a pose graph, where each node has an associated spherical view that stores the location of features from an omnidirectional image. Whenever a location is revisited, a state-machine modeled on human memory is used to decide which currently observed features are to be added to the these views: features first pass through short term memory, which serves to discard spurious observations, before they are admitted to long term memory. This is only one possible strategy for sampling useful features from a changing environment. Instead, we propose keeping features around between sessions, allowing us to take a batch offline processing approach to derive the best map based on all data. We then generate a reduced map for online usage on the vehicle, via sampling strategies that take into account associations between features across all sessions.

Similarly, Konolige and Bowman (2009) present an approach for selecting useful views from their graph-based map. For each metric neighbourhood in their map, they aim to identify a set of view-clusters that represent the various appearance states of the environment. Similar view-pairs are considered connected based on the maximal connected subsets of a graph that describes the local appearance-dynamics of a place. In order to manage map complexity, Konolige and Bowman propose a scheme that deletes views from a neighbourhood, while still preserving the diversity of clusters as far as possible. Again, this can be seen as an example for a strategy that samples useful features from an environment with variable appearance. While this strategy is limited to finding clusters in the appearance of frames, we propose scoring and discarding individual *landmarks*. This allows evaluating the usefulness of structures for localization on a sub-scene level.

Instead of using predefined image features, McManus et al. (2014b) propose to learn a set of detectors specific to each place in a map. These detectors are based on Support Vector Machine (SVM) classifiers, which are trained seperately for each place, based on sampled rectangular regions. The training data can include images from multiple visits, acquired under varying environmental conditions. This imbues the resulting bank of feature detectors with a high robustness to changes in visual appearance, sacrificing some of the accuracy of a point-feature based method. In contrast, we believe that varying appearance characteristics can equally be coped with by extending the representation of a place by adding newly observed features. This supports updating the map incrementally, as opposed to retraining a classifier. Furthermore, it is unclear how well the SVM training will perform in areas where large parts of the image are occupied by uninformative, but highly structured objects, such as in a car park.

The approach we take to long-term mapping is close in spirit to the method proposed by Johns and Yang (2013, 2014). They associate image features to real-world landmarks, that can be re-observed during multiple visits to the same place. Thus, appearance changes can be modelled at a sub-scene level, independently updating each landmark incrementally. While Johns and Yang focus on place recognition from images only, we apply a similar approach to the precise metric localization of an automated vehicle.

3 System Overview

There are a multitude of viable camera setups for automated vehicles. Configurations range from a single monocular camera (Lovegrove et al., 2011), to a lone stereo-pair (Ziegler et al., 2014a), to complex multi-camera rigs (Muehlfellner et al., 2013). Our map representation does not depend on the specifics of the camera system. The datasets and results presented in this paper are based on the setup shown in Figure 2.

Our lifelong mapping framework builds on the concept of *multi-session* maps. A *session* refers to a drive with the vehicle in a given environment, where data is acquired that can subsequently be used to build or extend a map. Sessions are recorded as discrete episodes, they can stem from automated or manual runs. *Datasets* are composed of images and wheel-tick odometry, associated with distinct time instants j and *vehicle frames* $\mathcal{F}_{V_j}$. The vehicle frames represent the 6-DOF pose (position and orientation) of the rear-axle center at time j. We assume that the cameras are triggered synchronously, and group the acquired images into *image-tuples*.

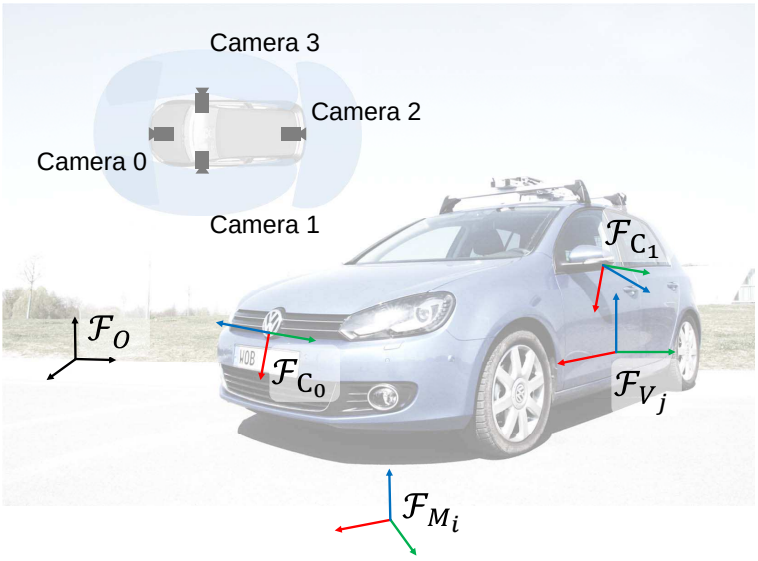


Figure 2: *Our driverless car, showing the location of the involved coordinate frames. Of the four wide-angle fisheye monocular cameras on the car, only the front and left cameras are visible, alongside their associated coordinate frames* $\mathcal{F}_{C_{0,1}}$*. Two additional cameras are arranged symmetrically on the right and the rear of the car.*

The mapping life cycle is shown in Figure 1 and can be split into four main steps.

1. Initialization populates the database with a fresh map based on data collected with a vehicle.
2. During regular online operation, vehicles localize in the environment and (optionally) log localization and sensor data for further subsequent processing.
3. The recorded data is transmitted to an off-vehicle computer, where the recorded data is fused into a large full-scale map which stores all the long-term information that is available about an environment.
4. Finally, from this full-scale map, a *Summary Map* can be created that is small enough for efficient and robust online operation.

The three steps after initialization can be repeated in a cycle.

The remainder of this section begins with an overview of the underlying map representation. We then describe our multi-session mapping pipeline in detail, covering initialization, localization, mapping, and the creation of summary maps.

3.1 Map Representation

Figure 3 depicts the coordinate frames contained in the map. To provide a better intuition about their location, the same coordinate frames are also indicated in Figure 2. In the following, we write ${}_A\mathbf{p}$ to describe the coordinates of a point p expressed in coordinate frame $\mathcal{F}_A$. $\mathbf{T}_{AB}$ refers to a 6-DOF transformation that takes points from $\mathcal{F}_B$ to $\mathcal{F}_A$, such that ${}_A\mathbf{p} = \mathbf{T}_{AB}\,{}_B\mathbf{p}$. The map consists of map frames, $\mathcal{F}_{M_i}$, vehicle frames, $\mathcal{F}_{V_j}$, and landmarks, ${}_{M_i}\mathbf{l}_k$. A relational database handles the interaction between the following elements:

Map frames are denoted $\mathcal{F}_{M_i}$. Estimation of vehicle poses and landmark positions takes place relative to these coordinate frames. They are initialized based on the first mapping dataset. An optimal estimate of each frame is stored as the transformation, $\mathbf{T}_{M_iO}$, which takes points from the arbitrary global frame $\mathcal{F}_O$ to frame $\mathcal{F}_{M_i}$. The transformation, $\mathbf{T}_{M_mM_n}$, between two arbitrary frames, $\mathcal{F}_{M_n}$ and $\mathcal{F}_{M_m}$, is computed via

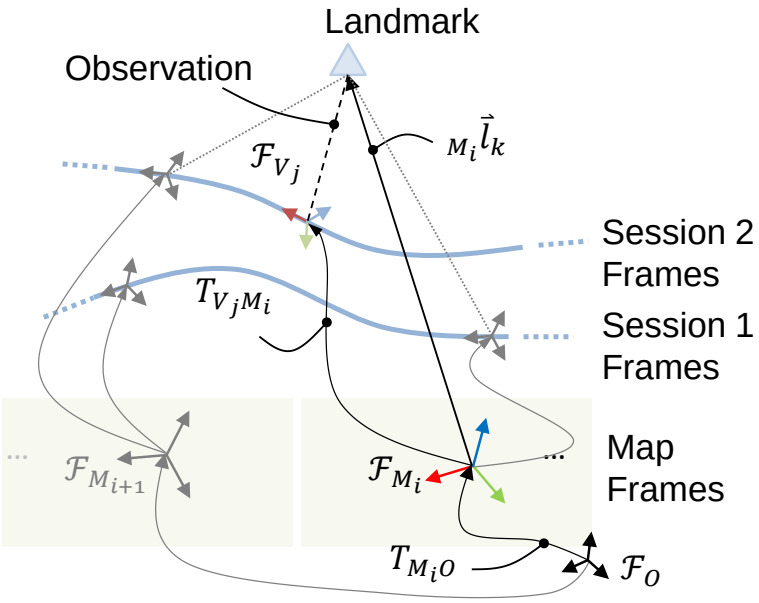


Figure 3: *The data structures and associations involved in the map. A single map and vehicle frame have been highlighted for clarity.*

the kinematic chain through the global frame

$$\mathbf{T}_{M_m M_n} = \mathbf{T}_{M_m O} \mathbf{T}_{M_n O}^{-1}. \tag{1}$$

One can think of the map frames as a skeleton graph of places where other geometric data (localized vehicle frames, landmarks) is stored.

Vehicle frames, $\mathcal{F}_{V_j}$, are attached to the vehicle at time j (see Figure 2), each with its associated image tuple and odometry information from that time. Each image gives rise to a set of sparse local image features, consisting of a 2D-image coordinate and a local descriptor. We refer to the features from a set of synchronized images as associated with a vehicle frame, rather than a specific camera image. Our representation is independent of the feature type. The results described in this paper rely on BRISK (Leutenegger et al., 2011), fixing the descriptor orientation to be aligned with the downward direction in the current vehicle frame. We chose BRISK, which uses a local binary pattern descriptor, for its fast detection, descriptor extraction and comparison, as well as minimal storage-size requirements. The choice of the exact image feature type is not crucial to the concepts descriped in this paper.

Each vehicle frame is associated with a set of intrinsic and extrinsic calibration parameters for the camera system. This is necessary because the calibration changes over time due to maintenance and normal wear and tear on the vehicle. We rely on the CamOdoCal library (Heng et al., 2014) to generate these parameters.

Localizing a vehicle coordinate frame refers to estimating the 6-DOF transformation (or simply *pose*) $\mathbf{T}_{V_j M_i}$ between it and the closest map frame, $\mathcal{F}_{M_i}$. During localization, associations between the features and the landmarks are made.

Landmarks represent 3D points in the real world. They arise from the triangulation of tracked image features. The k-th landmark is expressed as ${}_{M_i}\mathbf{l}_k$ in the map frame, $\mathcal{F}_{M_i}$, in which $\mathcal{F}_{V_j}$ is localized when the landmark is instantiated. A landmark's associated features are stored as *observations* in the map. Each landmark has a corresponding image descriptor that arises from its observations and that is used to associate it with further features.

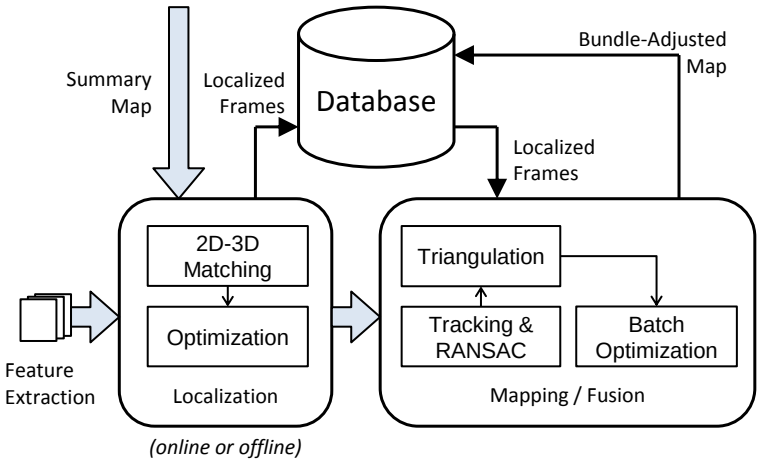


Figure 4: *The pipeline for the localization and mapping steps of the life cycle depicted in Figure 1.*

3.2 The Multi-Session Pipeline

At first, the map is based on a single dataset. Thus, the $\mathcal{F}_{M_i}$ initially correspond to a concrete dataset. Over time, as the map is extended with data from more sessions, an increasing number of vehicle frames end up referring to each map frame. This turns each $\mathcal{F}_{M_i}$ into an abstract place or small submap that can contain contributions from all sessions in the map. Localization is performed relative to these map frames, via landmarks potentially accumulated from all sessions. Semantic data, such as the road and lane definitions for navigation, labels for pedestrian crossings and traffic signs, etc. are also defined relative to these coordinate frames.

After initialization (Section 3.2.1), the map can be used for localization (Section 3.2.2). More importantly, the map can also be augmented based on data gathered in later sessions (Section 3.2.3). We refer to this process as multi-session mapping. It is depicted in Figure 4, which provides a more detailed view of the localization and mapping parts in the lifecycle. Multi-session mapping can be broken down into the following steps:

1. Localize each new vehicle frame and create data associations to the existing map. This involves matching 2D features with 3D points (landmarks) and optimizing the estimated pose.
2. Track 2D features between consecutive images. The resulting triangulated 3D points are then used to initialize new landmarks.
3. Perform global batch optimization over all sessions to ensure geometric consistency. In this step reprojection and odometry errors are minimized.

The creation of Summary Maps is a separate process (Section 3.3). It produces a map that is best suited for online operation and involves a separate set of objectives and design trade-offs.

3.2.1 Initialization

To build an initial set of vehicle and map frames from the first dataset, we first produce an intermediate form based on a state-of-the-art mapping pipeline (Lee, 2014). This pipeline sub-samples the dataset by picking vehicle frames and image-tuples such that consecutive frames are evenly spaced over the recorded trajectory, separated by a minimum baseline distance. Multi-camera visual odometry, loop-closure detection (Lee et al., 2013) and pose graph relaxation are performed in sequence on the selected vehicle frames. This produces a set of vehicle poses $\mathbf{T}_{V_jO}$ relative to some arbitrary global frame, as well as the features and odometry information associated with each retained vehicle frame $\mathcal{F}_{V_j}$.

The intermediate form is then transferred into the map representation described in section 3.1. A map frame $\mathcal{F}_{M_i}$ is created for each $\mathcal{F}_{V_j}$. The transformation $\mathbf{T}_{V_j M_i}$ between the two is initialized to identity, and similarly $\mathbf{T}_{M_i O} = \mathbf{T}_{V_j O}$. Finally, landmarks are initialized based on local image features tracked between the vehicle poses.

3.2.2 Localization

Localization means estimating a pose between a vehicle frame and one of the map frames. We use the same process for localization both online during an automated run and offline in batch mode. To estimate the pose, we formulate an optimization problem whose inputs are an initial guess and a set of correspondences between 2D image features and 3D landmarks. When initializing the localization pipeline on a new dataset, we assume that the vehicle starts at one of the map frames. This simple method works well in practice when the vehicle is always started in roughly (a 1-2 meter radius) at the same location. A more general scheme could be based on appearance-based global localization (e.g. (Cummins and Newman, 2008, 2011)) or a GNSS estimate.

To determine the correspondences, we first predict a set of likely-visible landmarks via an odometry pose

$$\mathbf{T}_{V_j M_i} = \hat{\mathbf{T}}_{V_j V_{j-1}} \mathbf{T}_{V_{j-1} M_{i-1}} \mathbf{T}_{M_{i-1} M_i}, \tag{2}$$

where $\mathbf{T}_{V_{j-1} M_{i-1}}$ is the previous pose, $\hat{\mathbf{T}}_{V_j V_{j-1}}$ is the odometry update, the new map frame $\mathcal{F}_{M_i}$ is chosen to be the one nearest in translation to the predicted pose, and $\mathbf{T}_{M_{i-1} M_i}$ is calculated by following the kinematic chain between $\mathcal{F}_{M_i}$ and $\mathcal{F}_{M_{i-1}}$. Then we select all landmarks that are associated with the vehicle frames that lie in a metric neighbourhood of the guessed pose and transform them to $\mathcal{F}_{M_i}$ by following the kinematic chains from their respective base frames, and from there into the vehicle frame via the predicted pose. Finally, each landmark is projected into the cameras and matched to observed features based on distances in image and descriptor space.

The resulting correspondences, also termed 2D-3D matches $n = (m, k)$, associate the measured image location of a 2D feature $\hat{\mathbf{x}}_m$ with a 3D landmark $\mathbf{l}_k$. Each match gives rise to a *reprojection error*

$$\mathbf{e}_n = \hat{\mathbf{x}}_m - \mathbf{h}(\mathbf{K}_{C_m}, \mathbf{T}_{V_j M_i} \, {}_{M_i}\mathbf{l}_k), \tag{3}$$

where the measurement model $\mathbf{h}(\cdot)$ takes landmarks ${}_{M_i}\mathbf{l}_k$ transformed to the vehicle frame using the current estimate $\mathbf{T}_{V_j M_i}$, and projects them into image space based on the parameters $\mathbf{K}_{C_m}$ of the camera C_m associated with the feature.

Pose optimization is performed using an iterative non-linear least-squares approach with re-weighting which revolves around the weighted sum of reprojection errors

$$J_{\text{loc}} = \sum_n \mathbf{e}_n^T \mathbf{W}_n \mathbf{e}_n, \tag{4}$$

where $\mathbf{W}_n$ is a weighting matrix based on the inverse measurement covariance of $\hat{\mathbf{x}}_m$ and a robustifying weight (m-estimator, see Zhang (1997)). The outcome of the optimization is an improved estimate of $\mathbf{T}_{V_j M_i}$. It also leads to a set of *inlier* feature-landmark correspondences defined by thresholding their reprojection error after optimization

$$\Omega_{\text{in}} = \{n : \|\mathbf{e}_n\| < \tau_r\} \tag{5}$$

Finally, localization is considered successful if there are sufficiently many inlier matches, i.e. if $|\Omega_{\text{in}}| \geq N_{\text{inlier}}$. Failed localizations are handled by propagating the odometry-based guess for $\mathbf{T}_{V_j M_i}$ instead of the optimized estimate.

3.2.3 Mapping

Mapping starts with features that are tracked during localization. As this involves triangulation, we use odometry to make sure that the images used by the mapping pipeline are separated by a minimum baseline distance. In practice on our setup, performance is not very sensitive to the exact value of the baseline within a fairly large range. The results presented in this paper use a baseline of 10cm. We use epipolar constraints arising from odometry, together with the extrinsic camera parameters, to create 2D-2D data-associations between image features from consecutive frames. These feature-feature correspondences form the input to a RANSAC estimation step. We use the iterative eigenproblem formulation proposed by Kneip and Lynen (2013) and implemented in the OpenGV library (Kneip and Furgale, 2014). The RANSAC-inliers are used to triangulate 3D-points. In turn, the triangulated points are used to initialize landmarks, bound to the map based on the pose-estimates from the localization step.

In the end, successfully localized frames have features that are associated to landmarks from the old map, as well as completely new landmarks. Both new and old landmarks are expressed in the same map frames, and are bound together by having been observed from the same vehicle frames. To ensure that they are geometrically consistent, the landmark positions ${}_{M_i}\mathbf{l}_k$ and vehicle poses $\mathbf{T}_{V_j M_i}$ are fed to a batch non-linear optimization step which minimizes

$$J = J_{\text{vis}} + J_{\text{odo}}, \tag{6}$$

where J_{odo} arises from the wheel-tick odometry measurements between frames (see below), and J_{vis} has the same expression as J_{loc} but uses the correspondences from the batch optimization process (i.e. all observations of landmarks in the map and their associated vehicle frames are included).

To compute J_{odo}, let Ω_{odo} be the set of all indices j of vehicle frames in the map, but excluding the first frame for each session. Each of the vehicle frames indexed by Ω_{odo} has an associated odometry update $\hat{\mathbf{T}}_{V_j V_{j-1}}$. The kinematic chain in the map between the current and the previous vehicle frames provides another estimate for the same motion $\mathbf{T}_{V_j V_{j-1}}$. To measure the difference between these two estimates, let

$$\mathbf{T}_{e_j} = \hat{\mathbf{T}}^{-1}_{V_j V_{j-1}} \mathbf{T}_{V_j V_{j-1}}. \tag{7}$$

and define odometry error terms $\mathbf{e}_j$ as

$$\mathbf{e}_j = \begin{pmatrix} \mathbf{t}_j \\ \mathbf{a}_j \end{pmatrix}, \tag{8}$$

which stacks the translational $\mathbf{t}_j$ and rotational $\mathbf{a}_j$ parts of $\mathbf{T}_{e_j}$. We express $\mathbf{a}_j$ in axis-angle form. The odometry error cost is then computed as

$$J_{\text{odo}} = \sum_{j \in \Omega_{\text{odo}}} \mathbf{e}_j^T \mathbf{W}_{\text{odo}} \mathbf{e}_j \tag{9}$$

where the weight matrix $\mathbf{W}_{\text{odo}}$ is set according to the inverse measurement covariance of the wheel-tick odometry.

Following batch optimization, feature-landmark observations that have reprojection errors larger than a threshold τ_m are removed from the map. This completes the mapping cycle. When more sessions are added, all landmarks are treated the same: they are matched with the new data and associated with additional frames from those sessions.

3.3 Summary Maps

The process described in section 3.2.3 solves the problem of adding new information to the map. Now we describe a framework for bounding its size while maximizing its usefulness for on-vehicle operation. The basic idea is to select a subset of pertinent landmarks for the map that gets downloaded by the vehicle. The resulting *Summary Map* describes the content of the full map in a simplified way. This concept can be seen as an analogy to the idea of a *Summary Statistics*, which concisely describe a set of observations.

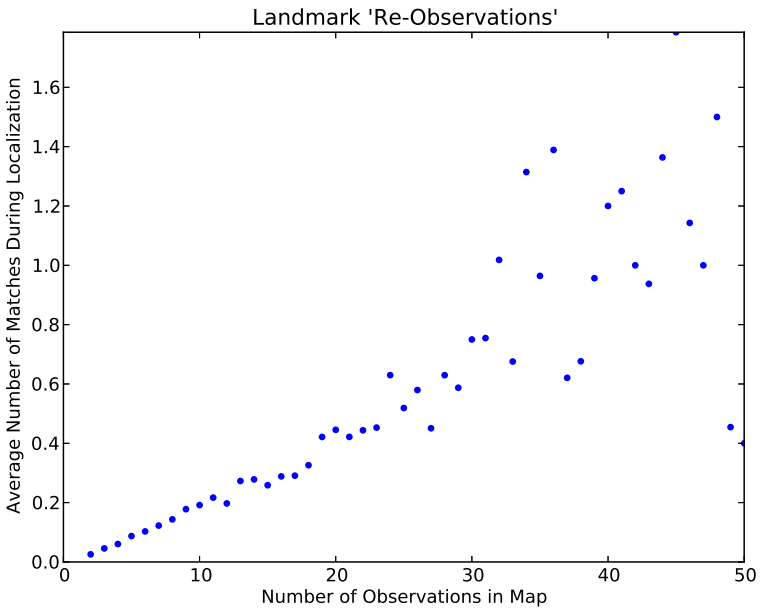


Figure 5: *The number of times a landmark is seen during mapping, versus the average number of times it is matched during localization, at a later point in time. X-Axis is truncated at 50 observations.*

We aim to build a compact map that robustly handles long-term environment dynamics. It is useful to first consider some of the phenomena that one encounters during long-term mapping, in order to come up with strategies for choosing landmarks from a multi-session map for the best localization results

Part of the information added with each mapping session comes from features that are visible only for a short time, such as parked cars or ephemeral shadows. Landmarks initialized from these features are not useful for localizing at a later point in time. This kind of *visual chaff* is already present in the initial map, and will bloat the map when more sessions are added to it.

Other features might not be visible during every session because their appearance depends on specific conditions, e.g. a certain incidence angle of light. Even though these *intermittent features* disappear, they are still caused by persistent structure in the environment and might be available during future sessions.

The appearance of the environment is likely to be subject to *long-term change*, causing features that are visible for some time and then disappear permanently. Seasonal changes that are not cyclic – such as bushes wilting during the winter and regrowing in a different way – as well as changes due to construction fall into this category.

In addition to the behaviour of individual features, we also need to consider the *spatial distribution of landmarks*. We expect some areas of the world to be relatively rich in persistent useful structure, while others are more deprived. Therefore, context-sensitive restrictions on the included landmarks should be applied, based on the properties of the local neighbourhood in the map.

To handle these requirements, we propose a framework of landmark *scoring functions* and *sampling policies*, which are applied to a multi-session map. Methods from literature for assessing the usefulness of Landmarks, such as the memory model proposed by Dayoub et al. (2011) and the appearance-based image clustering proposed by Konolige and Bowman (2009) can be expressed in this framework. In section 5 we report results based on the following concrete examples, designed to handle the phenomena identified above in the environment where we collected data. Note that our framework can easily be extended with different and more elaborate functions and policies.

A scoring function provides a prediction of the usefulness of an individual landmark for localization. Beyond the geometric point, ${}_{M_i}\mathbf{l}_k$, they are also associated with a set of observations, $\mathcal{O}_k$, that link the landmark to specific image features. Let $\mathcal{L}$ denote the set of all landmarks in the map, together with their associated observations.

A scoring function, f, assigns a real valued score to each landmark

$$f : \mathcal{L} \to \mathbb{R} \tag{10}$$

An intuitive choice for a scoring function is the number of observations, $|\mathcal{O}_k|$, for each landmark. This is backed by the data: Figure 5 shows the strong correlation between the number of times a landmark is seen in a mapping dataset and the number of times it is successfully matched to features from a test dataset. The aim of this function is to get rid of visual chaff in the form of landmarks that are visible only for very short time-spans. It does not distinguish between features that are tracked well throughout a single dataset, and features that are persistent over a longer time-span.

To allow this distinction, the set of sessions $\mathcal{D}_k$ associated to a landmark via its observations can be considered. Landmarks can then be scored by the number of distinct sessions $|\mathcal{D}_k|$ they appear in. This scoring function trades between removing intermittent landmarks, that have only been seen during few sessions but might reappear, and landmarks that only appeared during a single session.

Scoring landmarks according to the time since their last appearance in a mapping session provides a way of distinguishing features that are lost due to long term changes. A simple scoring function can be designed for this purpose, that counts the number, $\Delta\mathcal{D}$, of sessions since the landmark has last appeared.

The state-machine proposed by Dayoub et al. (2011) for modelling biological Short- and Long Term Memory behaviour can also be turned into a scoring function on the Landmark-level. This scoring function produces a binary output, returning 1 if a Landmark is present in memory and 0 if it is not. The state-machine which is used for producing these values initializes each Landmark in Short Term Memory. For each observation, the Landmark is advanced through a chain of stages until it is considered stable and passed on into Long Term Memory; if an observation is missed for a session, the memory state of the Landmark is reset to the start of the chain. When in Long Term Memory, Landmarks are kept around unless they are missed for a certain consecutive number of sessions. This process is run individually for each Landmark, and results in a binary score.

Sampling policies select a subset of $\mathcal{L}$ based on the various scores of individual landmarks. Furthermore, they take into account the geometric and topological relationships between landmarks (see section 3.1). Multiple scoring functions can be combined in a sampling policy. The selected subset of landmarks forms the Summary Map, that can be used for localization on-board the vehicle.

The simplest way to define a sampling policy is to use a threshold τ_s on the scoring function for each landmark. Landmarks that score lower than this threshold are discarded. This does not deal well with different distributions of the landmark score, for example due to regions, or whole maps, that are starved of good features.

A natural technique to work around the problem of different score distributions is to use a ranking-based approach. The landmarks are sorted (for instance lexicographically) according to the values assigned to them by the scoring functions, and then the top, N_s, ranked landmarks are selected. This approach allows to directly tune the number of landmarks, N_s, that remain in the map. The ranking can be applied globally, or landmarks can be ranked in a sliding window, moving over the map frames, $\mathcal{F}_{M_i}$. Such a sliding window-ranking approach is similar to the set-cover method proposed by Li et al. (2010) for localization in unordered image collections.

The least-recently used (LRU) View Deletion algorithm proposed by Konolige and Bowman (2009) also fits into the sampling-policy framework. For each local metric neighbourhood, defined as a neighbouring group

of map frames, we retrieve all associated vehicle frames. Next, a similarity-measure between vehicle frames is needed for clustering; in (Konolige and Bowman, 2009), appearance similarity it is defined as

$$c(\mathcal{F}_{V_a}, \mathcal{F}_{V_b}) = \frac{m}{\tilde{m}} - 1, \tag{11}$$

where, $\tilde{m}$, is the number of inliers matched between frames in the local neighbourhood, $\mathcal{F}_{V_a}$, and, $\mathcal{F}_{V_b}$, and m is the minimum number of features in either frame.

The number of inliers matched between frames, $\tilde{m}$, is equivalent to the number of Landmarks shared between the frames. Based on this similarity-measure, we follow Konolige and Bowman by defining a graph with edges between vehicle frames if $c(\mathcal{F}_{V_a}, \mathcal{F}_{V_b})$ is smaller than some threshold, and defining clusters as the maximally connected subsets of this graph. This is followed by deleting vehicle frames from each neighbourhood until a target-number of frames is reached; this is done according to the LRU algorithm described by Konolige and Bowman (2009). Landmarks that are associated with the deleted vehicle frames are also removed. Note that there is no scoring function associated associated with this sampling policy per se.

4 Experimental Setup

To evaluate the proposed multi-session mapping pipeline, we use datasets collected with our research vehicle for three different experiments. In this section, we give an overview of these datasets, followed by a description of the different summary strategies we will test on the data. We also describe the procedures we use for determining the values for *availability*, *accuracy* and *complexity*, that serve as the basis for evaluation in the next section.

All data was collected with our two similar research vehicles (Figure 2). Gray-scale images from four cameras were recorded at a resolution of 1280×800 pixels at a rate of 12.5Hz (at an interval of 80ms). For operating the pipeline, the images were sub-sampled to 640×400 pixels. Exposure for all cameras was triggered at the same time, but due to different exposure-times we observed a variance of up to 16ms between the actual arrivals of images. Wheel-tick odometry samples were collected at a rate of 100Hz and synchronized with the images via their common NTP-based timestamps. In addition to cameras and odometry, we also recorded data from an iMar-iTraceRT-F400 INS/GNSS, which was available on one of the two test vehicles. The INS provides pose-estimates with a precision in the order of centimeters, thus serving to provide 'ground-truth' estimates for evaluating the metric localization performance of the proposed pipeline.

In the car, we use an industrial-grade computer, with a desktop-architecture Intel Core i7 2600 CPU clocked at 3.4 GHz, 16 Gigabytes of RAM, and a Solid State Drive (SSD) for storing data. Processing for the localization pipeline was carried out offline on a copy of this machine with the same specifications. While we performed the localization experiments offline, we have been running the proposed localization pipeline on the vehicle for more than one year, successfully demonstrating automated driving based on the provided localization results on numerous occasions. For batch-optimization of the multi-session maps, we utilized the "Euler" high-performance computing cluster at ETH Zürich.

4.1 Datasets

The datasets we use in our experiments were captured in two locations. The two *Parking Lot* datasets are depicted in Figure 6, whereas the *Suburban* dataset is shown in Figure 7b. Detailed information about each recording, including the time and weather conditions, is given in Appendix A.

The **Year-Long Parking Lot** dataset was collected in the busy parking area close to our research campus. The DGPS tracks for the dataset route, which is of about 200 meters long, is shown in Figure 6c. A total

(a) Year-Long Parking Lot Dataset



(b) Day-Night Parking Lot Dataset





(c) IMU/DGPS tracks. White arrow indicates approximate pose and viewing direction for example images.

Figure 6: *Example images for the 'Parking Lot' datasets. Each image is cropped from the rear camera view at approximately the same location. The left column shows the year-long dataset-, the right column the day-night dataset. In the bottom left, the IMU/DGPS tracks associated with the datasets are shown superimposed on an aerial image.*

of 33 recordings were collected, starting in August 2013 and continuing until the end of July 2014, with a break in February–March 2014. Figure 6a illustrates the dynamics of an example scene, demonstrating the changes in parked cars, lighting and foliage.

This dataset contains a complete cycle of seasons, various weather and lighting conditions, as well as structural change—especially in the form of a different configuration of parked cars every day. While the traversed path is relatively short, the challenging aspect of this parking area is that it contains little permanent structure. The campus building is visible in the distance on in some places, but often the views are occluded by parked cars. Lamp posts and the (sparse) road markings appear largely unchanged. Some structure is offered by the



(a) *IMU/DGPS tracks. White arrow indicates approximate pose and viewing direction for example images.*



(b) *Year Long Suburban dataset.*

Figure 7: *Top: the IMU/DGPS tracks associated with the year-long 'Suburban' dataset. Bottom: example snapshots from the dataset. Each image is a cropped selection of the rear camera view, approximately at the same location.*

trees throughout- and bordering the area, but their appearance changes heavily during the different seasons.

The **Day-Night Parking Lot** dataset records the transition from bright daylight to darkness on the same path through the parking lot. Data was collected during a single day from mid-afternoon (15:00) to evening (17:45), with a higher density during sunset (16:20). This amounted to a total of 20 recordings. Much of the structure remains in place for this dataset—the parked cars are mostly unchanged—but the differences in lighting are dramatic. During this overcast night, only the parts of the environment that are illuminated by artifical light are visible to the cameras. The most notable light sources are street lights, and the illuminated windows of the campus building.

Finally, the **Year-Long Suburban** dataset was recorded in a residential area in Wolfsburg, Germany. The DGPS measurements for the approximately one-kilometer long route in this dataset are depicted in Figure 7a. Recordings were made from October 2013 to October 2014, with a longer break in winter from December-April, amounting to a total of 22 recordings. Again, example images that depict one scene throughout the year, are shown in Figure 7b.

In contrast ot the Parking Lot datasets, there is a lot more permanent structure in this suburban area. Single-family homes and apartment buildings, fences and driveways, remain largely unaffected by long-term structural change, except for some ongoing minor construction. Vegetation does change as the seasons move on, but it plays a minor role in the overall appearance of this dataset; so do parked cars and various dynamic objects. The visual variability is therefore dominated by changes due to weather and lighting. In addition, the roads in this area are relatively wide, compared to the parking area—changes in viewpoint occur more often, as we avoid oncoming traffic and street-parked cars.

4.2 Strategies and Parameters

Table 1: *Scoring functions, sampling policies and parameters employed for summarization. Combinations are grouped according to three general* strategies.

Strategy	Key	Scoring Function $f(l_k)$	Sampling Policy	Parameters	Description
Single Session	Single	-	-	-	Map built from a single dataset, not summarized.
Full Multi Session	Full	-	-	-	Map built from all available datasets, not summarized.
Latest Datasets	Latest	$\begin{cases} 1 & \Delta\mathcal{D} \leq \tau_s \\ 0 & \dots \end{cases}$	$f(l_k) = 1$	$\tau_s = 1$	Only accept landmarks that have been seen in the most recent dataset.
Fixed Threshold	D2	$\lvert\mathcal{D}_k\rvert$	$f(l_k) \geq \tau_s$	$\tau_s = 2$	Score by the number of distinct datasets, $\lvert\mathcal{D}_k\rvert$, associated with each landmark.
Local Ranking	LDO	$\lvert\mathcal{D}_k\rvert, \lvert\mathcal{O}_k\rvert$	Local Best N_s	$N_s = 1200$	First rank by $\lvert\mathcal{D}_k\rvert$, then by $\lvert\mathcal{O}_k\rvert$, pick the best 1200.
Memory Model	MM	Memory State Machine	$f(l_k) = 1$	$N_{STM} = 2$ $N_{LTM} = 4$	The state machine described in Dayoub et al. (2011), applied to each landmark with N_{STM} short-term- and N_{LTM} long-term- memory states.
Appearance Clustering	AC	-	LRU Removal on Local Frame Clusters	$\tau_s = 10$ $N_c = 3$ $N_l = 5$	Find clusters based on similarity threshold $\tau_s = 10.0$. Remove least recently used Vehicle Frames from neighbourhoods of consecutive N_l Map Frames until only N_c clusters are left. Described in Konolige and Bowman (2009).

For our evaluation, we consider a number of different Summary strategies. Each strategy is built by combining a scoring function, a sampling policy and a set of parameters for scoring/sampling. The evaluated strategies are shown in Table 1. The Table gives a key for each scoring function, that is used throughout the figures in Section 5. Entries for the "Single" and "Full" maps, which are not summarized but are nonetheless part of the evaluation, are also included in the table.

The *Latest Datasets* strategy chooses Landmarks that were seen during the most recent traversals of the environment. By adjusting the threshold, τ_s, the number of sessions since the last observation of the landmark is selected. This strategy is a simplified version of the way appearance-fusion is handled by pose graph pruning approaches such as (Johannsson et al., 2012).

The *Fixed Threshold* strategy involves picking landmarks based on a threshold on the scoring function. Landmarks are admitted to the map whenever their score exceeds the given threshold. We use the number of distinct datasets a Landmark was seen in, $|\mathcal{D}_k|$, as an example scoring function to go with this sampling policy.

While these first two strategies have been selected to demonstrate the basic principles of map summarization, we also compare several more advanced approaches. These include a ranking based approach of our own design, as well as two strategies adapted from literature.

The *Local Ranking* strategy selects a fixed number of Landmarks for each Map Frame. A lexicographical ordering, based on multiple scoring functions, is applied to the Landmarks visible at each Map Frame. The highest ranking N_s Landmarks are included in the summarized map. For this evaluation, we first use the number of distinct datasets, $|\mathcal{D}_k|$, and second the number of observations, $|\mathcal{O}_k|$, to build the ordering.

The *Memory Model* strategy is based on the approach described by Dayoub et al. (2011). A memory state is tracked per Landmark and per Map Frame; after a Landmark has initially been observed at a Map Frame, it is expected to be seen during all additional sessions. Starting out in Short Term Memory, Landmarks need to be seen N_{STM}-times consecutively, before they are admitted to Long Term Memory. Landmarks are kept in Long Term Memory, unless they are missed N_{LTM} consecutive times. Both Landmarks from Long Term as well as Short Term Memory are included in the Summary Map. By Dayoub et al. (2011), Long-Term Landmarks are matched first during localization. Instead, we perform robust matching of all Landmarks, regardless of their memory state.

The *Appearance Clustering* strategy applies the approach described by Konolige and Bowman (2009). We select local metric neighbourhoods as groups of N_l Map Frames. For each of these groups, the associated Vehicle Frames are clustered based on a similarity threshold, τ_s. N_c Vehicle Frames are selected for inclusion in the map, following the LRU deletion algorithm described by Konolige and Bowman (2009). All Landmarks that are associated with the selected Vehicle Frames are included in the Summary Map.

4.3 Evaluation Criteria

In this paper, we use three criteria to evaluate different localization and mapping strategies: *availability* measures the percentage of time the robot is able to localize within the map, *accuracy* measures the metric error of the robot's six degree of freedom (6-DOF) pose estimate with respect to ground truth, and *computational complexity* measures the computational cost of localizing within the map.

4.3.1 Availability

The availability score is based on the ability of the pipeline to decide when localization was successful. This is similar to the binary output for localizers proposed by Churchill and Newman (2013). To decide if localization for a given vehicle pose was successful, we classify landmark-feature matches as either inliers or outliers. This is done based on a threshold, τ_r, given in pixels, on the reprojection error,

$$r = \sqrt{\mathbf{e}_n^T \mathbf{W}_n \mathbf{e}_n}, \tag{12}$$

for each match (see Equation 3). Localization is considered successful if a minimum number of inlier matches, N_{inlier}, with $r < \tau_r$, is found over all cameras. The values for the reprojection error threshold and the minimum number of inliers have been manually selected based on our experience with automated driving in various scenarios. The used values were: $\tau_r = \sqrt{3}$ pixels and $N_{\text{inlier}} = 40$.

When considering availability, it is important to note that the proposed pipeline uses wheel-tick odometry in order to predict the vehicle pose between localizations. This introduces an interaction between the driving speed, localization rate, and the robustness of our system: larger gaps between localizations mean that larger relative errors are accumulated, and need to be corrected. However, the precision of wheel-tick odometry – which is usually excellent over short intervals – is not the sole source of error here. Localization errors in rotation, that are propagated from a previous point in time, also play an important role.

We eliminate the influence of driving speed and localization rate, by using only a subset of image-tuples from each test dataset. These samples are chosen at an evenly spaced intervals of 1.5m for all runs, where distance is measured by wheel-tick odometry. This roughly corresponds to a car driving at 27 km/h, while localizing at a rate of 5 frames/second. We evaluate the framerates achieved by our localization system separately.

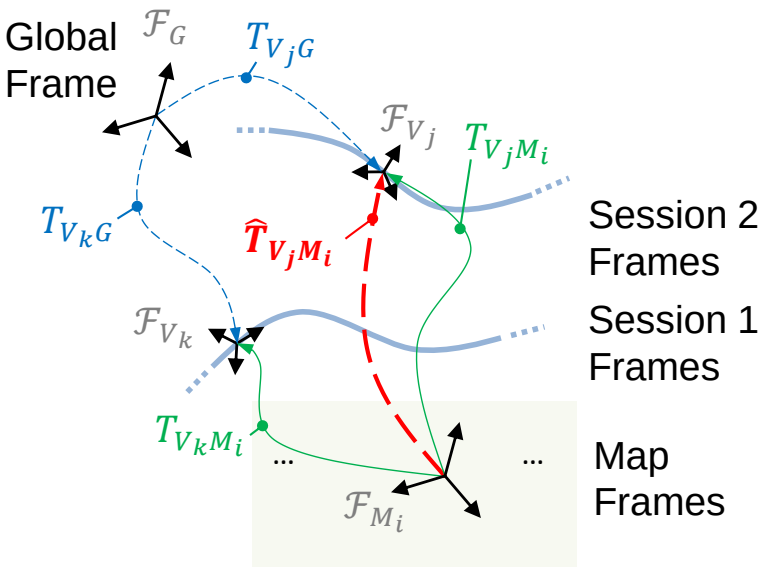


Figure 8: *Coordinate frames and transformations involved in calculating the local metric error. Transformations provided by INS are shown as dashed blue- and vision-based estimates in solid green arrows. The calculated 'reference' transformation* $\hat{\mathbf{T}}_{V_jM_i}$ *is shown in bold-dashed red.*

The availability of wheel-tick odometry also means that it is possible to recover from momentary failures in localization. In case a localization attempt is classified as a failure, we ignore the estimated pose, and use the prediction made by odometry instead. With the next frame, localization is attempted anew.

4.3.2 Accuracy

To evaluate the quality of our localization results, we calculate the *local* metric error. The local error quantifies the difference between the estimated transformation, $\mathbf{T}_{V_jM_i}$, and a ground truth transformation, $\hat{\mathbf{T}}_{V_jM_i}$, between the the local map coordinate frame, $\mathcal{F}_{M_i}$, and the current vehicle coordinate frame, $\mathcal{F}_{V_j}$. This error only evaluates the *local* accuracy of the estimate, including the localization and local mapping errors. Global metric accuracy of the map is disregarded this way, but in our framework, it is also not essential for automated path-planning and following (Bosse et al., 2003).

An overview of the coordinate frames and transformations involved in calculating the error is given by Figure 8. In order to calculate $\hat{\mathbf{T}}_{V_jM_i}$, time-synchronized INS estimates, $\mathbf{T}_{V_kG}$, relative to some[1] global coordinate frame, $\mathcal{F}_G$, are stored for all vehicle frames, $\mathcal{F}_{V_k}$, in the map. When localizing, the 'mapped' vehicle frame, $\mathcal{F}_{M_j}$, closest to the current vehicle frame, $\mathcal{F}_{V_j}$, is selected. $\hat{\mathbf{T}}_{V_jM_i}$ is then calculated as $\mathbf{T}_{V_jG}\mathbf{T}^{-1}_{V_kG}\mathbf{T}_{V_kM_i}$, where $\mathbf{T}_{V_jG}$ is the INS estimate for the current frame, and $\mathbf{T}_{V_kM_i}$ is the stored estimate for the pose of the closest vehicle frame in the map.

Finally, we calculate the error as $\hat{\mathbf{T}}^{-1}_{V_jM_i}\mathbf{T}_{V_jM_i}$ resulting in a 6-DOF transformation. To quantify the error values present in this transformation, we split it into translation and rotation components $\mathbf{t}_e$ and $\mathbf{R}_e$. We take the root mean squared error (RMSE) of the translation errors for a test-run with N samples as, $\frac{1}{N}\sum_{i=0}^{N}\mathbf{t}_e^T\mathbf{t}_e$). Rotational error is calculated by transferring $\mathbf{R}_e$ into an axis-angle representation and using the average magnitude of the rotation angle, $|\phi_e|$, as the error metric.

[1] Specifically, the origin of UTM 32U.

4.3.3 Computational Cost

When operating the proposed pipeline both the memory required for storing maps on the vehicle as well as the timing-performance of the system are important factors. A major concern is how these costs evolve, under the application of different summarization strategies, as we add more datasets to the map. For measuring the cost of storing maps, we consider the number of landmarks in each map; this dominates all other costs, and is directly proportional to the physical map size.

5 Results

We have described a pipeline for building a multi-session map in order to cope with long-term changes in an environment. Furthermore, we have gathered long-term datasets for two different scenarios, as well as a dataset that records the lighting transition between day and night. In this section, we want to answer two main questions:

1. Can we build a maps for the different scenarios, that are usable for localization in all seasons, weather and lighting conditions?
2. What is the tradeoff between the computational cost associated with these maps and the availability/error in localization, when we employ the proposed summarization procedures?

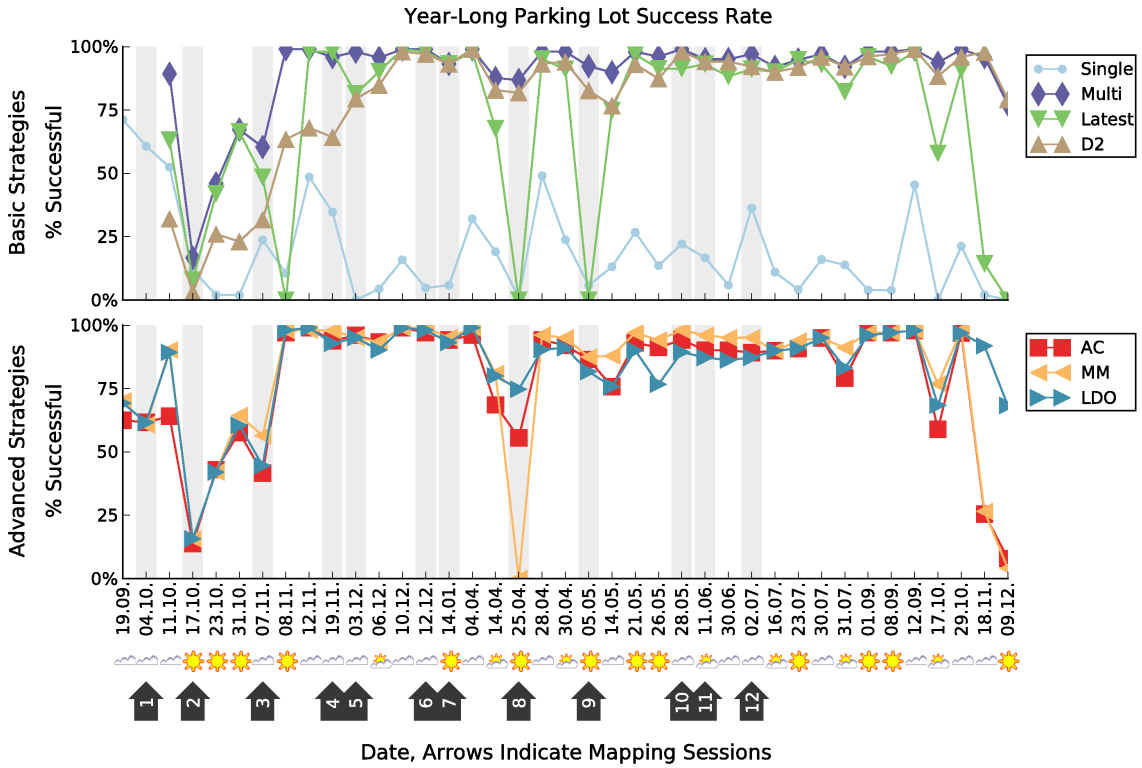
Referring to the first question, we analyze the availability of localization in the proposed framework. To determine if the generated maps are useful, we also analyze whether the metric error of the 6-DOF pose estimates they provide. With regards to the second question, we look at the size of the map, as well as the computational cost – expressed as the framerate of our localization system – related to it.

We build multi-session maps for each scenario, using manually selected recordings from the available pool of each dataset. The size of this selection was motivated primarily by limits on the computation time for the mapping process. These *mapping-sessions* were chosen first to sample regularly over the time spans covered by the datasets and second to reflect a range of environmental conditions. The mapping-sessions picked in this way are identified in table 2-4 in Appendix A. The other recordings were used as additional test data.

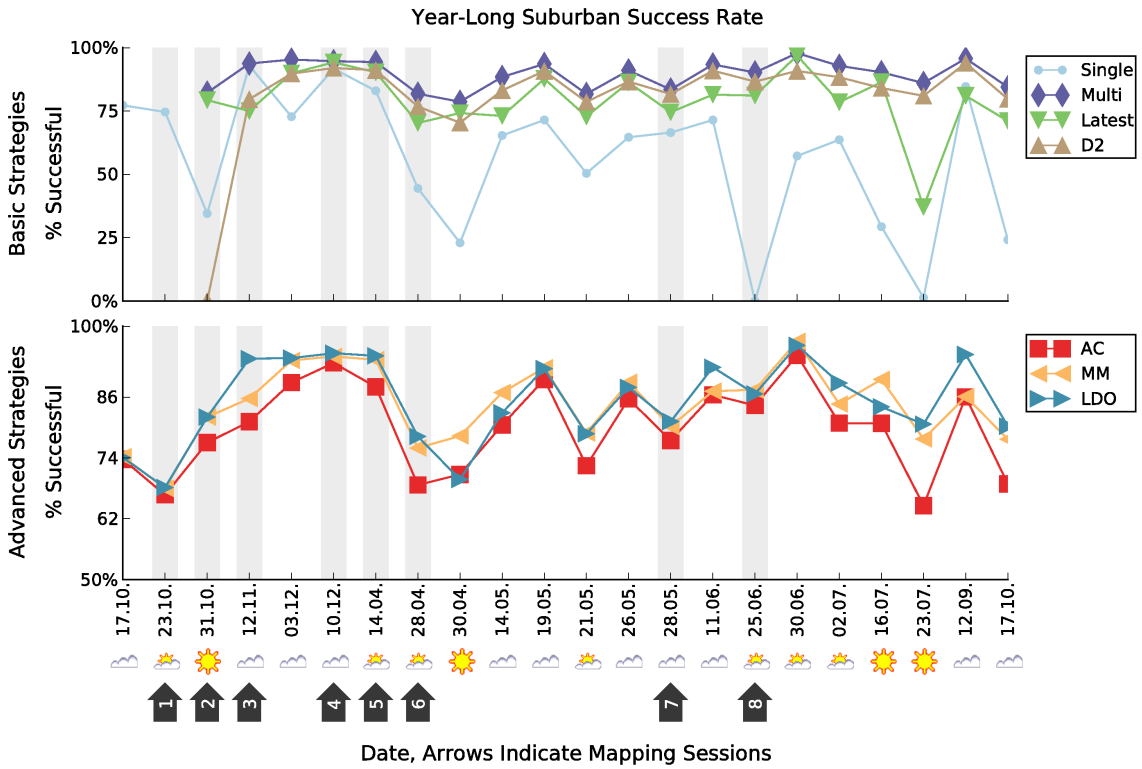
For the Year-Long Parking Lot and Suburban datasets, we used an *incremental* mapping and test procedure. Starting with an initial map base on the first mapping-session for each dataset, we built up the map by fusing all mapping-sessions in chronological order. At each time step, we also generated several Summary Maps based on the most recent full multi-session map. For this purpose, the Summary strategies described in Section 4.2 were used. All recordings were localized against the *most recently* available full and summarized maps, as well as the initial 'single-session' map.

The evaluation of the Day-Night dataset followed a different mapping process. We first built a *complete* map, based on all mapping-sessions added in chronological order, which we then summarized. The test datasets, which were not fused into the map, were used to evaluate the localization performance using the complete map. We chose this process in order to evaluate the performance of the complete day-and-night map under all lighting conditions.

In this section, we first provide a detailed description of the experimental results for our three evaluation criteria – availability, accuracy and computational cost. Afterwards, we will discuss the general findings that we identify based on these results.



(a) Availability for the Year-Long Parking Lot dataset.



(b) Availability for the Year-Long Suburban dataset.

Figure 9: *Availability as localization success rate for the Year-Long datasets. Calculated as the number of successful localizations (see 4.3.1) divided by the total of localization attempts, per map and dataset in chronological order. Split between basic and advanced summarization strategies. Arrows at the bottom indicate when a mapping dataset was injected into the multi-session map. Sun- and cloud-symbols indicate lighting conditions.*

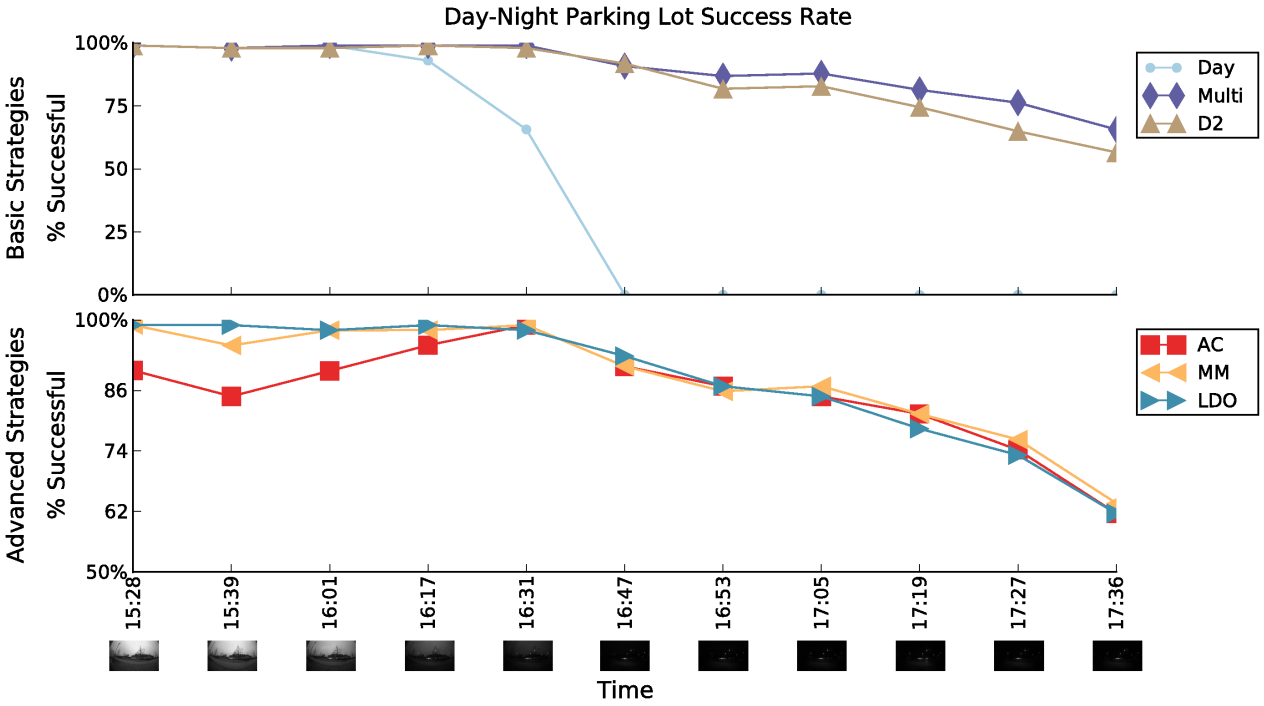


Figure 10: *Availability for the Day-Night dataset, calculated as the number of successful localizations divided by the total of localization attempts (see 4.3.1). Split according between basic and advanced summarization strategies. A downscaled image from the front-camera is shown at the bottom, to indicate overall scene illumination.*

5.1 Availability

The results for the Year-Long Parking Lot and Suburban experiments in figures 9a and 9b respectively. In each figure, the summary strategies (c.f. Table 1) are presented in two groups: the *basic strategies* (full, single, Latest and D2) on top, the *advanced strategies* (LDO, MM and AC) below. These figures relate each test recording to the localization success rate (c.f. Section 4.3.1). Recordings that were incorporated into the map, after testing, are marked an arrow and a grey background.

Below the recording timeline, a rough overview of the lighting conditions is given for each dataset. Appendix A seperately describes the weather conditions, azimuth and altitude of the sun. We found the nature of outdoor scene lighting difficult to classify according to a single discrete label. For Figure 9, we have chosen three categories for describing the illumination. Strong intensity (sun symbol) was chosen when drop-shadows, under a bright sun, were clearly visible in the images. Medium intensity (sun-and-cloud symbol) indicates that a single light source was clearly illuminating the scenes; shadows are not always visible. Diffuse lighting (cloud symbol) means that a single light source was seemingly absent and no shadows could be seen.

As evident from the figures, by incorporating new information throughout, we can build a map that is consistently useable for localization over a long period. In the parking lot, Figure 9a, the success rate for the single-session map quickly drops, in line with results reported for other localization systems (c.f. Furgale and Barfoot (2010), Muehlfellner et al. (2013)). The full multi-session map, which experiences the same drop, but recovers as new data is added to it. The Summary Maps generally perform well, but experience some problems during some periods (see below).

Availability of the single-session map for the suburban dataset, Figure 9b, is more erratic – fluctuating between high and low success rates. We believe this shows that the mode of visual change is different for this area. While there is still seasonal variation in the vegetation, and other minor long-term structural

alterations, there is also a lot of permanent structure visible throughout. Additionally, there are faster changes related to lighting and weather. The initial map was recorded during overcast weather, with diffuse lighting. Figure 9b indicates that there is a trend for this map to fail in when lighting is strong.

In contrast, the full multi-session map, and the maps derived from it, provide stable performance for the whole range of conditions. This consistently good performance makes the multi-session map preferrable also for this dataset.

Summary strategies profit from the ability to keep information around, even if it is not used within the most recent iterations of the map. This is evident for the parking lot dataset, Figure 9a, where the progressive change of seasons is influential. There are noticeable drops in the performance of all strategies, on April 25$^{\text{th}}$ of and on May 5$^{\text{th}}$ during spring 2014. These recordings correspond to a transitional period in the foliage, as exemplified by the snapshot from April 25$^{\text{th}}$ shown in the fourth row of Figure 6a.

The Latest map twice completely fails to localize during this period, as it does not retain information beyond that recorded during winter. Similarly, the MM map suffers from a breakdown; the long-term memory mechanics prohibit it from keeping Landmarks that were not seen for N_{LTM} (4) sessions, which only record information from November to January. The AC map, which tries to preserve appearance diversity but prefers newer clusters nonetheless, also seems more affected than the other strategies. This difference in failure rate indicates that some of the landmarks retained from the previous summer and autumn in the other maps, are actually useful for localization at this point in time.

This is further corroborated by the performance of the various strategies for the last recordings, from November and December 2014. At this point, information was injected into the map on July 2$^{\text{nd}}$ for the last time. The information avaiable to the *time-dependent* strategies—Latest, MM and AC—is dominated by Spring/Summer. They fail during the next transition to autumn.

There is also a peculiar drop in the performance of the Latest map, and all maps from the advanced category, on October 17$^{\text{th}}$. For this recording, the sun is hanging very low in the sky, casting long shadows. While the illumination is not as strong as on some of the other days, this configuration is atypical. The reduced models for these strategies do not seem to cope well with these conditions.

The different Summary strategies perform on par with each other for the suburban dataset. The performance of the Latest strategy is surprisingly good, except for one fluctuation during the—very sunny—recording made on July 23$^{\text{rd}}$. The prevalence of permanent structure seems to mitigate the effect of transitory periods.

The availability of localization for the Day-Night dataset is shown in Figure 10. All test recordings were taken on the same day, November 18$^{\text{th}}$; the recording time is given on the horizontal axis. Furthermore, a strongly downsampled snapshot from each dataset is shown below the time. This is intended to give a rough idea of the overall scene illumination for the dataset.

As the sun sets, between 16:17 and 16:47, the performance of the single-session map drops sharply. Considering the appearance change depicted in Figure 6b, this is not surprising. In contrast, the full multi-session map—which incorporates recordings from day and night—performs well throughout the whole timeframe. The artificial lights visible at night provide enough features to enable this. Transitioning from day to night is helped by the beacon-like nature of the street lamps.

The performance of the Summary Maps mirrors that of the full multi-session map. The simple D2 scheme performs proportional to the full map throughout. Similarly, the LDO and MM schemes perform well both during day and night. The success rate is generally lower at night, but so is the number of visible features throughout the recordings, while the success criteria remain the same. Section 5.2 shows that we are still able to obtain reasonable metric accuracy during the night.

The performance of the AC map is slightly worse during the day. This strategy prefers newer information

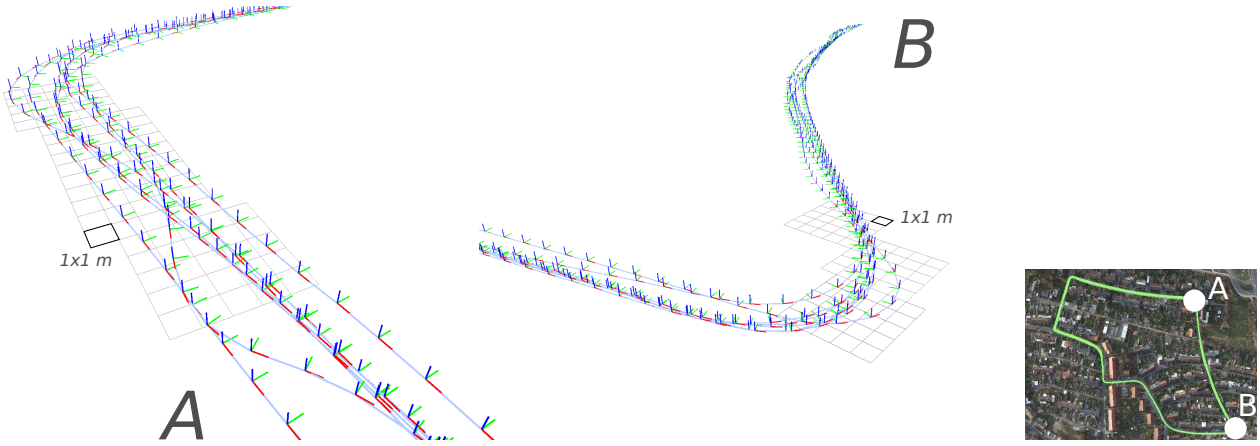


Figure 11: *Metric 6-DOF pose estimates, illustrated as oriented 3D axis-triples in a common reference frame per location. The results for several test runs from the Suburban dataset are shown; the example two locations are identified in the lower right corner. For scale, square grid cells with one meter side length are depicted. Viewpoint variations, such as the swerves in A which were necessary in order to avoid street-parked cars, are handled naturally by the proposed metric pipeline. These results are based on the map created using the LDO strategy.*

due to the employed LRU deletion scheme—since the night recordings were later than the day recordings, this property causes it to discard information from the day. Nonetheless, the performance is reasonable throughout.

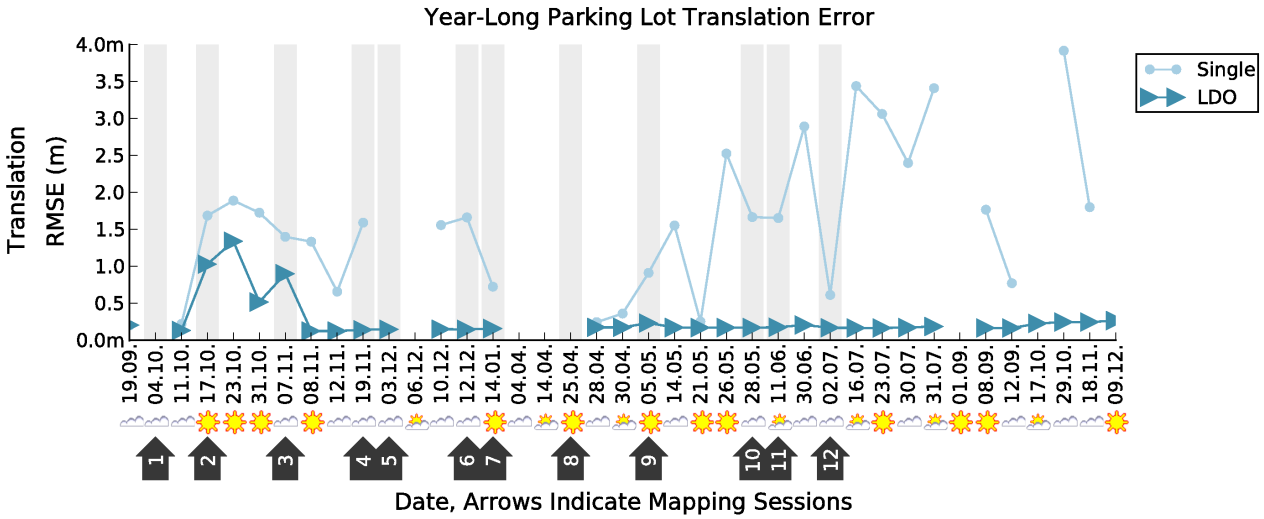
The evaluation of the Day-Night dataset shows us that the proposed mapping procedure can cope with extreme scene changes, in addition to the structural and seasonal variations captured by the long-term datasets. While the performance of the different summary strategies varies, we can nonetheless identify two candidates—LDO and AC—that provide good performance throughout. Strategies that do not preserve visual information in the long term, either by simply selecting the most recent samples (Latest) or by putting a hard limit on the long-term memory (MM) cope less well with the slow variation of seasonal transitions. Next, we evaluate the metric localization error, in order to determine whether localization is accurate and useful when it is available.

5.2 Metric Accuracy

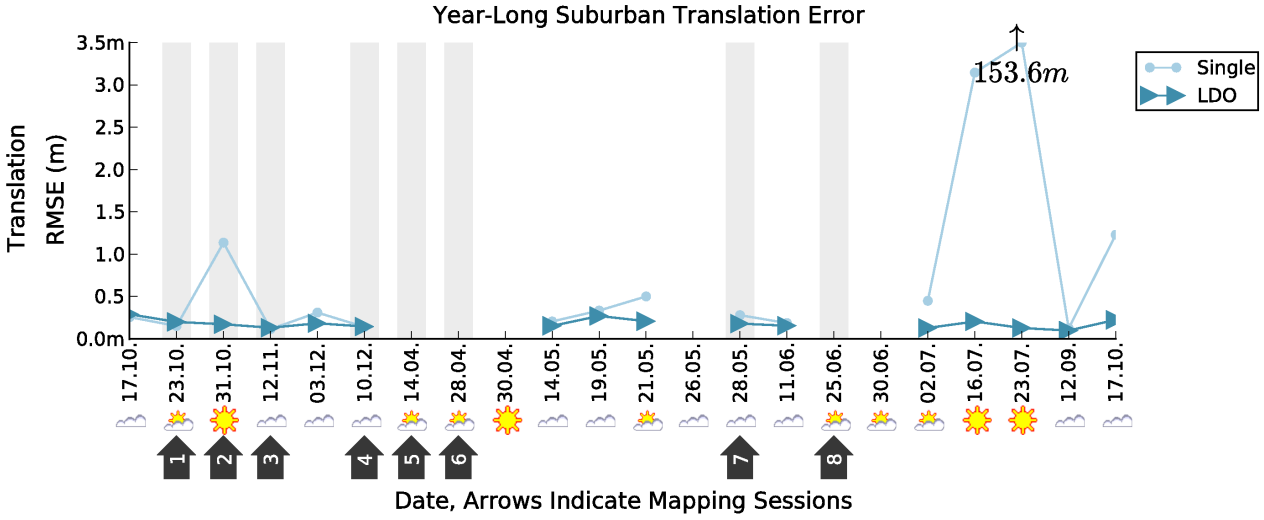
Providing locally accurate metric localization is one of the major goals of the proposed pipeline. Figure 11 illustrates the 6-DOF pose output produced, using the LDO map, at two example locations of the Suburban dataset. The Figure also shows the viewpoint variations in the dataset that result from the need to avoid obstacles.

We evalaute the accuracy (see Section 4.3.2) based on the INS reference system. This error is calculated for both successful and failed localizations. If there is a localization failure, the prediction from the odometry system is in place of the estimate by the visual localizer. For automated driving, this is the full system error that we are interested in.

Detailed values for the translational RMSE and the absolute angular deviation, for all datasets and Summary strategies, are given in Appendix B. If estimates are available—the success rate shown in figures 9 and 10 is high—the behaviour of the metric error for the various Summary maps is very similar. Given the INS employed as a reference sensor, these differences in error are too small to be meaningfully compared. This is to be expected, as the Summary Maps are based off the same, optimized, multi-session geometry. Therefore, we will simply compare the error of the single-session map to the error of the LDO strategy in the following,



(a) Translation Error for the Year-Long Parking Lot dataset.



(b) Translation Error for the Year-Long Suburban dataset.

Figure 12: *Evolution of the RMS translational error for the Year-Long datasets. Arrows and grey bars indicate mapping datasets. Sun- and cloud-symbols indicate lighting conditions. Error values are only shown if the reference IMU/DGPS system was available, and if there was at least one successful localization for the recording. On July 23rd, the translational error was exceedlingly high for the single-session map and the Suburban dataset; this is truncated in the figure.*

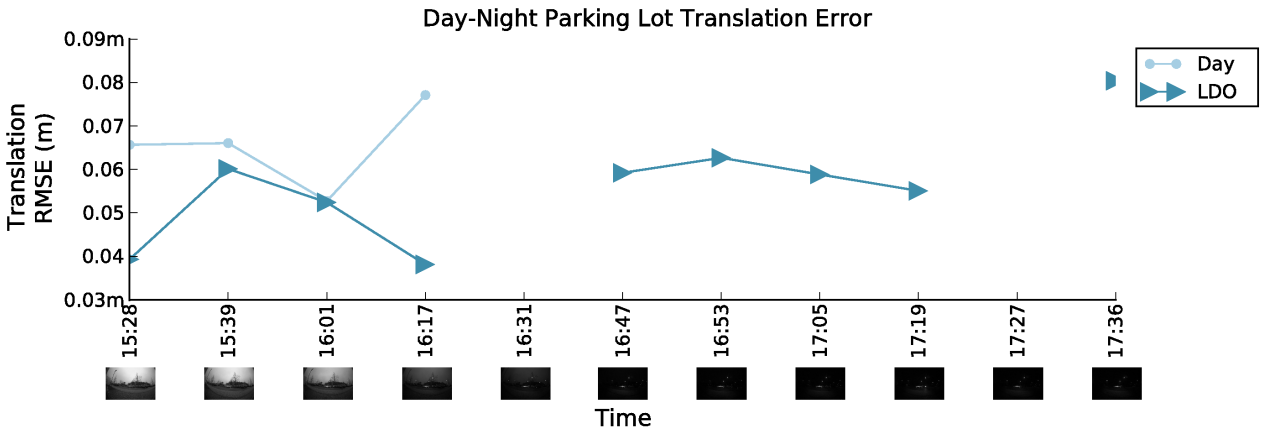


Figure 13: *Evolution of the RMS translational error for all datasets. To demonstrate illumination conditions, a downscaled image is show below each test-time. Note that the DGPS system suffered from intermittent problems at 16:31 and 17:27 – the final error for the LDO map at 17:36 should not be overlooked. Errors are also only shown if there was at least one successful localization.*

since this summary strategy generally provides high availability.

Generally, we have observed non-vanishing mean values in the Y and Z components of the error—which can not be seen from the depicted RMSE values directly. These point towards the difficulties in the long term-operation of the combined visual and INS-system. It is possible for both the quality of the INS calibration relative to the camera system to degrade, as well as for biases to be introduced by the nature of the DGPS system that is employed within the INS. Similar difficulties using DGPS for ground-truth assessment have been reported in other papers (Churchill and Newman, 2013, McManus et al., 2013). We have omitted several recordings from the evaluation, where clear biases, or missing samples were observed for the INS.

Figure 12a shows how the translation error develops for the Year-Long Parking Lot dataset. During late October and early November 2014 the error is high for both maps. This happens at the same time as the initial drop in availability depicted in Figure 9a. For the remainder of the observed time span, the LDO map consistently provides small errors, while the performance of the single-session map degrades. These low error values show that the map is internally consistent, and that introducing large amounts of new data through the course of a year does not degrade performance.

The translation error for the Year-Long Suburban dataset is shown in Figure 12b. Again, the multi-session LDO map provides low errors, while the single-session map is subject to high errors and complete failures. For this dataset, an especially large number of recordings had to be excluded from the metric error evaluation, due to to obvious DGPS biases.

Similarly, the metric error for the LDO map for the day-night dataset is very low. Once more, these results indicate general metric consistency of the map, even under very different illumination for the individual sessions. The error of the day single-session map is low when localization is available; during the night, no estimates could be provided.

To demonstrate how the system typically behaves during a single localization run, Figure 14 compares the translational error for the single-session and the LDO map for an example recording from the Year-Long Parking Lot dataset, from November 12$^{\text{th}}$ 2013. In this plot, the X-, Y- and Z-errors are summarized by the norm, $\|t\|$, of the translation error vector. For the single-session map it can clearly be seen how the estimate drifts, when localization fails for longer amounts of time; the estimate jumps back to a better value as soon as localization succeeds again. When localization succeeds, the error is small for both maps.

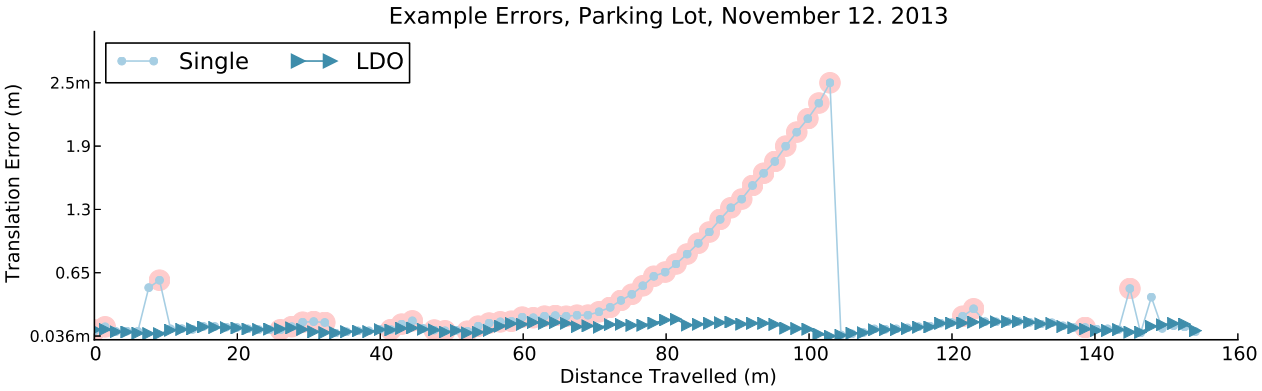


Figure 14: *Translational error, summarized as the norm, $\|t\|$, of the X-, Y-, Z-error vector over distance traveled during a single dataset. Localization failures (see* Availability*, Section 4.3.1) are marked with a faint-red background. This demonstrates how estimates drift when localization fails.*

By investigating the metric error, we have confirmed that the proposed map provides useful, accurate 6-DOF pose estimates. This is the case when many session are incorporated over a very long timeframe, and even under extreme changes in lighting. Next, we have to investigate the computational cost associated with the various Summary Maps.

5.3 Computational Cost

Figure 15 shows how the number of landmarks evolves, for the maps created based on the Year-Long Parking Lot dataset. The number of landmarks in a map dominates the storage cost of maps.

For both the full multi-session map and the Fixed-Threshold maps, the number of landmarks grows without bounds in an approximately linear fashion. Trivially, the number of landmarks stays the same or oscillates around a single value for the single-session map and the maps created with the Latest strategy.

For the advanced strategies – LDO, MM and AC – we have tried to choose parameters such that they reach comparable sizes. All advanced maps have mechanisms built in that lead to more-or-less constant map sizes. The MM strategy is limited by the size of the long-term memory, whereas the number of preserved clusters is the deciding parameter for the AC strategy.

Maps created with the LDO strategy also converge to a fixed size after some time. The drop in the number of landmarks for the LDO maps can be explained as an effect of the quantization of the ranking function. When the proposed integer-valued scoring functions are used, multiple landmarks can occupy the same rank. We implemented the sampling policy in a conservative fashion; if a landmark for a given rank is admitted, all other landmarks of the same rank are also included in the map. Since many landmarks occupy a limited number of ranks during the first sessions, this implementation leads to a significant overshooting in the number of desired landmarks. As more observations are made, the ranking for the landmarks becomes more fine grained, allowing the local-ranking maps to converge in size.

In general, we have no requirement for operating the proposed system at the framerate provided by our camera system. When the pipeline is used for automated navigation, localization runs in the background, providing the newest pose estimate to other software components whenever it becomes available. To output estimates at the high rate required by local path planning and control, wheel-tick odometry is used to extrapolate from the most recent estimate. However, for the accumulated error due to odometry integration to stay bounded, we require regular updates from the localization pipeline.

The framerate—the number of frames localized per second—of our localization pipeline for the various map

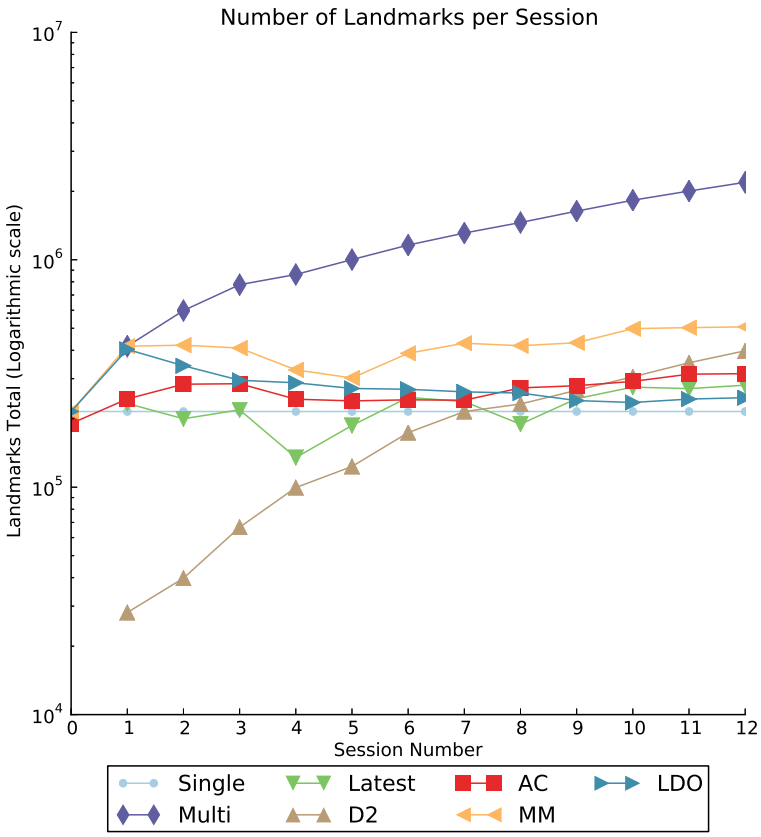


Figure 15: *The landmark total on a logarithmic scale, per session added to the map. Behaviour roughly falls into two groups: maps with a bounded number of landmarks, and maps with unbounded size that grow linearly. The special convergence behaviour of the Local-Ranking maps can be explained as a quantization effect.*

types and sessions is shown in Figure 16. These framerates were recorded during offline operation, on the computer described in section 4. Scaling, feature-extraction, and other image processing operations are not reflected in this figure. They are carried out in a separate processing step, and remain the same over all map types.

The behaviour of the framerate is roughly inversely proportional to the number of landmarks in the map. This indicates that the available number of landmarks dominates the complexity of the involved algorithms. The LDO map outperforms the other Summary strategies.

5.4 Discussion

In the above experiments, we have looked at the availability and metric accuracy of localization in several scenarios, as well as at the computational cost associated with different summary strategies. Based on the results of these experiments, we identify the following main findings:

It is possible to build maps that work for localization across a wide range of conditions (through seasonal or diurnal cycles) by merging data from multiple sessions. The multi-session maps we have built incorporate datasets that are recorded under disparate conditions. Likewise, we have performed tests of the resulting maps on a similarly wide range of recordings. The results of these experiments not only show us that localization is possible under these very different conditions (Section 5.1) but that we can maintain low metric errors throughout (Section 5.2).

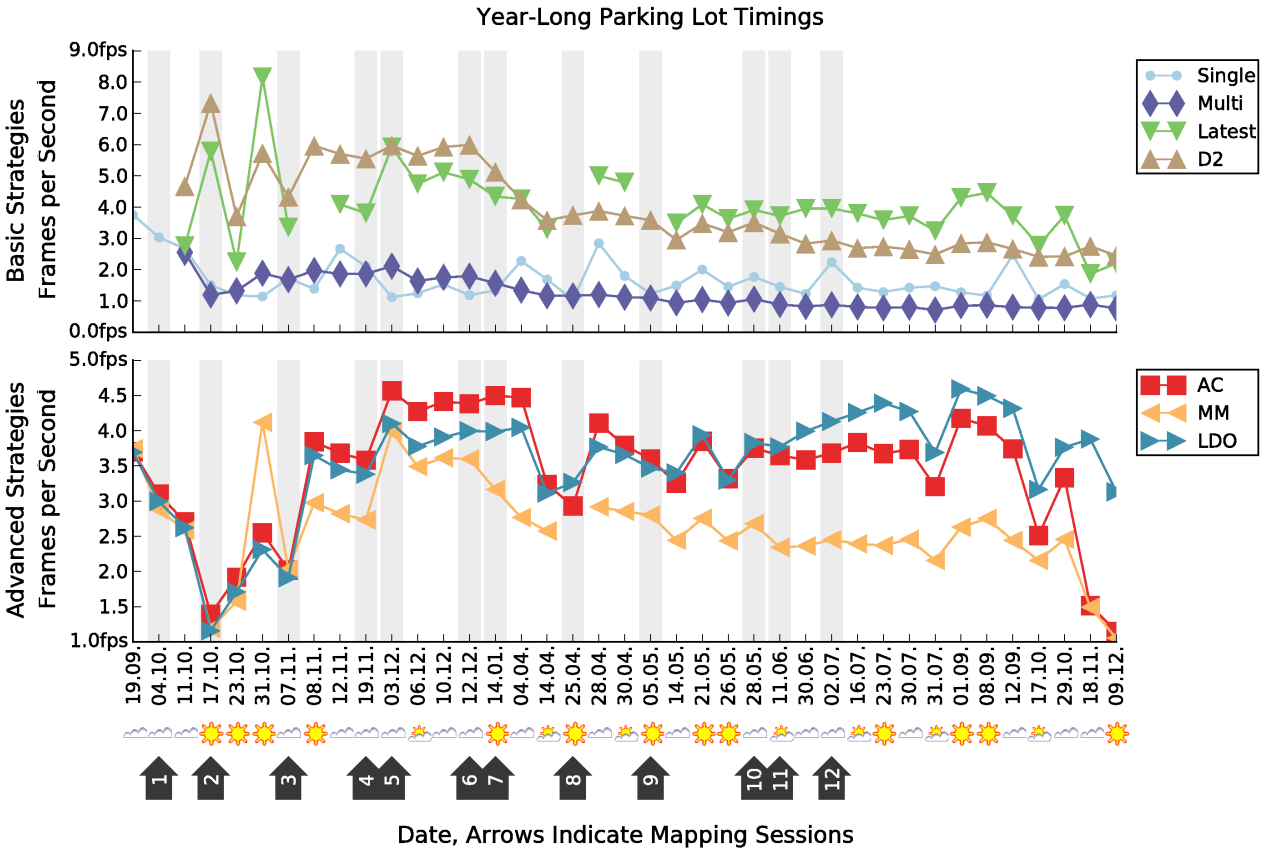


Figure 16: *The framerate of the full localization pipeline without feature extraction, for the various map types, over all datasets. Grey vertical lines indicate when new data was added to the map. The framerate of the unbounded maps steadily drops as new sessions are added, while the advanced strategies provide approximately constant framerates. Generally, the behaviour mirrors the growth in map size shown in Figure 15.*

We summarized the full map, reducing the number of landmarks contained in the resulting maps by a factor of, roughly, ten (Figure 15). Both the AC and LDO strategies, as well as the basic D2 strategy are still available throughout wide ranges of conditions. For these summary strategies, localization remains both accurate and highly available, even under a massive reduction in map size.

Highly available, constant size, maps can be built. Referring to both the localization availability of the various maps (Figure 9) and the map size (Figure 15), we can identify several distinct behaviours. Maps that are not limited in size—the full and the Fixed-Threshold D2 maps—have high success rates after accumulating several sessions. In comparison, the maps based on the Latest strategy, which have approximately constant size, intermittently suffer from complete failures. These failures also affect the MM map, which has a similar hard limit on the age of landmarks.

The LDO and AC maps are of approximately fixed size. They generally perform slightly worse than the unlimited maps, but still provide reasonable performance at a massive reduction of storage cost. This makes them preferrable to the other Summary strategies.

The results with maps of constant size also indicate that a certain number of landmarks can capture the visual variability in an environment (similar to the experience-saturation in Churchill and Newman (2013)). This even seems possible in highly dynamic car-parks and under day-night lighting changes. Fundamental changes, such as passing seasons, cause performance drops, but can be handled gracefully using the right Summary strategy.

Using all available information on a per-landmark basis proves beneficial. At times of significant change in environment appearance, during spring and autumn for the Year-Long Parking Lot dataset, the LDO strategy outpforms AC.

The AC strategy tries to select a diverse set of vehicle frames to reflect the visual appearance of each local neighbourhood, preferring newer information. In contrast, LDO has no special preference for data based on its timestamp, and does not enforce grouping landmarks into keyframes. Rather, two simple statistics – the number of datasets and the total number observations per landmark – are collected over all sessions and used to select the best individual landmarks. In this way, LDO provides a better landmark-selection for long-term localization, at the same map size.

6 Conclusion

We have described, implemented, and evaluated a full system to solve the problem of long-term visual localization. At the core of this system is a pipeline that alternates online localization and offline map-fusion to maintain a *multi-session map*. Information from different sessions is registered in common, metric, local coordinate frames. This enables the estimation of an unambiguous vehicle pose over a wide range of lighting conditions, seasonal change, and weather by mixing landmarks from multiple sessions.

While all data is aggregated for offline batch processing of the map, we propose producing a *summarized* version of it for online localization that contains only the most pertinent landmarks. For Summary Maps, a framework of scoring functions and sampling policies is defined. Scoring functions leverage the cross-session landmark data-associations, to predict the usefulness of landmarks for localization. Sampling policies select landmarks based on the scores assigned to them, possibly incorporating the local geometric properties of the map.

The proposed system was evaluated on an extensive collection of datasets, recorded over the span of one and a half years at two different locations. The datasets depict seasonal change and various weather conditions, as well as the transition for day to nighttime. We built multi-session maps for these different scenarios, with the goal of covering all environment conditions. From our evaluation, we saw that the multi-session map provided consistently good performance, with regards to availability for localization and metric error.

We explored the tradeoff between the computational cost, regarding both map size and framerate, and availability/error. While the full multi-session map provides the best performance at a very high cost, a compromise can be reached based on the proposed summarization strategies. We have compared several summary strategies, including appearance-based clustering (AC) based on the work Konolige and Bowman (2009), the memory model (MM) of Dayoub et al. (2011) and a local ranking strategy (LDO) of our own design. Both AC and LDO are able to provide maps that are consistently good for localization, while maintaining a constant map size. LDO outperforms the other strategies in the Year-Long Parking Lot environment, which is subject to strong seasonal changes.

We believe that the long-term mapping life cycle depicted in Figure 1 is an approach that can scale in all the ways necessary for lifelong operation of fleets of mobile robots in vast environments. This paper represents our first implementation of this concept and the excellent performance suggests that it is a clear way forward for future research.

As a first step, we would like to explore different strategies for producing the Summary Map. The strategies that we propose for map summary are only the very basics of what is possible with this approach. One could create new sampling strategies that ensure strong landmarks are well separated in image- or descriptor-space, or choose configurations of landmarks that are likely to produce the most accurate localization estimate. As more data is gathered, further offline processing could lead to place-specific data generated for the summary maps. For example, one could perform further processing of the image features, such as the quadratic

discriminant analysis used in Bosse and Zlot (2013), to create place-specific transformations of the feature space that produce the best localization results. One could even produce place-specific features such as the Scene Signatures proposed by McManus et al. (2014b). The take-away message is that processing the map data offline opens up the use of computationally expensive techniques to produce the summary map that gives the best localization result for the lowest computational cost.

Another focus will be to address the scalability of this concept with respect to a number of different aspects.

If we would like to scale this approach to larger areas, longer time scales, and more robots, we will have to deal efficiently with torrents of incoming data. Although we would like to produce maps using all data over all time, this is clearly impossible for any practical case. First, we must develop cluster-capable optimization algorithms (Agarwal et al., 2010, Ni et al., 2007, Snavely et al., 2008) and utilize windowed or relative map representations (Sibley et al., 2009). Second, we must also have the possibility to produce summary maps in the offline system. Inevitably, if the system is built to run continuously, some marginalization of raw data and sparsification of the pose graph will be necessary. Finding the best methods of summarizing data is an interesting avenue of research (Carlevaris-Bianco and Eustice, 2013, Johannsson et al., 2012, Mazuran et al., 2014). Furthermore, communication constraints may enforce that only a subset of data will make it back to the server for processing. Because of this, we would like to explore strategies for the online selection of data that is new or surprising.

Finally, we would like to implement this system on a team of field robots that is deployed continuously. This will involve significant work automating every step of the pipeline but it is clearly an important next step towards the goal of lifelong localization for mobile robots deployed in real-world environments.

Acknowledgement

This work is supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge) and grant #610603 (EUROPA2).

A Appendix: Data

Tables 2 to 4 give a detailed overview of the datasets that were used for the long-term mapping and localization experiments.

The *INS Status* column indicates whether the INS system was providing useable data for this run. Recordingss for which the INS provided a biased or inaccurate output, or for which there was no INS available on the vehicle, are marked with 'Defunct'. These recording where excluded from the metric error evaluation. The columns *Illumination*, *Foliage*, *Clouds* were filled in manually, to classify the state of the environment. The criteria for filling in the *Illumination* are given at the start of Section 5.1. *Solar Altitude* and *Azimuth* were determined using the *PySolar* library[2]. The final column indicates whether this recording was used for mapping, showing the number of the corresponding mapping session.

B Appendix: Local Metric Errors

The local metric errors for all test-datasets and sampling strategies are given in tables 5 - 7. These tables give the Root Mean Squared (RMS) Error for the translation, as well as the average magnitude of the overall angular error. Local metric errors are calculated as described in section 4.3.2. For a discussion of these values, see section 5.2.

[2]Freely Available Online

Table 2: *Details for the Year-Long Parking Lot dataset.*

#	*Date, Time*	*Duration (min)*	*INS Status*	*Illumination*	*Clouds*	*Solar Azimuth (deg)*	*Solar Altitude (deg)*	*Foliage*	*Mapping Session*
0	2013-08-20 16:29	01:39	Nominal	Diffuse	Overcast	284.9	26.47	Full	0 (Initial)
1	2013-09-19, 10:09	02:28	Defunct	Diffuse	Overcast	37.1	33.02	Full	
2	2013-10-04, 14:39	02:10	Nominal	Diffuse	Overcast	317.2	24.47	Full	
3	2013-10-11, 15:15	01:53	Nominal	Diffuse	Overcast	309.4	17.87	Transitional	1
4	2013-10-17, 15:24	01:47	Nominal	Strong	Scattered	308.1	14.69	Transitional	
5	2013-10-23, 14:08	01:54	Nominal	Strong	Scattered	326.5	20.43	Transitional	2
6	2013-10-31, 13:55	02:00	Nominal	Strong	Few	330.4	18.86	Transitional	
7	2013-11-07, 14:09	02:09	Nominal	Diffuse	Overcast	327.8	15.75	Bare	
8	2013-11-08, 10:25	02:11	Nominal	Strong	Few	23.86	17.97	Bare	3
9	2013-11-12, 15:17	02:57	Nominal	Diffuse	Broken	313.3	7.951	Bare	
10	2013-11-19, 12:21	02:24	Nominal	Diffuse	Overcast	355.3	17.95	Bare	
11	2013-12-03, 15:37	02:31	Defunct	Diffuse	Few	312.6	2.761	Bare	4
12	2013-12-06, 13:49	01:21	Nominal	Medium	Broken	336.1	11.88	Bare	5
13	2013-12-10, 10:54	02:23	Nominal	Diffuse	Overcast	17.73	12.94	Bare	
14	2013-12-12, 10:31	02:40	Nominal	Diffuse	Fog	23.17	11.51	Bare	
15	2014-01-14, 12:43	02:25	Defunct	Strong	Clear	355.8	16.27	Bare	6
16	2014-04-04, 15:03	02:37	Defunct	Diffuse	Overcast	309.2	32.57	Bare	7
17	2014-04-14, 10:49	02:28	Defunct	Medium	Broken	30.43	43.52	Bare	
18	2014-04-25, 11:27	01:13	Nominal	Strong	Clear	18.0	49.69	Transitional	
19	2014-04-28, 10:07	02:41	Nominal	Diffuse	Broken	45.3	44.21	Transitional	8
20	2014-04-30, 09:13	02:18	Nominal	Medium	Scattered	60.63	38.23	Transitional	
21	2014-05-05, 14:56	02:08	Nominal	Strong	Clear	302.9	41.8	Transitional	
22	2014-05-14, 10:18	02:34	Nominal	Diffuse	Broken	44.38	49.52	Full	9
23	2014-05-21, 17:07	02:04	Nominal	Strong	Clear	269.9	25.87	Full	
24	2014-05-26, 13:49	02:08	Nominal	Strong	Scattered	320.2	53.68	Full	
25	2014-05-28, 16:37	02:29	Nominal	Diffuse	Overcast	275.3	31.55	Full	
26	2014-06-11, 15:23	02:03	Nominal	Medium	Scattered	291.3	44.12	Full	10
27	2014-06-30, 18:10	02:10	Nominal	Diffuse	Overcast	257.2	19.64	Full	11
28	2014-07-02, 08:44	02:08	Nominal	Diffuse	Broken	75.78	39.74	Full	
29	2014-07-16, 18:46	01:27	Nominal	Medium	Few	252.3	13.39	Full	12
30	2014-07-23, 09:05	01:43	Nominal	Strong	Few	68.85	40.18	Full	
31	2014-07-30, 09:07	02:03	Nominal	Diffuse	Overcast	67.33	39.1	Full	
32	2014-07-31, 14:45	02:07	Defunct	Medium	Few	307.4	45.98	Full	
33	2014-09-01, 18:29	02:04	Nominal	Strong	Few	262.6	4.681	Full	
34	2014-09-08, 18:06	02:03	Nominal	Strong	Few	268.2	5.76	Full	
35	2014-09-12, 17:50	02:07	Nominal	Diffuse	Overcast	272.1	6.801	Full	
36	2014-10-17, 17:08	01:16	Nominal	Medium	Scattered	286.2	0.9232	Transitional	
37	2014-10-29, 10:21	01:26	Nominal	Diffuse	Overcast	25.81	20.76	Transitional	
38	2014-11-18, 14:19	02:01	Nominal	Diffuse	Overcast	326.8	12.36	Bare	
39	2014-12-09, 11:24	01:48	Nominal	Strong	Few	10.63	14.18	Bare	

Table 3: *Details for the Year-Long Suburban dataset.*

#	*Date, Time*	*Duration (min)*	*INS Status*	*Illumination*	*Clouds*	*Solar Azimuth (deg)*	*Solar Altitude (deg)*	*Foliage*	*Mapping Session*
0	2013-10-11, 15:20	29:28	Nominal	Diffuse	Overcast	308.1	17.23	Transitional	0 (Initial)
1	2013-10-17, 16:18	17:58	Nominal	Diffuse	Scattered	296.4	7.703	Transitional	
2	2013-10-23, 14:14	18:57	Nominal	Medium	Scattered	325.0	19.91	Transitional	
3	2013-10-31, 13:09	18:32	Nominal	Strong	Few	342.0	21.73	Transitional	1
4	2013-11-12 15:23	19:26	Nominal	Diffuse	Broken	312.2	7.345	Bare	2
5	2013-12-03 15:13	18:25	Nominal	Diffuse	Scattered	317.5	5.239	Bare	3
6	2013-12-10 09:59	17:34	Defunct	Diffuse	Overcast	30.08	9.586	Bare	
7	2014-04-14 09:04	11:47	Defunct	Medium	Broken	60.14	32.11	Transitional	4
8	2014-04-28 11:41	19:01	Defunct	Medium	Scattered	12.87	51.22	Transitional	5
9	2014-04-30 08:02	16:59	Nominal	Strong	Few	77.23	28.17	Transitional	6
10	2014-05-14, 09:34	11:17	Nominal	Diffuse	Broken	57.77	44.23	Full	
11	2014-05-19, 13:48	11:39	Nominal	Diffuse	Scattered	321.5	52.57	Full	
12	2014-05-21, 17:29	11:54	Defunct	Medium	Clear	265.8	22.65	Full	
13	2014-05-26, 13:55	11:54	Nominal	Diffuse	Scattered	318.2	53.11	Full	
14	2014-05-28, 16:45	11:06	Nominal	Diffuse	Overcast	273.7	30.41	Full	
15	2014-06-11, 15:36	10:20	Defunct	Diffuse	Scattered	287.9	42.15	Full	7
16	2014-06-25, 09:07	08:53	Defunct	Medium	Overcast	70.14	43.64	Full	
17	2014-06-30, 18:16	09:43	Nominal	Medium	Overcast	256.2	18.84	Full	8
18	2014-07-02, 08:50	09:25	Nominal	Medium	Broken	74.36	40.64	Full	
19	2014-07-16, 18:21	08:41	Nominal	Strong	Few	257.0	17.05	Full	
20	2014-07-23, 09:11	08:32	Nominal	Strong	Few	67.47	40.97	Full	
21	2014-09-12, 18:05	09:13	Nominal	Diffuse	Overcast	269.1	4.506	Full	
22	2014-10-17, 17:17	04:08	Nominal	Diffuse	Overcast	284.5	-0.171	Transitional	

Table 4: *Details for the Day-Night dataset.*

#	*Date, Time*	*Duration (min)*	*INS Status*	*Illumination*	*Clouds*	*Solar Azimuth (deg)*	*Solar Altitude (deg)*	*Foliage*	*Mapping Session*
0	2014-11-18, 14:19	02:01	Nominal	Day	Overcast	326.8	12.36	Bare	0 (Initial)
1	2014-11-18, 15:15	01:51	Nominal	Day	Overcast	314.7	7.039	Bare	
2	2014-11-18, 15:28	02:10	Nominal	Day	Overcast	311.9	5.594	Bare	
3	2014-11-18, 15:39	02:15	Nominal	Day	Overcast	309.7	4.375	Bare	
4	2014-11-18, 15:53	02:02	Nominal	Day	Overcast	306.9	2.769	Bare	
5	2014-11-18, 16:01	02:07	Nominal	Day	Overcast	305.3	1.873	Bare	1
6	2014-11-18, 16:12	02:00	Nominal	Dusk	Overcast	303.1	0.595	Bare	
7	2014-11-18, 16:17	02:02	Defunct	Dusk	Overcast	302.0	0.02923	Bare	2
8	2014-11-18, 16:24	02:03	Nominal	Dusk	Overcast	300.7	-0.725	Bare	
9	2014-11-18, 16:31	01:59	Nominal	Dusk	Overcast	299.2	-1.711	Bare	3
10	2014-11-18, 16:37	02:20	Nominal	Dusk	Overcast	298.1	-2.745	Bare	
11	2014-11-18, 16:42	02:09	Nominal	Night	Overcast	297.1	-3.621	Bare	4
12	2014-11-18, 16:47	02:06	Nominal	Night	Overcast	296.1	-4.474	Bare	5
13	2014-11-18, 16:53	02:06	Nominal	Night	Overcast	295.2	-5.25	Bare	
14	2014-11-18, 16:59	02:04	Nominal	Night	Overcast	294.0	-6.133	Bare	
15	2014-11-18, 17:05	02:03	Nominal	Night	Overcast	292.9	-6.987	Bare	6
16	2014-11-18, 17:11	02:16	Nominal	Night	Overcast	291.7	-7.883	Bare	
17	2014-11-18, 17:19	02:22	Defunct	Night	Overcast	290.2	-8.983	Bare	7
18	2014-11-18, 17:27	02:08	Nominal	Night	Overcast	288.6	-10.2	Bare	
19	2014-11-18, 17:36	02:06	Nominal	Night	Overcast	286.9	-11.46	Bare	

Table 5: *Year-Long Parking Lot Local Metric Error*

	First RMS (m)	*First* $\lvert\phi_e\rvert$ (deg)	*Multi RMS* (m)	*Multi* $\lvert\phi_e\rvert$ (deg)	*Latest RMS* (m)	*Latest* $\lvert\phi_e\rvert$ (deg)	*D2 RMS* (m)	*D2* $\lvert\phi_e\rvert$ (deg)	*AC RMS* (m)	*AC* $\lvert\phi_e\rvert$ (deg)	*MM RMS* (m)	*MM* $\lvert\phi_e\rvert$ (deg)	*LDO RMS* (m)	*LDO* $\lvert\phi_e\rvert$ (deg)
19.09.	0.20	0.85							0.22	1.00	0.20	0.86	0.20	0.90
04.10.														
11.10.	0.22	1.16	0.13	1.16	0.13	1.17	1.47	2.45	0.17	1.14	0.13	1.16	0.13	1.17
17.10.	1.69	3.29	1.05	2.10	1.25	2.65	1.44	2.84	1.40	2.71	1.03	2.07	1.03	2.08
23.10.	1.89	3.83	0.71	1.64	1.37	2.23	1.75	2.84	1.10	2.01	1.09	2.06	1.34	2.22
31.10.	1.72	3.67	0.32	1.26	0.25	1.09	1.42	2.33	0.34	1.26	0.34	1.26	0.52	1.44
07.11.	1.40	2.97	0.23	1.18	0.38	1.24	0.95	1.66	0.98	1.72	0.23	1.20	0.90	1.69
08.11.	1.33	2.89	0.12	1.22			0.16	1.09	0.12	1.21	0.12	1.20	0.12	1.20
12.11.	0.66	1.76	0.12	1.13	0.13	1.13	0.22	1.09	0.12	1.12	0.12	1.12	0.12	1.13
19.11.	1.59	2.74	0.14	1.01	0.14	1.00	0.16	0.92	0.14	1.00	0.14	1.00	0.14	1.01
03.12.			0.15	1.21	0.15	1.21	0.15	1.21	0.15	1.22	0.15	1.21	0.15	1.22
06.12.														
10.12.	1.56	3.13	0.15	1.23	0.15	1.24	0.15	1.22	0.15	1.22	0.15	1.23	0.15	1.23
12.12.	1.66	3.56	0.15	1.21	0.15	1.20	0.15	1.19	0.15	1.19	0.15	1.20	0.15	1.20
14.01.	0.72	2.71	0.16	1.20	0.16	1.19	0.16	1.19	0.16	1.19	0.16	1.18	0.16	1.19
04.04.														
14.04.														
25.04.														
28.04.	0.24	1.48	0.17	1.40	0.17	1.40	0.18	1.39	0.17	1.39	0.17	1.39	0.17	1.41
30.04.	0.36	1.72	0.18	1.31	0.17	1.30	0.18	1.30	0.17	1.29	0.17	1.30	0.17	1.29
05.05.	0.91	2.57	0.24	1.45			0.23	1.43	0.24	1.41	0.24	1.42	0.23	1.42
14.05.	1.55	3.40	0.17	1.44	0.18	1.43	0.17	1.39	0.17	1.37	0.17	1.39	0.17	1.40
21.05.	0.25	1.75	0.17	1.44	0.17	1.44	0.17	1.42	0.17	1.44	0.17	1.43	0.17	1.44
26.05.	2.53	7.46	0.17	1.41	0.17	1.42	0.17	1.40	0.17	1.38	0.17	1.42	0.17	1.40
28.05.	1.67	3.30	0.17	1.30	0.17	1.29	0.17	1.29	0.17	1.28	0.17	1.29	0.17	1.29
11.06.	1.65	5.53	0.18	1.38	0.18	1.39	0.18	1.37	0.18	1.38	0.18	1.40	0.18	1.39
30.06.	2.89	8.77	0.21	1.30	0.21	1.30	0.21	1.28	0.21	1.32	0.21	1.29	0.21	1.29
02.07.	0.61	2.49	0.17	1.33	0.17	1.34	0.17	1.32	0.17	1.34	0.17	1.34	0.17	1.30
16.07.	3.44	9.86	0.17	1.29	0.17	1.27	0.17	1.28	0.17	1.28	0.17	1.29	0.17	1.25
23.07.	3.06	8.81	0.17	1.35	0.17	1.36	0.17	1.36	0.17	1.36	0.17	1.37	0.17	1.33
30.07.	2.40	7.16	0.17	1.34	0.17	1.37	0.17	1.33	0.17	1.34	0.17	1.37	0.17	1.31
31.07.	3.41	9.38	0.19	1.44	0.19	1.43	0.19	1.42	0.19	1.47	0.19	1.44	0.19	1.40
01.09.														
08.09.	1.77	3.32	0.17	1.27	0.17	1.26	0.17	1.26	0.17	1.26	0.16	1.27	0.17	1.27
12.09.	0.77	2.03	0.17	1.25	0.17	1.26	0.17	1.24	0.17	1.26	0.17	1.25	0.17	1.25
17.10.			0.22	1.53	0.24	1.46	0.22	1.48	0.23	1.48	0.23	1.47	0.23	1.49
29.10.	3.92	6.10	0.25	1.52	0.25	1.48	0.25	1.50	0.25	1.49	0.25	1.48	0.25	1.52
18.11.	1.80	4.33	0.26	1.56	1.49	2.76	0.24	1.54	1.99	3.19	2.05	3.26	0.25	1.55
09.12.			0.27	1.72			0.25	1.66	2.15	4.04	2.25	4.46	0.26	1.72

Table 6: *Year-Long Suburban Local Metric Error*

	First *RMS* (m)	*First* $\lvert\phi_e\rvert$ (deg)	*Multi* *RMS* (m)	*Multi* $\lvert\phi_e\rvert$ (deg)	*Latest* *RMS* (m)	*Latest* $\lvert\phi_e\rvert$ (deg)	*D2* *RMS* (m)	*D2* $\lvert\phi_e\rvert$ (deg)	*AC* *RMS* (m)	*AC* $\lvert\phi_e\rvert$ (deg)	*MM* *RMS* (m)	*MM* $\lvert\phi_e\rvert$ (deg)	*LDO* *RMS* (m)	*LDO* $\lvert\phi_e\rvert$ (deg)
17.10.	0.26	0.82							0.31	1.10	0.28	1.08	0.29	1.08
23.10.	0.15	0.60							0.25	1.06	0.21	1.03	0.20	1.03
31.10.	1.14	1.82	0.17	0.88	0.17	0.90			0.17	0.92	0.17	0.89	0.17	0.89
12.11.	0.11	0.36	0.13	0.75	0.31	1.14	0.29	1.14	0.14	0.82	0.27	1.04	0.13	0.75
03.12.	0.31	0.66	0.19	0.87	0.20	0.89	0.18	0.87	0.19	0.88	0.19	0.88	0.18	0.87
10.12.	0.14	0.52	0.14	0.37	0.16	0.37	0.14	0.36	0.18	0.37	0.16	0.37	0.15	0.37
14.04.														
28.04.														
30.04.														
14.05.	0.21	0.71	0.16	0.44	0.17	0.49	0.16	0.45	0.17	0.46	0.16	0.46	0.15	0.44
19.05.	0.33	0.79	0.27	0.44	0.28	0.49	0.28	0.45	0.28	0.46	0.28	0.46	0.27	0.45
21.05.	0.50	1.00	0.21	0.61	0.25	0.66	0.23	0.64	0.24	0.66	0.21	0.61	0.21	0.62
26.05.														
28.05.	0.28	0.83	0.20	0.73	0.18	0.79	0.19	0.74	0.21	0.78	0.19	0.75	0.18	0.73
11.06.	0.19	0.73	0.17	0.42	0.30	0.60	0.17	0.42	0.18	0.44	0.18	0.42	0.15	0.41
25.06.														
30.06.														
02.07.	0.45	1.11	0.12	0.45	0.68	0.86	0.13	0.48	0.18	0.56	0.34	0.67	0.13	0.47
16.07.	3.15	2.78	0.13	0.54	0.21	0.60	0.22	0.64	0.23	0.69	0.18	0.57	0.21	0.61
23.07.	153.63	49.53	0.12	0.57	42.30	11.19	0.15	0.65	0.32	0.83	0.48	0.82	0.13	0.62
12.09.	0.12	0.66	0.10	0.47	0.17	0.56	0.10	0.47	0.12	0.51	0.14	0.52	0.10	0.47
17.10.	1.23	1.89	0.22	0.71	0.70	0.90	0.25	0.74	0.81	0.97	0.30	0.76	0.23	0.74

Table 7: *Day-Night Parking Lot Local Metric Error*

	Day *RMS* (m)	*Day* $\lvert\phi_e\rvert$ (deg)	*Multi* *RMS* (m)	*Multi* $\lvert\phi_e\rvert$ (deg)	*D2* *RMS* (m)	*D2* $\lvert\phi_e\rvert$ (deg)	*AC* *RMS* (m)	*AC* $\lvert\phi_e\rvert$ (deg)	*MM* *RMS* (m)	*MM* $\lvert\phi_e\rvert$ (deg)	*LDO* *RMS* (m)	*LDO* $\lvert\phi_e\rvert$ (deg)
15:28	0.07	0.33	0.04	0.34	0.03	0.34	0.05	0.33	0.04	0.33	0.04	0.34
15:39	0.07	0.38	0.06	0.40	0.06	0.40	0.06	0.37	0.06	0.39	0.06	0.38
16:01	0.05	0.37	0.05	0.38	0.05	0.38	0.05	0.38	0.04	0.39	0.05	0.38
16:17	0.08	0.47	0.04	0.42	0.04	0.42	0.05	0.43	0.04	0.42	0.04	0.42
16:31												
16:47			0.06	0.40	0.06	0.40	0.05	0.40	0.06	0.40	0.06	0.40
16:53			0.06	0.41	0.08	0.41	0.07	0.41	0.06	0.41	0.06	0.41
17:05			0.06	0.51	0.06	0.50	0.06	0.51	0.06	0.51	0.06	0.50
17:19			0.06	0.65	0.05	0.64	0.06	0.65	0.06	0.65	0.06	0.64
17:27												
17:36			0.08	0.48	0.08	0.48	0.08	0.48	0.08	0.48	0.08	0.48

References

Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *Computer Vision–ECCV 2010*, pages 29–42. Springer, 2010.

Peter Biber and Tom Duckett. Dynamic maps for long-term operation of mobile service robots. In *Robotics: science and systems*, pages 17–24, 2005.

M. Bosse and R. Zlot. Place recognition using keypoint voting in large 3D lidar datasets. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2677–2684, May 2013. doi: 10.1109/ICRA.2013.6630945.

M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1899–1906. IEEE, 2003.

Marcus A Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3057–3064. IEEE, 2013.

Nicholas Carlevaris-Bianco and Ryan M. Eustice. Long-term simultaneous localization and mapping with generic linear constraint node removal. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1034–1041, Tokyo, Japan, November 2013. doi: http://dx.doi.org/10.1109/IROS.2013.6696478.

Nicholas Carlevaris-Bianco and Ryan M Eustice. Learning visual feature descriptors for dynamic lighting conditions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, IEEE*. IEEE, 2014.

Winston Churchill and Paul Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2085–2092. IEEE, 2013.

Mark Cummins and Paul Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

Mark Cummins and Paul Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

Feras Dayoub, Grzegorz Cielniak, and Tom Duckett. Long-term experiments with an adaptive spherical view representation for navigation in changing environments. *Robotics and Autonomous Systems*, 59(5): 285–295, 2011.

P. Furgale and T.D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010.

Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Muhlfellner, Stefan Wonneberger, Julian Timpner, Stephan Rottmann, Bo Li, et al. Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 809–816. IEEE, 2013.

Lionel Heng, Paul Furgale, and Marc Pollefeys. Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig. *Journal of Field Robotics*, 2014. (video), (code).

H. Johannsson, M. Kaess, M.F. Fallon, and J.J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *RSS Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, Sydney, Australia, Jul 2012.

Edward Johns and Guang-Zhong Yang. Feature co-occurrence maps: Appearance-based localisation throughout the day. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3212–3218. IEEE, 2013.

Edward Johns and Guang-Zhong Yang. Generative methods for long-term place recognition in dynamic scenes. *International Journal of Computer Vision*, 106(3):297–314, 2014.

Laurent Kneip and Paul Timothy Furgale. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, Hong Kong, China, May 3 – June 7 2014. URL `bib/kneip_icra14.pdf`. (code).

Laurent Kneip and Simon Lynen. Direct optimization of frame-to-frame rotation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2352–2359. IEEE, 2013.

K. Konolige and J. Bowman. Towards lifelong visual maps. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1156–1163. IEEE, 2009.

Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přeučil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, 2010.

Tomas Krajnik, Sol Pedre, and Libor Preucil. Monocular navigation for long-term autonomy. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6. IEEE, 2013.

Tomas Krajnik, Jaime P Fentanes, Oscar M Mozos, Tom Duckett, Johan Ekekrantz, and Marc Hanheide. Long-term topological localisation for service robots in dynamic environments using spectral maps. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4537–4542. IEEE, 2014.

H. Lategahn and C. Stiller. City GPS using stereo vision. In *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pages 1 –6, july 2012. doi: 10.1109/ICVES.2012.6294279.

H. Lategahn and C. Stiller. Vision-only localization. *Intelligent Transportation Systems, IEEE Transactions on*, 15(3):1246–1257, June 2014. ISSN 1524-9050. doi: 10.1109/TITS.2014.2298492.

Henning Lategahn, Johannes Beck, Bernd Kitt, and Christoph Stiller. How to learn an illumination robust image feature for place recognition. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 285–291. IEEE, 2013.

Gim Hee Lee. *Visual mapping and pose estimation for self-driving cars*. PhD thesis, Eidgenössische Technische Hochschule ETH Zürich, Nr. 21858, 2014, 2014.

Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 564–571. IEEE, 2013.

Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In *Computer Vision–ECCV 2010*, pages 791–804. Springer, 2010.

Steven Lovegrove, Andrew J Davison, and Javier Ibanez-Guzmán. Accurate visual odometry from a rear parking camera. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 788–793. IEEE, 2011.

Will Maddern, Michael Milford, and Gordon Wyeth. Towards persistent localization and mapping with a continuous appearance-based topology. *Robotics*, page 281, 2013.

Will Maddern, Alexander D Stewart, and Paul Newman. LAPS-II: 6-DOF day and night visual localisation with prior 3D structure for autonomous road vehicles. In *Intelligent Vehicles Symposium, 2014 IEEE*, 2014.

Mladen Mazuran, Tipaldi Gian Diego, Spinello Luciano, and Wolfram Burgard. Nonlinear graph sparsification for SLAM. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

C McManus, P T Furgale, B E Stenning, and T D Barfoot. Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287, 2013. doi: 10.1002/rob.21444.

Colin McManus, Winston Churchill, Will Maddern, Alexander D Stewart, and Paul Newman. Shady dealings: Robust, long-term visual localisation using illumination invariance. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014a.

Colin McManus, Ben Upcroft, and Paul Newman. Scene signatures: Localised and point-less features for localisation. In *Proceedings of Robotics Science and Systems (RSS)*, Berkley, CA, USA, July 2014b.

Michael Milford, Eleonora Vig, Walter Scheirer, and David Cox. Vision-based simultaneous localization and mapping in changing outdoor environments. *Journal of Field Robotics*, 31(5):814–836, 2014.

Michael J Milford and Gordon Fraser Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649. IEEE, 2012.

Peter Muehlfellner, Paul Furgale, Wojciech Derendarz, and Roland Philippsen. Evaluation of fisheye-camera based visual multi-session localization in a real-world scenario. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 57–62. IEEE, 2013.

Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and Cyrill Stachniss. Robust visual robot localization across seasons using network flows. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2014.

Peer Neubert, Niko Sunderhauf, and Peter Protzel. Appearance change prediction for long-term navigation across seasons. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 198–203. IEEE, 2013.

Kai Ni, Drew Steedly, and Frank Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

Edward Pepperell, Peter I Corke, and Michael J Milford. All-environment visual place recognition with smart. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1612–1618. IEEE, 2014.

G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, 2010.

Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Adaptive relative bundle adjustment. In *Robotics Science and Systems (RSS)*, Seattle, USA, June 2009.

Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *CVPR*, volume 1, page 2, 2008.

Alexander D Stewart and Paul Newman. Laps-localisation using appearance of prior structure: 6-DOF monocular camera localisation using prior pointclouds. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2625–2632. IEEE, 2012.

Christoffer Valgren and Achim J Lilienthal. Sift, surf & seasons: Appearance-based long-term localization in outdoor environments. *Robotics and Autonomous Systems*, 58(2):149–156, 2010.

Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997.

J. Ziegler, H. Lategahn, M. Schreiber, C.G. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller. Video based localization for Bertha. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 1231–1238, June 2014a. doi: 10.1109/IVS.2014.6856560.

Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, C Keller, et al. Making Bertha drive? an autonomous journey on a historic route. *Intelligent Transportation Systems Magazine, IEEE*, 6(2):8–20, 2014b.

D Leveraging Image-Based Localization for Infrastructure-Based Calibration of a Multi-Camera Rig

Authors: L. Heng, P. Furgale, M. Pollefeys

Link to Table of Contents.

Leveraging Image-Based Localization for Infrastructure-Based Calibration of a Multi-Camera Rig

Lionel Heng*
Computer Vision and Geometry Lab
ETH Zürich
Universitätstrasse 6, 8092 Zürich
hengli@inf.ethz.ch

Paul Furgale
Autonomous Systems Lab
ETH Zürich
Tannenstrasse 3, 8092 Zürich
paul.furgale@mavt.ethz.ch

Marc Pollefeys
Computer Vision and Geometry Lab
ETH Zürich
Universitätstrasse 6, 8092 Zürich
marc.pollefeys@inf.ethz.ch

Abstract

Most existing calibration methods for multi-camera rigs are computationally expensive, use installations of known fiducial markers, and require expert supervision. We propose an alternative approach called *infrastructure-based calibration* that is efficient, requires no modification of the infrastructure (or calibration area), and is completely unsupervised. In infrastructure-based calibration, we use a map of a chosen calibration area and leverage image-based localization to calibrate an arbitrary multi-camera rig in near real-time. Due to the use of a map, before we can apply infrastructure-based calibration, we have to run a survey phase once to generate a map of the calibration area. In this survey phase, we use a survey vehicle equipped with a multi-camera rig and a calibrated odometry system, and SLAM-based self-calibration to build the map which is based on natural features. The use of the calibrated odometry system ensures that the metric scale of the map is accurate. Our infrastructure-based calibration method does not assume an overlapping field of view between any two cameras, and does not require an initial guess of any extrinsic parameter. Through extensive field tests on various ground vehicles in a variety of environments, we demonstrate the accuracy and repeatability of the infrastructure-based calibration method for calibration of a multi-camera rig. The code for our infrastructure-based calibration method is publicly available as part of the CamOdoCal library at `https://github.com/hengli/camodocal`.

1 Introduction

There is an ongoing and rapid growth in the use of cameras on robotic and human-operated vehicles. From the robotic perspective, a camera is a rich source of appearance information. Furthermore, recent advances

*`http://www.inf.ethz.ch/personal/hengli`



(a) The Volkswagen Golf platforms. The cameras are integrated into the car body.

(b) The Toyota Prius platform. The cameras are mounted on the roof rack.

Figure 1: Each platform is equipped with a set of four fish-eye cameras that provides a surround view of the environment. These platforms facilitate the demonstration of two unsupervised methods, self-calibration and infrastructure-based calibration, to estimate the extrinsic transformations of these multi-camera systems without operator input. No specialized fiducial markers such as chessboards are used.

in both structure-from-motion techniques and computing hardware enable the extraction of geometric information from images in real time. This abundance of sensory information together with the use of multiple cameras to increase the robot's field of view greatly enhances robot perception. From the automotive perspective, multiple cameras are useful for driver assistance applications which help improve road safety. For example, major automotive manufacturers are making surround view systems standard on an increasing number of car models; each of these surround view systems typically comprises four outward-looking cameras. Our three test platforms follow this trend; each platform is equipped with a four-fisheye-camera system that provides a surround view. Two of our test platforms are Volkswagen Golf cars modified for autonomous driving under the auspices of the V-Charge project (Furgale et al., 2013b) while the third platform is a Toyota Prius car retrofitted for sensor data collection. These platforms are shown in Figure 1.

For vehicles equipped with multi-camera systems, so-called *calibration parameters*---transformations between cameras, camera lens parameters, etc.---must be known with a high degree of precision to ensure safe and robust operation in the presence of pedestrians and other vehicles. Although systems can be calibrated in the factory, several parameters will change during extended operation due to normal wear and tear. To everyone, particularly in research and development groups and the automotive after-market industry, calibration and recalibration of multi-camera systems is a continual burden that requires expertise, specialized equipment, and many man-hours. Consequently, it is necessary that we seek accurate, fast, robust, and unsupervised calibration algorithms.

One possibility for calibration of multi-sensor systems is to build "calibration areas" that include special instrumentation for different sensors such as known fiducial markings for cameras or known structural geometry for lasers. The use of specially designed fiducials can resolve appearance and geometric ambiguities and reduce the computational complexity of system calibration. For example, this was the strategy adopted by Geiger et al. (2012) as they were collecting an extensive multi-sensor dataset over a number of months (Geiger et al., 2013). They installed a number of chessboards covering the full field of view of the cameras and laser scanner and used them to recalibrate the vehicle before every run of data collection. As attractive as this method is, it still requires modification of the infrastructure, which could make deployment on the large scale complicated and expensive, and represent yet another barrier to the deployment of autonomous systems equipped with multiple sensors.

In this paper, we introduce a method called *infrastructure-based calibration* that shares positive aspects with the above-mentioned method but relaxes the requirement to modify the infrastructure. This relaxation is made possible by using natural features instead of known fiducial markings. Our infrastructure-based calibration method requires a map of a chosen calibration area, and hence, a prior one-time run of a survey

phase which builds a high-fidelity map of the calibration area. Mapping is made possible via SLAM-based self-calibration which makes use of data collected from a survey vehicle equipped with a multi-camera rig and a calibrated odometry system. SLAM-based self-calibration allows us to build an accurate map of the calibration area with a multi-camera rig for which we do not know the extrinsic parameters beforehand. Once the map is available, we can apply infrastructure-based calibration in which we use the map and leverage image-based localization to calibrate an unlimited number of arbitrary multi-camera rigs. Sections 4 and Section 5 describe in detail the SLAM-based self-calibration and infrastructure-based calibration respectively. We note that each infrastructure-based calibration incurs a small fraction of the high computational cost that comes with SLAM-based self-calibration. This high cost is due to the presence of two computationally expensive steps in SLAM-based self-calibration methods: exhaustive feature matching between different cameras and bundle adjustment. In contrast, these two steps are not required in image-based localization which is used by our infrastructure-based calibration method. Both the SLAM-based self-calibration and infrastructure-based calibration methods are unsupervised; they neither require expert supervision nor a special calibration setup. Furthermore, no initial guesses for the extrinsic parameters or overlapping fields of view between any two cameras are required. Infrastructure-based calibration is therefore a both financially and computationally efficient solution to the continuous unsupervised calibration of large numbers of multi-camera rigs.

The accuracy of our infrastructure-based calibration method is dependent on the accuracy of image-based localization, and in turn, the accuracy of the map of the calibration area. If we are to estimate extrinsic parameters with correct metric scale, the metric scale of the map of the calibration area generated by our SLAM-based self-calibration method must be accurate. Metric accuracy is ensured via a calibrated odometry system on the survey vehicle. We note that the SLAM-based self-calibration method used in the survey phase requires a calibrated odometry system to infer metric scale, but we do not see this requirement as restrictive since odometry data is commonly available on robots and vehicles. Furthermore, the map must be globally consistent which is ensured via loop closures. To illustrate the importance of global consistency, we give an example case in which a location is visited twice during the survey phase. No loop closures are performed, and hence, this location is represented as two distinct places in the map of the calibration area. We calibrate a two-camera rig, and it is possible that during the calibration, we use data associated with the first visit for the first camera, and data associated with the second visit for the second camera. Since the geometric discrepancy between the first and second visits is not rectified, the calibration inaccurately estimates the extrinsics such that the two cameras appear further from each other than expected.

In this paper, we expand on our previous work on infrastructure-based calibration (Heng et al., 2014) by including details of the survey phase. To make infrastructure-based calibration as accurate as possible, we focus on maximizing the accuracy of the map generated by the survey phase. Specifically, we present a significantly improved version of our SLAM-based self-calibration method (Heng et al., 2013) that is able to generate a metrically accurate and globally consistent map. Such a map enables our infrastructure-based calibration method to estimate extrinsic parameters with high accuracy and repeatability for arbitrary multi-camera rigs.

We briefly discuss the improvements to our SLAM-based self-calibration method used in the survey phase. We add robust 6D pose graph optimization so that wrong loop closures can be identified and removed. Furthermore, with globally consistent vehicle pose estimates from pose graph optimization, the joint optimization step is more likely to converge to the correct solution. We also improve the accuracy of feature matching between images from different cameras. Other improvements focus on the joint optimization step. Here, landmarks seen in different cameras tend to account for a small percentage of all landmarks in the map, and to maximize the accuracy of the poses of the cameras with respect to one another, a higher weighting is assigned to landmarks seen in different cameras. We also make use of data from the intrinsic calibration to enforce the constraint that lines are straight in rectified pinhole images; otherwise, overfitting of intrinsic parameters in the joint optimization step may lead to warped lines. Furthermore, we make use of a calibrated odometry system to infer metric scale. In addition, 6D vehicle poses are optimized instead of 3D vehicle poses to account for non-planar ground surfaces in real-world environments. Field experiments are performed to quantify the accuracy and repeatability of the camera extrinsics that are estimated by

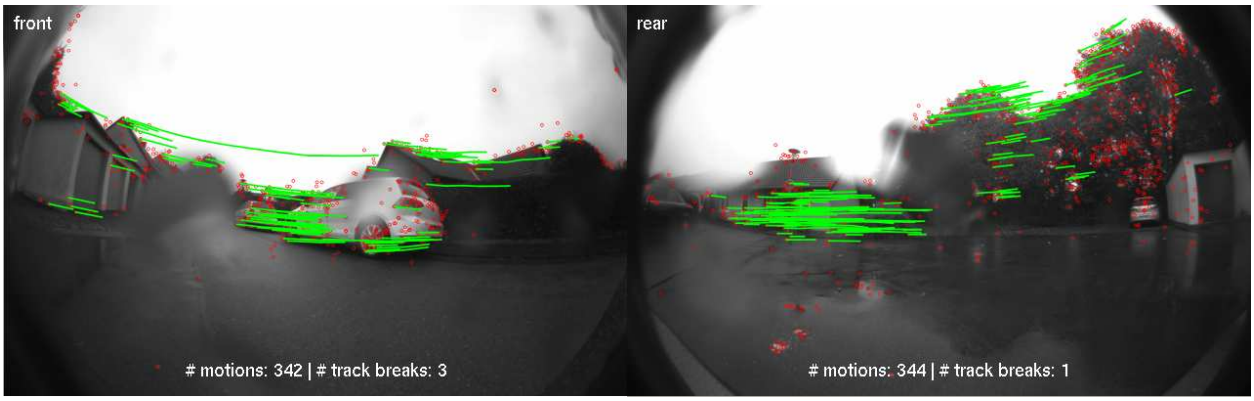


Figure 2: In rainy conditions, water droplets on the camera lens cause significant parts of the image to be obscured, and render such areas unusable for feature tracking. However, our calibration methods are still able to estimate the camera extrinsics with high accuracy.

the improved SLAM-based self-calibration method. Furthermore, we also evaluate the accuracy of the map generated by the improved SLAM-based self-calibration method.

Our work is novel in the following aspects:

1. Our SLAM-based self-calibration and infrastructure-based calibration methods are able to estimate in an unsupervised fashion and without special calibration objects the inter-camera transforms with metric scale for a multi-camera rig. Here, we do not assume overlapping fields of views. To the best of our knowledge, there is no other existing target-free calibration method for multi-camera rigs that is able to estimate the inter-camera transforms with metric scale.

2. We present extensive experiments to show the accuracy of both calibration methods for multi-camera rigs. To the best of our knowledge, no other work on calibration for multi-camera rigs shows experimental validation on a similar scale.

3. To the best of our knowledge, there is no other existing method for multi-camera rigs that makes use of a prior map based on natural features to calibrate a multi-camera rig.

Our calibration methods are designed to be robust; the SLAM-based self-calibration method used in the survey phase explicitly handles breaks in visual odometry and wrong loop closures, and the infrastructure-based calibration method only estimates camera poses given a minimum number of 2D-3D correspondences. For example, in rainy conditions, water droplets can land on the camera lenses, causing significant parts of the images to be obscured and rendered unusable for feature tracking as shown in Figure 2.

2 Related Work

There has been much research on the hand-eye calibration problem (Daniilidis, 1999; Brookshire and Teller, 2012). In general, published generic solutions to the hand-eye calibration problem do not work for vehicles with planar motions, as the height of the camera with respect to the odometry is unobservable. Hence, camera-odometry calibration requires specialized solutions (Antonelli et al., 2010; Censi et al., 2013). Antonelli et al. (2010) finds the extrinsics, camera intrinsics, and odometry parameters simultaneously for a differential drive robot; however, a set of known landmarks is required. Similarly, Censi et al. (2013) finds the extrinsics and odometry parameters for a differential drive robot without the need for known landmarks, but only obtain the three degrees of freedom of the camera-odometry transform. In the case of multi-camera

rigs with minimal overlapping fields of view, simply computing the camera-odometry transform for each camera based on camera and odometry motions is not sufficient to achieve accurate calibration parameters for a multi-camera rig. A feature point may be observed by multiple cameras at different times, and if such information is not made use of, the resulting calibration parameters may not be sufficiently accurate for reprojection of one point in a camera into another camera with a sub-pixel reprojection error. Hence, we look at state-of-the-art methods for calibrating multi-sensor rigs.

2.1 Self-Calibration

We look at unsupervised methods that do not require specific calibration patterns. In the case of lasers, Maddern et al. (2012) proposes a method to estimate the calibration parameters between at least one 3D laser and multiple 2D lasers. They construct a point cloud using a 3D laser, and use this point cloud to obtain the relative transforms between the 3D laser and each 2D laser. Here, they require a 3D laser, and an initial guess of the extrinsic parameters. In the case of both lasers and cameras, Levinson and Thrun (2012) exploit depth discontinuities in laser data and edges in images to estimate the 6-DoF transform between a Velodyne sensor and a camera.

In the case of cameras, Carrera et al. (2011) uses natural features in the environment to estimate the extrinsic parameters up to scale. They propose an extrinsic calibration of a multi-camera rig by using a modified version of MonoSLAM to build a globally consistent sparse map of landmarks for each camera, finding feature correspondences between each pair of maps via thresholded matching between SURF descriptors, and using the 3D similarity transform together with RANSAC to find inlier feature correspondences. At the end, a joint optimization is run to optimize the camera poses, 3D scene points, and robot poses. Our approach is similar in spirit; however, we estimate the extrinsic parameters with metric scale whereas Carrera et al. (2011) does not. Furthermore, we optimize the intrinsics; we find that if the intrinsics are not optimized as part of the bundle adjustment, the calibration accuracy is suboptimal. Our self-calibration method scales well to large environments unlike the approach of Carrera et al. (2011). Here, they use MonoSLAM which is known not to scale to large environments. As a result, our self-calibration method is able to generate a map of a large calibration area that can be used for efficiently calibrating multi-camera rigs. Furthermore in Carrera et al. (2011), the 3D similarity transform step can fail in outdoor environments where the majority of natural features are located far away from the cameras, and their estimated 3D locations can have substantial noise as a result, leading to few inliers.

We note that since SLAM-based calibration methods do not assume a prior map, they have to perform an exhaustive search of feature correspondences between images from different cameras. By relying on a pre-existing map for infrastructure-based calibration, we remove the need to find inter-camera feature correspondences. Furthermore, we do not have to do costly bundle adjustment. Hence, infrastructure-based calibration is far simpler, more robust, and requires a much shorter time compared to SLAM-based calibration methods. We can think of our infrastructure-based calibration method as requiring a one-time fixed cost in terms of map generation but entailing a much lower variable cost per calibration. Hence, our infrastructure-based calibration method is advantageous over SLAM-based calibration methods if many calibrations are needed within a short time span.

2.2 Infrastructure-Based Calibration

All known infrastructure-based calibration methods make use of calibration setups that include known fiducial markings. The use of specially designed fiducials can resolve appearance and geometric ambiguities and reduce the computational complexity of system calibration.

In the case of lasers, Gao and Spletzer (2010) use pairs of retro-reflective markers with known baselines for extrinsic calibration of a multi-laser system. Features are detected from these pairs of markers, and after a vehicle is driven along multiple loops, inter-loop feature correspondences are found and used to estimate the

extrinsic parameters via second-order cone programming.

In the case of cameras, a majority of existing work (Kumar et al., 2008; Lebraly et al., 2011; Li et al., 2013) requires a pattern board. The additional use of a mirror in Kumar et al. (2008) creates a limitation in which the mirror has to be in the camera's field of view, while at the same time, the entire pattern is visible in the camera. Lebraly et al. (2011) uses two pattern boards, and requires the rig to manoeuvre such that each camera sees both pattern boards at different times. Li et al. (2013) requires neighboring cameras to see some part of the pattern at the same time. We note that the use of a pattern board comes with a viewing constraint that makes calibration of multi-camera rigs non-straightforward. To overcome the viewing constraint associated with the use of a pattern board, Geiger et al. (2012) uses a room setup in which many chessboards are mounted in a configuration that covers the full field of view of the cameras and 3D laser. This room setup was used to recalibrate the multi-sensor setup before every run of data collection for an extensive dataset over a number of months (Geiger et al., 2013). This method requires modification of the infrastructure. In contrast, our infrastructure-based calibration method does not require modification of the infrastructure as it makes use of natural features instead of chessboards.

Existing methods for localization of multi-camera rigs (Tariq and Dellaert, 2004; Kneip et al., 2013; Lee et al., 2013b) assume that the extrinsic parameters are known with a high degree of accuracy. and hence, are not applicable in infrastructure-based calibration. Furthermore, these multi-camera localization methods are sensitive to errors in the extrinsic parameters; a slightly incorrect estimate of the subset of extrinsic parameters corresponding to one camera may lead to the labeling of all inlier 2D-3D feature correspondences observed by that camera as outliers. As a result, that camera is rendered redundant in the multi-camera localization. Our infrastructure-based calibration method estimates all extrinsic parameters accurately such that reliable multi-camera localization is possible.

3 Vehicle Platforms

Each of our vehicle platforms uses four CCD cameras with global shutter and GigE interfaces. On the Volkswagen Golf cars, the cameras built by Allied Vision Technologies provide 1280×800 monochrome images at a user-configured rate of 12.5 Hz, while on the Toyota Prius car, the cameras built by Matrix Vision provide 1280×960 monochrome images at a user-configured rate of 10 Hz. Each camera is fitted with a fisheye lens that has a 185° field of view. For each multi-camera setup, one camera is designated as a master camera which triggers all the other cameras. In this way, the multi-camera setup captures images from all cameras simultaneously.

The calibrated odometry system on each vehicle platform is a black box provided by the car manufacturer; apart from the fact that the odometry system reads in wheel tick data from each wheel and steering wheel angle measurements, and outputs odometry poses, we do not know the internal workings of the odometry system. On each platform, odometry pose data is published at 50 Hz.

RTI Connext DDS[1] and ROS[2] are used for message transmission of image and odometry data on the Volkswagen Golf car and Toyota Prius car respectively. A computer located in the boot logs all data to disk, and the data logs are subsequently processed by our calibration methods.

4 Survey Phase - SLAM-Based Self-Calibration

This section describes our SLAM-based self-calibration method that is used in the survey phase. This method simultaneously produces a map of the calibration area and estimates the inter-camera transforms for the

[1] `http://www.rti.com/products/dds/index.html`
[2] `http://www.ros.org`

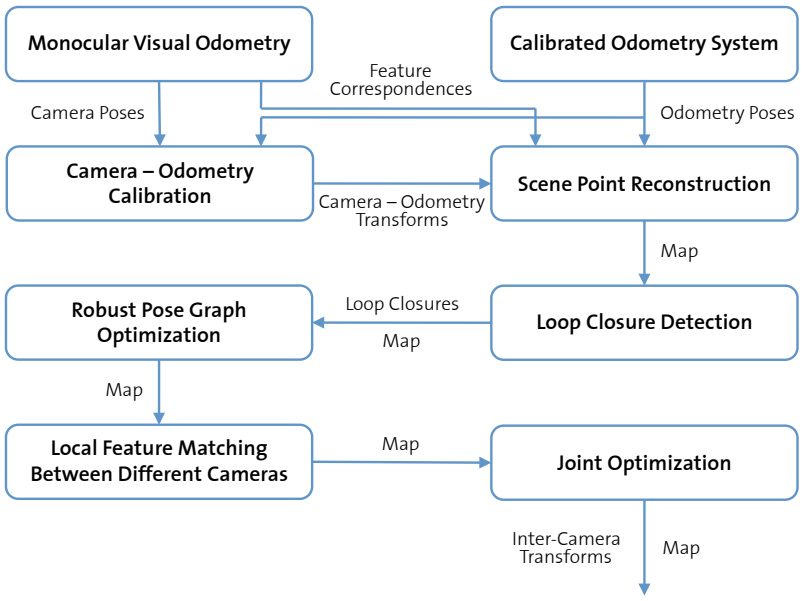


Figure 3: Images and odometry data are input to the self-calibration pipeline which estimates the camera extrinsics, and at the same time, generates a map of the calibration area.

multi-camera rig mounted on the survey vehicle. A calibrated odometry system on the survey vehicle is used to infer metric scale for the map. The generated map is subsequently used for the infrastructure-based calibration method described in Section 5.

Figure 3 shows the pipeline for our self-calibration method. Here, we give a brief outline of the method followed by detailed descriptions in the subsections below.

The monocular visual odometry step detailed in Section 4.1 estimates sets of camera poses and inlier feature tracks for each camera. The camera poses are used by the camera-odometry calibration described in Section 4.2 together with odometry poses to obtain an initial estimate of the camera-odometry transforms. This initial estimate is used together with the inlier feature tracks and odometry data by the scene point reconstruction described in Section 4.3.

By then, we have a map of the calibration area and estimates of the camera-odometry transforms. However, the map is not globally consistent as there are no loop closures. The inter-camera transforms inferred from the camera-odometry transforms are not accurate as we have not taken camera-to-camera constraints into account when computing the camera-odometry transforms in Section 4.2.

To make the map globally consistent, we detect loop closures as described in Section 4.4. Subsequently, we use robust pose graph optimization described in Section 4.5 to identify wrong loop closures and remove them, and at the same time, optimize the pose graph. Furthermore, using the globally consistent vehicle pose estimates from robust pose graph optimization instead of the odometry poses as an initial guess for the joint optimization leads to better convergence behavior. To obtain accurate inter-camera transforms such that a feature point in one camera can be reprojected to another camera with sub-pixel accuracy, we obtain temporal camera-to-camera constraints in the form of feature correspondences between different cameras as described in Section 4.6. Subsequently, the joint optimization described in Section 4.7 uses the temporal camera-to-camera constraints to optimize the inter-camera transforms in addition to intrinsic camera parameters, vehicle poses, and 3D scene points. We now have a metrically accurate and globally consistent map, and accurate inter-camera transforms.

Table 1: A table row corresponds to a step in the self-calibration pipeline. A bullet in cell (r, c) indicates that variable c is modified by step r.

	Camera Intrinsics	Camera-Odometry Transforms	Vehicle Poses	Scene Points
Monocular Visual Odometry				
Camera-Odometry Calibration		•		
Scene Point Reconstruction		•		•
Loop Closure Detection				•
Robust Pose Graph Optimization			•	
Local Feature Matching Between Different Cameras				•
Joint Optimization	•	•	•	•

Table 1 shows which variables are modified by each step in the pipeline. In summary, the pipeline produces optimized estimates of the intrinsic camera parameters and inter-camera transforms for a rig with an arbitrary number of cameras, and a metrically accurate and globally consistent map of the calibration area which can reliably be used by infrastructure-based calibration.

4.1 Monocular Visual Odometry (VO)

The monocular VO step takes images from the multi-camera system and time-synchronized odometry as input, and outputs sets of camera poses and inlier feature tracks for each camera.

In this step, we run monocular VO for each camera in order to obtain a set of camera motions together with inlier feature tracks. The set of camera motions is required for the subsequent step of computing an initial estimate of the camera-odometry transforms in Section 4.2. Similarly, the inlier feature tracks are required for scene point reconstruction in Section 4.3. It is possible for the VO to break occasionally in poorly-textured areas, producing sets of VO estimates with different scales. Nevertheless, in the camera-odometry calibration stage described in Section 4.2, we use all sets of VO estimates to find an initial estimate of the camera-odometry transform for each camera.

In our monocular VO, we use OpenCV's GPU implementation of SURF[3] to extract feature points and their descriptors. We extract a synchronized set of keyframes from all cameras when the corresponding odometry pose is at least 0.2 m away from the odometry pose at which the last set of keyframes was taken. At each iteration, we use the P3P method (Kneip et al., 2011) together with RANSAC to compute the camera pose and identify inlier feature tracks, and subsequently, apply sliding window bundle adjustment. We show the inlier feature tracks and the estimated camera poses for the four fish-eye cameras in Figures 4 and 5 respectively.

4.2 Camera - Odometry Calibration

The camera-odometry calibration step uses the camera poses computed by the monocular VO step and time-synchronized odometry. The output is an initial estimate of the camera-odometry transforms.

[3] `http://docs.opencv.org/modules/nonfree/doc/feature_detection.html`

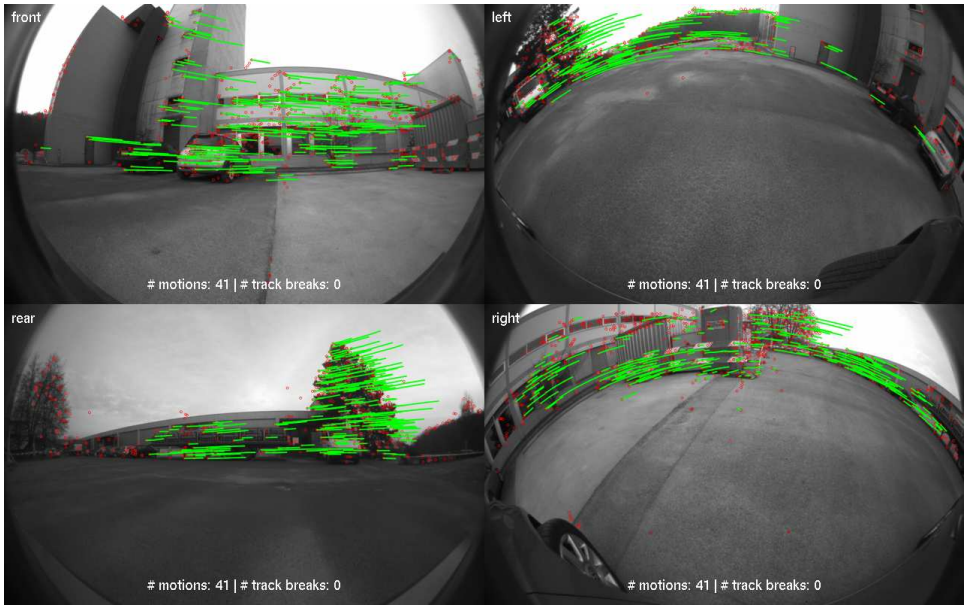


Figure 4: Inlier feature tracks.

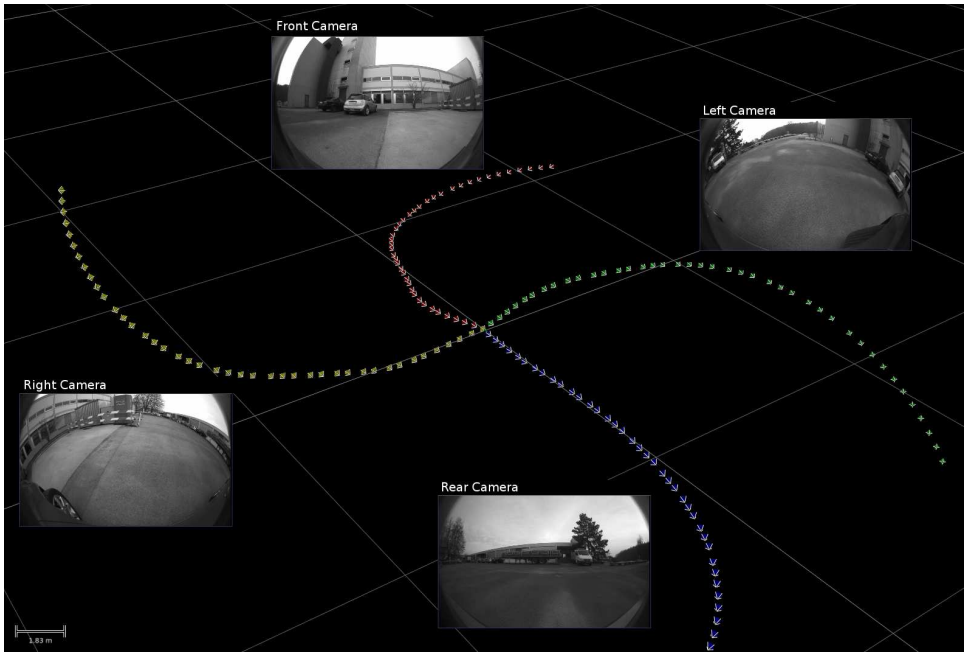


Figure 5: Camera poses estimated by monocular VO for the four fish-eye cameras when the car is making a gradual right turn. Note that the scale of the estimated poses for each camera is different. Our camera-odometry calibration step jointly solves for the different scales and the camera-odometry transforms.

To obtain an initial estimate of each camera-odometry transform, we solve a planar version of the hand-eye calibration problem which relates camera motions to odometry motions via the camera-odometry transform. Camera motions are obtained from the previous monocular VO step. We cannot use existing solutions for the 6-DoF hand-eye calibration problem, as the car moves on a $x-y$ plane most of the time, and thus, the z-component of the camera-odometry translation is unobservable.

It is common for the VO estimation to break occasionally, resulting in several segments of camera poses with different scales from a single camera. The method of Guo et al. (2012) is only able to use one segment of camera poses at most to estimate the camera-odometry transform. Hence, we make modifications to the method to estimate a single camera-odometry transform from an arbitrary number of segments of camera poses. In the first step which is identical to that in Guo et al. (2012), our approach solves for the pitch and roll components of the camera-odometry rotation by minimizing a least-squares cost function. The second step differs from Guo et al. (2012) in the aspect that we estimate multiple scales instead of a single scale; we simultaneously solve for the yaw component of the camera-odometry rotation, the camera-odometry translation, and the scales for all sparse maps by minimizing another least-squares cost function. As the vehicle motion is planar, the z-component of the camera-odometry translation is unobservable, and therefore, set to zero.

For each camera, in the first step, we estimate $\mathbf{q}_{yx}$, and in the second step, we estimate $\mathbf{q}_z$, ${}^{O}\mathbf{t}_C = [t_x, t_y, 0]^T$, and s_j for $j = 1, ..., m$ where ${}^{O}\mathbf{q}_C = \mathbf{q}_z\mathbf{q}_{yx}$ and ${}^{O}\mathbf{t}_C$ are the unit quaternion and translation respectively that transform the camera frame to the odometry frame, $\mathbf{q}_z$ corresponds to the yaw quaternion that corresponds to the yaw component of the camera-odometry rotation, $\mathbf{q}_{yx}$ corresponds to the pitch-roll quaternion which is the product of the pitch and roll quaternions corresponding to the pitch and roll components of the camera-odometry rotation respectively, and s_j is the scale for each of the m sparse maps.

We first start with the well-known hand-eye problem using the quaternion representation:

$$ {}^{O_{i+1}}\mathbf{q}_{O_i}\, {}^{O}\mathbf{q}_C = {}^{O}\mathbf{q}_C\, {}^{C_{i+1}}\mathbf{q}_{C_i}, \tag{1} $$

$$ (\mathbf{R}({}^{O_{i+1}}\mathbf{q}_{O_i}) - \mathbf{I})\, {}^{O}\mathbf{t}_C = s_j \mathbf{R}({}^{O}\mathbf{q}_C)\, {}^{C_{i+1}}\mathbf{t}_{C_i} - {}^{O_{i+1}}\mathbf{t}_{O_i}, \tag{2} $$

where ${}^{O_{i+1}}\mathbf{q}_{O_i}$ and ${}^{O_{i+1}}\mathbf{t}_{O_i}$ are the unit quaternion and translation respectively that transform odometry frame i to odometry frame $i+1$, ${}^{C_{i+1}}\mathbf{q}_{C_i}$ and ${}^{C_{i+1}}\mathbf{t}_{C_i}$ are the unit quaternion and translation respectively that transform camera frame i to camera frame $i+1$, $\mathbf{R}(\mathbf{q})$ is the rotation matrix that corresponds to $\mathbf{q}$, and $\mathbf{I}$ is a 3×3 identity matrix. Given ${}^{O_{i+1}}\mathbf{q}_{O_i}$, ${}^{O_{i+1}}\mathbf{t}_{O_i}$, ${}^{C_{i+1}}\mathbf{q}_{C_i}$, and ${}^{C_{i+1}}\mathbf{t}_{C_i}$, we use equations 1 and 2 to estimate ${}^{O}\mathbf{q}_C$ and ${}^{O}\mathbf{t}_C$.

4.2.1 Finding the pitch-roll quaternion

The method to find the pitch-roll quaternion is identical to that in Guo et al. (2012). To estimate $\mathbf{q}_{yx}$, we substitute ${}^{O}\mathbf{q}_C = \mathbf{q}_z\mathbf{q}_{yx}$ into equation 1 and use the fact that rotations around the z-axis commute to get

$$ {}^{O_{i+1}}\mathbf{q}_{O_i}\mathbf{q}_{yx} - \mathbf{q}_{yx}\, {}^{C_{i+1}}\mathbf{q}_{C_i} = 0, \tag{3} $$

which can be written as a matrix vector equation:

$$ \underbrace{(\mathcal{L}({}^{O_{i+1}}\mathbf{q}_{O_i}) - \mathcal{R}({}^{C_{i+1}}\mathbf{q}_{C_i}))}_{\mathbf{S}}\, \mathbf{q}_{yx} = 0, \tag{4} $$

where the matrix which we call $\mathbf{S}$ is a 4×4 matrix and

$$ \mathbf{q} = [w_q\ x_q\ y_q\ z_q]^T, $$

$$ \mathcal{L}(\mathbf{q}) = \begin{bmatrix} w_q & -z_q & y_q & x_q \\ z_q & w_q & -x_q & y_q \\ -y_q & x_q & w_q & z_q \\ -x_q & -y_q & -z_q & w_q \end{bmatrix}, \quad \mathcal{R}(\mathbf{q}) = \begin{bmatrix} w_q & z_q & -y_q & x_q \\ -z_q & w_q & x_q & y_q \\ y_q & -x_q & w_q & z_q \\ -x_q & -y_q & -z_q & w_q \end{bmatrix}. $$

We have two constraints on the unknown $\mathbf{q}_{yx}$:

$$x_{q_{yx}} y_{q_{yx}} = -z_{q_{yx}} w_{q_{yx}}, \quad \text{and} \quad \mathbf{q}_{yx}^T \mathbf{q}_{yx} = 1. \tag{5}$$

Given $n \geq 2$ motions, we build the $4n \times 4$ matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{S}_1^T & ... & \mathbf{S}_n^T \end{bmatrix}^T, \tag{6}$$

which has rank 2 in the absence of noise. We find the singular value decomposition $\mathbf{T} = \mathbf{U}\mathbf{S}\mathbf{V}^T$; the last two right-singular vectors $\mathbf{v}_3$ and $\mathbf{v}_4$ which are the last two columns of $\mathbf{V}$ span the null space of $\mathbf{T}$:

$$\mathbf{q}_{yx} = \lambda_1 \mathbf{v}_3 + \lambda_2 \mathbf{v}_4. \tag{7}$$

We use the two constraints from equation 5 to solve for λ_1 and λ_2, and therefore, obtain $\mathbf{q}_{yx}$.

4.2.2 Finding the yaw quaternion, scales, and translation

Given that the z-component of the camera-odometry translation is unobservable due to planar motion, we obtain a planar version of equation 2 by setting all z-components to zero:

$$\begin{bmatrix} \cos\phi - 1 & -\sin\phi \\ \sin\phi & \cos\phi - 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} - s_j \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + {}^{O_{i+1}}\mathbf{t}'_{O_i} = 0, \tag{8}$$

where ϕ is the relative yaw between odometry frames i and $i+1$, $[p_1\ p_2]^T$ are the first two elements of the vector $\mathbf{R}(\mathbf{q}_{yx})\,{}^{C_{i+1}}\mathbf{t}_{C_i}$ and ${}^{O_{i+1}}\mathbf{t}'_{O_i}$ denotes the first two elements of the vector ${}^{O_{i+1}}\mathbf{t}_{O_i}$. $\mathbf{q}_z$ is a function of α, the yaw component of the camera-odometry rotation, such that

$$\mathbf{q}_z = \begin{bmatrix} \cos\frac{\alpha}{2} & 0 & 0 & \sin\frac{\alpha}{2} \end{bmatrix}^T. \tag{9}$$

Given ϕ, $[p_1\ p_2]^T$, and ${}^{O_{i+1}}\mathbf{t}'_{O_i}$, our goal is to estimate t_x, t_y, s_j and α.

We reformulate equation 8 as a matrix vector equation:

$$\begin{bmatrix} \mathbf{J} & \mathbf{K} \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ -s_j \cos\alpha \\ -s_j \sin\alpha \end{bmatrix} = -\,{}^{O_{i+1}}\mathbf{t}'_{O_i}, \tag{10}$$

where $\mathbf{J} = \begin{bmatrix} \cos\phi - 1 & -\sin\phi \\ \sin\phi & \cos\phi - 1 \end{bmatrix}$ and $\mathbf{K} = \begin{bmatrix} p_1 & -p_2 \\ p_2 & p_1 \end{bmatrix}$.

As the VO for a single camera may break occasionally, we may have more than 1 VO segment. Each VO segment contributes one instance of equation 10. For the m VO segments with $n_1 \geq 2$, ..., $n_m \geq 2$ motions respectively where $n = \sum_{i=1}^{m} n_i$, we stack the m instances of equation 10 into the $2n \times (2 + 2m)$ matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{J}_1^1 & \mathbf{K}_1^1 & ... & 0 & 0 & ... & 0 & 0 \\ ... & ... & ... & 0 & 0 & ... & 0 & 0 \\ \mathbf{J}_{n_1}^1 & \mathbf{K}_{n_1}^1 & ... & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & ... & \mathbf{J}_1^j & \mathbf{K}_1^j & ... & 0 & 0 \\ 0 & 0 & ... & ... & ... & ... & 0 & 0 \\ 0 & 0 & ... & \mathbf{J}_{n_j}^j & \mathbf{K}_{n_j}^j & ... & 0 & 0 \\ 0 & 0 & ... & 0 & 0 & ... & \mathbf{J}_1^m & \mathbf{K}_1^m \\ 0 & 0 & ... & 0 & 0 & ... & ... & ... \\ 0 & 0 & ... & 0 & 0 & ... & \mathbf{J}_{n_m}^m & \mathbf{K}_{n_m}^m \end{bmatrix}, \tag{11}$$

where the matrices $\mathbf{J}_i^j$ and $\mathbf{K}_i^j$ correspond to the ith motion in VO segment j, and

$$\mathbf{G}\begin{bmatrix} t_x \\ t_y \\ -s_0\cos\alpha_0 \\ -s_0\sin\alpha_0 \\ ... \\ -s_m\cos\alpha_m \\ -s_m\sin\alpha_m \end{bmatrix} = -\begin{bmatrix} {}^{O_1}\mathbf{t}'_{O_0} \\ ... \\ {}^{O_{n+1}}\mathbf{t}'_{O_n} \end{bmatrix}, \tag{12}$$

for which we use the least squares method to find the solution to ${}^{O}\mathbf{t}_C = [t_x\ t_y]^T$, s_j and α_j. We have estimated the scale s_j for each VO segment, and the translation ${}^{O}\mathbf{t}_C$; however, we have m hypotheses of α. We choose the best hypothesis that minimizes the cost function

$$\mathbf{C} = \sum_{i=0}^{n_j}((\mathbf{R}({}^{O_{i+1}}\mathbf{q}_{O_i}) - \mathbf{I})\,{}^{O}\mathbf{t}_C - s_j\mathbf{R}(\alpha)\mathbf{R}(q_{yx})\,{}^{C_{i+1}}\mathbf{t}_{C_i} + {}^{O_{i+1}}\mathbf{t}_{O_i}). \tag{13}$$

We then refine the estimate of $\mathbf{q}_{yx}$, α, ${}^{O}\mathbf{t}_C$, and s_j for $j = 1, ..., m$ by using non-linear optimization to minimize $\mathbf{C}$.

4.3 Scene Point Reconstruction

The scene point reconstruction step takes as input inlier feature tracks found during the monocular VO step, time-synchronized odometry, and the initial estimate of the camera-odometry transforms from the camera-odometry calibration step, and outputs 3D scene points. This step is necessary as 3D scene information is required to compute the 6D transform measurement for each loop closure edge in the pose graph.

We initialize the vehicle poses to be the same as the odometry poses. For each camera, we iterate through every frame, and at each frame, we find all features which are visible in the previous and current frames, and do not correspond to a previously initialized 3D scene point. We triangulate each feature correspondence; if the reprojection error of the resulting 3D scene point in either frame does not exceed a threshold of 2 pixels, we associate the 3D scene point with the feature correspondence. Subsequently, each feature track is associated to the same 3D scene point which the first two features already correspond to. We illustrate the scene point reconstruction step in Figure 6. We use the Ceres solver (Agarwal et al., 2013) to run a bundle adjustment which optimizes the camera-odometry transforms and 3D scene points by minimizing the image reprojection error across all frames for all cameras while keeping the vehicle poses fixed.

4.4 Loop Closure Detection

The loop closure detection step takes as input the camera-odometry transforms, images from the multi-camera system, and 3D scene points computed by the scene point reconstruction step, and outputs loop closure measurements, each of which is a relative 6D transform between two vehicle poses. Here, a loop closure can be between either the same camera, or different cameras.

The 6D transform measurements are used in the robust pose graph optimization step. The loop closure detection step also generates an inlier set of 2D-3D correspondences for each loop closure; if a loop closure is verified by the robust pose graph optimization step to be correct, the corresponding set of 2D-3D correspondences is used to merge duplicate 3D scene points that initially were created from the vehicle visiting the same area multiple times.

We use the DBoW2 implementation of the vocabulary tree to find similar images, and the P3P method (Kneip et al., 2011) together with RANSAC for geometric verification. We add a loop closure between two

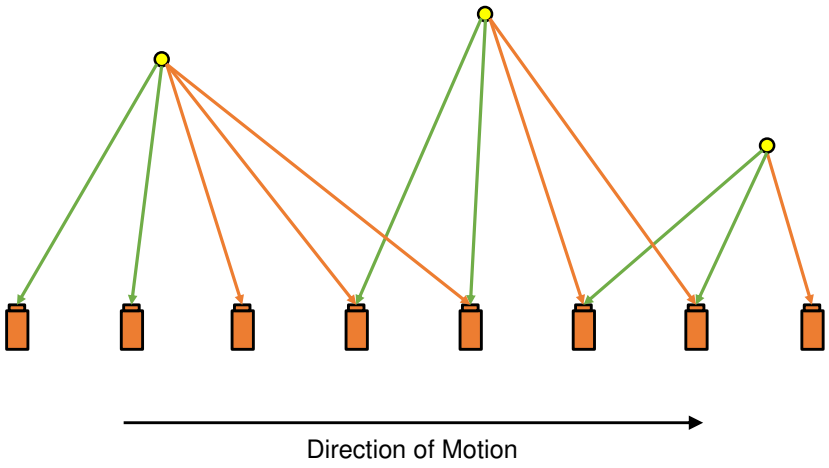


Figure 6: Yellow circles represent 3D scene points. An arrow between a camera and a 3D scene point indicates that the scene point is observed by the camera. Green arrows indicate that the corresponding feature observations are used to triangulate the associated scene point.

frames if the number of inlier 2D-3D feature correspondences exceeds a threshold. These two frames may belong to either the same camera or different cameras. Furthermore, we use information from the 2D-3D feature correspondences to merge duplicate 3D scene points after the loop closure is verified by the robust pose graph optimization step to be correct.

4.5 Robust Pose Graph Optimization

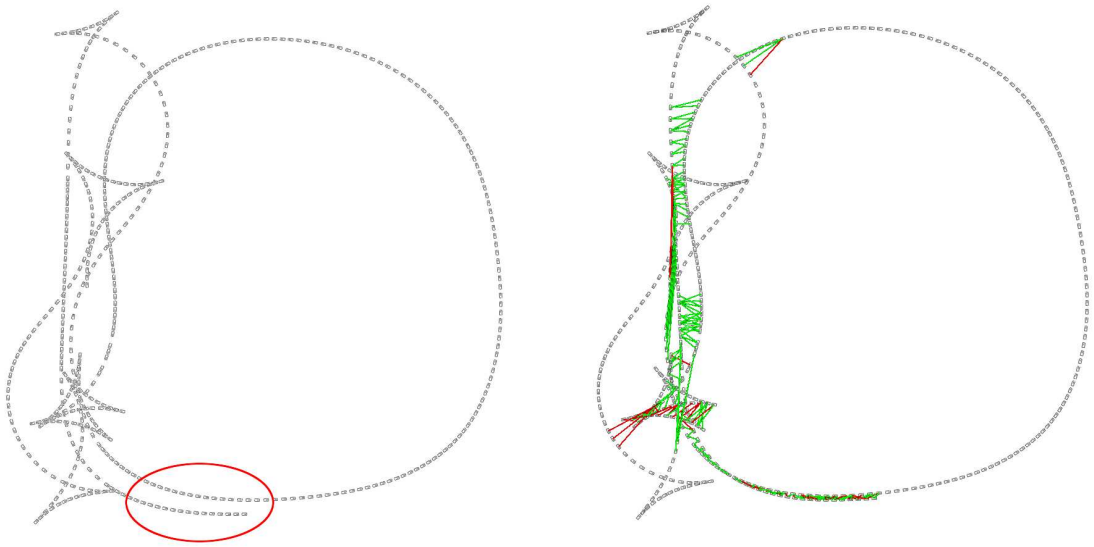
The robust pose graph optimization takes as input time-synchronized odometry, vehicle poses, and loop closure measurements computed by the loop closure detection step, and outputs the corrected vehicle poses.

We build a pose graph in which the nodes are the vehicle poses, and edges between nodes correspond to measurements of relative 6D transforms obtained from either odometry data or loop closures. We use the approach in Lee et al. (2013a) to optimize this pose graph, and at the same time, switch off false loop-closure edges. An example is shown in Figure 7. Figure 7a shows the initial pose graph, and Figure 7b shows the optimized pose graph with all loop closures edges identified as either correct or wrong. Here, we observe that the loop is closed correctly near the bottom of Figure 7b.

After the pose graph optimization, we rigidly move the scene points such that they are consistent with the new vehicle poses. At this point, the camera-odometry transforms are not sufficiently accurate for reprojection of feature points from one camera into another camera with sub-pixel reprojection errors. We solve this issue by finding local feature point correspondences between different cameras, and give details on this step in the next section.

4.6 Local Feature Matching Between Different Cameras

The local feature matching step reads as input vehicle poses computed from the robust pose graph optimization step, images from the multi-camera system, and 3D scene points computed from the 2nd scene point reconstruction step. The output is a set of 3D-3D correspondences between different cameras; as each 3D-3D correspondence corresponds to a pair of duplicate scene points, we merge these two scene points into a single scene point.



(a) Pose graph before optimization.

(b) Pose graph after robust optimization.

Figure 7: The pose of the vehicle at each time instance is marked with a black rectangle. In (a), the vehicle drives along the same road at two different times as marked by a red circle. Due to odometry drift, the corresponding trajectories do not coincide. After loop closures in this area, the trajectories are now aligned as shown in (b). In (b), loop-closure edges identified as correct by the pose graph optimization are marked as green while wrong loop closures are marked as red.

Although inter-camera feature correspondences are available from the loop closure detection step, it is typical for the image pair corresponding to an inter-camera feature correspondence to be captured far apart in time, and thus, such a feature correspondence acts as a very weak prior for the accurate estimation of relative camera poses. In contrast, the image pair corresponding to each inter-camera feature correspondence detected in this step is captured close in time, and thus, these feature correspondences act as strong priors.

Here, the cameras are assumed to have minimal overlapping fields of view at most. We iterate over each odometry pose in order of increasing timestamp, and maintain a local frame history for each camera. Note that the larger the local frame history, the longer time the local feature matching step takes. In our implementation, the local frame history for each camera spans a distance of 3 meters as measured from the vehicle poses associated with the frames in the local frame history.

At each odometry pose, for each possible pair of cameras, we do feature matching between the first camera's current frame and those in the second camera's frame history, and find the frame in the history that yields the highest number of inlier 2D-2D correspondences. Instead of using previously extracted features that correspond to existing 3D scene points, we extract and use features from rectified images. Before the feature matching, the image pair is rectified on a common image plane which corresponds to the average rotation between the first camera's pose and the second camera's pose; the camera poses are computed using the current extrinsic estimate and the odometry poses. An example of a rectified image pair between the right and front cameras is shown in Figure 8. This rectification step is to ensure a high number of inlier feature point correspondences. In contrast, we get a very low number of inlier feature point correspondences between images that are distorted and unrectified. These images inherently have large distortion in the case of fisheye cameras, and often, each pair of cameras has significantly different viewpoints. Hence, given a pair of feature descriptors corresponding to the same feature but to two different cameras, it is common for these feature descriptors to be dissimilar, and thus, be not identified as a valid feature correspondence.

For each feature point correspondence (f_1, f_2) obtained from a pair of rectified pinhole images, we find the

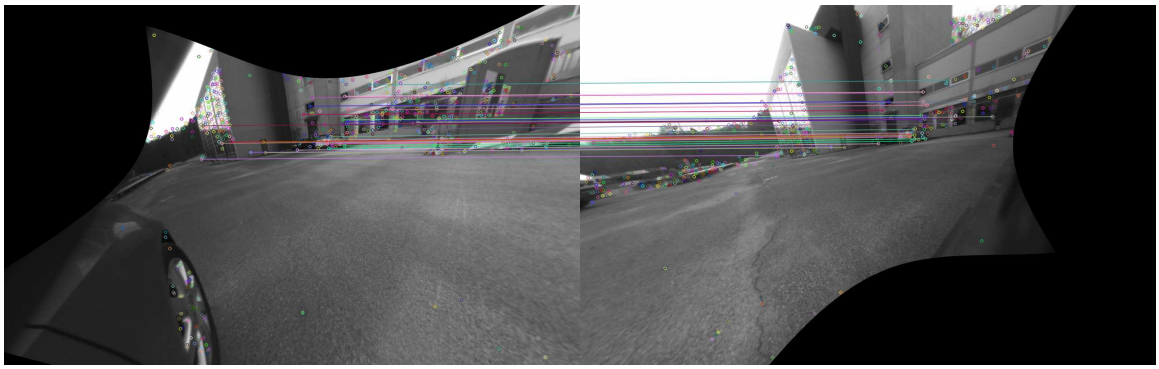


Figure 8: Inlier feature point correspondences between rectified pinhole images in the right and front cameras.



Figure 9: The subset of the same feature point correspondences from Figure 8 that correspond to 3D scene points in the maps is shown with the original images from the right and front cameras.

image coordinates of the two feature points: p_1 and p_2 in the corresponding original images. For feature point f_i where $i = 1, 2$, we look for a feature point g_i in the original image that is associated with a 3D scene point X_i in the map and whose image coordinates are within 1 pixel of p_i. If there is an associated 3D scene point for each of f_1 and f_2, and both scene points are different, we see one scene point as a duplicate of the other, and simply merge the two scene points. We show an example of inlier feature correspondences with associated 3D scene points in Figure 9.

Figure 10 shows a top-level illustration of how our local feature matching between different cameras works. Here, F_t is the current frame F_t captured by the front camera at time t. The local frame history for the right camera contains the last 5 frames captured by the right camera: $R_{t-4}, ..., R_t$. From these 5 frames, we find the frame that has the highest number of feature correspondences with F_t. As the camera pose corresponding to R_{t-2} has the closest viewing direction to the camera pose corresponding to F_t, R_{t-2} has the highest number of feature correspondences with F_t.

This step is critical to finding optimal camera-odometry transforms that allow us to project one scene point observed by one camera into another camera with minimal reprojection error.

4.7 Joint Optimization

We perform joint optimization that optimizes all intrinsics, camera-odometry transforms, vehicle poses, and 3D scene points. We note that the height of each camera with respect to the odometry frame is not observable due to the planar motion of the vehicle. However, this joint optimization recovers the relative heights of the cameras due to the use of feature correspondences between different cameras which are found from Section

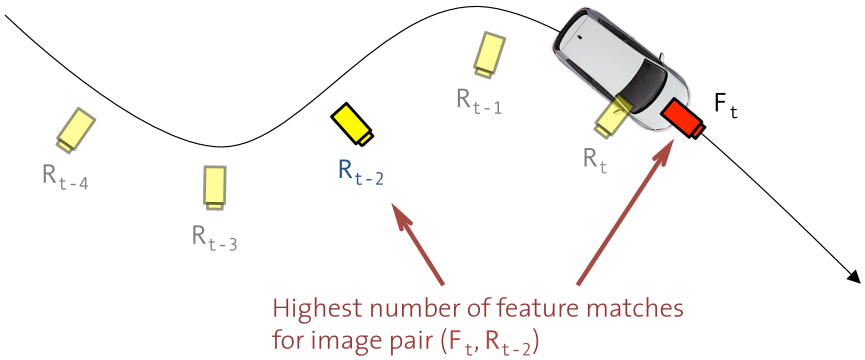


Figure 10: In this example, we do feature matching between the front camera's current frame F_t and those in the right camera's frame history, and find the frame in the history that yields the highest number of feature correspondences. Here, R_{t-2} has the highest number of feature correspondences with F_t since they have the closest viewing directions.

4.6.

The joint optimization is set up as a least squares minimization problem that minimizes a cost function. The cost function is designed to fulfil the following objectives:

1. The map should be accurate.
2. The pose of any camera with respect to all other cameras should be accurate.
3. Lines should be straight in rectified pinhole images.

Here, the cost function comprises three sets of inverse-covariance-weighted residuals:

$$\min_{\mathbf{C}_c,\mathbf{V}_i,\mathbf{Q}_j,\mathbf{T}_c,\mathbf{X}_p} \sum_{c,i,p} w_p \rho\left(\Delta\mathbf{z}_{c,i,p}{}^T \mathbf{W}_1 \Delta\mathbf{z}_{c,i,p}\right) + \sum_{c,j,q} \rho\left(\Delta\mathbf{z}_{c,j,q}{}^T \mathbf{W}_2 \Delta\mathbf{z}_{c,j,q}\right) + \sum_i \mathbf{u}_i{}^T \mathbf{W}_3 \mathbf{u}_i. \tag{14}$$

where

$$\begin{aligned} \Delta\mathbf{z}_{c,i,p} &= \boldsymbol{\pi}_1(\mathbf{C}_c, \mathbf{V}_i, \mathbf{T}_c, \mathbf{X}_p) - \mathbf{p}_{cip} \\ \Delta\mathbf{z}_{c,j,q} &= \boldsymbol{\pi}_2(\mathbf{C}_c, \mathbf{Q}_j, \mathbf{Y}_q) - \mathbf{p}_{cjq} \\ \Delta\mathbf{u}_i &= \mathbf{h}(\mathbf{V}_i, \mathbf{V}_{i+1}) - \mathbf{Z}_{i,i+1} \end{aligned} \tag{15}$$

As long as the residuals are inverse-covariance-weighted and assuming that the residuals have zero-mean Gaussian distributions, the joint optimization outputs a maximum likelihood estimate (Triggs et al., 2000).

$\boldsymbol{\pi}_1$ is a projection function that predicts the image coordinates of the scene point $\mathbf{X}_p$ seen in camera c given the camera's intrinsic parameters $\mathbf{C}_c$, the vehicle pose $\mathbf{V}_i$, and the rigid body transformation from the camera frame to the odometry frame $\mathbf{T}_c$. $\mathbf{p}_{cip}$ is the observed image coordinates of $\mathbf{X}_p$ seen in camera c with the corresponding vehicle pose $\mathbf{V}_i$. Similarly, $\boldsymbol{\pi}_2$ is a projection function that predicts the image coordinates of the chessboard corner point $\mathbf{Y}_q$ seen in camera c given the camera's intrinsic parameters $\mathbf{C}_c$, and the camera pose $\mathbf{Q}_j$. $\mathbf{p}_{cjq}$ is the observed image coordinates of $\mathbf{Y}_q$ seen in camera c whose pose is $\mathbf{Q}_j$. ρ is a robust cost function used for minimizing the influence of outliers. h is a function that computes the relative pose given two consecutive vehicle poses $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$, and $\mathbf{Z}_{i,i+1}$ is the observed relative pose between $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$ as measured from odometry data. We visualize the reference frames and variables in Figure 11.

In equation 14, the first set of residuals corresponds to the sum of image reprojection errors from all feature observations, the second set of residuals corresponds to the sum of image reprojection errors associated with

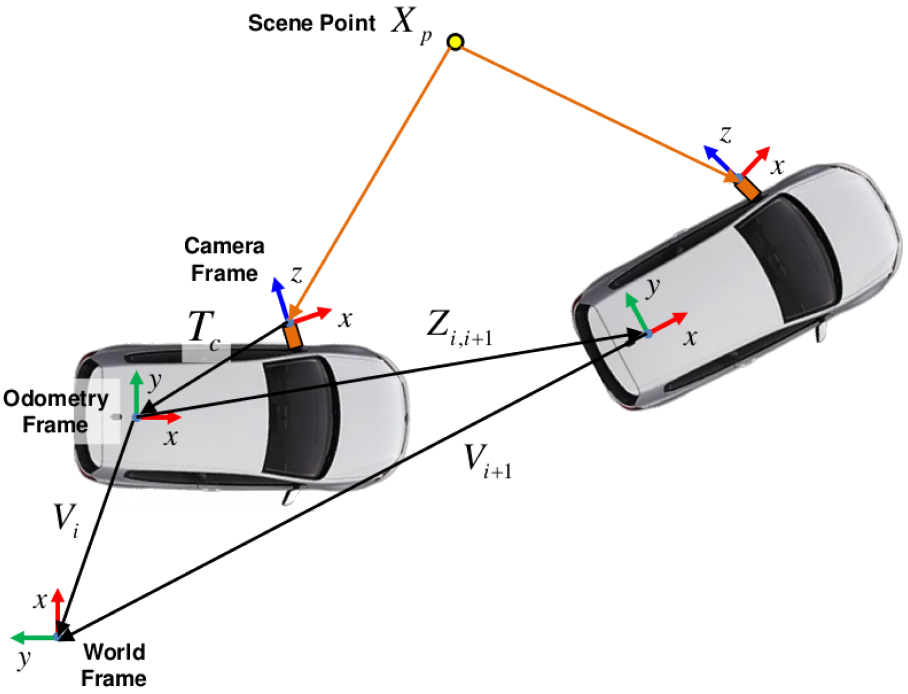
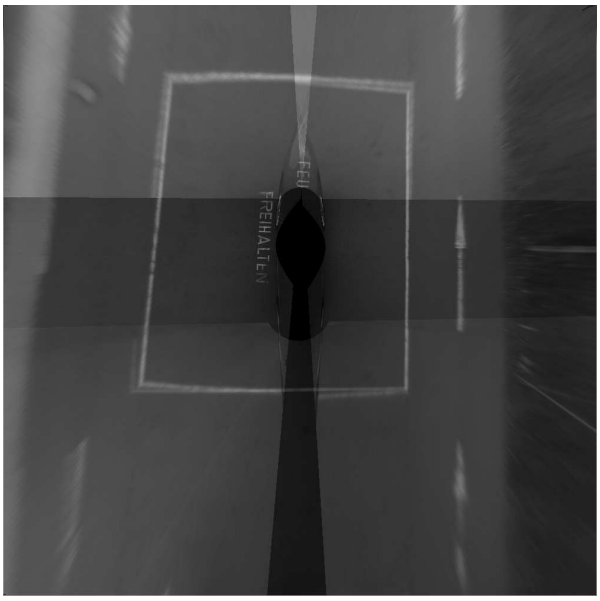


Figure 11: Visualization of the different reference frames and variables described in the joint optimization step.

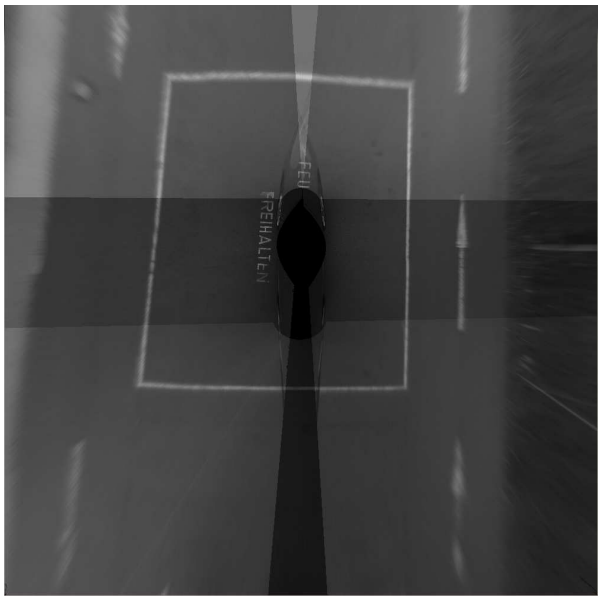
the chessboard corner points, and the third set of residuals corresponds to the sum of relative pose errors as measured by odometry data. Note that no measurement is used more than once.

w_p is a weight assigned to a feature observation associated with scene point $\mathbf{X}_p$. If $\mathbf{X}_p$ is observed by multiple cameras, we choose $w_p = 1$. Otherwise, if $\mathbf{X}_p$ is only observed by one camera, we choose $w_p = \frac{n_m}{n_s}$ where n_m is the number of feature observations associated with scene points observed by multiple cameras and n_s is the number of feature observations associated with scene points observed by a single cameras. This weighting scheme is chosen because scene points observed by multiple fractions usually form a small fraction of all scene points in the map, and the lower weighting of scene points seen by a single camera ensures that the relative camera poses are accurate. $\mathbf{W}_2$ is the inverse of the measurement covariance corresponding to the chessboard corner points; this measurement covariance is computed during the intrinsic camera calibration. Given the fact that feature detectors estimate keypoint coordinates with less accuracy compared to sub-pixel chessboard corner detectors, we obtain $\mathbf{W}_1$ by multiplying $\mathbf{W}_2$ by a fudge factor, which in our case, is 0.1. Similarly, $\mathbf{W}_3$ is the inverse of the measurement covariance corresponding to the relative odometry poses; we compute $\mathbf{W}_3$ using the odometry data and estimated vehicle poses from the robust pose graph optimization step described in Section 4.5.

The camera intrinsics are optimized in the joint optimization. The reason for this optimization is that the intrinsic camera calibration estimates sub-optimal values of the intrinsic parameters due to a biased distribution of chessboard poses and overfitting. The number of 3D scene points observed by the cameras in the extrinsic self-calibration is many times higher than the number of chessboard corner points observed in the intrinsic camera calibration, and by making use of the numerous scene points, we can improve the accuracy of the camera intrinsics in the joint optimization. Figure 12a shows the composite top-view image created from images from four cameras with the intrinsics originally estimated by the chessboard-based intrinsic camera calibration while Figure 12b shows the composite top-view image created with the optimized intrinsics. We clearly see a difference between the two; the composite top-view image created with the optimized intrinsics



(a) A composite top-view image created with original intrinsic parameters.



(b) A composite top-view image created with optimized intrinsic parameters.

Figure 12: Both composite images cover a 20 m × 20 m area. The four camera images blend seamlessly with one another in the composite image on the right, but not in the composite image on the left. This obvious visual difference proves that optimization of the intrinsic parameters in the joint optimization step significantly improves the accuracy of these parameters.

show seamless blending between the four camera images. This visual difference shows that the optimized intrinsics are more accurate than those estimated by the intrinsic camera calibration.

In experiments, we observed that overfitting of camera intrinsic parameters can occur, causing lines to look curved in rectified pinhole images. To avoid this situation, we include the residuals from the intrinsic camera calibration, and thus, enforce the constraint that lines should look straight in rectified pinhole images.

5 Infrastructure-Based Calibration

Given a map of the calibration area which is generated by the SLAM-based self-calibration method in the survey phase, the infrastructure-based calibration method leverages image-based localization to obtain camera poses, and in turn, estimates both the rig poses and camera extrinsics which are further optimized via non-linear refinement. We use a vocabulary tree for image-based localization; it is possible to use other state-of-the-art methods (Sattler et al., 2011; Lim et al., 2012) for image-based localization. In contrast to the SLAM-based self-calibration method, the infrastructure-based calibration method does not require a calibrated odometry system as the map has metric scale, and hence, enables us to directly infer the extrinsic parameters with metric scale via image-based localization.

One advantage of the infrastructure-based calibration method is that it runs in near real-time. In contrast, the self-calibration method in the survey phase and other SLAM-based calibration methods typically takes a few hours. Hence, the infrastructure-based calibration method allows us to perform many calibrations within a short time.

We assume that the cameras used in the infrastructure-based calibration have been intrinsically calibrated beforehand. The pipeline behind the infrastructure-based calibration method is shown in Figure 13. We

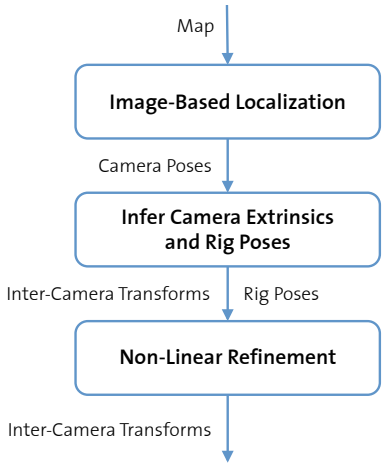


Figure 13: A map and images are input to the infrastructure-based calibration pipeline which generates the camera extrinsics.

describe the calibration process in detail below.

5.1 Image-Based Localization

The image-based localization step takes images from the multi-camera rig and the map as input, and outputs camera poses with respect to the map's reference frame. Image-based localization allows us to infer the camera poses for a given set of frames captured simultaneously from all cameras. First, for each image, we use the vocabulary tree to find the k most similar images. For each candidate, we obtain 2D-2D feature correspondences from matching features between the query image and the candidate, and rejecting probable false matches using the distance ratio test. As the feature points in the candidate image already have corresponding 3D scene points from the map, it is trivial to obtain 2D-3D feature correspondences. We use the P3P method (Kneip et al., 2011) together with RANSAC to find the camera pose together with the inlier set of 2D-3D feature correspondences. For the query image, we choose the camera pose associated with the candidate which has the highest number of inlier 2D-3D feature correspondences. The camera pose is defined to be unknown if the highest number of inlier correspondences does not exceed a threshold, which in our case, is 25. We denote the number of cameras on the multi-camera rig as n. For a particular time step t, we store the set of camera poses $\mathcal{S}_t$ if the following two conditions are met:

1. if at least 2 camera poses are found such that $2 \leq |\mathcal{S}_t| \leq n$, and
2. the minimum distance between the current camera pose and the previous camera pose over all cameras exceeds a threshold, which in our case, is 0.3 m.

The former condition is necessary for the set of camera poses to be useful for calibration, and the latter condition minimizes bias by avoiding situations where the majority of sets of camera poses is concentrated in a few locations. These situations occur when the car is at a standstill or moving slowly. For brevity, we denote the collection that contains all sets of camera poses meeting the two conditions as $\mathcal{W}$.

5.2 Inferring Camera Extrinsics and Rig Poses

In this step, we use $\mathcal{W}$, the collection of sets of camera poses inferred from image-based localization, to obtain an initial estimate for the camera-rig transforms and the rig poses. Since the cameras are rigidly fixed to the

rig, for $\mathcal{S}_t \in \mathcal{W}$, we can express $\mathcal{S}_t$, the set of camera poses, as a function of the rig pose and the camera-rig transforms. Here, the rig pose is the pose of the rig with respect to the map's reference frame, and each camera-rig transform is a rigid body transform from the camera's reference frame to the rig's reference frame. Without loss of generality, we choose the reference frame of the rig to coincide with that of a designated camera that makes up part of the multi-camera rig to be calibrated.

From $\mathcal{W}$, we generate hypotheses of the camera-rig transforms and rig poses. We choose the best hypothesis that minimizes the sum of reprojection errors over all inlier 2D-3D feature correspondences which were used to infer the camera poses in $\mathcal{W}$. This hypothesis gives us the initial estimate of the camera-rig transforms and rig poses. Algorithm 1 provides details on how we infer this initial estimate from $\mathcal{W}$. In the next step, we optimize the initial estimate via non-linear refinement.

Algorithm 1: Algorithm for obtaining an initial estimate of camera-rig transforms and rig poses from $\mathcal{W}$.

input : $\mathcal{W}$

output : Initial estimate of camera-rig transforms and rig poses

$\mathcal{A} = \emptyset$

for $\mathcal{S}_t \in \mathcal{W}$ *such that* $|\mathcal{S}_t| = n$ **do**

- compute a hypothesis of camera-rig transforms $\{\mathbf{H}_1, ..., \mathbf{H}_n\}$ from $\mathcal{S}_t$ where $\mathbf{H}_c$ is the rigid body transformation from camera c to the rig's reference frame
- create an empty hypothesis of rig poses
- **for** $\mathcal{T}_u \in W$ **do**
  - compute $|\mathcal{T}_u|$ estimates of the rig pose from $\mathcal{T}_u$ and $\{\mathbf{H}_1, ..., \mathbf{H}_n\}$
  - use the $|\mathcal{T}_u|$ estimates to obtain an average estimate of the rig pose $\mathbf{P}_u$ at time step u by using:
    1. quaternion averaging (Markley et al., 2007) to obtain the average rotation, and
    2. simple averaging to obtain the average translation
  - add $\mathbf{P}_u$ to the hypothesis of rig poses
- add the hypothesis of camera-rig transforms and rig poses to $\mathcal{A}$

select from $\mathcal{A}$ the best hypothesis of camera-rig transforms and rig poses that minimizes the sum of reprojection errors over all inlier 2D-3D feature correspondences used to infer the camera poses in $\mathcal{W}$

5.3 Non-Linear Refinement

In this step, we minimize the sum of all reprojection errors by optimizing the camera-rig transforms and the rig poses while keeping the coordinates of all 3D scene points fixed.

Formally, we solve the optimization problem

$$\min_{\mathbf{P}_t, \mathbf{T}_c} \sum_{c,t,p} \rho\left(||\boldsymbol{\pi}(\mathbf{C}_c, \mathbf{P}_t, \mathbf{H}_c, \mathbf{X}_p) - \mathbf{p}_{ctp}||^2\right). \tag{16}$$

$\boldsymbol{\pi}$ is a projection function that predicts the image coordinates of the scene point $\mathbf{X}_p$ seen in camera c given the camera's intrinsic parameters $\mathbf{C}_c$, the rig pose $\mathbf{P}_t$, and the rigid body transformation from the camera to the rig's reference frame $\mathbf{H}_c$. $\mathbf{p}_{ctp}$ is the observed image coordinates of $\mathbf{X}_p$ seen in camera c with the corresponding rig pose $\mathbf{P}_t$. ρ is a robust cost function used for minimizing the influence of outliers. We use the Cauchy cost function in this case. We visualize the reference frames and variables in Figure 14.

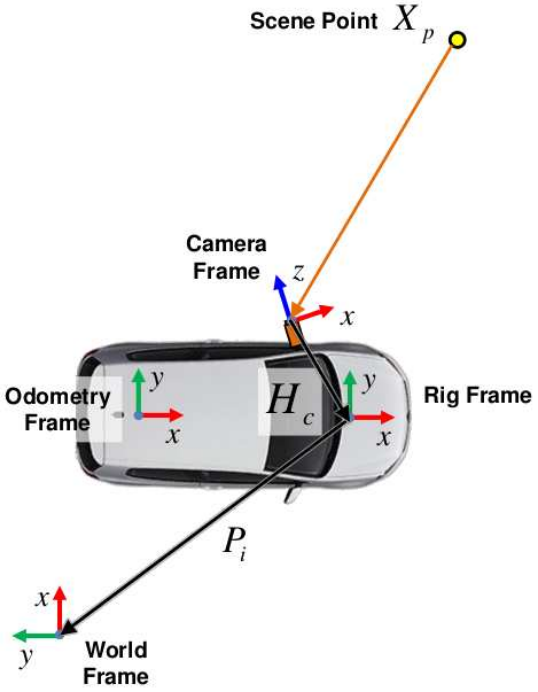


Figure 14: Visualization of the different reference frames and variables described in the non-linear refinement step.

6 Experiments and Results

This section presents a series of experiments to validate our self-calibration method (Section 6.1) and our infrastructure-based calibration method (Section 6.2).

6.1 Self-Calibration

This section presents a variety of experiments to validate the accuracy and repeatability of the calibration parameters estimated by our self-calibration method. Furthermore, we perform additional experiments to validate the metric accuracy and global consistency of the map produced by the self-calibration method.

The first experiment measures the accuracy of the self-calibration method. In this experiment, we run an AR-marker-based extrinsic calibration to provide ground truth data. We compare the results from our self-calibration method against those from the AR-marker-based extrinsic calibration. The second experiment measures the repeatability of the results from the self-calibration method over multiple loops in the same area, and aims to show that the self-calibration method provides stable results. The third experiment evaluates the accuracy of the vehicle poses estimated by our self-calibration method based on the use of GPS/INS measurements as ground truth, and thus, the metric accuracy and global consistency of the map. The fourth experiment evaluates the accuracy of the map based on a visual comparison with aerial imagery.

On an 2.80 GHz Intel Core i7 PC, our self-calibration pipeline takes approximately 3 hours to compute the calibration parameters using 500 frames for each camera; the majority of the time is spent on the local feature matching and joint optimization steps.

Table 2: Kermit: Estimated angles in degrees between the front camera (reference camera) and the other three cameras.

	Our self-calibration	**AR-marker-based calibration**	**Difference**
Left Camera	87.6419°	88.3278°	0.6859°
Rear Camera	177.0829°	177.2843°	0.2014°
Right Camera	89.2839°	89.6838°	0.3999°

Table 3: Grobi: Estimated angles in degrees between the front camera (reference camera) and the other three cameras.

	Our self-calibration	**AR-marker-based calibration**	**Difference**
Left Camera	93.1446°	92.5857°	0.5589°
Rear Camera	168.0953°	168.5607°	0.4654°
Right Camera	91.0565°	90.8517°	0.2048°

6.1.1 Experiment 1 - Accuracy of Calibration Parameter Estimates

We compare our extrinsic calibration results against those generated by an AR-marker-based extrinsic calibration. In this AR-marker-based calibration, three different types of AR markers are placed around the vehicle: camera markers that are visible in at least one camera, floor markers that are placed on the ground around the car, and wheel markers that are placed on the rear wheels to help identify the nominal odometry frame. The cameras on the vehicle together with a calibrated handheld camera capture images of the markers. Subsequently, the markers are detected in all images, and a bundle adjustment is run to optimize the poses of the markers and cameras. At the end of the bundle adjustment, the poses of the cameras are extracted with respect to the nominal odometry frame; the locations of the floor markers define the ground plane, and the wheel markers define the rear axis of the vehicle.

We compare the results for two different Volkswagen Golf cars which we name Kermit and Grobi respectively. For both cars, we tabulate in Tables 2 and 3 the estimated angles between the front camera and the other three cameras for both our estimated extrinsics and the AR-marker-based extrinsics. Similarly, we tabulate in Tables 4 and 5 the unit estimated translations between the front camera and the other three cameras to ignore the effect of scale. For Kermit, we show a visual comparison of the final estimated extrinsics with the extrinsics computed by the AR-marker-based calibration in Figure 15. A green sphere marks the origin of the odometry frame. We observe that for both cars, both sets of extrinsics estimated by our self-calibration method and the AR-marker-based calibration almost coincide; the scale of the extrinsics estimated by our self-calibration method is 2.7% and 0.8% smaller on average for Kermit and Grobi respectively.

Table 4: Kermit: Estimated unit translations in meters between the front camera (reference camera) and the other three cameras.

	Our self-calibration	**AR-marker-based calibration**	**Difference**
Left Camera	$\begin{bmatrix} -0.5464 \\ -0.2159 \\ -0.8092 \end{bmatrix}$ m	$\begin{bmatrix} -0.5502 \\ -0.2420 \\ -0.7992 \end{bmatrix}$ m	$\begin{bmatrix} 0.0038 \\ 0.0261 \\ 0.0100 \end{bmatrix}$ m
Rear Camera	$\begin{bmatrix} -0.0074 \\ -0.1070 \\ -0.9942 \end{bmatrix}$ m	$\begin{bmatrix} -0.0160 \\ -0.1343 \\ -0.9908 \end{bmatrix}$ m	$\begin{bmatrix} 0.0086 \\ 0.0273 \\ 0.0034 \end{bmatrix}$ m
Right Camera	$\begin{bmatrix} 0.5306 \\ -0.2261 \\ -0.8169 \end{bmatrix}$ m	$\begin{bmatrix} 0.5274 \\ -0.2454 \\ -0.8134 \end{bmatrix}$ m	$\begin{bmatrix} 0.0032 \\ 0.0193 \\ 0.0035 \end{bmatrix}$ m

Table 5: Grobi: Estimated unit translations in meters between the front camera (reference camera) and the other three cameras.

	Our self-calibration	**AR-marker-based calibration**	**Difference**
Left Camera	$\begin{bmatrix} -0.5059 \\ -0.3085 \\ -0.8055 \end{bmatrix}$ m	$\begin{bmatrix} -0.5273 \\ -0.2906 \\ -0.7985 \end{bmatrix}$ m	$\begin{bmatrix} 0.0214 \\ 0.0179 \\ 0.0070 \end{bmatrix}$ m
Rear Camera	$\begin{bmatrix} 0.0372 \\ -0.1835 \\ -0.9823 \end{bmatrix}$ m	$\begin{bmatrix} 0.0145 \\ -0.1788 \\ -0.9838 \end{bmatrix}$ m	$\begin{bmatrix} 0.0227 \\ 0.0047 \\ 0.0015 \end{bmatrix}$ m
Right Camera	$\begin{bmatrix} 0.5732 \\ -0.2614 \\ -0.7766 \end{bmatrix}$ m	$\begin{bmatrix} 0.5513 \\ -0.2804 \\ -0.7858 \end{bmatrix}$ m	$\begin{bmatrix} 0.0219 \\ 0.0190 \\ 0.0092 \end{bmatrix}$ m

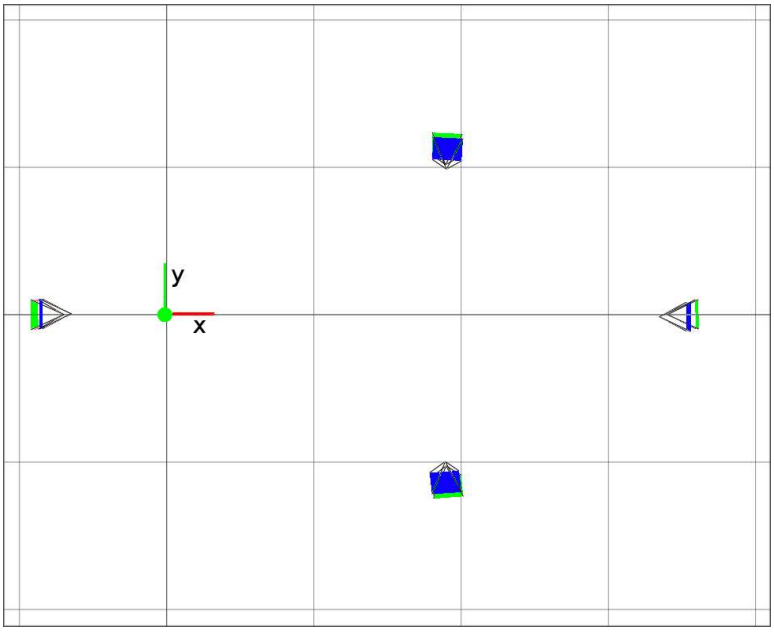
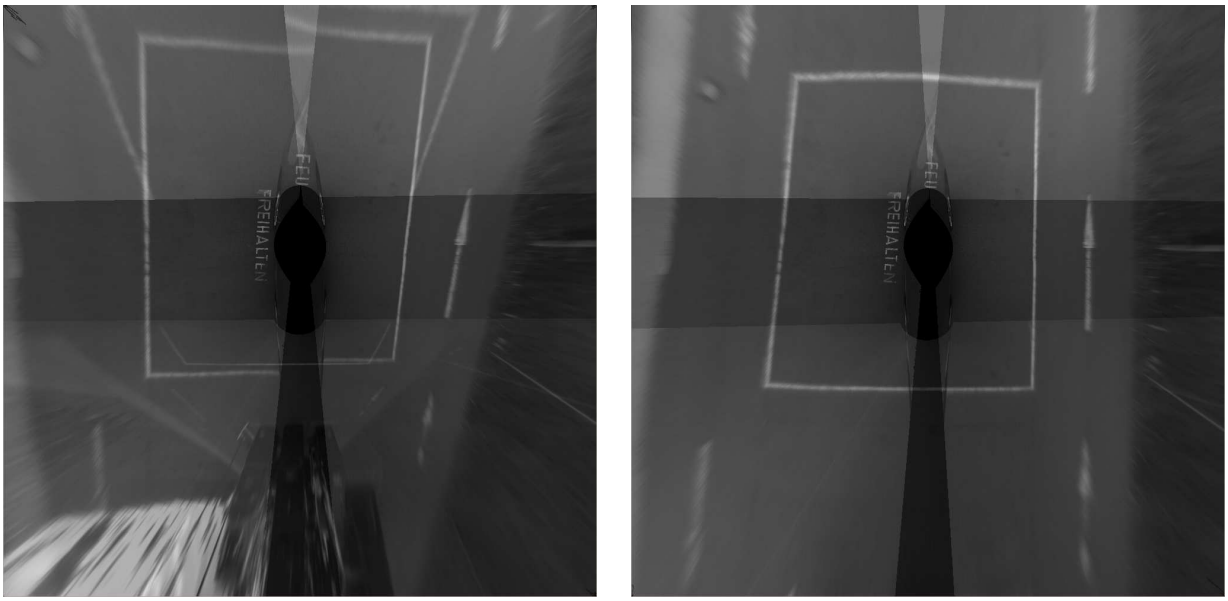


Figure 15: Comparison of extrinsics generated by our pipeline and the AR-marker-based calibration method for Kermit. The green sphere marks the origin of the odometry frame while the green and red axes represent the x-axis and y-axis of the odometry frame respectively. The cameras corresponding to the extrinsics estimated by our self-calibration method are colored blue while those corresponding to the AR-marker-based extrinsics are colored green. The grid resolution is 1 m.

We are unable to compare our approach with Carrera et al. (2011) as their 3D similarity transform step fails in our experimental settings. The reason is that in outdoor environments, a vast majority of scene points are usually located far away from the cameras, and thus, the estimates of their 3D coordinates have significant noise.

Calibration algorithms are notoriously difficult to evaluate because of the difficulty of obtaining ground truth values for the parameters being estimated. Hence, we resort to evaluating our method by demonstrating the enhanced performance of algorithms that use the calibration parameters. We argue that the extrinsics estimated by our self-calibration method are more accurate than those estimated by the AR-marker-based extrinsic calibration. We propose two qualitative tests to support our argument: creating a composite top-view image that is formed by projecting the image from each camera onto the ground plane using the given calibration parameters, and performing motion stereo based on the plane sweep algorithm which takes three consecutive images and corresponding odometry poses and computes a depth map based on these three images. In the first test, the calibration parameters are assumed to be perfect if images from all cameras blend seamlessly. Similarly, in the second test, the calibration parameters are assumed to be accurate if the computed depth map from motion stereo is dense and accurate.

Composite Top-View Image: We create a composite top-view image by using the calibration parameters to project the image from each of the four cameras onto the ground, and merging the resulting 4 images via alpha blending. The image covers a 20 m $\times$ 20 m area with the vehicle's odometry frame located at the center of the image. Figure 16 shows the composite top-view images created with both sets of calibration parameters. Seamless blending is observed in the image that corresponds to our self-calibration method, while no seamless blending is observed at all in the image that corresponds to the AR-marker-based extrinsic calibration. This visual difference shows that our self-calibration method produces very accurate calibration parameters.



(a) A composite top-view image created with calibration parameters estimated by the AR-marker-based calibration.

(b) A composite top-view image created with calibration parameters estimated by the self-calibration.

Figure 16: Both composite top-view images are of a 20 m $\times$ 20 m area centered on the car. The four camera images blend seamlessly with one another in the composite image on the right, but not in the composite image on the left. This visual difference shows that our self-calibration method produces very accurate calibration parameters.

Motion Stereo: We perform an experiment to show that the estimated calibration parameters are more accurate than that of the AR-marker-based parameters; we use a multi-view plane-sweep-based stereo algo-

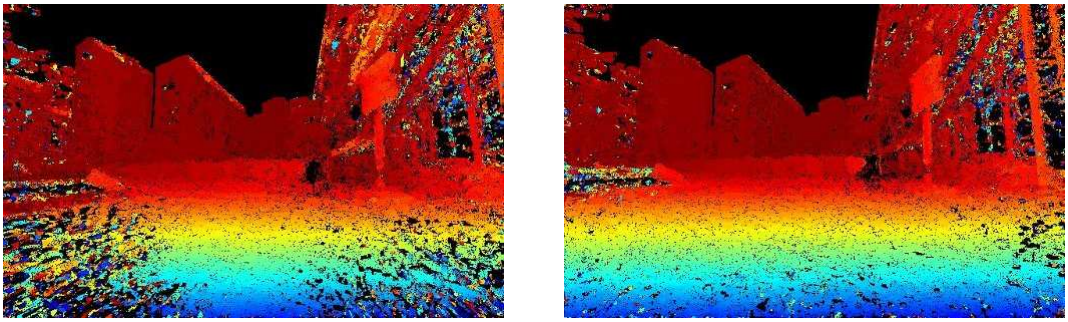


Figure 17: 3-view plane sweep stereo with the front camera. Forward camera motion is challenging for motion stereo. The left depth map is computed using the AR-marker-based parameters while the right depth map is computed using our estimated parameters.

Table 6: Mean and standard deviation of the rotation and translation vectors for each camera.

	Rotation Vector			**Translation Vector**		
Left Camera	$\begin{bmatrix} -0.0155 \\ -1.4958 \\ -0.0116 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0007 \\ 0.0047 \\ 0.0007 \end{bmatrix}$	$\begin{bmatrix} -0.6026 \\ -0.1225 \\ -1.3954 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0043 \\ 0.0012 \\ 0.0033 \end{bmatrix}$
Rear Camera	$\begin{bmatrix} -0.0063 \\ 3.0932 \\ 0.0435 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0002 \\ 0.0034 \\ 0.0018 \end{bmatrix}$	$\begin{bmatrix} 0.1065 \\ -0.0024 \\ -1.6114 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0039 \\ 0.0013 \\ 0.0034 \end{bmatrix}$
Right Camera	$\begin{bmatrix} 0.0103 \\ 1.5602 \\ -0.0143 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0006 \\ 0.0056 \\ 0.0008 \end{bmatrix}$	$\begin{bmatrix} 0.7073 \\ -0.0060 \\ -1.3041 \end{bmatrix}$	$\pm$	$\begin{bmatrix} 0.0043 \\ 0.0012 \\ 0.0027 \end{bmatrix}$

rithm (Yang and Pollefeys, 2003) with semi-global block matching to compute dense depth maps from rectified pinhole images for each camera using only the odometry poses and calibration parameters without any camera pose optimization. The plane sweep is extremely sensitive to the accuracy of the calibration parameters; we compare the depth maps computed by the plane sweep using the calibration parameters estimated by our self-calibration method with the depth maps computed with the calibration parameters generated by the AR-marker-based calibration. We show in Figure 17 that the use of our estimated parameters results in a more accurate dense map especially on the ground compared to the use of the AR-marker-based parameters. This result demonstrates the higher accuracy of our estimated parameters, and can be attributed to the fact that our pipeline explicitly makes use of odometry data to estimate the camera-odometry transforms, and makes use of a high number of features over a wide range of depths to ensure optimal accuracy.

6.1.2 Experiment 2 - Repeatability of Calibration Parameter Estimates

This experiment looks at the repeatability of the camera extrinsics estimated by our self-calibration method. We drive 25 loops with the same camera configuration in a slowly changing environment in which cars and pedestrians move around. For each loop, we use the same set of camera intrinsics, and estimate the camera-odometry transforms. In turn, for each camera, we compute the rotation and translation vectors that correspond to its pose with respect to the front camera. We tabulate the means and standard deviations of the rotation and translation vectors for each camera in Table 6. In addition, we illustrate the distribution of the rotation and translation vectors over the 25 loops in the form of a box plot in Figure 18. For ease of comparison, we subtract the vectors from their means in this box plot.

We note that the standard deviations of the rotation and translation vectors are very small, and therefore, conclude that the extrinsics estimated by our self-calibration method are highly repeatable.

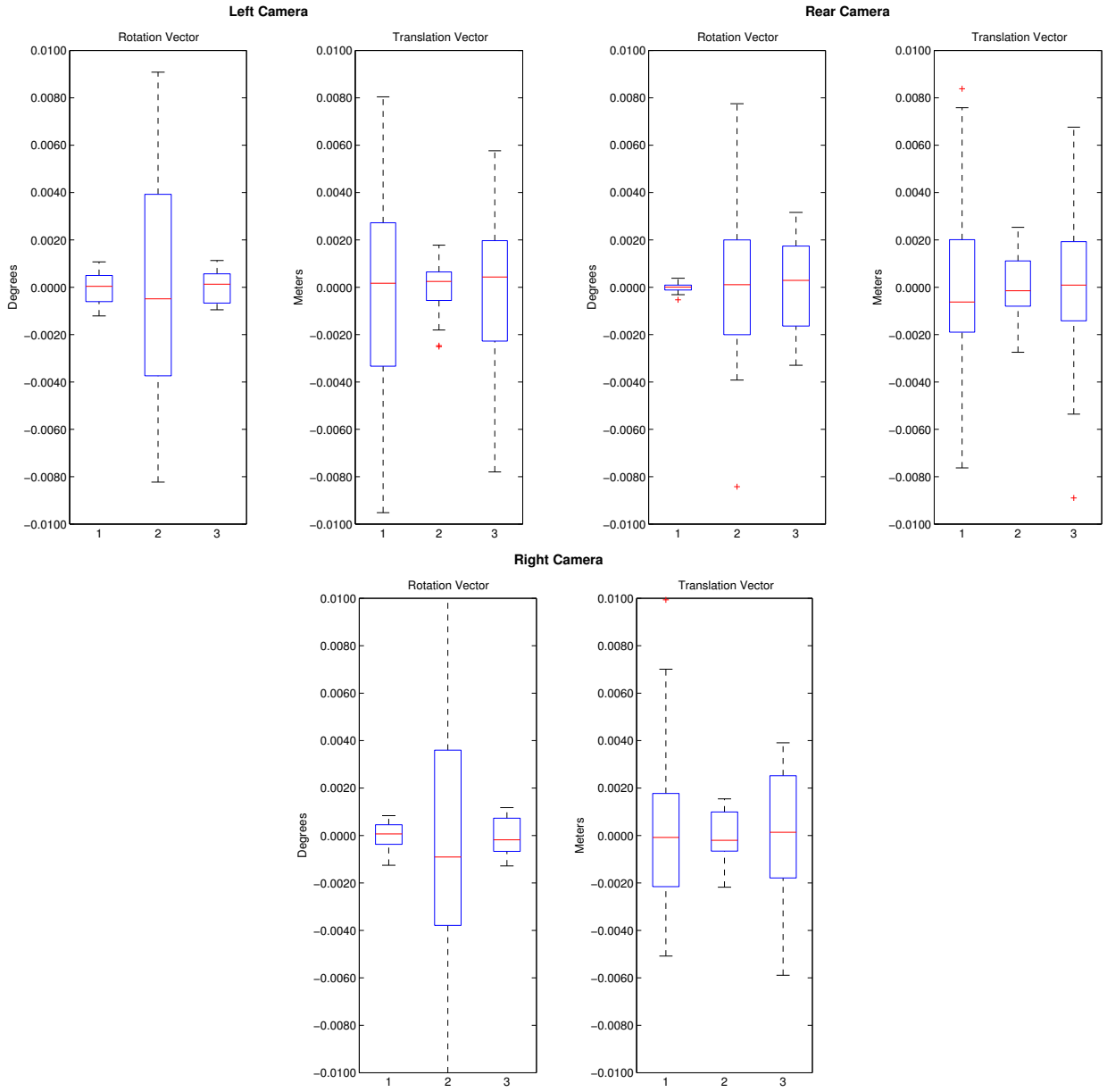


Figure 18: A box plot of the rotation and translation vectors about the mean for the left, rear, and right cameras with respect to the front camera.

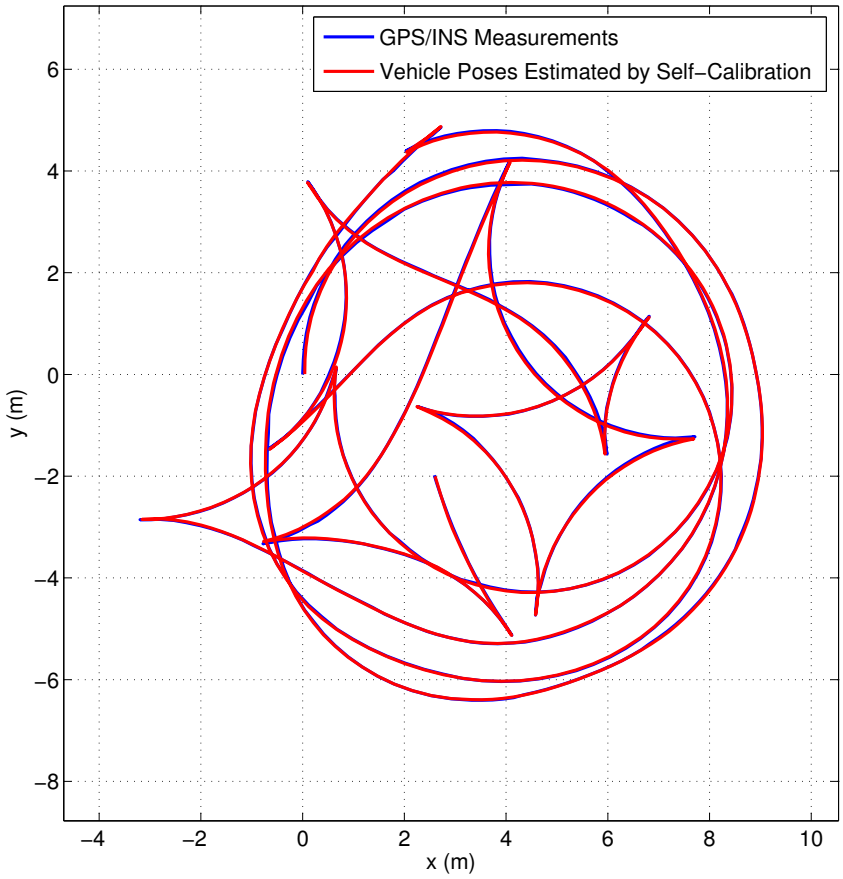


Figure 19: A top-view plot of the vehicle poses (red) estimated by our self-calibration method together with corresponding GPS/INS measurements (blue).

6.1.3 Experiment 3 - Pose Accuracy

In this experiment, we perform a quantitative evaluation of the accuracy of the vehicle poses estimated by our self-calibration method. We use 6-DoF pose measurements from a GPS/INS system as ground truth.

We use our self-calibration method to generate a map from a 180-meter trajectory in an outdoor environment. 652 sets of synchronized images from all cameras were used in the self-calibration, and in turn, 652 vehicle poses were estimated. We show a top-view plot of the vehicle poses together with the corresponding GPS/INS measurements in Figure 19. The vehicle poses and GPS/INS measurements are marked as red and blue respectively.

Based on the GPS/INS measurements, the vehicle poses estimated by our self-calibration method have an average rotation and translation error of 0.140° and 3.51 cm respectively. These low errors show that the estimated poses are accurate, and in turn, show that our self-calibration method produces a metrically accurate and globally consistent map. This global accuracy is extremely important for our infrastructure-based calibration method as it gives the method the ability to estimate accurate extrinsics for a multi-camera rig.

6.1.4 Experiment 4 - Metric Map Accuracy

We use our self-calibration method to generate a map from a 308-meter trajectory. This map has 885734 2D feature points and 197836 3D scene points, and an average reprojection error of 0.37 pixels. In Figure 20, we show this map together with its predecessor which was generated at the beginning of the pipeline with intrinsics estimated from the intrinsic camera calibration, and extrinsics estimated from the camera-odometry calibration step. In the visualized map, each 3D scene point is colored according to the camera it was first observed in. 3D scene points observed by the front, left, rear, and right cameras are colored red, green, blue, and yellow respectively.

We observe in Figure 20 that the optimized intrinsics and extrinsics correspond to a more clearly-defined map. This observation shows that the optimized parameters are more accurate than the initial parameters. Furthermore, in this figure, we also show the map overlaid on aerial imagery which is manually aligned. This overlay shows that the map aligns well with the aerial imagery on a global scale, and therefore, is metrically accurate.

6.2 Infrastructure-Based Calibration

This section presents an evaluation of our infrastructure-based calibration method. We perform experiments in both an indoor parking garage and outdoor urban environment on the ETH campus. Figure 21 shows images of both areas taken by the front camera. We design our experiments to demonstrate that our infrastructure-based calibration exhibits a high level of accuracy in both indoor and outdoor environments in the presence of moving cars and pedestrians.

Before we conduct the experiments, we run the survey phase once to build a map for both areas as shown in Figures 20 and 22a. To determine the accuracy of our estimated camera extrinsics, we compare the estimated camera extrinsics against those estimated by the self-calibration method in the survey phase. We first compute the pose of each camera with respect to the first camera for both sets of extrinsics. Then, we use these relative poses to compute the rotation error and two types of translation errors: the angle between the two translation vectors, and the norm of the difference between the two translation vectors. These three error metrics are used to give a quantitative measure of the accuracy of the camera extrinsics estimated by our infrastructure-based calibration method.

On average, it takes 0.5 seconds to estimate the camera poses from each set of images assuming that each camera has its own dedicated thread which processes all images coming from that camera. Computing the initial estimate of the camera extrinsics together with non-linear refinement typically takes no more than 5 seconds.

6.2.1 Experiment - Indoor Parking Garage

In this experiment, a vehicle is driven along one loop with the same camera configuration that was used to generate the map. This loop trajectory differs from that taken by the vehicle during the data collection for building the map. This experiment aims to show that our estimated extrinsics and those estimated by the self-calibration method are the same.

Figure 22a shows the map generated by the self-calibration method that is used in the infrastructure-based calibration. Figure 22b shows the subset of scene points that are from the map and used for the calibration. Furthermore, Figure 22b shows the estimated camera poses which are used to infer the camera extrinsics.

We tabulate in Table 7 the three error metrics for our estimated extrinsics, using the extrinsics estimated by the self-calibration method as ground truth. The infrastructure-based calibration estimated the extrinsics from 167 sets of camera poses with an average of 3.05 camera poses per set. A total of 37860 2D-3D

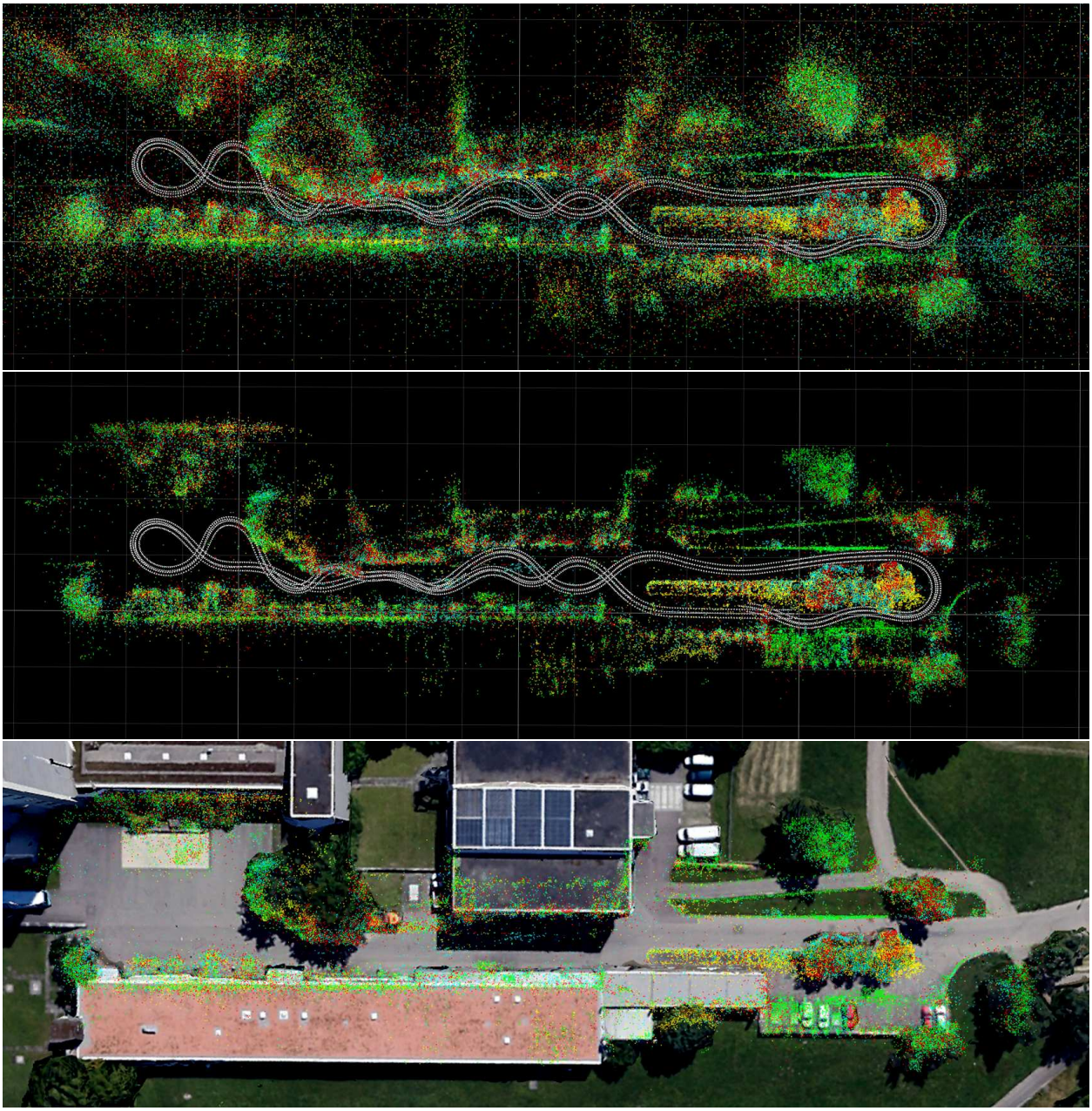
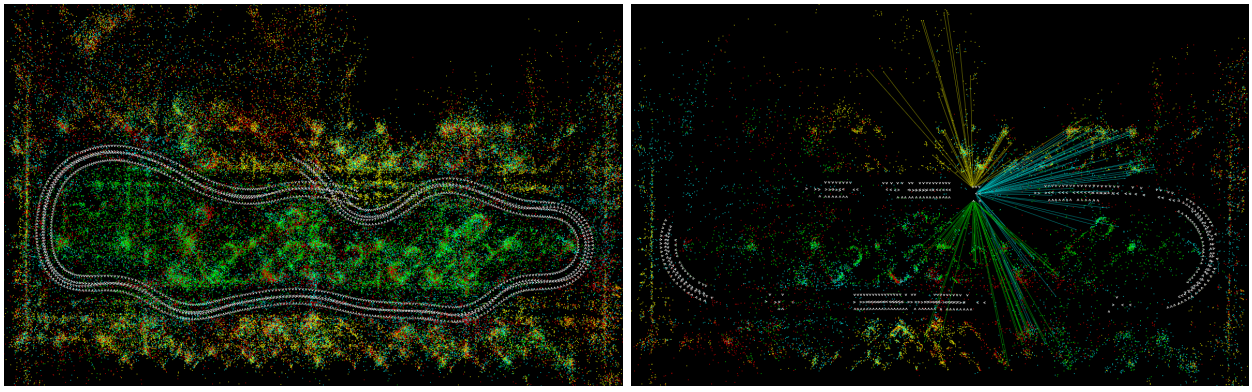


Figure 20: Each 3D scene point is colored based on the camera it was first observed in. The distance between consecutive major grid lines marked by thick white lines is 50 meters. The distance between consecutive minor grid lines marked by thin white lines is 10 meters. (a) The map with intrinsics estimated from the intrinsic camera calibration, and extrinsics estimated from the camera-odometry calibration step. (b) The map with optimized intrinsics and extrinsics. (c) The map is overlaid on aerial imagery.



(a) (b)

Figure 21: (a) Indoor parking garage and (b) outdoor urban environment as seen from the front camera



(a) (b)

Figure 22: An experiment in an indoor parking garage as shown in Figure 21a. (a) The map generated by the self-calibration method. The camera poses marked in white illustrate the twisting path taken by the car. (b) The subset of scene points from the map and which is used for infrastructure-based calibration is shown. The camera poses inferred from PnP are marked as white triangles. Here, the car approaches the end of a loop, and the current set of camera poses are derived from 2D-3D correspondences visualized as lines from the camera poses to the scene points.

Table 7: Indoor experiment: error metrics for the extrinsics estimated by our infrastructure-based calibration method.

	Left Camera	**Rear Camera**	**Right Camera**
Rotation error (deg)	0.0044	0.0032	0.0088
Translation error (deg)	0.0563	0.0468	0.0349
Translation error (m)	0.0016	0.0022	0.0021



Figure 23: The left camera is mounted on a three-way tripod head, which in turn, is mounted on a sliding plate. This sliding plate is attached to the car roof via screws.

correspondences were used. The initial estimates of the extrinsics and rig poses had an associated average reprojection error of 0.99 pixels which reduced to 0.69 pixels after non-linear refinement. It is observed from the results in Table 7 that the extrinsics estimated by our method are virtually the same as those estimated by the self-calibration method.

6.2.2 Experiments - Outdoor Urban Environment

We run a total of four experiments. In each of the first three experiments, the Prius is driven in one loop in the same scene and with a different camera configuration. As in the indoor experiment, this loop trajectory differs from that taken by the Prius during the data collection for building the map. These three experiments aim to show that our calibration can reliably estimate the camera extrinsics for a camera configuration different from that used for building the map. For the fourth experiment, we drive the Prius over 25 loops for 1 hour with the same camera configuration that was used to generate the map. This experiment shows the impact of a changing environment on the calibration accuracy; during this one hour, cars and pedestrians continually move around, and plants and trees sway significantly in moderate wind conditions. From odometry measurements, the average distance of each loop in all experiments is 308 m.

We use a three-way tripod head and a sliding plate in Figure 23 to ensure that camera configuration changes can be measured as precisely as possible.

Experiment 1: The left and right cameras are moved towards the front of the car by 10.0 cm as measured with a ruler. The infrastructure-based calibration used 418 sets of camera poses with an average of 3.26 camera poses per set. Based on the results from our method, the left and right cameras are deemed to have moved 10.08 cm and 10.36 cm respectively. These estimates closely agree with the hand measurements.

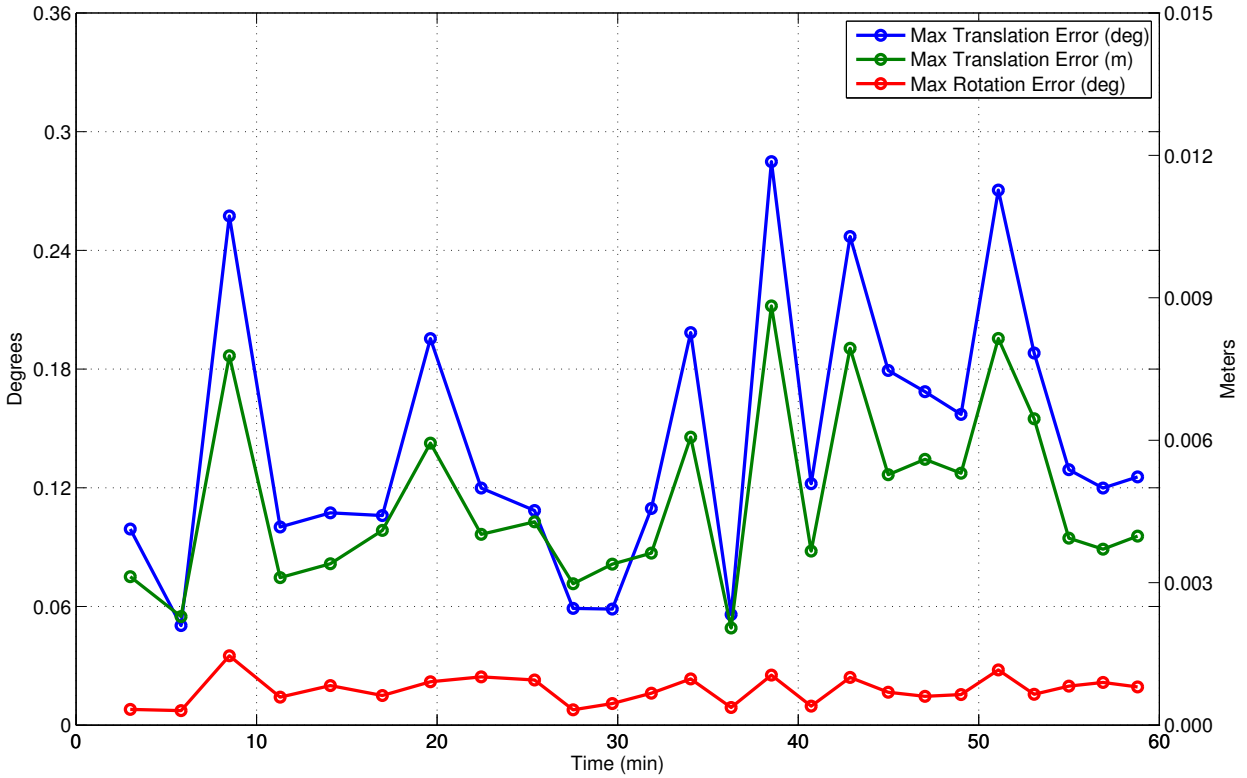


Figure 24: A plot of the maximum errors against the time at which each loop was completed.

Experiment 2 The left camera is rotated around its x-axis towards the ground by 30° as measured with a scale that is available for each axis of movement of the three-way tripod head. The infrastructure-based calibration used 411 sets of camera poses with an average of 3.14 camera poses per set. Our method estimates the left camera to have rotated about its x-axis by 29.9°. This estimate closely agrees with the hand measurement.

Experiment 3 The left camera is rotated around its z-axis towards the front of the car by 15°. The infrastructure-based calibration used 381 sets of camera poses with an average of 3.20 camera poses per set. Our method estimates the left camera to have rotated around its x-axis by 14.3°. This estimate closely agrees with the hand measurements.

Experiment 4 For each of the 25 loops, we estimate the extrinsics, and find the maximum of the three error metrics among all cameras. We then plot in Figure 24 these maximum errors against the time at which each loop was completed. Furthermore, we use a box plot in Figure 25 to show the distribution of the maximum errors over the 25 loops. These plots show that our calibration method is still very accurate regardless of changes in the environment, as the maximum rotation and translation errors do not exceed 0.035°, 0.28°, and 0.88 cm respectively. We note that these errors are higher than those from the indoor parking garage experiment as the average depth of the scene points in the indoor parking garage is smaller. In other words, the nearer the scene points to the camera, the more accurate the calibration parameters.

7 Discussions

Our past use of an AR-marker-based extrinsic calibration method motivated the development of our infrastructure-based calibration method. The AR-marker-based calibration method required us to prepare

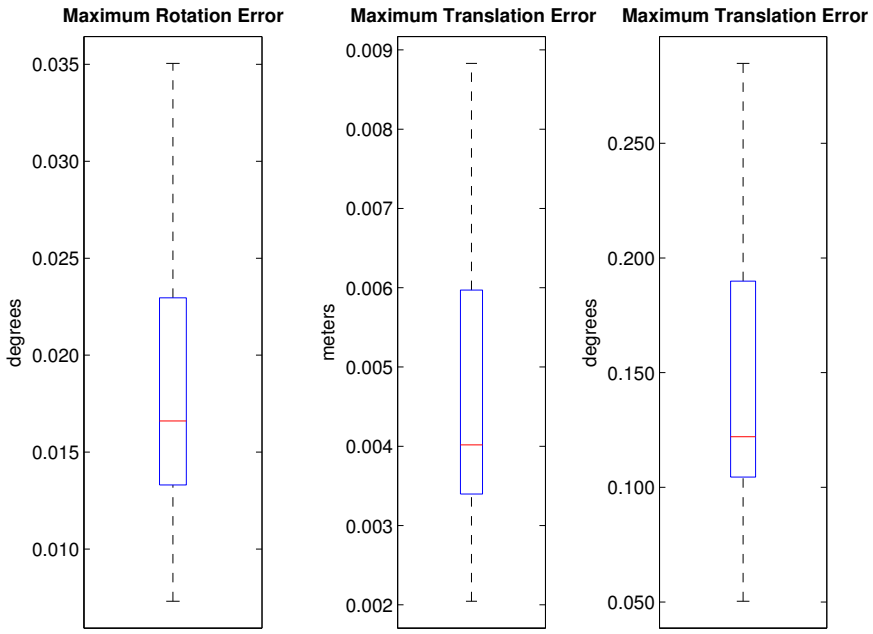


Figure 25: A box plot of the maximum errors for 25 loops.

many AR markers and a calibrated camera for taking images of the scene in conjunction with the multi-camera system. This method was time-consuming, and required a great deal of manual and repetitive work. Hence, we set out to develop a calibration pipeline that was unsupervised, and minimized operator involvement.

Our original implementation of the SLAM-based self-calibration method (Heng et al., 2013) had several flaws that led to an inaccurate map of the calibration area, and in turn, inaccurate estimates of the extrinsic parameters. On some occasions, wrong loop closures caused the map to become distorted, and the extrinsics to become inaccurate. The relative inter-camera poses were not sufficiently accurate for a seamless composite top-view image. Furthermore, optimization of the intrinsics in the bundle adjustment led to overfitting; lines in rectified pinhole images were no longer straight after joint optimization. This line warping issue caused problems for multi-view stereo applications that depended on rectified pinhole images. Lastly, scale drift occurred during joint optimization, causing the scale of both the extrinsics and the map to become inaccurate. Hence, in this paper, we added robust pose graph optimization to solve the false loop closure problem, and added multiple sets of inverse-covariance-weighted residuals in the joint optimization to improve the accuracy of the relative inter-camera poses, prevent overfitting of the intrinsics, and avoid scale drift.

Our infrastructure-based calibration method is able to produce accurate calibration parameters such that the resulting composite top-view image created from a multi-camera system shows seamless blending between images from all cameras. The infrastructure-based calibration consistently produces accurate calibration parameters as long as the map is metrically accurate, and is an accurate representation of the environment.

However, the infrastructure-based calibration method can fail in three cases:

1. If a map of a dynamically changing environment is used for the infrastructure-based calibration long after it was created, the image-based localization step may yield either too few or erroneous camera poses, and produce inaccurate calibration parameters as a result. Here, the image-based localization will not work well as expected because the map does not accurately represent the calibration area.

2. The map is metrically inaccurate and not globally consistent. There are three causes which cause a

degradation in the quality of the map provided by our self-calibration method:

(a) Wrong loop closures are considered in the construction of the map. During self-calibration, robust pose graph optimization may fail in self-similar environments in which a wrong loop closure occurs more often than a correct loop closure. As a result, wrong loop closures may be misclassified as correct, and in turn, the map becomes distorted, and the estimates of the extrinsics are inaccurate. However, we use a visualization tool to visually inspect the map after the completion of the self-calibration, and we observe that if the map visually looks correct given knowledge of the environment layout, the extrinsics are usually accurate.

(b) The odometry system on the survey vehicle is not calibrated. The scale accuracy of the map generated by our self-calibration method is closely tied to the scale accuracy of odometry data. If odometry data reports inaccurate scale information such that the distance travelled by the vehicle is either overestimated or underestimated, the scale of the map will be inaccurate. In turn, the scale of the extrinsics estimated by the infrastructure-based calibration is inaccurate. Nevertheless, if we ensure that the odometry system on the survey vehicle is always well-calibrated, the resulting map and extrinsics will be accurate in terms of scale.

(c) There are too few features in the environment due to lack of texture. Monocular visual odometry may break very often, loop closures may be difficult to detect, and feature correspondences between different cameras may be few. Hence, without sufficient features in the environment, we cannot obtain a metrically accurate and globally consistent map.

8 Conclusions

Through extensive field experiments, our SLAM-based self-calibration method is proven to provide a metrically accurate and globally consistent map of the calibration area during the survey phase. As a result, our infrastructure-based calibration method computes highly accurate calibration parameters for a multi-camera rig with non-overlapping fields of view. Due to the use of natural features, our infrastructure-based calibration method can be used in an arbitrary environment without the need for special calibration setups or environment modifications. Most importantly, the calibration is quick and unsupervised. Hence, no substantial time-related and material-related costs are incurred from using our method to calibrate a multi-camera rig.

Our extensive experiments clearly demonstrate the high accuracy of both our self-calibration and infrastructure-based calibration methods. Both methods are demonstrated to work both indoors and outdoors. We show that our infrastructure-based calibration method can infer calibration parameters for camera configurations which significantly differ from that used to build the map during the survey phase. We also show changes in the environment to have no impact on the calibration accuracy.

We see our infrastructure-based calibration method as having huge potential benefits for the robotics community and automotive industry as the method is unsupervised, robust, easy to use, and produces accurate calibration parameters. The implementations for the self-calibration and infrastructure-based calibration methods are available as part of the CamOdoCal library at `https://github.com/hengli/camodocal`.

9 Future Work

We plan to work on an online version of the calibration. The motivation is that several cameras on our car platforms are mounted on movable parts, for example, a car door, and that vision applications will fail when the cameras are moved. Furthermore, wear and tear causes the camera extrinsics to change slowly over time. An online calibration avoids the need for drivers to periodically bring their vehicles equipped with multi-camera systems to the workshop for recalibration. To deal with unobservable parameters during online calibration, we will explore the use of the method proposed by Maye et al. (2013), which locks unobservable

directions in parameter space, and chooses informative measurements to keep the computational complexity limited.

In addition, we will look at extending the usable lifespan of the map used for the infrastructure-based calibration by exploring time-invariant feature descriptors and illumination-invariant imaging.

Finally, we would like to investigate the possibility of integrating our recent work on continuous-time state estimation into the calibration pipeline. This would allow the estimation of the time offsets between the odometry and image measurement streams (Furgale et al., 2013a), and enable the pipeline to work with rolling shutter cameras (Oth et al., 2013) that are ubiquitous in the automotive industry.

Acknowledgments

The first author is funded by the DSO National Laboratories Postgraduate Scholarship. In addition, this work is supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge). We thank Christian Häne for the plane sweep stereo implementation, and Mathias Bürki and Jerome Maye for their hard work in setting up the Prius experimental platform for data collection.

References

Agarwal, S., Mierle, K., and Others (2013). Ceres solver. `https://code.google.com/p/ceres-solver/`.

Antonelli, G., Caccavale, F., Grossi, F., and Marino, A. (2010). Simultaneous calibration of odometry and camera for a differential drive mobile robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5417--5422.

Brookshire, J. and Teller, S. (2012). Extrinsic calibration from per-sensor egomotion. In *Robotics: Science and Systems (RSS)*.

Carrera, G., Angeli, A., and Davison, A. (2011). Slam-based automatic extrinsic calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2652--2659.

Censi, A., Franchi, A., Marchionni, L., and Oriolo, G. (2013). Simultaneous calibration of odometry and sensor parameters for mobile robots. *IEEE Transactions on Robotics*, 29(2):475--492.

Daniilidis, K. (1999). Hand-eye calibration using dual quaternions. *International Journal of Robotics Research (IJRR)*, 18(3):286--298.

Furgale, P., Rehder, J., and Siegwart, R. (2013a). Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280--1286, Tokyo, Japan.

Furgale, P., Schwesinger, U., Rufli, M., Derendarz, W., Grimmett, H., Mühlfellner, P., Wonneberger, S., Rottmann, J. T. S., Li, B., Schmidt, B., Nguyen, T. N., Cardarelli, E., Cattani, S., Brüning, S., Horstmann, S., Stellmacher, M., Mielenz, H., Köser, K., Beermann, M., Häne, C., Heng, L., Lee, G. H., Fraundorfer, F., Iser, R., Triebel, R., Posner, I., Newman, P., Wolf, L., Pollefeys, M., Brosig, S., Effertz, J., Pradalier, C., and Siegwart, R. (2013b). Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 809--816, Gold Coast, Australia.

Gao, C. and Spletzer, J. (2010). On-line calibration of multiple lidars on a mobile vehicle platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 279--284.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231--1237.

Geiger, A., Moosmann, F., Car, O., and Schuster, B. (2012). Automatic calibration of range and camera sensors using a single shot. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936--3943.

Guo, C., Mirzaei, F., and Roumeliotis, S. (2012). An analytical least-squares solution to the odometer-camera extrinsic calibration problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3962--3968.

Heng, L., Bürki, M., Lee, G. H., Furgale, P., Siegwart, R., and Pollefeys, M. (2014). Infrastructure-based calibration of a multi-camera rig. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Heng, L., Li, B., and Pollefeys, M. (2013). Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1793--1800.

Kneip, L., Furgale, P., and Siegwart, R. (2013). Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3770--3776.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2969--2976.

Kumar, R., Ilie, A., Frahm, J., and Pollefeys, M. (2008). Simple calibration of non-overlapping cameras with a mirror. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1--7.

Lebraly, P., Royer, E., Ait-Aider, O., Deymier, C., and Dhome, M. (2011). Fast calibration of embedded non-overlapping cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 221--227.

Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013a). Robust pose-graph loop-closures with expectation-maximization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 556--563.

Lee, G. H., Li, B., Pollefeys, M., and Fraundorfer, F. (2013b). Minimal solutions for pose estimation of a multi-camera system. In *International Symposium on Robotics Research (ISRR)*.

Levinson, J. and Thrun, S. (2012). Automatic calibration of cameras and lasers in arbitrary scenes. In *International Symposium on Experimental Robotics (ISER)*.

Li, B., Heng, L., Köser, K., and Pollefeys, M. (2013). A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1301--1307.

Lim, H., Sinha, S., Cohen, M., and Uyttendaele, M. (2012). Real-time image-based 6-dof localization in large-scale environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1043--1050.

Maddern, W., Harrison, A., and Newman, P. (2012). Lost in translation (and rotation): Rapid extrinsic calibration for 2d and 3d lidars. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3096--3102.

Markley, F., Cheng, Y., Crassidis, J., and Oshman, Y. (2007). Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(1):12--28.

Maye, J., Furgale, P., and Siegwart, R. (2013). Self-supervised calibration for robotic systems. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 473--480, Gold Coast, Australia.

Oth, L., Furgale, P., Kneip, L., and Siegwart, R. (2013). Rolling shutter camera calibration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1360--1367.

Sattler, T., Liebe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 667--674.

Tariq, S. and Dellaert, F. (2004). A multi-camera 6-dof pose tracker. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 296--297.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment a modern synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298--372. Springer Berlin Heidelberg.

Yang, R. and Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 211--217.

References

[1] Hugo Grimmett, Rohan Paul, Rudolph Triebel, and Ingmar Posner. Knowing when we don't know: Introspective classification for mission-critical decision making. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[2] Rudolph Triebel, Hugo Grimmett, Rohan Paul, and Ingmar Posner. Driven learning for driving: How introspection improves semantic mapping. In *International Symposium on Robotics Research (ISRR)*, Singapore, December 2013.

[3] Christian Häne, Christopher Zach, Jongwoo Lim, Ananth Ranganathan, and Marc Pollefeys. Stereo depth map fusion for robot navigation. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[4] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Muhlfellner, S. Wonneberger, B. Li, B. Schmidt, T. Nghia Nguyen, E. Cardarelli, S. Cattani, S. Bruning, S. Horstmann, M. Stellmacher, S. Rottmann, H. Mielenz, K. Köser, J. Timpner, 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 *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2013.