Information and Communication Technologies

Collaborative Project

“V-Charge”

Autonomous Valet Parking and Charging for e-Mobility

Grant Agreement Number 269916



Deliverable Number	D3.5
Deliverable Name	Final integration and evaluation
Dissemination Level	Restricted
WP Number	WP 3
Date of Preparation	15/09/2015
Date of Submission	30/09/2015
Editor/Beneficiary	K. Mimura / Robert Bosch GmbH

Executive summary

This deliverable reports on the progress within work package 3 of the V-Charge Project for Autonomous Valet Parking and Charging for e-Mobility. The main objective of the overall work package is to develop and test novel approaches towards online, robot-local localization and environment perception using camera-based, ultrasonic and probably radar based sensors. The sensors are selected with an eye kept to a resulting close-to-market setup.

The goal of this deliverable is to report final integration status and results of performance evaluation for all components involved in the work package 3.
In detail the deliverable addresses the following tasks from the description of work:

"Task 3.6: Testing and Integrity Check (RBG (lead), VW, Oxford, ETHZ)
Bosch, in cooperation with VW, Oxford and ETHZ, is responsible for integrating and testing the developed localization and scene analysis algorithms on the test vehicle. This validation will be an ongoing task during the entire project, adapted to its different stages. In particular, it requires the availability of accurate ground-truth data, which is one major goal in this task. Furthermore, we will formulate evaluation criteria for the quality of the semantic maps (delivered from WP2) and the scene analysis module based on the resulting performance improvement of the localization algorithm.

First of all, final status of sensor / software-module integration in the V-Charge system is explained in Section 1. Integrated software-modules are categorized into three groups: localization, perception in static environment and perception in dynamic environment.

In the group of localization, a module of "Visual Localization" developed by ETHZ-ASL and VW is included (Section 2). The localization algorithm works based on mono-camera images. Performance of the Visual Localization module is evaluated in Section 2.1.

Four modules are categorized into the second group, perception in static environment: "Stereo-Cameras" developed by RBG, "Structure from Motion based on Mono-Cameras" developed by ETHZ-CVG, "Ultrasonic Sensors" from RBG and "Grid-Map Fusion" by VW (Section 4). Evaluation results for accuracy of obstacle positions detected by these modules are shown in the sub sections.

In the last group, perception in dynamic environment, five modules are included: “Object Tracking” by VW and “Stereo Hypothesis Classification”, “Mono Object Detection by Classification”, “Dynamic Object Detection at Intersections” and “Parking Spot Refinement” developed by Parma (Section 5). Performance for these modules is evaluated in the sub sections.

In addition to evaluations for each module, some problems observed during test drives have been analyzed. Problems appearing in vehicle behavior (unexpected or “strange” behavior) need to be analyzed by tracing back on the module chain in the system. A problem which occurred in the ACC scenario is taken as an example and “cross-module” analysis is introduced in Section 6.

In a final conclusion (Section 7), an overview on performance of all perception-related modules integrated in the V-Charge system is summarized.

1 Introduction 5
1.1 Sensor Integration 5
1.2 Software-Module Integration 6
2 Localization 8
2.1 Visual Localization 8
3 Perception in Static Environment 9
3.1 Stereo-Cameras 10
3.1.1 Evaluation Method and Data-Set 11
3.1.2 Evaluation Result 12
3.2 Structure-from-Motion based on Mono-Cameras 13
3.3 Ultrasonic Sensors 14
3.3.1 Evaluation Criteria and Methodology 15
3.3.2 Data-Set used for Evaluation 17
3.3.3 Evaluation Results and Discussion 18
3.4 Grid-Map Fusion 20
3.4.1 Evaluation Methodology 20
3.4.2 Results 23
4 Perception in Dynamic Environment 28
4.1 Object Tracking 28
4.2 Stereo Hypothesis Classification 31
4.3 Mono Object Detection by Classification 33
4.4 Dynamic Object Detection at Intersections 34
4.5 Parking Spot Refinement 35
5 Analysis of Problems during Test Drives 36
5.1 Problem at ACC Scenario 36
6 Conclusion 37
7 References 38

1 Introduction

V-Charge is a collaborative project to investigate and provide a profound impact on the state-of-art in automated driving in low-speed environments such as parking lots, for example, automated valet-parking. The project concept is to combine an automated valet-parking system with a topic of battery charging for near-future E-Mobility life.

In combination with very restricted integration effort on the infrastructure side, this goal shall be achieved with entry-level, low-cost sensors for the task of online perception (Work Package 3). Only vision based sensors (cameras) and ultrasonic sensors which are already in or close to series-production are integrated.

In this section, final status of sensor (hardware) integration and software-module integration is explained.

1.1 Sensor Integration

Keiji Mimura, RBG

There are three types of sensors integrated in the V-Charge system: stereo-cameras (front and rear), fisheye mono-cameras and ultrasonic sensors (Fig. 1, Fig. 2).

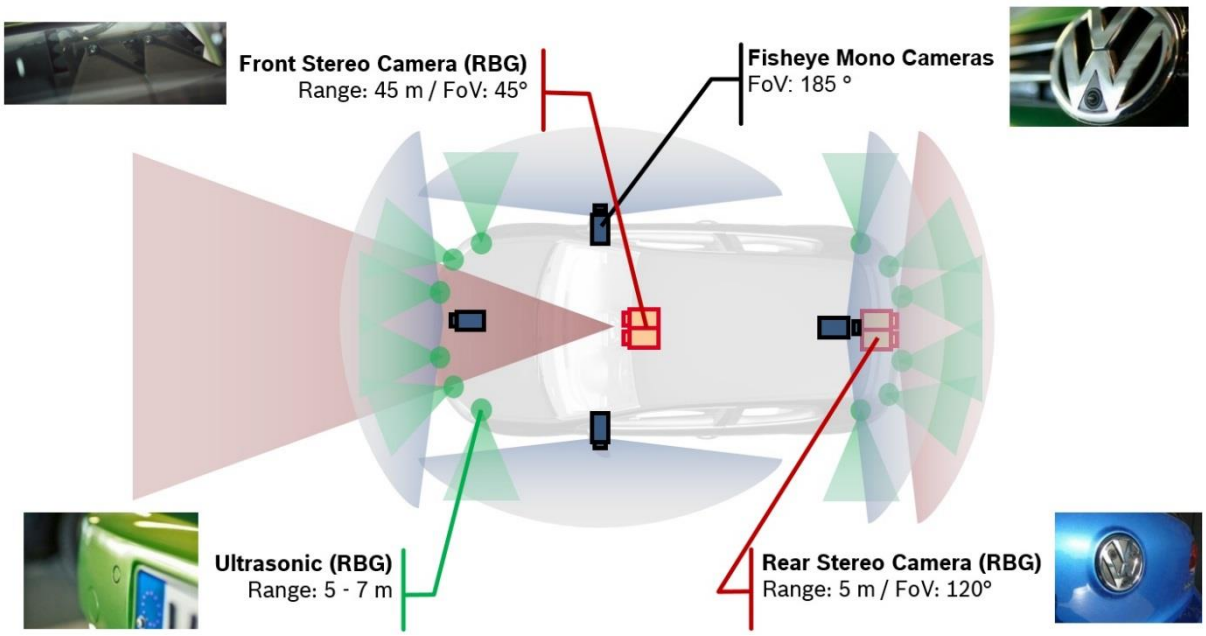


Fig. 1: Overview of integrated sensors in V-Charge test vehicles



Fig. 2: Integrated sensors (Top: Front Stereo-Camera, Bottom-Left: an Ultrasonic Sensor, Bottom-Right: a Fisheye Mono-Camera)

A front stereo-camera is installed in the front to detect objects (Fig. 2, top). Its range is 45 m and field-of-view (FoV) is 45 degree. Reference images have resolution of 512 x 219 Pixel. A prototype rear stereo-camera module is installed in the rear. The lenses for the rear stereo-camera are same fisheye lenses as ones for the mono cameras. Its stereo baseline is 5 cm so that the module can be integrated into the VW-logo in the rear. Stereo-cameras (front and rear) deliver distance data readings at 12.5 [Hz].

In addition to stereo-cameras, four fisheye mono-cameras are installed in the front VW-logo (Fig. 2, bottom-right), in the rear VW-logo and in both side mirrors. They have a nominal FoV of 185 degree and 1.3 Megapixel resolution and synchronously triggered at 12.5 Hz. Partners get raw images from mono-cameras for their software-modules.

Beside vision based sensors, twelve ultrasonic sensors are installed in front-/rear-bumpers (6 in the front, 6 in the rear) to detect objects in short range (Fig. 2, bottom-left). Each sensor has an aperture angle of horizontally 120 degree and vertically 60 degree and a range up to 4.2 m. Partners have access to raw distance readings and preprocessed object distance readings.

1.2 Software-Module Integration

Keiji Mimura, RBG

In the work package 3, seven software-modules have been developed by different partners, integrated into the test vehicles and tested. An overview of the modules is shown in Tab. 1.

Category	Software-Module	Responsible Partners	Source Sensors	Integration Status
Localization	Visual-Localization	ETHZ-ASL, VW	Fisheye Mono-Cameras	Integrated
Perception in Static Environment	Stereo-Cameras (Front and Rear)	RBG	Stereo-Cameras (Front and Rear)	Integrated
	Structure-from-Motion based on Mono-Cameras	ETHZ-CVG	Fisheye Mono-Cameras	Integrated
	Ultrasonic Sensors	RBG	Ultrasonic Sensors	Integrated
	Grid-Map Fusion	VW	(Outputs from Stereo-Cameras, SfM, US)	Integrated
Perception in Dynamic Environment	Object Tracking	VW	Stereo-Cameras	Integrated
	Stereo Hypothesis Classification	Parma	Fisheye Mono-Cameras	Integrated
	Mono Object Detection by Classification	Parma	Fisheye Mono-Cameras	Integrated
	Dynamic Object Detection at Intersections	Parma	Fisheye Mono-Cameras	Integrated
	Parking Spot Refinement	Parma	Fisheye Mono-Cameras	Integrated

Tab. 1: Overview of software modules

The software-modules are categorized into three groups: localization, perception in static environment and perception in dynamic environment. In the group of localization, a module of “Visual Localization” developed by ETHZ-ASL and VW is included. The localization algorithm works based on the fisheye mono-camera images. Four modules are categorized into the second group, perception in static environment: “Stereo-Cameras” developed by RBG, “Structure from Motion based on Mono-Cameras” developed by ETHZ-CVG, “Ultrasonic Sensors” from RBG and “Grid-Map Fusion” by VW. The module of Grid-Map Fusion takes outputs from other three modules (Stereo-Cameras, Structure from Motion, Ultrasonic Sensors). In the last group, perception in dynamic environment, two modules are included: “Object Tracking” by VW and “Stereo Hypothesis Classification”, “Mono Object Detection by Classification”, “Dynamic Object Detection at Intersections” and “Parking Spot Refinement” developed by Parma. The Object Tracking module works based on the output from the Stereo-Cameras and all components developed by Parma process

the output from the Object Tracking and the images from the Fisheye Mono-Cameras.

All components listed in Tab. 1 have been integrated into V-Charge test vehicles.

2 Localization

As a localization system in the V-Charge system, a component “Visual Localization” has been integrated. The Visual Localization module finds vehicle’s position and orientation through matching of natural landmarks based on images from fisheye mono-cameras through matching.

2.1 Visual Localization

Mathias Buerki, ETH-ASL

The visual localization system has been developed in close collaboration between ASL and VW, and is related to the mapping activities of CVG and OX defined in WP2. Four mono-fisheye cameras, attached to the car in all cardinal directions, are used together with wheel odometry sensors to build vision-based maps against which the car can robustly and reliably localize itself (Fig. 3). The performance of this visual localization system has been tested and analysed extensively in the past sixteen months. For this purpose, 77 datasets over a total length of 30km were collected and evaluated. The analysis focuses on robust metric localization in outdoor conditions subject to change in weather and lighting. A key element to reach this goal and a major contribution of this work forms the summary-mapping technique, which applies various simple as well as more sophisticated landmark selection strategies to downsize multi-session maps. Through this, efficient and reliable localization is made possible at constant map sizes (wrt. the mapped area). The results have been submitted and accepted for publication in Journal of Field Robotics [1].

Key Evaluation Results:

- Localization accuracy < 10cm (local error, wrt. DGPS)
- Localization possible outdoor over full yearly cycle
- Constant map size while localization success rate with summarization techniques
- Localization possible over day-night cycle (with limitations)

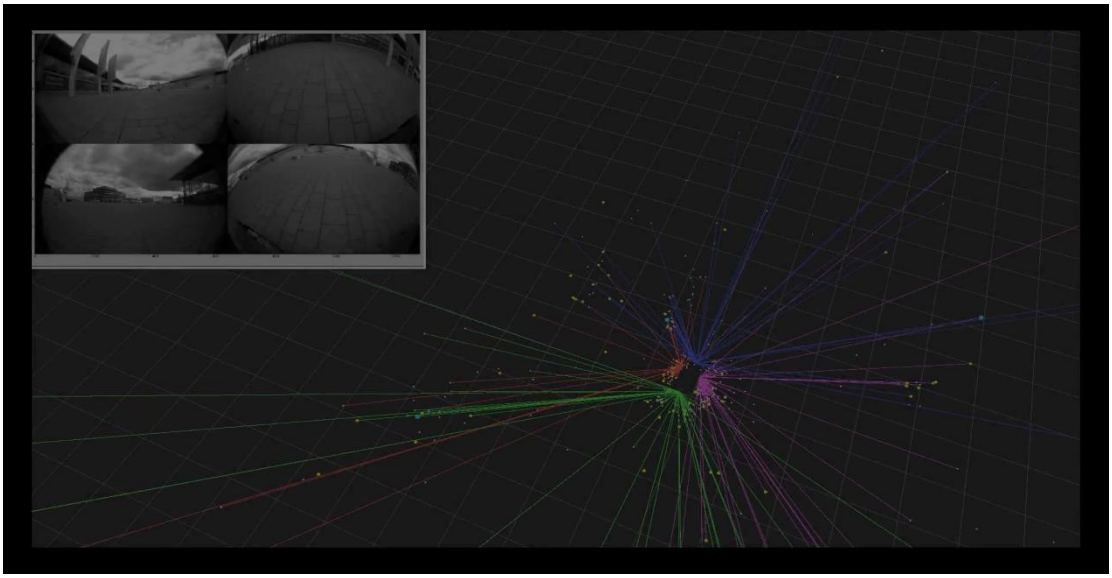


Fig. 3: An Example scene of a matching process in Visual Localization

3 Perception in Static Environment

In order to realize collision-free drive, obstacles around the vehicle need to be detected and avoided. Static objects are detected by various sensors and fused to represent the environment in a form of Grid-Map. All related modules are summarized in Fig. 4.

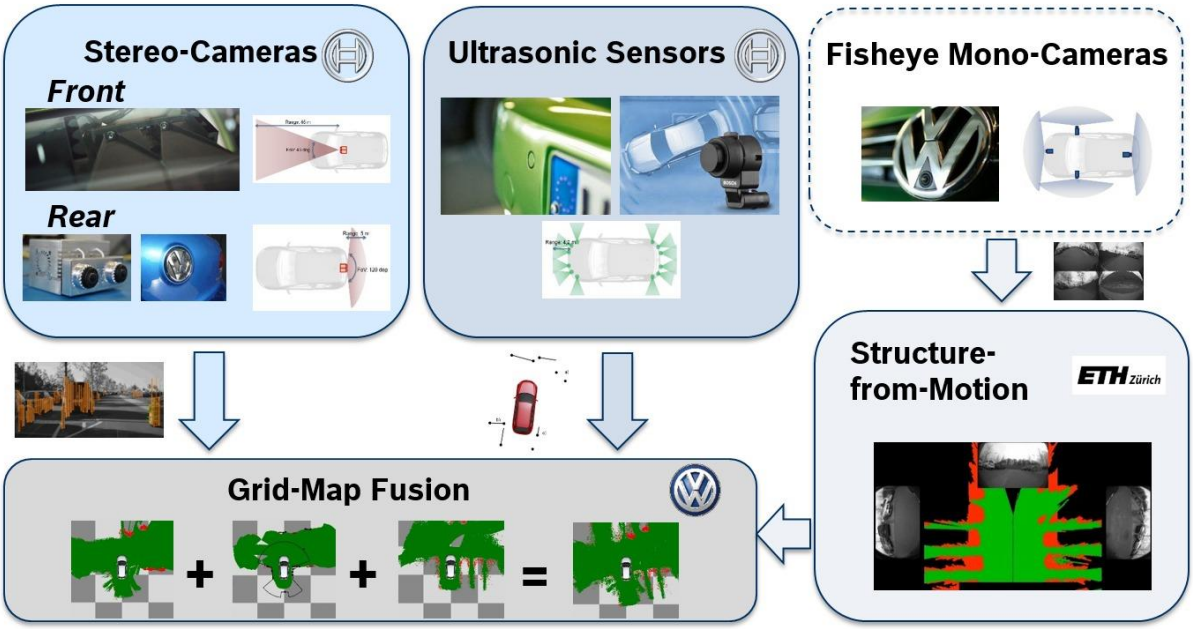


Fig. 4: Overview of modules in Perception in Static Environment

3.1 Stereo-Cameras

Bernd Göbelsmann, RBG

A stereo-camera is installed in the front to detect objects. Its range is 45 m and its field-of-view (FoV) is 45 degree (Fig. 5). There are two operation modes for the front stereo-camera:

a) Deliver distance data readings + left reference image

b) Deliver left and right reference images

These two operation modes are switchable online per software. Reference images have resolution of 512 x 219 Pixel. Stereo-cameras (front and rear) deliver distance data readings at 12.5 [Hz].

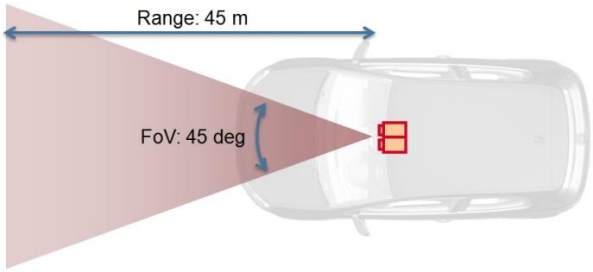


Fig. 5: Range and Field-of-View (FoV) of Stereo-Camera Front

A prototype stereo-camera module is installed in the rear. The lenses for the rear stereo-camera are same fisheye lenses as ones for the mono cameras. Its stereo baseline is 5 cm so that the module can be integrated into the VW-logo in the rear. In order not to make the over-all system complex, the mono cameras and the rear stereo-camera remain separate sub-systems. As a consequence, there are 3 lenses integrated in the rear VW-logo: 1 lens for the mono cameras and 2 for the rear stereo-camera. This new rear stereo-camera is connected to the same PC as the front stereo-camera which results in no additional components except the stereo-camera module itself. Its range is 5 m and its field-of-view (FoV) for the stereo-camera function is 120 [deg] (Fig. 6).

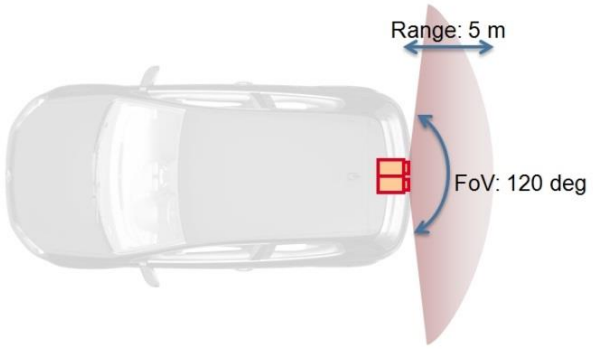


Fig. 6: Range and Field-of-View (FoV) of Stereo-Camera Rear

Both stereo-cameras deliver up to several hundreds of preprocessed object distance readings, together with additional attributes like height over ground or instantaneous motion direction (Fig. 7). The rear stereo-camera is conceptually similar to the front stereo-camera. Main difference is its wide FoV. As lenses with wide FoV used in the rear stereo-camera, necessary adaptations to manage visual distortions are included in the processing for the rear stereo-camera.

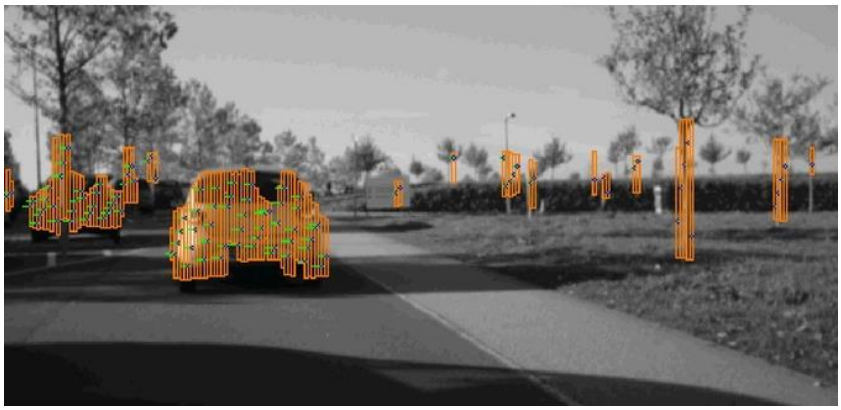


Fig. 7: Objects detected by Stereo-Camera Front

3.1.1 Evaluation Method and Data-Set

For evaluation we generated datasets with ground truth positions coming from laser rangefinder. The dataset for front-stereo consist of measurements of a car standing in front of the ego-vehicle. Several distances from 5m to 25m in camera coordinate system were analyzed (Fig. 8). Based on these static scenes we made an evaluation of the object distance readings. Target was to determine the relative difference of the measured distance of the car in front to ground truth position.



Fig. 8: Evaluation setup of Stereo-Camera Front

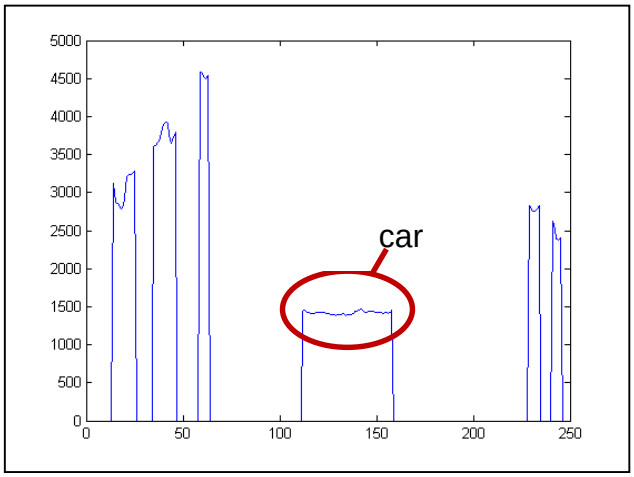


Fig. 9: Min. distance (y-axis) of each column segment (x-axis) in cm. Ground-truth distance is 15m.

The evaluation method of rear stereo system is very similar to front stereo. Here the dataset consists of a row of sequences with targets positioned in a grid back of the car. The lateral expansion of the analyzed surface extends from -2m to +2m and the longitudinal expansion extends from 0m up to 4m in camera coordinate system. A Sample image is shown in Fig. 10.



Fig. 10: Sample image Stereo-Camera Rear

3.1.2 Evaluation Result

As mentioned above our target from our front stereo image processing was to determine the relative difference of the measured distance of the car in front to ground truth position. For this we analyzed the minimum distance detected in 100 frames per column segment (Fig. 9). Because the measurements are entered into the grid map, we need to evaluate the minimal distance measured. The mean deviation over all column segments is lower than 2% in all tested distances. We can say that the stereo camera is a suitable sensor for this use case.

For the rear stereo camera the longitudinal- and lateral deviation is analyzed separately. The longitudinal deviation from ground truth measurements is shown in Fig. 11. Four rows of objects (distances from 1m to 4m) were placed in back of the car and separately analyzed. The mean deviation in lateral direction is shown in Fig. 12. We also used four rows of objects but this time rotated 90°. The mean deviation in both directions is about five percent. The accuracy should be symmetry, but small errors in calibration can lead to asymmetry.

Beside evaluation for distance measurement accuracy, the height of objects to be detected was evaluated: the minimum height of objects is c.a. 60cm.

In Chapter 3.4 “Grid-Map Fusion”, a grid-map layer based on stereo-camera output is evaluated in different scenarios (Fig.17 and Fig.18). The evaluation results on the “stereo-camera” grid-map layer (Tab.7 and Tab.8) basically match well with the evaluation results here.

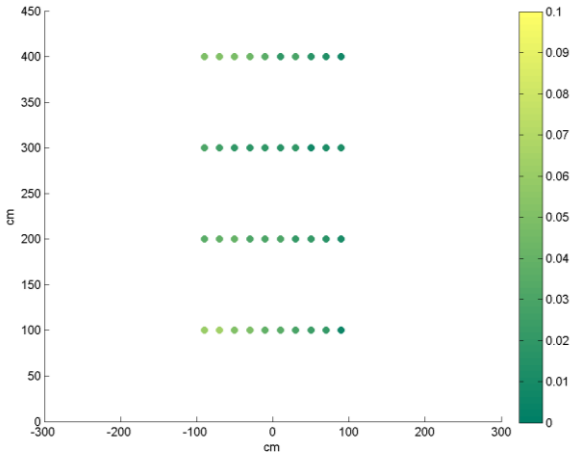


Fig. 11: Longitudinal deviation of stereo measurements in ratio (0.1 = 10%)

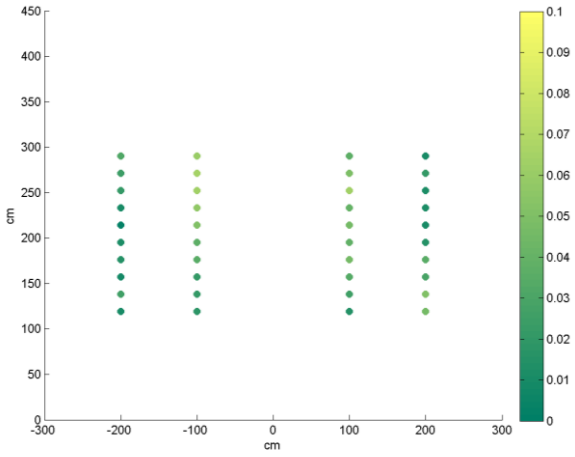


Fig. 12: Lateral deviation of stereo measurements in ratio (0.1 = 10%)

3.2 Structure-from-Motion based on Mono-Cameras

Christian Haene, ETH-CVG

We developed and integrated a system that computes obstacle information based on the fisheye camera images and the motion of the car. For the motion of the car we solely rely on the vehicle's wheel odometry.

The system works by first computing depth maps and then subsequently analyzing the depth maps to find the relevant obstacles. In order to compute depth maps, several images of a slightly different known viewpoint are necessary. This is achieved by treating each camera independently and using several subsequent camera images while the car moves. The camera pose is computed by applying the extrinsic camera calibration on the vehicle pose from the wheel odometry.

The details of our system are described in two scientific papers [2] [3] one was published at the 3D Vision conference (3DV) 2014 and the other one is currently under review.

[2] describes how the depth maps are computed based on the camera images. The algorithm is implemented on a graphics processing unit (GPU) and achieves real-time performance. One of the novelties of the approach is that the fisheye images are directly matched without extracting pinhole images.

[3] describes how the obstacle information is extracted from the depth maps and evaluates the accuracy. To be able to make meaningful statements about the accuracy we implemented a fusion approach for the extracted obstacle information. The evaluation was done by measuring the width of a parking spot and the distance between two obstacles using a tape measure and then comparing this ground truth measurements with the measurements extracted from the fused result. We achieve measurements which are within ± 10 cm from the ground truth, which is accurate

enough for the anticipated tasks. Note that the fusion which is used here is different from the grid map fusion used in the car. It is specifically tuned to handle the output of the structure-from-motion module. The grid-map fusion used on the car needs to be able to fuse different sensors such as binocular stereo camera data, ultrasound obstacle information and the output of the structure-from-motion module. Therefore, the results reported here are slightly more accurate than the ones from the grid-map fusion module.

For the details of our system we would like to refer to the attached scientific articles.

3.3 Ultrasonic Sensors

Keiji Mimura, RBG

Ultrasonic sensors are range sensors based on a time-of-flight concept using ultrasonic waves. A Bosch Ultrasonic system was used in V-Charge. The system is used for near-range environment perception.

The Bosch system has multiple modes. V-Charge makes use of a mode with preprocessed data. The preprocessing algorithm provides ultrasonic objects to represent the environment. Raw sensor echoes are preprocessed to achieve higher accuracy. To increase spatial resolution neighboring sensors are combined. The echoes are also filtered over time to gain a more detailed understanding of the environment, which is modeled using obstacle points and line segments. Specialized parking spot detection algorithms based on raw sensor echoes allow far better results in this special application. However the ultrasonic objects where used to create a higher level environment model together with other sensors, so the evaluation also focuses on the same data. Fig. 13 illustrates a typical parking scenario showing the ultrasonic objects.

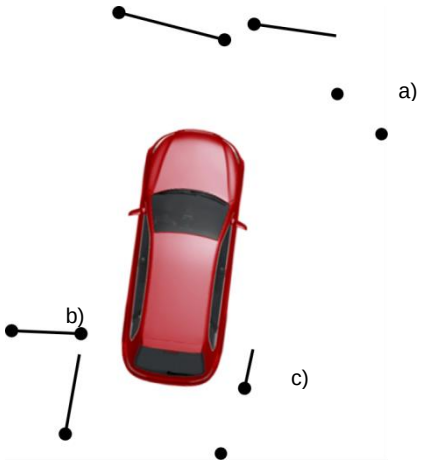


Fig. 13: Ultrasonic measurements in a typical parking situation (The environment is modeled using points a) and line segments b). In the figure points on line segment endings define confident segment endings. The absence of a point on an ending signals an ongoing segment c).)

3.3.1 Evaluation Criteria and Methodology

Ultrasonic information is evaluated as it is integrated into the static environment grid map, disregarding higher level information such as the line between the coordinates itself or the information that the line is ongoing at the end point.

Parking

The ultrasonic sensors' accuracy is generally very dependent on scenarios. Evaluations in simplified scenarios show only limited insight into accuracy of measurement in parking scenarios.

To evaluate the sensor performance in parking application accurately, the parking spot width defined by ultrasonic sensor readings is calculated for different parking scenarios. The reference information is the pre-defined parking position and parking spot dimensions measured manually with laser measurement equipment.

To calculate a parking spot width based on ultrasonic sensor readings, points from each side of the parking spot must be available. This is not the case in each measurement cycle. In order to calculate a mean error of the estimated widths, they are separated into equally sized bins over the remaining path. The mean of the width bins is then calculated per bin.

Part of the evaluation is done at crucial positions of events during the parking maneuver, called evaluation spots. The evaluation was performed at the following events. See also Fig. 15 for the illustration of typical those positions on the vehicle's trajectory.

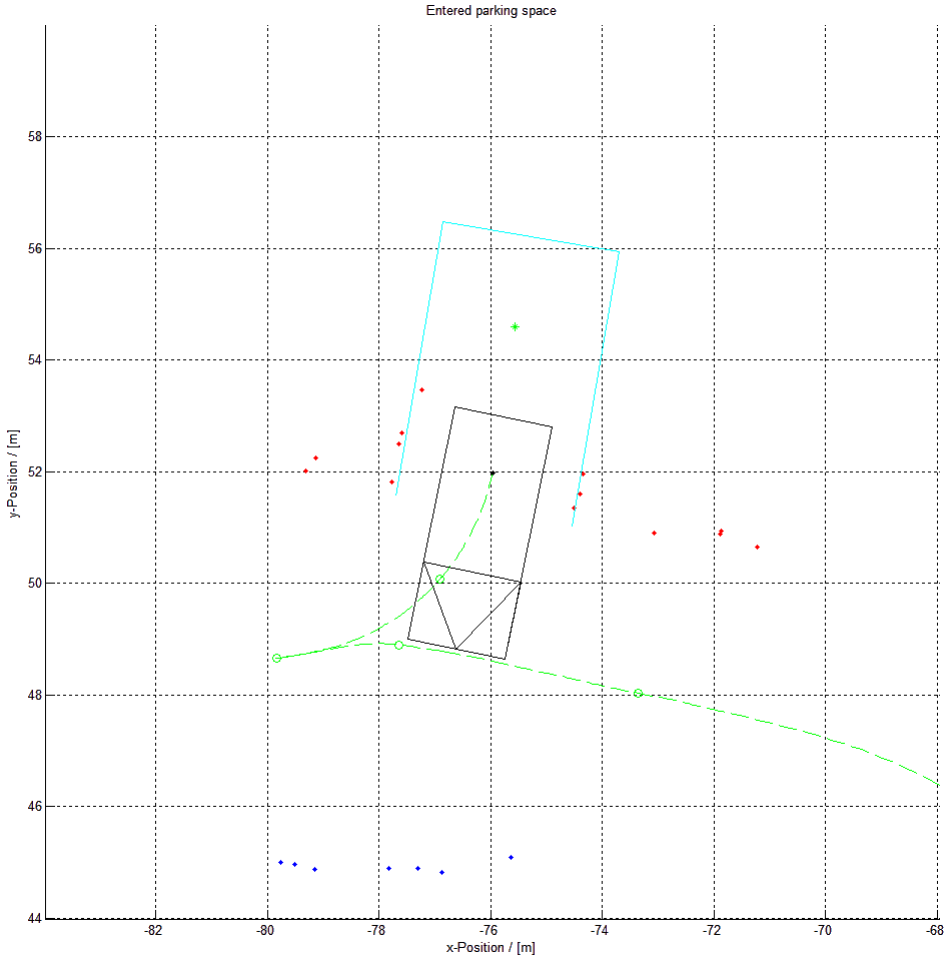


Fig. 14: Evaluation of a typical parking run (The vehicle - visualized in black color, arrow points to front direction - parks in the parking spot - cyan color - by driving the green trajectory. Ultrasonic points in red color are considered belonging to the parking spot. The desired pose within the parking spot is illustrated in green.)

1) Approaching the parking spot

The event occurs when the first ultrasonic objects are associated to the parking spot boundary. This given an initial guess about the expected parking spot width. The vehicle is usually several meters away from the parking position.

2) Crossing the center line with front

In all scenarios backward parking is performed. That means the vehicle crosses the center axis of the parking spot. This evaluation event is reached when the vehicle's front passed the center line the first time.

3) Crossing the center line with rear end

This event corresponds to the event before, but occurs when the rear end of the vehicle crosses the center line.

4) First stop

In this event a specialized trajectory planning algorithm may be executed and find a suitable collision-free path into the parking spot. Accuracy of the parking spot

width is crucial. However since in this evaluation only current measurements are used

5) Entered parking spot

The event occurs as soon as vehicle shape and parking spot overlap. It represents a time frame where re-planning of the trajectory might be necessary based on new sensor information.

6) In parking position

This final event is reached when the vehicle reaches its final position on the trajectory. In this position the sensors perceive usually good results



Fig. 15: Evaluation points in a typical parking maneuver

3.3.2 Data-Set used for Evaluation

The data-set was recorded from 11.11.2014 – 13.11.2014 in Wolfsburg, Germany with the V-Charge demonstrator vehicle.

The evaluation data-set consists of 3 different cross-parking spots in which the vehicle is parked backward in automated mode. For each spot there are different goal poses that had been previously recorded by the localization system of the vehicle. A pose is defined by its position and orientation with respect to world coordinates. The parking spot geometry was measured by the reference system, a handheld laser sensor. Tab. 2 shows an overview of the data-set.

The parking algorithm planned the trajectories to the goal poses and parks the vehicle while trying to avoid collisions with the environment, which is primarily detected by the ultrasonic sensors.

Parking spot	Reference width	Number of runs
PS1	3,27	36
PS2	3,19	11
PS3	3,21	3

Tab. 2: Overview of the data-set.

3.3.3 Evaluation Results and Discussion

Evaluation

Fig. 16 shows the measured parking spot width based on ultrasonic points objects over the remaining path to the parking spot. Additionally the mean positions of the crucial spots over all runs are displayed as vertical lines.

Tab. 3 shows the corresponding mean error at the evaluation spots.

A fast decay of the error can be observed after the vehicle enters the parking spot. This is caused by two effects. First, the obstacles are measured in a much shorter distance when entering the parking spot than when driving by. Second, the position and orientation of sensors with respect to the obstacles and the obstacles planar surface allow more accurate measurements when entering the spot.

Discussion

As a result it can be stated that even with an ultrasonic environment perception that is not focused on parking situations the accuracy of the data is suitable to measure the parking spot width, especially in shorter distances where collisions must be prevented.

Position	Parking spot width error [m] (absolute value)
Near parking position	1.68
Crossing the center line with front end	1.27
Crossing the center line with rear end	1.19
First stop	0.99
Entered parking spot	0.77
In parking position	0.04

Tab. 3: Mean error measured at the evaluation points over all runs.

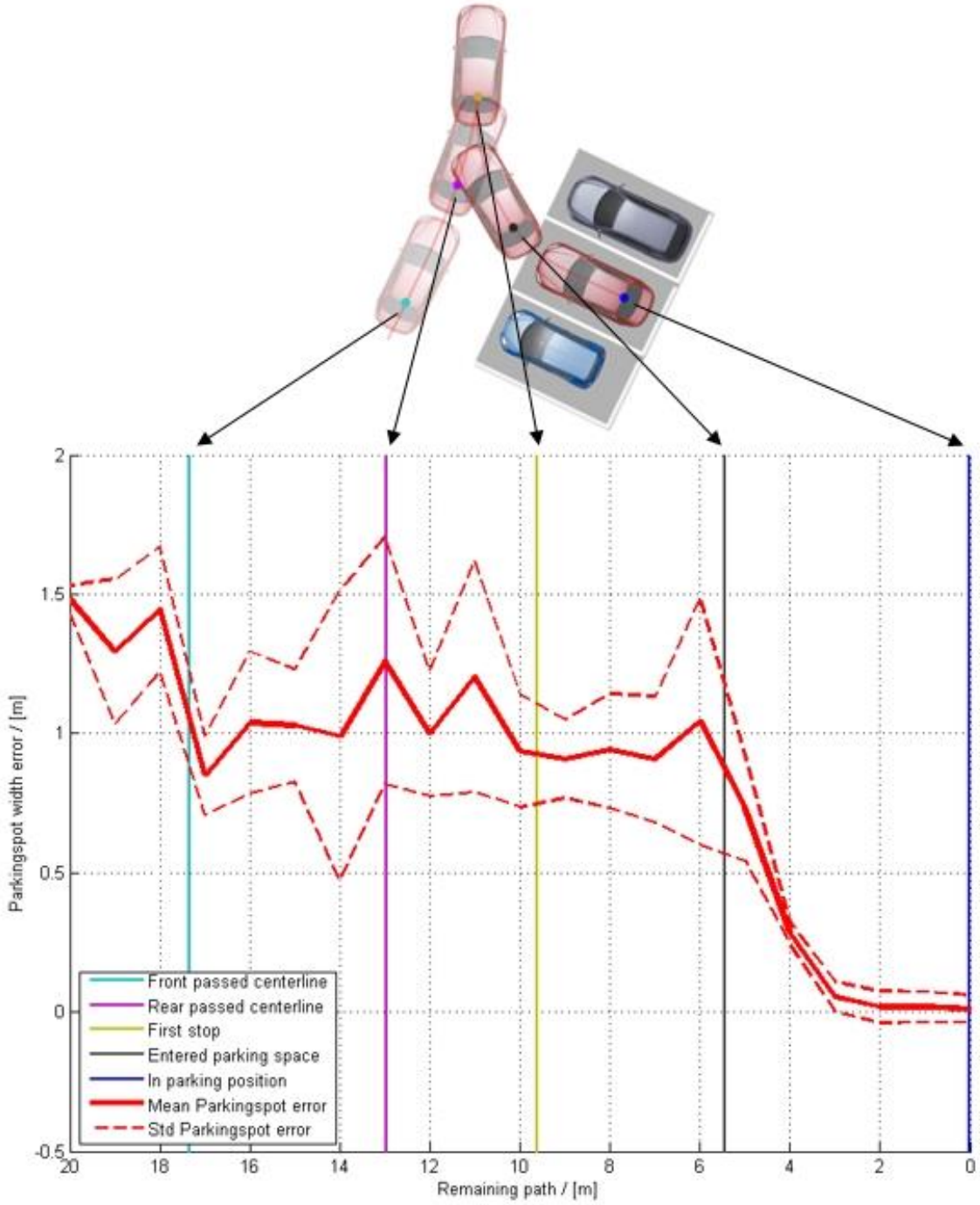


Fig. 16: Parking spot width error and its standard deviation (dashed lines) over the remaining path length of all available runs and scenarios.

In Chapter 3.4 “Grid-Map Fusion”, a grid-map layer based on ultrasonic sensors outputs is evaluated in different scenarios (Fig.17 and Fig.18). The evaluation results on the “ultrasonic sensors” grid-map layer (Tab.7 and Tab.8) basically match well with the evaluation results here.

3.4 Grid-Map Fusion

Peter Muehlfellner, VW

In V-Charge, an occupancy grid map [4] is used to fuse information from different sensors and provide an obstacle representation for use in local planning. Incoming obstacle information from all available sensors is fused into a two-dimensional lattice. Each grid point on this lattice stores the probability that the corresponding section of the world is occupied by an obstacle. These occupancy probabilities are updated from sensor measurements using standard Bayesian methods [4]. The resulting *occupancy grid map* describes a metric region surrounding the vehicle, at a resolution that is defined according to application requirements.

In practice, we keep track of two separate grid maps that are parametrized to allow both navigation on the parking lot and precision parking maneuvers. The navigation map covers a larger area at a lower resolution, whereas the parking map provides high resolution for a small region. The exact parameters for each grid map, as well as the sensors used to generate it, are given in Tab. 4. The values for these parameters were selected to allow representing the highest resolution sensor that is used for each of the grid maps.

Layer Name	Cell size (m)	Size (m)	Sensors
Driving Layer	0.1	50	Front Stereo Camera
Parking Layer	0.05	25	Dense Motion Stereo, Ultrasonic, Rear Stereo Camera, Front Stereo Camera

Tab. 4: Parameters and sensors for the two occupancy grid maps

In the following we will evaluate the suitability of the occupancy grid map as a basis for automated driving in static environments. We design a methodology for evaluation, define a set of experiments and evaluate the results of these experiments.

3.4.1 Evaluation Methodology

The obstacle sensors we employ each have different capabilities and limitations. Each sensor has a limited field of view, some sensors suffer from false-positive or false-negative measurements and they all provide metric obstacle information with various precisions. The goal for our occupancy grid map is to combine the strengths of the individual sensors, while getting rid of their weaknesses. To evaluate how well we perform this task, we first consider the *completeness* of the fused grid representation: do all obstacles appear in the grid map, how are well are difficult obstacles handled? Second, we consider the *precision* of the grid map: what is the error made in measuring the metric position of obstacles?

Completeness

To evaluate the *completeness* of the generated occupancy grid maps, we look at several scenes that include elements that are typically difficult for the sensors we

employ. These scenes include obstacles with repetitive texture, such as wooden and wire-link fences, which are notoriously difficult for the stereo and structure from motion based obstacle sensors. We also consider objects with high ultrasonic absorbance and small obstacles, such as traffic cones and poles. Based on fused occupancy grid snapshots of the example scenes, we determine whether objects are missing and whether safe driving would be possible given the input.

The *range* at which obstacles appear in the grid is equally important, to allow safe driving and parking maneuvers. In order to measure the effective range of each sensor, we set up a set of well textured foam-block dummy obstacles. As we approach these obstacles, we continuously measure the distance between them, and the vehicle rear-axis. We note down at what distances the obstacles appear in the occupancy grid map. Fig. 17 gives an overview of the setup for this experiment.

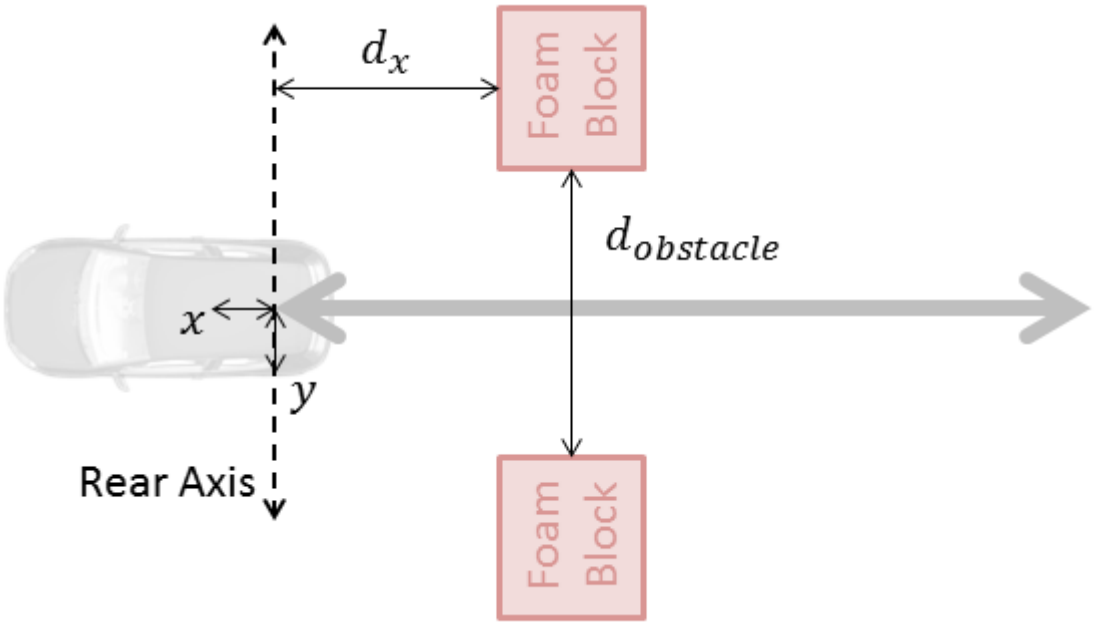


Fig. 17: Evaluation procedure for the suitability of the occupancy grid map for navigation

The vehicle is driven straight through the gap between two foam-block obstacles, forward and backwards. For evaluating sensor-range, we note down at what distance, d_x, the obstacles are visible in the grid map. Grid map precision is measured by comparing the distance, d_obstacle, between the foam-blocks as manually determined using a tape measure, and as estimated via the occupancy grid map. This comparison is performed for various distances between the foam block obstacles, at fixed points after driving forward and backwards between them.

Precision

In V-Charge we aim to perform parking maneuvers with high precision, into narrow parking spots. Similarly, we need to safely navigate narrow passages in a parking lot. Precise obstacle information in the occupancy grid map is necessary to enable such maneuvers. In this report, we will evaluate the metric performance of the grid

representation, built on the basis of our sensor setup, regarding parking and navigation.

For evaluating the precision of the occupancy grid map in the driving scenario, we use a similar setup to the one shown in Fig. 17. The vehicle drives through the gap between two well-textured foam blocks. The gap width is varied in a range from 2.4 meters (barely wider than the vehicle) and 6 meters. We read out the gap width as estimated in the occupancy grid map and compare it to the known values.

For evaluating performance during a parking maneuver, we collect information about the width of a parking spot, as shown by the grid map, at various fixed points during a parking maneuver. This estimated width is then compared with ground truth data that we manually measured for the parking spot, using a tape measure. Cars will be parked on either side of the parking spot, providing obstacles that restrict the width of the free space. Fig. 18 gives an overview of this procedure.

The metric width of a real world parking spot is compared to the measured width with the grid map during various phases of a parking maneuver. Phase A, at t_A, is defined as the point in the time when the vehicle's front passes the entrance to the parking spot, Phase B, at t_B , when the vehicle vehicle's rear has passed the second obstacle, just bevor the reverse-parking maneuver is started. Phase C, at t_C, is defined as the point where the vehicle's rear enters the parking spot (as indicated by the yellow circle), Phase D, at t_D, occurs when the vehicle comes to rest within the parking spot.

All experiments were performed using the V-Charge car on the parking lot of our research campus in Wolfsburg. The occupancy grid map was calculated onboard the vehicle, satisfying the required output rate of ten samples per second and receiving live input from the sensors; the calculated maps were stored for later evaluation.

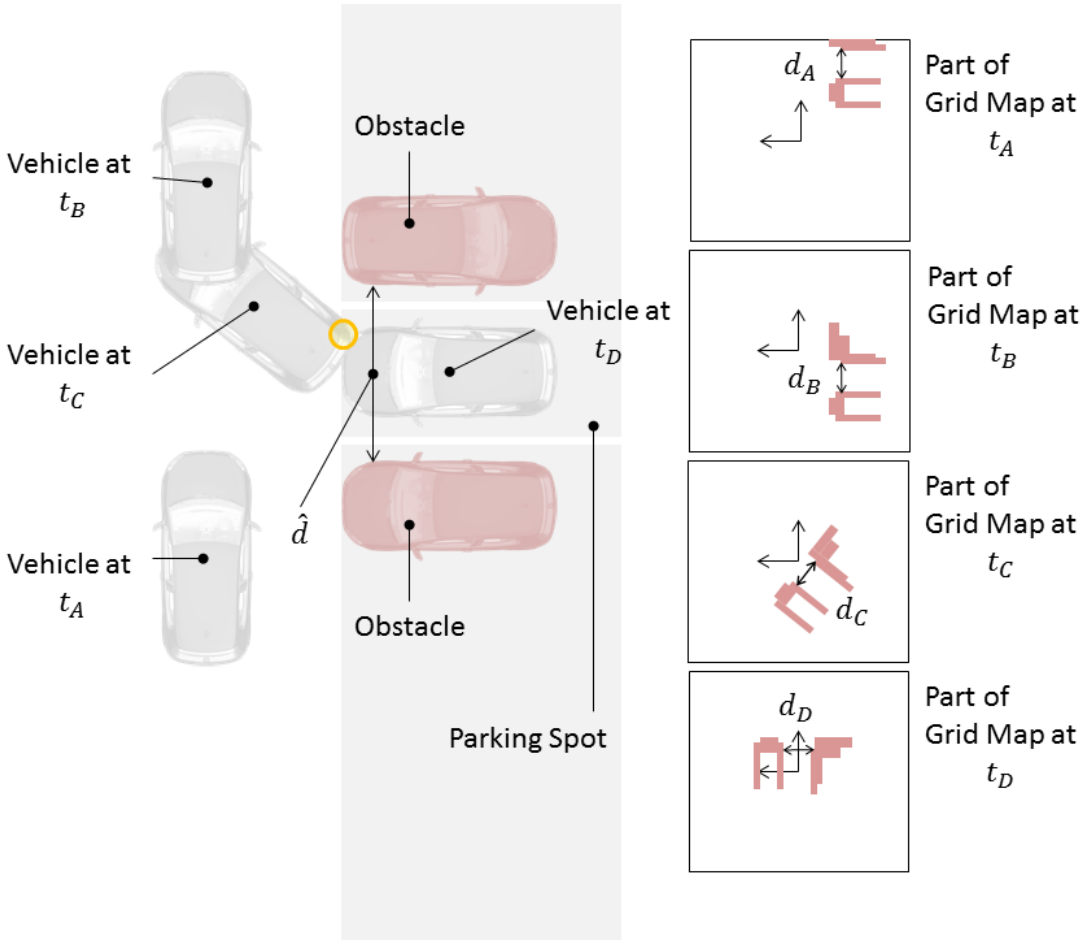


Fig. 18: Evaluation procedure for parking spot width in the occupancy grid map

3.4.2 Results

The qualitative results for the completeness of the occupancy grid map in the evaluated scenarios are given in Tab. 5. The table shows types of obstacles we have considered, shown in an example snapshot from the monocular vehicle cameras. These obstacles occur in the parking lot of our research campus, but we have aimed the selection to be representative of the wider categories of objects one would expect to occur in parking lot settings.

- Parking spots, both while driving and while parking form an important baseline for the parking task. Later in this report, we will also investigate the performance of our pipeline in estimating the width of parking lots.
- The wooden fence and the earth wall represent the typical, repetitively/weakly textured walls one might encounter in a parking garage or parking lot.
- The wired fence is an especially difficult structure; it is barely visible in images. Due to its small physical cross-section, it could be expected to be difficult to detect for active (e.g. ultrasonic) sensors as well.
- The gate arm also has a narrow cross section. Furthermore, it is not connected to the ground plane, potentially posing a problem for sensors/algorithm that work on that assumption.
- The gate housing has proven to be surprisingly difficult to perceive as well, probably due to its weak texture.

- The cones and poles have a small horizontal cross section and relatively low height, which would be expected to be difficult to detect.
- The ash tray has a weak texture, and might therefore be difficult to detect for image-based sensors.
- The foam blocks, finally, serve as our reference – they should be easy to see with all sensors.

For each sensor, we give our assessment of whether this obstacle was detectable, based on a series of datasets collected in Spring 2015, as well as our own experience from numerous parking trials; we also look at how these obstacles were represented in the overall fused representation.

We assess the overall system to be able to detect most types of obstacles, as documented in Tab. 5. Exceptions are the arms of the parking lot gates, which are not perceived by a single sensor – this makes it necessary to communicate the opening/closing state of the gate arms to the vehicle from the parking lot server.

The repetitively/weakly textured obstacles pose little problems to the camera-based sensors. The traffic cones pose a problem for the stereo cameras; they are apparently too small to detect reliably with this sensor. The wire-link fence is only perceived by the ultrasonic sensors.

For parking, the sensor fields of view and position is a deciding factor for their usability. Here, the surround-view based Dense Motion Stereo system proves invaluable, providing excellent performance. The field of view of the front stereo camera is too narrow to reliably allow looking into the parking spots while driving by them; during the later parking stages, the front stereo camera does not provide any useful information. While the field of view of the rear stereo camera is wider, it only provides reliable estimates at very close range. Similarly, the ultrasonic sensors work very reliably, but are also limited by their range.

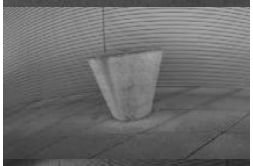
The range, at which obstacles appear for each sensor, and for the overall fused representation, is shown in Tab. 6. As expected, front stereo is the only sensor we can rely on for providing information regarding obstacles ahead of the vehicle; we rely on this information for navigating the parking lot. For perceiving the immediate, 360 degree, surroundings of the vehicles, the other sensors are used. Ultrasonic and Structure from Motion information can be used throughout the whole parking maneuver, rear stereo is useful for the final phase of parking in.

The results for the precision of the occupancy grid map in the foam-block driving scenario are shown in Tab. 7, and Tab. 8 shows the results for estimating the width of the parking lot. The statistics for the parking maneuver were collected over ten trials.

Both stereo sensors have relatively similar accuracy for grid estimation, in the range of 11-21 centimeters. It is possible that a higher accuracy can be reached by the stereo measurements themselves. For fusing their information in the occupancy grid, the precision of the calibration, time synchronization and wheel-tick odometry

are additional influences. The measurements provided by the ultrasonic sensor are not accurate in the early stages of the parking maneuver. Overall though, the accuracy and precision of the ultrasonic sensors is excellent, for both the foam blocks and for the later stages of the parking maneuver. Finally, Dense SfM shows mediocre precision. Due to its indirect measurement principle, it is the sensor that is most susceptible to calibration-, synchronization and pose-estimation errors.

Overall, we see that the proposed sensor setup does an adequate job of both perceiving the different obstacle classes encountered a parking lot, as well as precisely measuring their metric extents. We have combined a set of precise sensors, with limited fields of view or range - stereo and ultrasonic - with the 360° surround view Dense SfM. This gives us good metric performance while estimating the space between obstacles, and the extent of parking spots. At the same time, the sensor setup completely covers the vehicle surroundings, enabling safe navigation of the parking lot.

Scenario, Sensor		Stereo Front	Stereo Rear	Dense SfM	Ultrsonic	Fused Repr.
Parking Spots While Driving		-	-	+	-	+
Parking Spots While Parking		-	-	+	+	+
Wooden Fence		+	+	+	+	+
Wire Fence		-	-	-	+	+
Earth Wall		+	+	+	-	+
Horizontal Gate Arm		-	-	-	-	-
Gate Housing		+	+	+	+	+
Poles		+	+	+	-	+
Cones		-	-	+	+	+
Ashtray		+	+	+	+	+
Foam Block		+	+	+	+	+

Tab. 5: Qualitative evaluation of the completeness of the occupancy grid map with regard to various obstacle types

Distance (m),	Stereo Front	Stereo Rear	Dense	Ultrasonic	Fused Repr.
25	-	-	-	-	-
20	-	-	-	-	-
15	yes	-	-	-	yes
10	yes	-	yes	-	yes
7.5	yes	-	yes	-	yes
5	yes	-	yes	yes	yes
4	-	-	yes	yes	yes
3	-	-	yes	yes	yes
2	-	-	yes	yes	yes
1	-	-	yes	yes	yes
0	-	-	yes	yes	yes
-1	-	yes	yes	yes	yes
-2	-	yes	yes	yes	yes
-3	-	yes	yes	yes	yes
-4	-	yes	yes	-	yes
-5	-	yes	-	-	yes
-7.5	-	-	-	-	-
-10	-	-	-	-	-

Tab. 6: Availability of obstacle information at various ranges, for each sensor

Error / Block Dist. (m)	Stereo Front		Stereo Rear		Dense SfM		Ultrasonic		Fused Repr.	
	Fwd	Bwd	Fwd	Bwd	Fwd	Bwd	Fwd	Bwd	Fwd	Bwd
2.4	0.04	0.1	0.1	0.1	0.26	0.18	0.06	0.05	0.01	0.11
2.7	0.14	0.13	0.16	0.08	0.17	0.19	0.04	0.0	0.03	0.14
3	0.06	0.17	0.13	0.08	0.04	0.26	0.03	0.01	0.03	0.24
3.3	0.13	0.09	0.16	0.11	0.10	0.2	0.02	-0.09	0.1	0.12
3.6	0.07	0.14	0.12	0.16	0.32	0.37	-0.02	0.04	0.05	0.15
3.9	0.06	0.08	0.15	0.13	0.29	0.29	-0.02	-0.02	0.12	0.10
4.2	0.06	n/a	0.11	0.13	0.35	0.38	-0.03	0.02	0.05	0.34
4.5	0.14	n/a	0.1	0.1	0.34	0.26	-0.01	-0.02	0.15	0.29
4.8	0.14	n/a	0.2	0.06	0.38	0.31	0.04	0.03	0.23	0.23
5.1	0.17	n/a	0.23	0.1	0.5	0.37	0.01	0.0	0.18	0.35
5.4	0.16	n/a	0.15	0.0	0.26	0.34	-0.01	0.0	0.19	0.33
5.7	0.13	n/a	0.02	-0.2	0.24	0.36	-0.04	-0.04	0.23	0.35
6.0	0.18	n/a	-0.03	-0.03	0.49	0.3	n/a	n/a	0.25	0.36
RMSE	0.12	0.12	0.14	0.11	0.32	0.3	0.03	0.04	0.15	0.26

Tab. 7: Errors for the distance estimated between foam blocks in the occupancy grid, all units are meters

Mean Error (m) (Std. Dev. (m)) / Phase	Stereo Front	Stereo Rear	Dense SfM	Ultrasonic	Fused Repr.
Phase A	-0.10(0.21)	n/a	-0.05(0.10)	n/a	0.01(0.20)
Phase B	-0.10(0.21)	n/a	0.09(0.05)	0.66(0.19)	0.12(0.05)
Phase C	-0.10(0.21)	-0.13(0.07)	0.08(0.06)	0.24(0.09)	0.21(0.09)
Phase D	-0.10(0.21)	-0.12(0.07)	0.05(0.15)	0.07(0.03)	0.08(0.03)

Tab. 8: Mean error and standard deviation for the parking spot width estimated in the occupancy grid

4 Perception in Dynamic Environment

Beside static obstacle detection, moving objects need to be detected, classified and tracked through onboard sensors, in order to react properly in a dynamic environment. All modules included in the topic of dynamic environment perception are summarized in Fig. 19 and five modules here are evaluated.

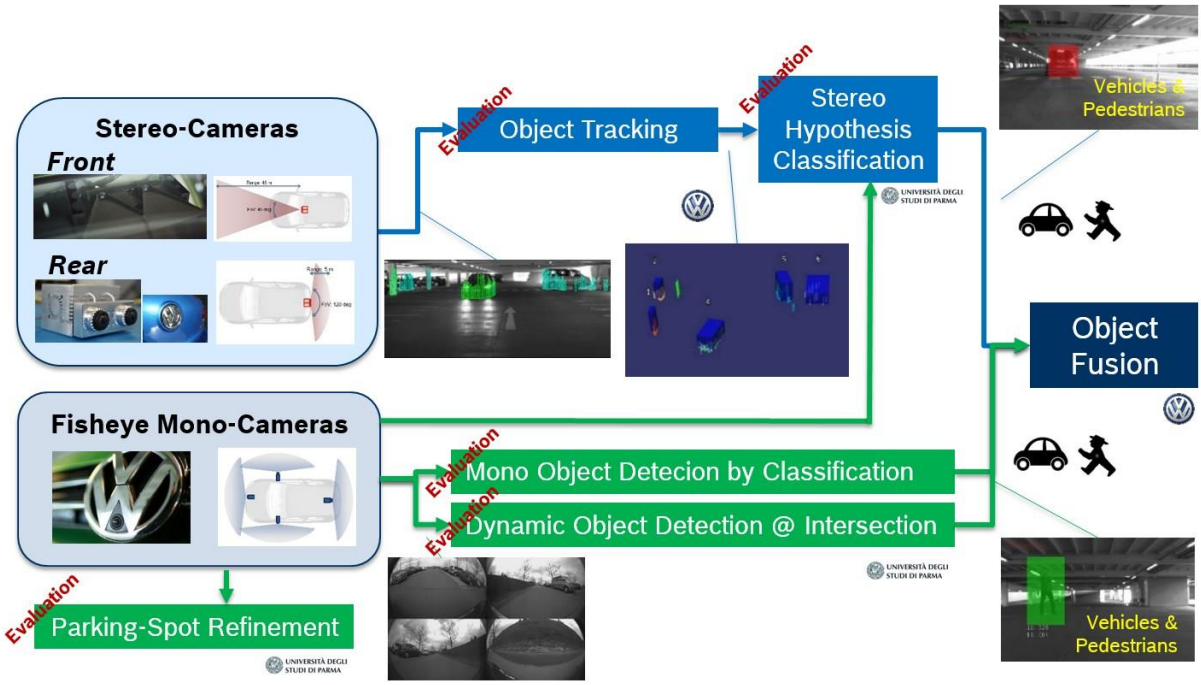


Fig. 19: Overview of modules in Perception in Dynamic Environment

4.1 Object Tracking

VW

To allow planning in mixed traffic scenarios, we have to gather information about other traffic participants, such as pedestrians or vehicles. The stereo camera provides the primary sensor input for detecting these traffic participants. For this purpose, the stereo segments provided by the camera (see Chapter 3.1) are clustered into objects, according to proximity. Objects are associated between samples using a metric nearest neighbor search. We implement a classic predict-measure-update cycle (c.f. [5]) for object tracking. An Extended Kalman Filter is used to estimate the parameters of the objects.

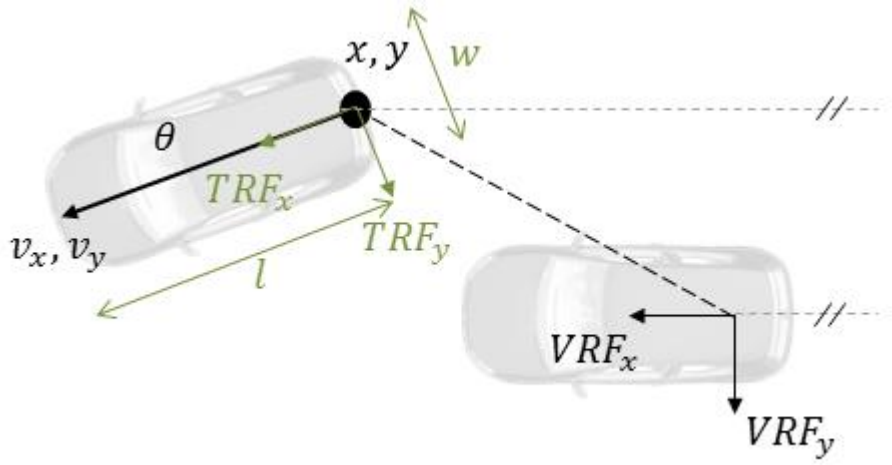


Fig. 20: The parameters x, y, v_x and v_y,and their covariance are estimated in the Vehicle Reference Frame (VRF). This reference frame is attached to the center of the vehicle rear axis of the V-Charge vehicle. Object estimation is performed in a 2D setting, no Z, roll or pitch coordinates are estimated. Length and width of the target vehicle, l, w, are estimated in the target reference frame (TRF, shown in green).

Both vehicles and pedestrians are modeled using the same set of parameters. These parameters are defined in the vehicle reference frame (VRF), as shown in Figure 20. For objects that currently move at low velocities and for objects that have just been initialized, the parameter vector, $\boldsymbol{x}$, is given as,

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \\ l \\ w \end{pmatrix},$$

where x and y represent the object position in the VRF, v_x and v_y represent the velocity in the VRF, l and w represent the length and width of the target vehicle. A full covariance matrix is used to represent the uncertainties of these parameters. The orientation of the tracked object in the VRF, θ, is estimated separately in an additional one-dimensional Kalman Filter, based on the orientation of the stereo segment clusters. Since measurements of this rotation can be very unreliable, it is not used in the prediction step of the EKF.

By itself, the object tracking pipeline does not distinguish between different object classes. All objects clustered from the stereo measurements are treated equal. In order to obtain a class label for an object the predicted object position in the VRF is passed on to the classification pipeline (see Chapter 4.2). This pipeline returns a

label for the objects based on their image appearance in the monocular fisheye camera images. The class information is used by the local planner in order to react correctly to the given traffic situation. Tracked objects are handed over to the planning pipeline if they have been tracked for some minimum amount of time, if their estimated parameter variances are low and if they have been successfully classified.

We have performed experiments in order to determine the metric accuracy and precision of the parameters estimated for the tracked objects. We use two Inertial Navigation Systems (INS), coupled to Differential GPS, to provide ground truth data for these experiments. An INS is mounted on both the V-Charge vehicle, as well as a target passenger car. Both vehicles are driven manually, and the target car was tracked on board the V-Charge vehicle using our object tracking system. For each time step, the INS measurements – made in an arbitrary global frame – are transformed to provide the pose of the target vehicle relative to the V-Charge vehicle; these reference measurements are compared to the output provided by the object tracker.

Parameter	Min. Error	Max. Error	Median Error	Mean Error	Std. Error	
x	0.22	5.66	1.19	1.62	1.33	(m)
y	-0.02	0.11	0.04	0.03	0.04	(m)
v_x	-1.85	-0.14	-0.93	-0.91	0.4	(m/s)
v_y	-0.28	-0.12	-0.2	-0.2	0.04	(m/s)
w	0.02	0.1	0.08	0.08	0.02	(m)

Tab. 9: Errors of the estimated parameters as measured in our experiments. All results are in meters

The results we have obtained for the parameters x, y, v_x and v_y, using this method, are shown in Table 9. Note that the parameters are given in the Vehicle Reference Frame, errors in x, v_x correspond to longitudinal deviations with respect to the V-Charge vehicle, errors in y, v_y to lateral deviations.

At this point, the difficulty of using two separate INS / Differential GPS systems should be noted. Calibration of both reference systems and the object sensors on vehicles, using different sources for GPS correction data, as well as latencies between the GPS and the remaining system introduce additional errors into these measurements. The above numbers can therefore be taken only as a rough indicator of general system behavior. The evaluation of the metric accuracy of the stereo camera itself, as shown in chapter 3.1, can be used as an alternative source for the supposed metric accuracy. More work on the evaluation methodology for the object tracking pipeline could be beneficial.

Nonetheless, we can glean some observations from the performed experiments.

As can be seen from Table 1, the longitudinal (x, v_x) estimates have a much worse error than the lateral (y, v_y) estimates. This is a characteristic of the stereo camera that we used as the primary sensor for providing metric object pose estimates – which has a high angular resolution but is prone to errors in measuring distance. The system also relies on the assumption that the world beneath the vehicle is flat; this is one possible source for the bias (non-zero mean value) that we see in the measurements. Furthermore, the results are skewed to be less accurate as the distance between the vehicles increases. The longitudinal estimates also suffer from bad outlier at large distances, which skew the results. The lateral estimates are surprisingly accurate; they are well within the range necessary in order to unambiguously assign a tracked object to a lane.

Similar to the lateral position and velocity estimates, the width of the target vehicle is also estimated with a small error. Due to the nature of the scenario – following another vehicle – we are unable to get reliable observations of the target vehicle's length.

Even though the above experiments show biased errors, in practice, the system provides adequate performance for enabling automated driving in the presence of other traffic participants. As we have seen in several live vehicle demonstrations, we are able to engage in stop-and-go driving behind another vehicle or a pedestrian, overtake static vehicles, and stop before crossing pedestrians. Perhaps, the metric accuracy of the estimated parameters does not prove to be critical for the types of scenarios we participate in. For example, we are able to smoothly follow another car, even though we have biased estimates for longitudinal positions and velocities.

4.2 Stereo Hypothesis Classification

Pietro Versari, Parma

We developed and integrated a system that is able to distinguish between static and dynamic obstacles. In order to achieve this capability we chose to classify object hypothesis obtained by a clustering step of stereo stixels.

This module re-projects obstacles onto a virtual plane in order to obtain a region of interest, which is used by a classifier to validate or reject the hypothesis. More details on this system are described in a scientific paper [6].

To be able to make meaningful statements about the accuracy of this system we evaluated the classifier performance on commonly used automotive data-sets. It is important to note the following results are obtained without tracking nor exploiting the stereo information: for this reason they have to be intended as lower bound.

The classifier performance is described using different metrics: in the first two plots we use the *precision/recall* metric, where precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved (Fig. 21). The plot on the left is related to the vehicle classifier while on the right to the pedestrian classifier.

Vehicle (left) and pedestrian (right) classifier evaluation: precision/recall metric

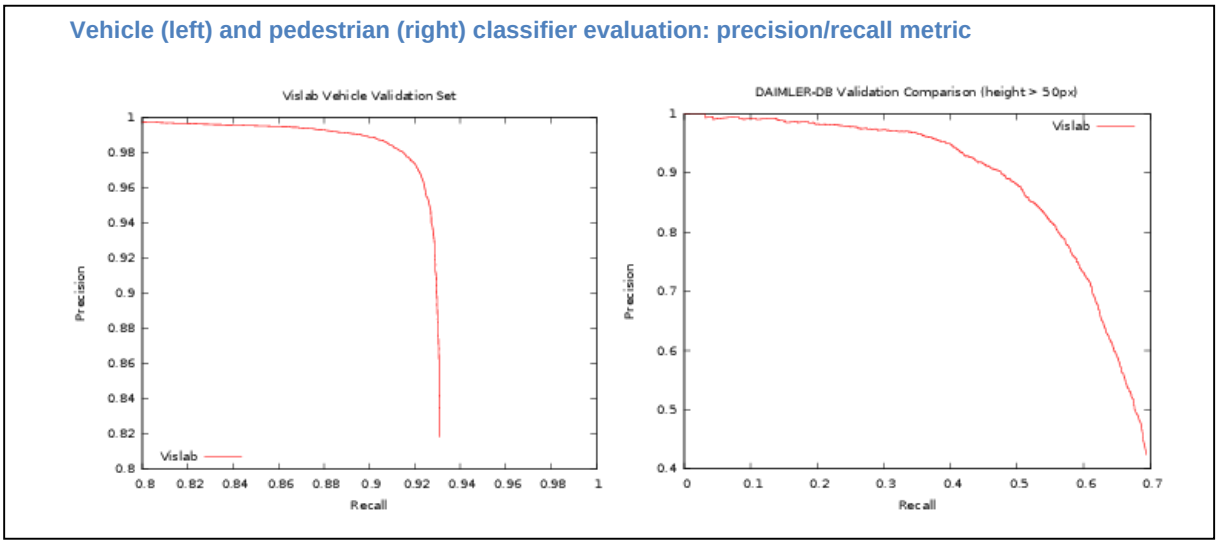


Fig. 21: Classifier performance – Precision vs Recall

The second metric used is the *miss-rate* metric (Fig. 22): as before the plot on the left is related to the vehicle classifier while on the right to the pedestrian classifier.

Vehicle (left) and pedestrian (right) classifier evaluation: miss-rate metric

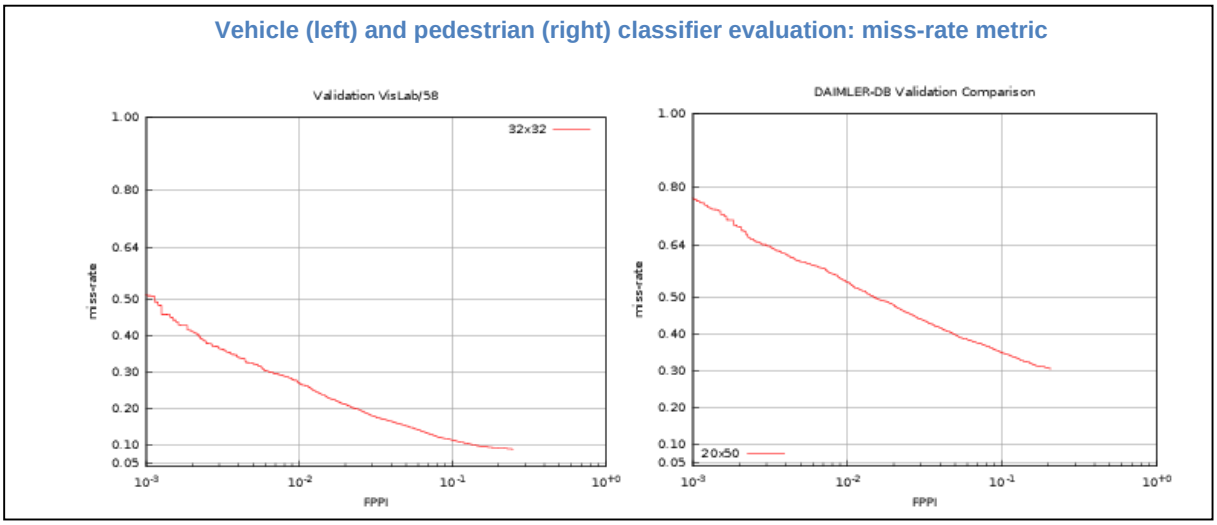


Fig. 22: Classifier performance – Miss-Rate vs False-Positive-Per-Image (FPPI)

The presented results are in line with the state-of-the-art.

As part of the final demonstration preparation many systems have been tested for a long time, in order to reach the reliability needed by public demonstrations; the scenarios where this system is involved have been tested with a success rate higher than 95%.

4.3 Mono Object Detection by Classification

Pietro Versari, Parma

We developed and integrated a system that is able to detect and track both pedestrian and vehicle all around the car. Three main steps are performed:

- **Image un-warping**: the fisheye images are un-warped in order to both correct the lens distortion and obtain a wide-angle view without strong aberrations.
- **Classification**: a Soft-Cascade+ACF classifier is run on each camera image to detect vehicles and pedestrians on all the surrounding space.
- **Multi-camera tracking**: in order to perform the 360° tracking, firstly an inter-camera association algorithm is performed then, an Unscented Kalman Filter is used to exploit those information to track and generate an all-around description, as an unique high-level sensor with 360° field of view

More details on this system are described in the scientific paper [7].

Since there aren't suitable public data-sets with fish-eye cameras in a similar configuration, we analyzed how the cylindrical model impacts on classifier performance. In [7] is presented a theoretical analysis and following are presented the results; the results related to the cylindrical model have been obtained by warping the original pinhole image using its intrinsic parameters (Tab. 10). The dataset used for this comparison is *KITTI Vision Benchmark*.

Camera model / classifier dimension / class	True Positive	True Negative	False Negative	False positive	Correct Detection	**Difference**
Pinhole / 32x64 / pedestrian	2521	123222	4272	2	125743 (96.7127%)	**0.0023%**
Cylindrical / 32x64 / pedestrian	2524	123222	4269	2	125746 (96.7150%)	
Pinhole / 48x96 / pedestrian	1142	64750	1266	0	65892 (98.1148%)	**-0.0104%**
Cylindrical / 48x96 / pedestrian	1135	64750	1273	0	65885 (98.1044%)	
Pinhole / 44x44 / vehicle	524	47382	10214	0	47906 (82.4260%)	**0.0636%**
Cylindrical / 44x44 / vehicle	487	47382	10251	0	47869 (82.3623%)	

Tab. 10: Classification performance on pinhole / cylindrical images

The results obtained with these two camera models are extremely similar: the cylindrical model low impacts on classifier performance allows us to assume the classifier precision presented for the “stereo hypothesis classification” module do not decrease.

The developed algorithm exploits the system characteristics – like cameras overlapping field-of-view – and tracks obstacles over the time: these techniques increase the reliability, the accuracy and the performance of the whole system. We evaluated the number of correct detection and the false detection on different sequences obtaining the following results (Tab. 11).

	Correct detections	False detections
Pedestrian	1179 (95.9%)	51 (4.1%)
Vehicle	2920 (98.1%)	57 (1.9%)

Tab. 11: Classification performance of overall pipeline (Distortion Compensation, Classification, Multi-Camera Association, Tracking)

4.4 Dynamic Object Detection at Intersections

Pietro Versari, Parma

To perform the intersection scenario, the vehicle has to detect all dynamic objects in the scene with a particularly high reliability due to the high relative speeds.

For this scenario we specifically developed and integrated an algorithm exploiting its characteristic: while the vehicle is standing all the moving features in the scene are related only to dynamic obstacles. To detect these obstacles the features are clustered, then each cluster is grouped with its neighbors and tracked over the time; subsequently the three different point of views (left, front and right cameras) are exploited to increase both the accuracy and reliability.

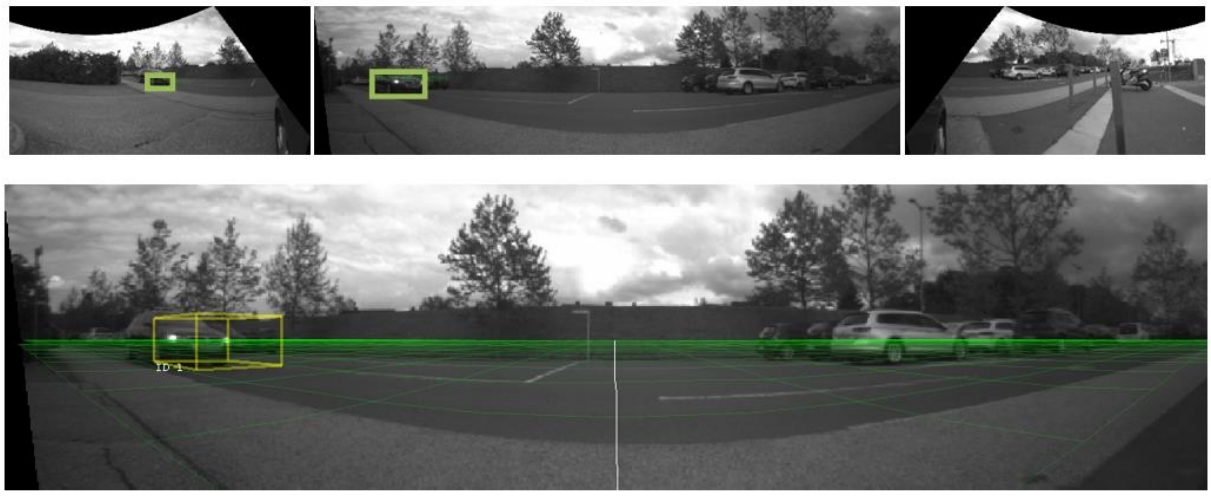


Fig. 23: Dynamic Object Detection at Intersections

The working principle is shown in Fig. 23. The first row of images show the different point of views and the features clustered, meanwhile the bottom image shows the 3D obstacle detected.

As part of the final demonstration preparation many systems have been tested for a long time, in order to reach the reliability needed by public demonstrations. For this specific scenario more than 15 GB of data have been analyzed counting the number of events with correct detection, miss-detection and false detection, obtaining the following results (Tab. 12).

	Dynamic Obj.	Detected	Not Detected	Wrongly Detected
Near lane	18	17 (94.4%)	1 (5.5%)	0
Far lane	20	20 (100%)	0	0

Tab. 12: Detection performance at intersections

4.5 Parking Spot Refinement

Pietro Versari, Parma

Errors in vehicle position estimation impacts the parking maneuver accurateness. We developed and integrated two different algorithms to update and refine the target parking spot information, to mitigate the impact of the vehicle positioning error on the parking maneuver.

Two different algorithms have been implemented to manage different types of parking spot: the first one is able to detect “tile grid” parking spots as shown in Fig. 25, meanwhile the second one works with “white stripes” bounded parking spot depicted in Fig. 24.

The evaluation was done by measuring the angle between the parking spot and the vehicle, and comparing it with the angle obtained by the algorithms.

The “tile grid” parking spot algorithm gets an average error of 0.11° with 0.78° maximum error. The “white stripes” parking spot algorithm gets an average error of 0.54° with 0.94° maximum error.

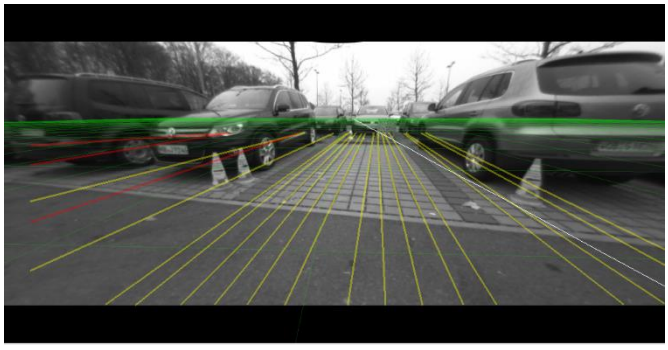


Fig. 25: Algorithm for „Tile Grid“ parking spots

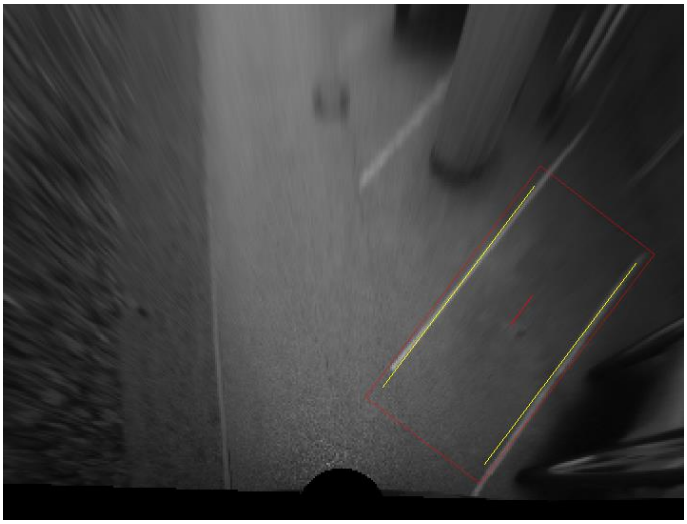


Fig. 24: Algorithm for parking spots with „White Stripes“

5 Analysis of Problems during Test Drives

In the past, a lot of test drives have been done with the test vehicles and some problems have been observed. Problems appearing in vehicle behavior (unexpected or “strange” behavior) need to be analyzed by tracing back on the module chain in the system. In this section, the “cross-module” analysis for some issues observed during test drives are introduced.

5.1 Problem at ACC Scenario

In the ACC scenario, the ego vehicle is required to follow a leading vehicle. In order to realize this behavior, the perception pipelines need to provide a correct classification of the leading vehicle as a “vehicle”.

However, the ego vehicle sometimes showed maneuvering behavior (tried to overtake the leading vehicle) during test drives. The problem was analyzed on the side of the on-lane motion planner in WP4 (refer to D4.4) first of all. The analysis showed that the leading vehicle was not classified as a “vehicle” correctly but partially detected as static obstacles. A part of the leading vehicle stays as static obstacles on the grid map (Fig. 26). That’s why the ego vehicle tried to avoid the leading vehicle which partially appears on the grid map as a static obstacle.

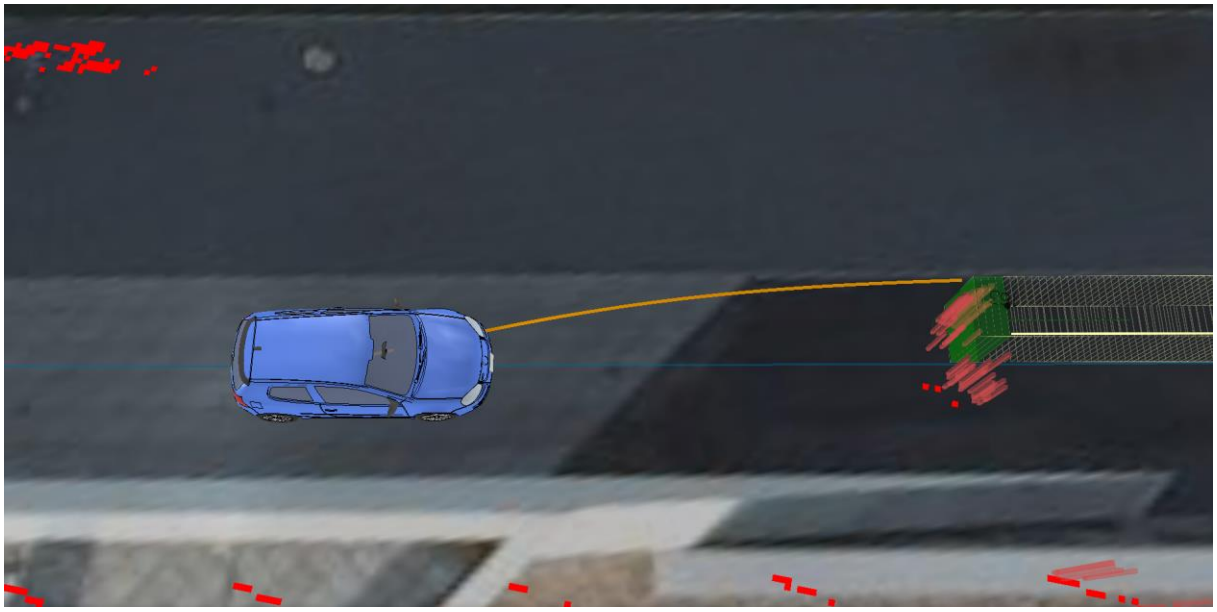


Fig. 26: Unexpected maneuvering at ACC scenario

The reason why the leading vehicle was not classified as a “vehicle” correctly was analyzed in WP3 perception pipeline. After tracing back outputs from different modules on the perception pipeline, it was found that the stereo segments (output from the stereo camera) in the middle of the leading car were missing (Fig. 27). In this case, the leading vehicle is not classified as a “vehicle”, but two separate static obstacles. An example of the stereo segments output which results in a correct behavior (to follow the leading vehicle) in the ACC scenario is shown in Fig. 28.

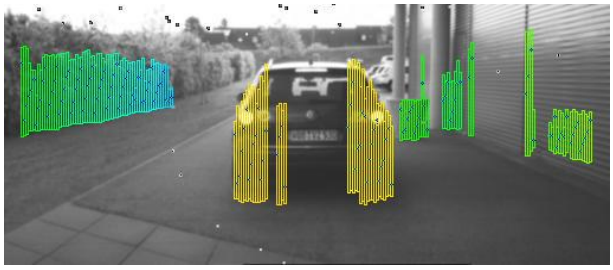


Fig. 27: Missing stereo segments in the middle of the leading vehicle

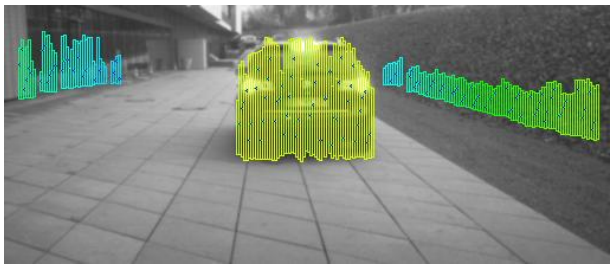


Fig. 28: Stereo segments resulting in a „correct“ behavior in the ACC scenario

There are several reasons why stereo processing pipeline could behave like this. First of all, cars in general have a lot of non-structured surfaces at the rear but the stereo measuring principle needs structures. Second point is that the leading vehicle was dark gray. Such dark body colors might have a problem to distinguish car bodies from dark asphalt ground especially in dark light condition. Finally, a prototype camera installed here might have impact from the weather conditions and mud on the windshield.

There was a countermeasure implemented in the module of “Object Tracking” by VW for such cases: if there is an unclassified object next to an object classified as “vehicle” and the outer edges of the objects would create an object with the width of a car, then the unknown object is also classified as a “vehicle”. This countermeasure solved about 80% of the described problems. As a further countermeasure to make sure that the system works 100% correctly in the ACC scenario, a dark colored vehicle is avoided as a leading vehicle in demonstrations.

6 Conclusion

This deliverable has reported on the final integration status and results of performance evaluation for all components involved in the work package 3.

All necessary components required to realize targeted use-cases have been integrated successfully as shown in Section 1.

Performance of components have been evaluated and the results are described in Section 2 – 4. Outputs from the perception pipelines (localization, grid map and dynamic object list) are provided to the navigation modules in WP4 and test vehicles have behaved as designed in targeted use-cases.

All in all, research and integration works in WP3 have been successful during the project V-Charge and the targeted use-cases have been realized based on the outputs from the WP3 pipelines, which resulted in successful final demonstration in Wolfsburg and press demonstrations in Amsterdam.

7 References

[1] P. Muehlfellner, M. Buerki, M. Bosse, W. Derendarz, R. Philippsen and P. Furgale, "Summary Maps for Lifelong Visual Localization," *Journal of Field Robotics,* 2015.

[2] C. Häne, L. Heng, G. H. Lee, A. Sizov and M. Pollefeys, "Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo, Proc Int. Conf. on 3D Vison (3DV) 2014".

[3] C. Häne, T. Sattler and M. Pollefeys, "Obstacle Detection for Self-Driving Cars Using Only Monocular Cameras And Wheel Odometry, submitted," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[4] H. P. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," in *IEEE International Conference, Procesdings "Robotics and Automation"*, 1985.

[5] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, MIT Press, 2005.

[6] U. Schwesinger, P. Versari, A. Broggi and R. Siegwart, "Vision-Only Fully Automated Driving in Dynamic Mixed-Traffic Scenarios," *Journal of Information Technology: it - Information Technology, Special Issue Autonomous Driving,* 2015.

[7] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti and P. Versari, "360° Detection and Tracking Algorithm of Both Pedestrian and Vehicle using Fisheye Images," in *IEEE Intelligent Vehicle Symposium (IV)*, 2015.

Real-Time Direct Dense Matching on Fisheye Images Using Plane-Sweeping Stereo

Christian Häne*, Lionel Heng, Gim Hee Lee†, Alexey Sizov, Marc Pollefeys

Department of Computer Science, ETH Zürich, Switzerland

Abstract

In this paper, we propose an adaptation of camera projection models for fisheye cameras into the plane-sweeping stereo matching algorithm. This adaptation allows us to do plane-sweeping stereo directly on fisheye images. Our approach also works for other non-pinhole cameras such as omnidirectional and catadioptric cameras when using the unified projection model. Despite the simplicity of our proposed approach, we are able to obtain full, good quality and high resolution depth maps from the fisheye images. To verify our approach, we show experimental results based on depth maps generated by our approach, and dense models produced from these depth maps.

1. Introduction

A camera is a low-cost and information-rich sensor that is suitable for many robotics applications such as obstacle detection and motion estimation. In these applications, a fisheye camera prevails over the conventional camera because it has a wider field-of-view that offers more coverage of the environment. Examples of successful use of fisheye cameras for robotics applications are Zingg $et.\ al$ [25] and Lee $et.\ al$ [14, 15], where fisheye cameras are used to perform optical flow for obstacle detection on a Micro-Aerial Vehicle (MAV) and motion estimation on a self-driving car. In these works, additional coverage provided by the fisheye cameras proved to be critical for the MAVs to gain maximum information of its surrounding for safe navigation, and increases the number of feature correspondences which improves motion estimation for the car. The use of fisheye cameras is not only limited to robotics applications. In recent years, we have seen an increasing number of mass-market cars such as the Nissan Qashqai and Honda Odyssey that are equipped with multiple fisheye cameras to provide drivers with a surround view for parking assistance. Another recent application of fisheye cameras is on hand-held mobile devices[1].

As compared to robotics and driving assistance applications, the use of fisheye cameras is limited in dense mapping applications probably due to the difficulty in handling the large distortion inherent in fisheye images for dense matching. Existing works in dense mapping such as the Urbanscape project [17] use multiple pinhole cameras to maximize coverage. The advantages of fisheye cameras for robotics applications, the convenience from the availability of these cameras on mass-market cars and the limited usage of fisheye cameras for dense mapping due to the large distortions motivate us to work on a simple yet effective model to produce depth maps from direct dense matching on fisheye images. We show that depth maps can be directly obtained from the fisheye camera by a simple adaptation of suitable fisheye camera models into the widely used plane-sweeping algorithm [4, 6]. The only requirement we need for the fisheye model is that both projection and unprojection can be done efficiently. In this work we use two different camera models, the unified projection model [8, 2, 16] and the field-of-view (FOV) model [5].

We choose to do plane-sweeping for dense matching because, in comparison to the rectification approach [18], the plane-sweeping approach can be done on more than two images. In addition, the plane-sweeping approach is particularly well-suited to Graphics Processing Unit (GPU) implementations for achieving real-time performance. We show experimentally that the alternative approach that first converts the fisheye images into pinhole images followed by plane-sweeping leads to a significant reduction in the coverage area. Our approach is only slightly slower than the standard plane sweep method, which uses planar homographies on the pinhole model, and thus still runs in real-time on a desktop computer. For mobile devices, high performance mobile GPUs have been released recently, for example the NVIDIA Tegra K1. Using devices equipped with such GPUs, there is a potential to reach interactive frame rates on mobile devices. Most importantly, we show that despite the simplicity of our proposed approach, we are able

*chaene@inf.ethz.ch

†Now at Mitsubishi Electric Research Laboratories

[1] www.google.com/atap/projecttango

to obtain full, good quality and high resolution depth maps from the fisheye images in arbitrary camera configurations without the need of prior rectification. It should be noted that the unified projection model [8, 2, 16], which we use for our approach also works for other non-pinhole cameras such as omnidirectional and catadioptric cameras.

In contrast to the existing works mentioned earlier in MAV navigation, motion estimation and sparse mapping for self-driving cars, which leveraged only sparse features from the fisheye images, our depth maps computed directly from the fisheye images provide more comprehensive information about the surrounding while retaining the advantages of fisheye cameras. We further demonstrate the potential of our approach by using the depth maps computed directly from the fisheye images to build dense maps of the environment. In comparison to the dense model built from the depth maps computed by the plane-sweeping algorithm on pinhole images [6] in the Urbanscape project [17], the dense model built from the depth maps which are computed directly from the fisheye images in our approach are shown to cover a larger part of the scene with fewer images and similar runtime performance.

1.1. Related Work

Rectified two-view and multi-view stereo matching are well established algorithms in computer vision. The main difference between these two methods is that the former works on only two rectified images which simplifies image matching along the aligned epiploar lines on the rectified images, while the latter works for any number of images. More details of the rectified two-view stereo can be found in [18]. In cases with more than two images, image matching along aligned epipolar lines is possible only in some special camera configurations. In addition, rectification fails when the epipoles fall within or too close to the borders of the images and this problem is exacerbated when wide field-of-view images are used as it is more probable that the motion direction lies inside the image boundaries. To overcome these problems, the plane-sweeping algorithm was introduced in [4]. This approach allows direct matching on multiple images without the need for rectification. Gallup $et.\ al$ [6] showed that the quality of the depth maps is improved by aligning the sweeping plane with the predominant orientations of the scene. In addition to the ease of matching images taken from arbitrary configurations, the plane-sweeping algorithm has the advantage of real-time performance which can be achieved by utilizing the texturing capabilities of GPUs [22, 6]. With such high performance stereo matching algorithms, dense reconstruction of whole cities becomes feasible [17].

While most of the works on stereo matching made use of pinhole images, limited works are shown on non-pinhole images such as omnidirectional, catadioptric or fisheye cameras. In [12, 24], Ishiguro $et.\ al$ and Zhu $et.\ al$ showed the first approaches on omnidirectional stereo vision that created a horizontal 360° panorama stereo pair by rotating a camera around a center point. In a later work [9], multiple catadioptric cameras were used to produce similar horizontal omnidirectional stereo pairs. These approaches, however, share the limitation that the field-of-view is large in only one direction. Catadioptric and fisheye cameras are able to capture images with an opening angle of 180° and beyond. One straightforward way to match the whole field-of-view of omnidirectional images would be to extract a set of pinhole images that cover the whole field of view. However, this leads to a significant increase in running time as many more images have to be matched. For specific tasks such as obstacle detection in automotive applications, [7] proposed to extract multiple pinhole images out of a single omnidirectional image for dense stereo matching only in important directions for obstacle detection. In [1], Abraham $et.\ al$ proposed an approach that rectifies fisheye images and preserves most of the field-of-view. However, such approaches do not generalize to arbitrary camera configurations.

2. Stereo Matching on Fisheye Images

In this section, we first introduce the generic notation for the camera projections that are suitable for our method. The details for the specific models can be found in [8, 2, 16] for the unified projection model and in [5] for the FOV model. Next, we describe in detail how the fisheye projection model is adapted into the plane-sweeping stereo matching algorithm.

2.1. Camera Projection Model

The camera projection model describes how a point is mapped to the image plane and its inverse function computes the corresponding ray in 3D space for a given image pixel. For our application, it is important that both of the functions can be evaluated efficiently.

Given a 3D point $\mathbf{X} = [X_x, X_y, X_z]^T$ in the camera frame F_C, the function $\mathbf{x}_u = \hbar(\xi, \mathbf{X})$ maps the 3D point $\mathbf{X}$ to an undistorted image point $\mathbf{x}$ on the normalized image plane. ξ is a single scalar parameter that models the fisheye lens. Imperfections of the camera lens cause additional radial and tangential distortions to the image. This transforms the normalized undistorted image coordinate $\mathbf{x}$ to the normalized distorted image coordinate $\mathbf{x}'$ according to the plumb blob model [3], which we denote as $\mathbf{x}' = D(\mathbf{x})$. Applying the camera projection matrix $\mathbf{K}$ on $\mathbf{x}'$, we get the image coordinate $(\tilde{x}', \tilde{y}')$ that we observed from an image captured with the fisheye camera. The intrinsic parameters $\boldsymbol{\phi} = [\xi,\ k_1,\ k_2,\ k_3,\ k_4,\ k_5,\ f_x,\ f_y,\ C_x,\ C_y]$ are obtained from the camera calibration. $[k_1,\ k_2,\ k_3,\ k_4,\ k_5]$ and $[f_x,\ f_y,\ C_x,\ C_y]$ are from the distortion model and

camera projection matrix respectively. We denote the unprojection, which maps a point $\mathbf{x}$ on the undistorted image plane to a ray by $\mathbf{X} = \hbar^{-1}(\xi, \mathbf{x})$.

The details of the individual camera models can be found in [16] and [5] for the unified projection model and the FOV model, respectively. For our work the important nature of these two camera models is that both the projection $\hbar$ and unprojection $\hbar^{-1}$ can be stated in closed form and hence be computed efficiently on the GPU. This is important as on the GPU a look up table is slower than the direct computation.

2.2. Plane-Sweeping for Fisheye Images

Plane-sweeping stereo matching [4] allows for dense stereo matching between multiple images without prior stereo rectification. Existing implementations such as [6] work for cameras that follow the standard pinhole model. We propose to extend the plane-sweeping stereo matching for fisheye cameras by incorporating the fisheye projection model directly into the plane-sweeping stereo matching algorithm. We closely follow [6] in the description of our algorithm.

The inputs to our plane-sweeping algorithm for fisheye cameras are a set of fisheye images, $\mathcal{I}' = \{I'_1, \cdots, I'_n, \cdots, I'_N\}$, and the camera parameters, $\mathcal{P} = \{P_1, \cdots, P_n, \cdots, P_N\}$ associated with the respective images. Each camera parameter $P_n = \{\boldsymbol{\phi}_n, \mathbf{R}_n, \mathbf{C}_n\}$ is made up of the camera intrinsics $\boldsymbol{\phi}_n$ and pose $[\mathbf{R}_n, \mathbf{C}_n]$. To keep our plane-sweeping algorithm for fisheye cameras simple and avoid a costly look up table on the GPU, we do a pre-processing step to remove the tangential and radial distortions from the fisheye images. Each pixel $[\tilde{x}, \tilde{y}, 1]^T$ on the set of undistorted images $\mathcal{I} = \{I_1, \cdots, I_n, \cdots, I_N\}$ is mapped onto $\mathcal{I}'$ by

$$[\tilde{x}', \tilde{y}', 1]^T = \mathbf{K} D\left(\mathbf{K}^{-1}[\tilde{x}, \tilde{y}, 1]^T\right) \tag{1}$$

This allows us to efficiently do a look-up for the corresponding pixel $[\tilde{x}', \tilde{y}', 1]^T$ in the fisheye image for each pixel $[\tilde{x}, \tilde{y}, 1]^T$ in the undistorted image $\mathcal{I}$.

Let us choose $I_{\text{ref}} \in \mathcal{I}$ as a reference view for the computation of a depth map. Additionally, we define a set of plane hypotheses $\Pi = \{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_m, \cdots, \boldsymbol{\Pi}_M\}$. The main idea of the plane-sweeping algorithm is to use these planes Π as the hypothetical local reconstructions of the scene, where we choose the best plane for each pixel by evaluating an image dissimilarity measure. Each plane hypothesis is defined as $\boldsymbol{\Pi}_m = [\mathbf{n}_m^T, d_m]$, where $\mathbf{n}_m$ is the unit length normal vector of the plane pointing towards the camera center and d_m the distance of the plane to the camera center. To evaluate a plane hypothesis $\boldsymbol{\Pi}_m$, the pixels of the image I_n are mapped onto the reference view by projecting the image I_n onto plane $\boldsymbol{\Pi}_m$ followed by rendering it onto the reference view P_{ref}, we call this $I^{\text{ref}}_{n,m}$. In projective geometry, such a planar mapping is described by a homography. Using the fisheye projection model described in the previous section, the mapping that directly uses the pixels $[\tilde{x}_n, \tilde{y}_n, 1]^T$ from the undistorted fisheye image is given by

$$\begin{aligned}\mathbf{H}^{\text{ref}}_{n,m} &= \mathbf{R}^T_n \mathbf{R}_{\text{ref}} + \frac{1}{d_m}(\mathbf{R}^T_n \mathbf{C}_n - \mathbf{R}^T_n \mathbf{C}_{\text{ref}})\mathbf{n}^T_m \\ [\tilde{x}_n, \tilde{y}_n, 1]^T &= \mathbf{K}_n \hbar(\mathbf{H}^{\text{ref}}_{n,m} \hbar^{-1}(\mathbf{K}^{-1}_{\text{ref}}[\tilde{x}_{\text{ref}}, \tilde{y}_{\text{ref}}, 1]^T))\end{aligned} \tag{2}$$

We now warp image I_n to an image $I^{\text{ref}}_{n,m}$ with Equation 2, which we compare against the reference image I_{ref}. In particular, the image dissimilarity is evaluated between all the warped $I^{\text{ref}}_{n,m}$ and reference I_{ref} image, $\forall n \neq \text{ref}$ and for all planes $\Pi = \{\boldsymbol{\Pi}_1, \cdots, \boldsymbol{\Pi}_m, \cdots, \boldsymbol{\Pi}_M\}$. In our experiments, we use the negative zero mean normalized cross correlation (ZNCC) over a correlation window $\mathcal{W}$ for the evaluation of image dissimilarity (see Equation 3).

Evaluating the ZNCC matching scores can be done in constant time independent of the window size by using integral images [21]. As memory access on GPUs is often slower than computing the required values, Stam [19] proposes the use of a box filter which maintains a running sum in the vertical direction and sums up the horizontal direction by summing up the values in shared memory. We utilize a sequence of five times box filtering to compute the ZNCC scores according to [21] leading to a running time that increases linearly with the matching window width.

2.3. Cost Aggregation

We use the aggregated cost $\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y})$ in cases where more than one image gets matched to the reference image. There are different options to compute the aggregated cost, with the choice of taking occlusions into account [13]. For this work we used two of them:

1. All matching costs are averaged. For example, with $2k$ images, k images before and k after the reference image,

$$\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y}) = \frac{1}{2k} \sum_{n=\text{ref}-k}^{n=\text{ref}+k} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}), \quad n \neq ref \tag{4}$$

2. Average cost from best half of the sequence. This assumes that the images are taken in a sequence with a predominant motion direction. First, we take the average cost over the half sequences on both sides of the reference view. Next, the final matching cost is the minimum of the two averages.

$$\mathcal{D}^{\text{ref}}_m(\tilde{x}, \tilde{y}) = \frac{1}{k} \min\Big\{ \sum_{n=\text{ref}-k}^{n=\text{ref}-1} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}), \sum_{n=\text{ref}+1}^{n=\text{ref}+k} \mathcal{D}^{\text{ref}}_{n,m}(\tilde{x}, \tilde{y}) \Big\} \tag{5}$$

$$\mathcal{D}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y}) = \frac{-\sum_{(i,j)\in\mathcal{W}} \{I_{\text{ref}}(\tilde{x}+i,\tilde{y}+j) - \overline{I}_{\text{ref}}(\tilde{x},\tilde{y})\}\{I^{\text{ref}}_{n,m}(\tilde{x}+i,\tilde{y}+j) - \overline{I}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y})\}}{\sqrt{\sum_{(i,j)\in\mathcal{W}} \{I_{\text{ref}}(\tilde{x}+i,\tilde{y}+j) - \overline{I}_{\text{ref}}(\tilde{x},\tilde{y})\}^2 \sum_{(i,j)\in\mathcal{W}} \{I^{\text{ref}}_{n,m}(\tilde{x}+i,\tilde{y}+j) - \overline{I}^{\text{ref}}_{n,m}(\tilde{x},\tilde{y})\}^2}} \tag{3}$$

$$\overline{I}(\tilde{x},\tilde{y}) = \frac{1}{|\mathcal{W}|}\sum_{(i,j)\in\mathcal{W}} I(\tilde{x}+i,\tilde{y}+j)$$

This option handles occlusion with the assumption that each half of the sequence sees a different view of the occlusion.

2.4. Depth Extraction and Sub-Pixel Interpolation

We adopt the winner-takes-all strategy to decide the final depth value for fast processing. In particular, the plane $\mathbf{\Pi}_m$ with the lowest aggregated matching cost for each pixel in the reference view is chosen.

$$\hat{m}(\tilde{x},\tilde{y}) = \underset{m}{\text{argmin}}\ \mathcal{D}^{\text{ref}}_m(\tilde{x},\tilde{y}) \tag{6}$$

Finally, the final depth Z is computed as

$$Z_m(\tilde{x},\tilde{y}) = -d_m\left(\mathbf{n}_m^T \hbar^{-1}\left(\mathbf{K}^{-1}_{\text{ref}}[\tilde{x},\tilde{y},1]^T\right)\right)^{-1} \tag{7}$$

with the fisheye model taken into account.

Depths are extracted at discrete plane positions using Equation 7 and this leads to non-smooth surfaces. We circumvent this problem with sub-pixel interpolation [20]. The idea is to look at the matching scores of the neighboring planes, fit an interpolation function through the matching scores and extract the depth at the extremal point of the interpolated matching cost. We fit a parabola using the scores of the two neighboring planes in our implementation.

3. Experimental Evaluation

3.1. Camera Setups

We use three different camera setups to experimentally validate the dense matching on fisheye images. The first camera setup consists of four CCD cameras with fisheye lenses that each capture a 185° field-of-view. These cameras provide 1280×800 images at 12.5 Hz, and are integrated into a Volkswagen Golf car in such a way that the camera setup provides a surround view with a minimal overlapping field of view between any two cameras. Similarly, the second camera setup comprises four CMOS cameras with 185° fisheye lenses. These cameras provide 754×480 images at 15 Hz, and are installed on an AscTec Firefly hexacopter. The first and second cameras are arranged in a forward-looking stereo configuration while the third and fourth cameras are arranged in a rear-looking stereo configuration. The third camera setup consists of a single fisheye lens which is installed on a mobile hand-held platform. For all camera setups, the cameras are hardware-synchronized. Where necessary, we use SLAM-based calibration to obtain both the camera intrinsics and extrinsics [11, 10].

The camera poses that are given to our plane-sweeping algorithm are computed using visual odometry combined with wheel odometry and inertial meaturements where available.

3.2. Run-Time Performance

In this section, we evaluate the performance of the fisheye plane sweep. We implemented both the standard pinhole plane sweep and our proposed fisheye extension using Nvidia CUDA. Our test platform is equipped with a GeForce GTX 680 GPU with 4GB of memory.

For all the experiments, we precomputed the poses and stored them to disk as a preprocessing step. In the actual experiment, we load the poses and the images from disk. As for many applications, the images do not need to be loaded from disk because they are directly captured by a camera. Hence, we do not include in our measurements the time spent on loading the images from disk. The reported times include the time to upload the images to the GPU, undistort them or extract pinhole images respectively, run the plane sweep, and download the depth map. As our images are taken in sequence, we only upload each image to the GPU once, and use the same image to compute multiple depth maps.

In the first experiment, we use images taken in a sequence from a moving car. The native resolution of the images is 1280×800. In the process of undistortion or pinhole image extraction, we downsample them by half such that we obtain an image resolution of 640×400. We match two consecutive images using 64 planes, ZNCC matching scores, no occlusion handling, and subpixel interpolation. Some example depth maps and the running times are shown in Fig. 1. For a 9×9 ZNCC matching window, the fisheye and pinhole versions generate depth maps at 59.8 Hz and 65.1 Hz respectively. The fisheye version has a 8.96% longer running time but we obtain depth maps that cover the entire field of view of the camera.

In the next experiment, we measure the performance using half-sequence occlusion handling. Again, we use

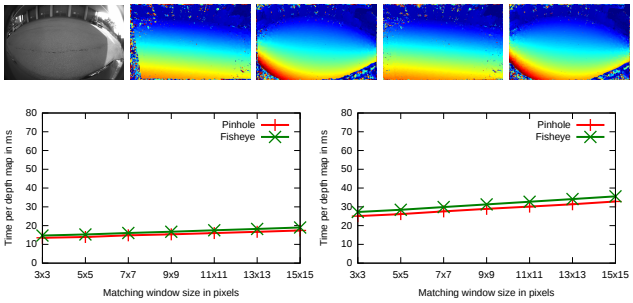


Figure 1. First row (from left to right): An input image, example pinhole and fisheye depth maps with 9×9 ZNCC for the two-view and three-view case. Second row (from left to right): Running times for the two-view and three-view cases averaged over 1667 and 1666 depth maps respectively.

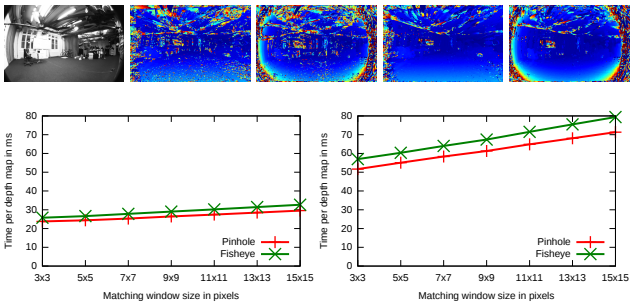


Figure 2. First row (from left to right): An input image, example pinhole and fisheye depth maps with 9x9 ZNCC for the two-view and four-view case. Second row (from left to right) Running times for the two-view and four-view cases averaged over 569 depth maps.

the half-resolution images with 64 planes, ZNCC matching scores, and subpixel interpolation. We match three images, choose the middle image as the reference image, and do half-sequence occlusion handling. The results can be seen in Fig. 1. For a 9×9 window, the fisheye and pinhole versions generate depth maps at 32.0 Hz and 34.6 Hz respectively. The fisheye version has a 8.15% longer running time.

We also use images from a MAV. The native resolution of the images is 754×480. We undistort the images or extract pinhole images with the same resolution. Each pair of cameras is set up in a stereo configuration. For each stereo camera, we use two images captured at the same time and match them. We use 96 planes, ZNCC matching scores, no occlusion handling, and subpixel interpolation. An example image and depth maps can be seen in Fig. 2. We notice that some noise in the depth map occurs due to some parts of the MAV being visible in the images. With a 9×9 ZNCC window, depth maps are generated at 37.8 Hz and 34.4 Hz for the pinhole and fisheye versions respectively. In this case, the running time for the fisheye version is longer by 9.8%.

In the last experiment, we use two stereo image pairs captured consecutively. Again, we use the full 754×480 resolution with 96 planes. We use ZNCC matching scores, no occlusion handling, and subpixel interpolation. Example depth maps and a plot of time measurements can be found in Fig 2. The performance for a 9×9 ZNCC window is 16.3 Hz for the pinhole version and 14.8 Hz for the fisheye version. Here, the running time for the fisheye version is longer by 9.9%.

All our measurements show that the increase in running time from using our proposed plane sweep with the unified projection model on fish eye images is below 10%. However, from running one plane sweep, we obtain a depth map that covers the entire field of view of the fisheye camera, and this advantage far outweighs the performance penalty.

3.3. Coverage

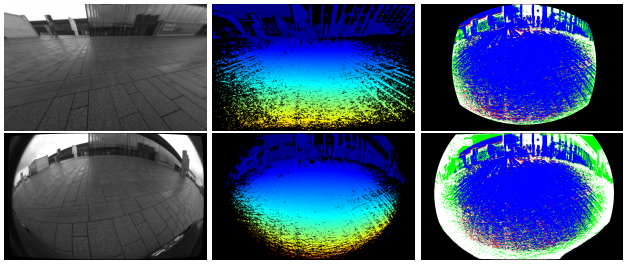


Figure 3. Coverage comparison of locally fused and filtered pinhole depth maps and fisheye depth maps. The first and second rows correspond to the pinhole and fisheye images respectively. The three columns show example input images, depth maps, and coverage images. In each coverage image, a blue pixel indicates the presence of a depth value for both depth maps, a red pixel indicates that a depth value is only present in the pinhole depth map, and a green pixel indicates that a depth value is only present in the fisheye depth map. The non-black area indicates the field of view of the camera. For the fisheye images, a mask was applied to mask out parts of the car visible in the images.

In order to quantitatively determine the increase in coverage obtained from fisheye images, that is, the additional information we gain with a bigger field of view, we conduct an experiment using images from the four cameras mounted on the car. In the first step, we run three-view stereo matching using half-sequence occlusion handling on full-resolution 1280×800 images, 128 planes, 11×11 ZNCC matching costs, and subpixel interpolation. We cannot directly compare the resulting depth maps because they contain a significant number of outliers. Therefore, we compute locally fused depth maps by using 11 raw depth maps and warping them into the middle view. To extract a high-quality depth map, we only accept depths in which at least 3 out of the 11 raw depth values per pixel agree. For each of the pinhole

	Avg coverage (pinhole)	Avg coverage (fisheye)
Pinhole area (front)	36.15%	45.38%
Pinhole area (left)	65.24%	73.07%
Pinhole area (rear)	32.21%	45.38%
Pinhole area (right)	73.22%	80.80 %
Fisheye area (front)	25.91%	37.48%
Fisheye area (left)	49.56%	61.56%
Fisheye area (rear)	24.17%	39.55%
Fisheye area (right)	54.38%	67.79%

Table 1. Average coverage of the depth maps over 200 frames.

and fisheye depth maps, we compute how many pixels have a depth estimate. To make the comparison consistent, we compare both depth maps in two ways; we look at both the portion of the depth maps visible in a pinhole image, and the portion of the depth maps visible in the fisheye image. Example depth maps are shown in Fig. 3, and the average coverage is given in Table 1. Here, we define the coverage to be the number of pixels associated with a depth value divided by the total number of pixels in the portion of the depth map to be compared (pinhole or fisheye area).

In a second experiment, we compute a dense point cloud reconstructed from 31 depth maps for both the fisheye and pinhole models. The resolution is 640×480 and each depth map is computed by locally fusing five depth maps and keeping the points that are consistent in at least three depth maps. Example input images and top-view renderings can be seen in Fig. 4. With the fisheye model, the ground is more complete, and the point cloud includes overhanging parts of buildings which are not visible in the point cloud corresponding to the pinhole model. Note that the camera is pointing roughly $45°$ towards the ground which makes reconstructing the elevated structures difficult. Close-up renderings of the point clouds are depicted in Fig. 5. We observe that when compared to the pinhole model, the truck on the left and the building facade on the right look more complete with the fisheye model.

3.4. Dense Modelling

One important use case of depth maps is to build dense 3D models. In this section, we compare the dense models we obtain from both pinhole depth maps and fisheye depth maps. To fuse the depth maps, we use in both cases the volumetric TV-Flux fusion of [23].

We use images from the MAV, and compute 20 depth maps using two consecutive stereo image pairs. From each fisheye image, we extract two different pinhole images; the first image has a horizontal field of view of $65°$ which corresponds to the field of view of standard pinhole cameras, and the second image has a wide field of view of $114.8°$. The depth maps are computed with a resolution of 754×480 pixels using 96 planes, 9×9 ZNCC matching scores, no occlusion handling, and subpixel interpolation. The resulting dense 3D models are visualized in Fig. 6. All the dense models are computed with the same settings. We observe that the $65°$ pinhole images are not able to see the ground or the ceiling, and thus, the reconstruction covers a small volume. Using the wide-angle $114.8°$ pinhole images, the reconstruction becomes more complete. The reconstruction is most complete with the fisheye images as we directly do dense matching on the fisheye images using our proposed matching algorithm.

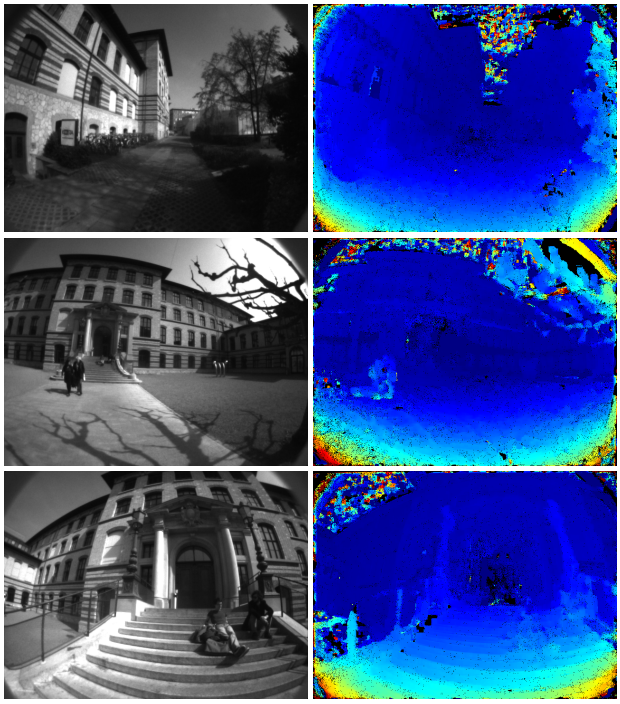


Figure 7. Results from the mobile device.

3.5. Mobile Devices

Another application of our algorithm is on hand-held mobile devices. We use the Google Tango phone[2]. This device contains a depth sensor. This sensor has a standard pinhole camera field-of-view and works indoors only. Our algorithm makes use of the fisheye camera from the device. It is able to compute depth information with a large field-of-view based on the motion of the device and the fisheye images. The camera poses are directly computed on the device in real-time. Our depth map computation is run offline on a desktop computer as a post processing step.

The settings of the plane sweep where as follows. We match three images where the last one is used as reference view. Occlusion handling is turned off and the image dissimilarity measure is an 11x11 ZNCC score. Some example depth maps are shown in figure 7.

[2]www.google.com/atap/projecttango

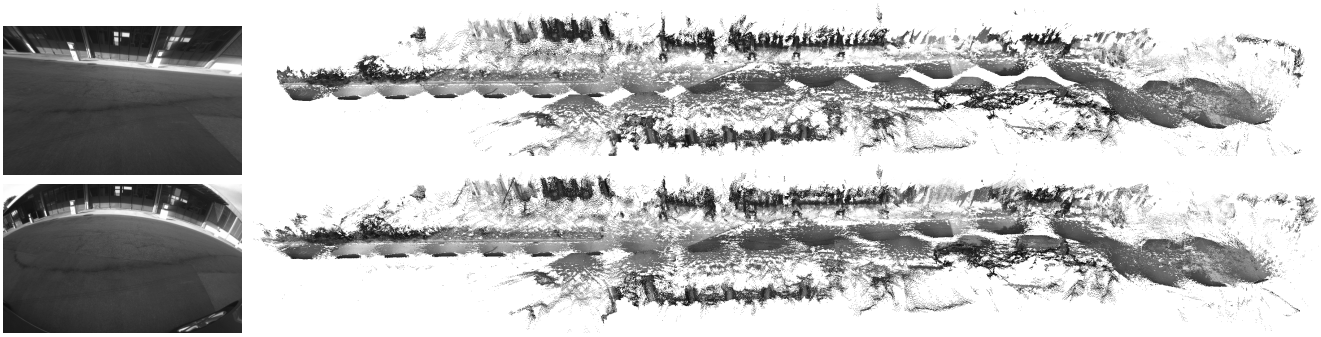


Figure 4. Top: An example pinhole image is shown on the left, and a top-view rendering of the generated point cloud corresponding to the pinhole model is shown on the right. Bottom: An example fisheye image is shown on the left, and a top-view rendering of the generated point cloud corresponding to the fisheye model is shown on the right.

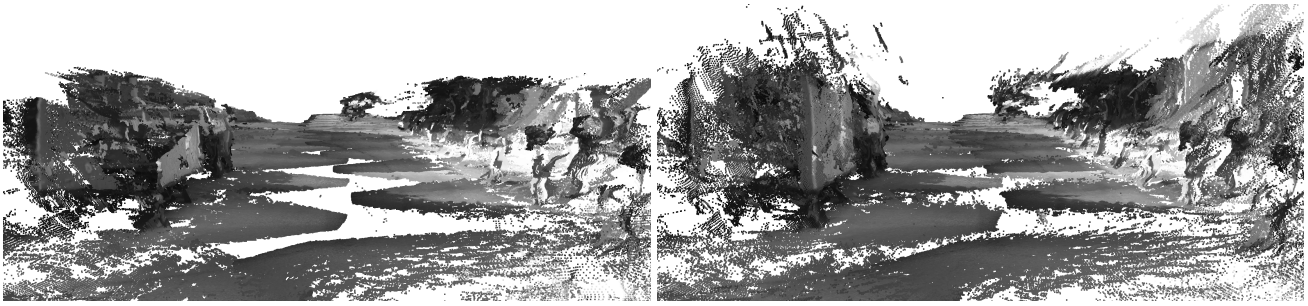


Figure 5. Close-up renderings of the generated point clouds. The left and right images correspond to the pinhole and fisheye models respectively.

4. Conclusions and Future Work

We have shown that our plane-sweeping stereo directly applied to fisheye images is capable of producing high-quality depth maps that cover the entire field-of-view of the fisheye camera. We make a simple adaptation to the widely-used plane-sweeping stereo algorithm for pinhole images by including the fisheye model such that our plane-sweeping stereo algorithm directly works with fisheye images without losing coverage. Our plane-sweeping stereo algorithm is well-suited for GPU implementations. Furthermore, our GPU implementation realizes real-time applications based on fisheye depth maps. In future work, we will implement our plane-sweeping stereo in a car equipped with a surround-view camera system, and generate a textured dense 3D model in real-time via volumetric fusion. In contrast to top-view composite 2D images that surround-view camera systems typically produce, a textured dense 3D model significantly improves the driver's situational awareness. Similarly, we will apply our plane- sweeping stereo to a robot platform equipped with a surround-view camera system such that the robot has an instantaneous 360° view of obstacles in the vicinity. Another future work is to implement the algorithm on a mobile GPU such as the NVIDIA Tegra K1.

Acknowledgements: We thank Torsten Sattler for providing the data sets from the mobile device. Furthermore we acknowledge the support of the V-Charge grant #269916 under the EC's FP7/2007-2013 and Google's Project Tango. Lionel Heng is funded by the DSO National Laboratories Postgraduate Scholarship.

References

[1] S. Abraham and W. Förstner. Fish-eye-stereo calibration and epipolar rectification. *Journal of photogrammetry and remote sensing (ISPRS)*, 2005. 2

[2] J. P. Barreto and H. Araujo. Issues on the geometry of central catadioptric image formation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 1, 2

[3] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 1971. 2

[4] R. T. Collins. A space-sweep approach to true multi-image matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1996. 1, 2, 3

[5] F. Devernay and O. Faugeras. Straight lines have to be straight. *Machine vision and applications*, 2001. 1, 2, 3

[6] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 1, 2, 3

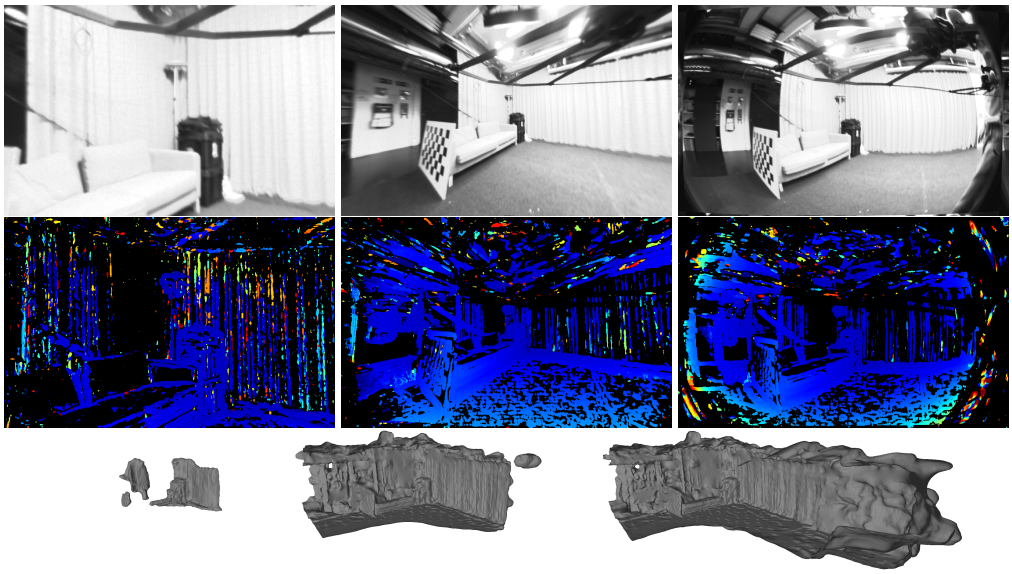


Figure 6. The first, second, and third columns correspond to the 65° pinhole model, 114.8° pinhole model, and the fisheye model respectively. The first row shows the input images, the second row shows example depth maps, and the third row shows the resulting dense 3D models.

[7] S. Gehrig. Large-field-of-view stereo for automotive applications. In *Proceedings of the 6th Workshop on Omnidirectional Vision (OMNIVIS)*, 2005. 2

[8] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical implications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2000. 1, 2

[9] J. Gluckman, S. K. Nayar, and K. J. Thoresz. Real-time omnidirectional and panoramic stereo. In *Proceedings of the Image Understanding Workshop*, 1998. 2

[10] L. Heng, P. Furgale, and M. Pollefeys. Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig. *Journal of Field Robotics (JFR)*, 2014. 4

[11] L. Heng, G. H. Lee, and M. Pollefeys. Self-calibration and visual slam with a multi-camera system on a micro aerial vehicle. In *Proceedings of Robotics: Science and Systems (RSS)*, 2014. 4

[12] H. Ishiguro, M. Yamamoto, and S. Tsuji. Omni-directional stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1992. 2

[13] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 3

[14] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Motion estimation for a self-driving car with a generalized camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1

[15] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013. 1

[16] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007. 1, 2, 3

[17] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salami, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision (IJCV)*, 2008. 1, 2

[18] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision (IJCV)*, 2002. 1, 2

[19] J. Stam. Stereo imaging with cuda. Technical report, NVIDIA Corporation, 2008. 3

[20] Q. Tian and M. N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics and Image Processing*, 1986. 4

[21] D. Tsai and C. Lin. Fast normalized cross correlation for defect detection. *Pattern Recognition Letters*, 2003. 3

[22] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 2

[23] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008. 6

[24] Z. Zhu. Omnidirectional stereo vision. In *Proceedings of the Workshop on Omnidirectinal Vision (OMNIVIS)*, 2001. 2

[25] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. Mav navigation through indoor corridors using optical flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. 1

Obstacle Detection for Self-Driving Cars Using Only Monocular Cameras and Wheel Odometry

Christian Häne[1], Torsten Sattler[1], and Marc Pollefeys[1]

***Abstract*— Mapping the environment is crucial to enable path planning and obstacle avoidance for self-driving vehicles and other robots. In this paper, we concentrate on ground-based vehicles and present an approach which extracts static obstacles from depth maps computed out of multiple consecutive images. In contrast to existing approaches, our system does not require accurate visual inertial odometry estimation but solely relies on the readily available wheel odometry. To handle the resulting higher pose uncertainty, our system fuses obstacle detections over time and between cameras to estimate the free and occupied space around the vehicle. Using monocular fisheye cameras, we are able to cover a wider field of view and detect obstacles closer to the car, which are often not within the standard field of view of a classical binocular stereo camera setup. Our quantitative analysis shows that our system is accurate enough for navigation purposes of self-driving cars and runs in real-time.**

I. Introduction

Reliably and accurately detecting obstacles is one of the core problems that need to be solved to enable autonomous navigation for robots and vehicles. For many use cases such as micro aerial vehicles (MAVs) or self-driving cars, obstacle detection approaches need to run in (near) real-time so that evasive actions can be performed. At the same time, solutions to the obstacle detection problem are often restricted by the type of vehicle and the available resources. For example, a MAV has restricted computational capabilities and can carry only a certain payload while car manufacturers are interested in using sensors already built into series vehicles in order to keep self-driving cars affordable.

There essentially exist two approaches for obstacle detection. Active methods use sensors such as laser scanners, time-of-flight, structured light or ultrasound to search for obstacles. In contrast, passive methods try to detect obstacles based on passive measurements of the scene, e.g., in camera images. They have the advantage that they work over a wide range of weather and lighting conditions, offer a high resolution, and that cameras are cheap. At the same time, a wide field of view can be covered using for example fisheye cameras. In this paper, we therefore present an obstacle detection system for static objects based on camera images. We use stereo vision techniques [1], [2], [3] to obtain a 3D model of the scene. Our main motivation is to enable self-driving cars to detect static objects such as parked cars and signposts, determine the amount of free space around them, and measure the distance between obstacles, e.g., to determine the size of an empty parking spot. We therefore detect obstacles as objects obtruding from the ground [4]. Most existing stereo vision-based techniques rely on classical forward facing binocular stereo cameras with a relatively narrow field of view and visual (inertial) odometry (VIO) systems to provide accurate vehicle poses. These systems are mainly targeted for detecting objects which are in front of the car and are therefore used in standard on road forward driving situations. For many maneuvers such as parking into a parking spot or navigating in a narrow parking garage, a full surround view is very important. We show that for such situations accurate obstacle detections can be obtained from a system that uses only monocular fisheye cameras and the less accurate poses provided from the wheel odometry of the car, if the noisy individual detections are properly fused over time. The resulting system does not require complex VIO systems, but simply exploits information already available while running in real-time on our test vehicle, we thus avoid any unnecessary delay a VIO system might introduce.

This paper makes the following contributions: We describe the overall obstacle detection system and explain each part in detail, highlighting the rationale behind our design decisions. We demonstrate experimentally that highly precise vehicle poses are not required for accurate obstacle detection and show that a proper fusion of individual measurements can compensate for pose errors. Self-driving cars are currently a very active field of research and we believe that the proposed system and our results will be of interest to a significant part of researchers working in this field. To our knowledge, ours is the first system that uses monocular fisheye cameras and only relies on the wheel odometry.

The paper is structured as follows: The remainder of this section discusses related work. Sec. II provides an overview over both our vehicle setup and our obstacle detection system. Sec. III explains the computation of the depth maps. Obstacle extraction is described in Sec. IV, while Sec. V details how to fuse detections from multiple depth maps. Sec. VI experimentally evaluates the proposed method.

A. Related Work

In contrast to motion stereo systems, active sensors such as lidar [5] and binocular stereo cameras [6] can provide a depth map of the environment at any time, even when the vehicle is not moving. Stereo cameras offer the advantage of being cheap to produce while providing high-quality measurements in real-time [6]. Thus, many obstacle detection systems rely on a stereo setup [7], [8], [9]. Obstacles are usually detected in an occupancy grid [10], [11], a digital elevation [12] or

*This work has been supported by EU's 7th Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge).

[1]Department of Computer Science, ETH Zürich, Switzerland {chaene, sattlert, pomarc}@inf.ethz.ch

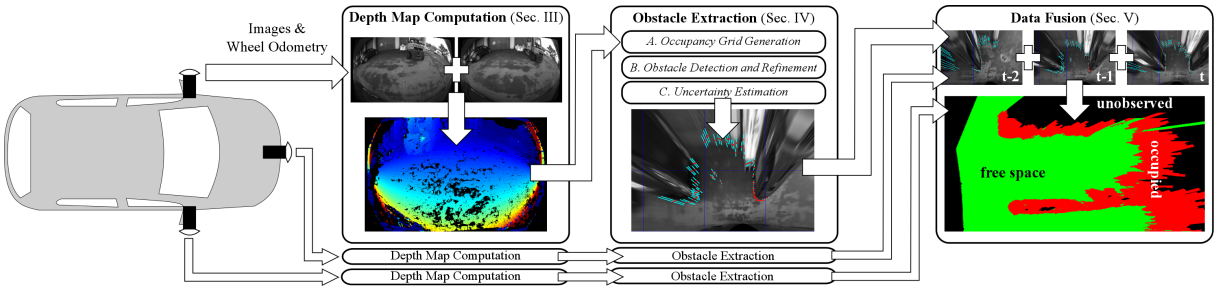


Fig. 1. Overview over the obstacle detection system proposed in this paper.

height map [4], [9], [13], or a volumetric data structure [14] into which the depth measurements are fused. Occupancy grids are probably the most popular scene representation as they not only provide information about the positions of the occupied space but also about free space [15].

In the case of ground-bound vehicles, e.g., cars moving on planar surfaces, obstacles correspond to objects obtruding from the ground: [9] compute a height profile from stereo measurements and subsequently estimate for each 2D occupancy grid cell whether it corresponds to free space or occupied space. While [9] use a probabilistic model, [16] employ dynamic programming on the grid cells to determine free space. Given the free space, *stixels* can be used to compactly represent obstacles as boxes standing on the ground [4]. Overhanging structures and archways can be handled by allowing multiple height map layers [13].

In this paper, we follow a similar approach and model obstacles as objects extruding from the ground. While existing work requires high-precision sensors [16] or visual odometry methods to obtain precise vehicle poses before fusing the depth data [9], [17], we show that using the less accurate wheel odometry readily available in every car is sufficient.

The disadvantage of most stereo cameras is their limited field of view (FOV). Thus, car manufacturers are beginning to additionally integrate wide FOV [7] or fisheye cameras [18] to cover the entire scene surrounding the vehicle. The extremely wide FOV of fisheye cameras enables them to also observe objects close to the vehicle, i.e., they are well-suited for obstacle detections. In this paper, we thus use such cameras and show experimentally that we are able to accurately detect obstacles through motion stereo. [19] use motion stereo, based on very accurate poses from GPS/INS measurements, to generate large-scale urban reconstructions by estimating the 3D geometry of the scene. In contrast, our approach determines which parts of the environment are occupied by obstacles and which are free space.

II. Overview

As illustrated in Fig. 1, our obstacle detection framework consists of three main stages. First, we extract a depth map for each camera mounted on the car using multi-view stereo matching on a sequence of camera frames recorded by the moving car. The camera poses required for this step are directly obtained from the wheel odometry and the extrinsic calibration of the cameras. No visual odometry system is employed to refine the poses. The depth maps provide a 3D reconstruction of the surroundings of the car but do not offer any information about which structures are obstacles that need to be avoided and which parts correspond to free space that the car can move through. In the second stage, we thus detect and extract both obstacles and free space for each individual depth map. Since obstacles are objects protruding from the ground, obstacle detection is performed in 2D.

One of the shortcomings of using wheel odometry is that the rotation of the wheels is only discretely sampled, which leads to a slight oscillation around the true traveled distance. This is not a problem in terms of determining the position of the car but leads to uncertainty for the estimated depth maps due to a slightly inaccurate baseline in the stereo matching. Therefore, the third stage fuses the obstacle detections over several camera frames to obtain a more accurate estimation of the occupied space around the car. Depending on the use case, the fusion can be done independently per camera or as a single fusion that combines data from multiple cameras.

We integrated our obstacle detection system into two VW Golf VI cars that are equipped with a system of four fisheye cameras mounted on the car with minimally overlapping FOVs. The cameras are installed in the side mirrors as well as in the front and back of the car and each have a 185° FOV, jointly covering the whole field of view around the car. They record at a frame rate of 12.5Hz and are triggered synchronously. Besides the cameras, we also utilize the wheel odometry of the car, which provides a 3 degrees-of-freedom pose of the car on the ground plane by measuring the rotation of the wheels. The cameras are calibrated with respect to the odometry frame using a simultaneous localization and mapping-based calibration pipeline [20]. The cars are the test platforms of the V-Charge project [18], whose aims are fully automated navigation and parking. As such they are equipped with other environment perception sensors which are not used for our system. However, since the output of our method is simply a set of occupied and free space estimations, our results can easily be fused into a combined obstacle map together with similar data from other sensors. All used sensors are either already built into today's cars or close-to-market. For the processing of the sensor data and to perform the navigational tasks required for automated driving, the cars have a cluster of 6 PCs installed in the trunk.

III. Depth Map Computation

The first step of our algorithm is to compute a depth map out of the fish eye camera images and the camera poses provided by the wheel odometry. Often, the ground plane is hard to match as it is weakly textured and affected by

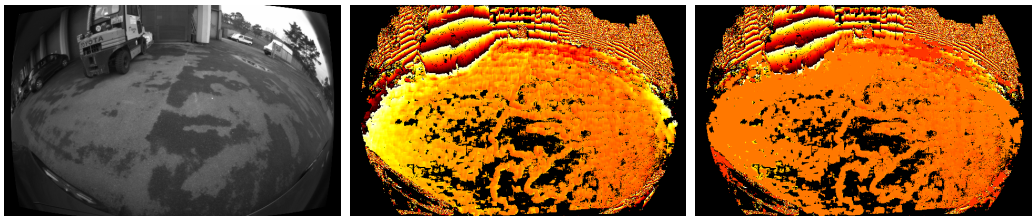


Fig. 2. Results of our depth map computation procedure. (Left) image of the reference view, the depth maps computed (middle) without ground direction plane-sweep and (right) the complete depth map computation procedure including the ground directions. The depth maps are coloured with respect to height above ground instead of depth (color pattern repeating every 0.5 meters), to illustrate the quality of the ground. Without the ground direction plane-sweep, the ground is reconstructed rather bumpy (cf. middle). In contrast, our complete procedure obtains a much smoother reconstruction.

motion blur due to the close proximity to the car. To reduce noise on the areas where the ground plane is visible in the images, we introduce a two stage approach which first checks if the ground plane matches sufficiently well in the images and only reconstructs other structures if a matching patch is unlikely to belong to the ground plane.

Standard binocular stereo matching takes two stereo rectified images as an input and computes a depth map for one of the two images. In our case, images are recorded with a continuous frame rate. This means we have the option to match more than one image to a single reference image to increase the quality of the computed depth maps by increasing the baseline. For general camera configurations, it is not possible to stereo rectify a set of more than two images. To avoid this problem, plane-sweeping stereo [1] is usually used since it does not require any rectification. The main idea of plane-sweeping is to match a set of images to a reference image by projecting them onto a plane hypothesis and then back to the reference image. The images warped through this procedure and the reference image will then be compared using an image dissimilarity measure, which is evaluated over a small matching window. If the tested plane hypothesis is close to the true depth of a pixel in a reference image, the corresponding dissimilarity value will be low. Testing many plane hypotheses and taking the depth induced by the best matching plane for each pixel then produces a depth map for the reference image. Originally, plane-sweeping has been proposed for pinhole images, where the images are warped into the reference view through a planar homography. In order to cover a wider field of view with each camera, and thus enable obstacle detection around the car, our setup uses fish eye cameras. While this increases the complexity of the warping process slightly, depth maps can still be computed in real-time on a graphics processing unit (GPU) [3].

The plane-sweeping algorithm locally approximates the reconstructed 3D structure as a plane. If the normal direction of the plane hypothesis is not well aligned with the actual surface direction, the warped image will be locally distorted with respect to the reference image, which increases the dissimilarity score even for matching patches. This can be overcome by aligning the sweeping plane directions with the predominant directions in the scene [2]. We therefore sweep planes in two directions; Since we are interested in detecting obstacles on the ground, one sweep direction uses planes parallel to the ground plane while the others are fronto-parallel to the camera. The extrinsic camera calibration, which gives us the height of the cameras with respect to the surface the car drives on, provides a good initial estimate of the ground plane. Hence, we only need to test very few planes for the ground direction; we used 10 in our experiments. For the fronto-parallel sweep, we space the planes inversely proportional to the depth, resulting in an even sampling in disparity space. For each pixel in the reference image and each plane, we obtain one dissimilarity score from each image that is matched against the reference image. To obtain one single cost per plane for each pixel in the reference image, we simply average these dissimilarity scores. Notice, this means we do no utilize any occlusion handling (c.f. [3]). Occlusion handling could slightly improve the quality of the depth maps around occlusion boundaries but this small gain comes with a set of disadvantages. Firstly, occlusion handling slows down the depth map computation process. Secondly, doing occlusion handling the way it was initially proposed in [21] would mean that we would compute the depth map for at least one frame before the newest one available. This would lead to an unnecessary delay of the data. As we could not observe any significant gains with it and since it comes with the aforementioned problems, we refrained from using occlusion handling in our experiments. The depth map is finally extracted as the depth of the plane with minimal cost, utilizing a winner takes all strategy.

The last step of our depth map computation procedure is to combine the potentially incomplete depth map computed for each sweeping direction. For this, we need to define a bit more formally the output of the plane-sweeping algorithm. For the sweeps in the ground and fronto-parallel direction, we obtain a depth map $Z_{\mathrm{g}}(x,y)$ and $Z_{\mathrm{f}}(x,y)$, respectively. The computation of the depth maps provides for each pixel (x,y) the image dissimilarity cost for the best matching plane $C_{\cdot}(x,y)$ and the uniqueness ratio $U_{\cdot}(x,y)$ encoding the uniqueness of this match. Let $C_{\cdot,2}(x,y)$ be the cost of the second-best matching plane for pixel (x,y). The uniqueness ratio is then defined as the ratio $C_{\cdot}(x,y)/C_{\cdot,2}(x,y)$. Both of these additional values can be used to filter the depth maps: A low matching cost implies that the image patches match well. A large uniqueness ratio indicates ambiguities, e.g., uniformly textured regions or repeating patterns will result in a ratio close to 1. We use two sets of thresholds for the two sweeping direction $T_{C_{\mathrm{g}}}$, $T_{U_{\mathrm{g}}}$ and $T_{C_{\mathrm{f}}}$, $T_{U_{\mathrm{f}}}$, respectively. Let

$$\alpha_g(\mathbf{x}) := \big(C_{\mathrm{g}}(\mathbf{x}) < T_{C_{\mathrm{g}}}(\mathbf{x})\big) \wedge \big((U_{\mathrm{g}}(\mathbf{x}) < T_{U_{\mathrm{g}}}(\mathbf{x}))\big)$$
$$\alpha_f(\mathbf{x}) := (C_{\mathrm{f}}(\mathbf{x}) < T_{C_{\mathrm{f}}}(\mathbf{x})) \wedge ((U_{\mathrm{f}}(\mathbf{x}) < T_{U_{\mathrm{f}}}(\mathbf{x})))$$

denote that a pixel $\mathbf{x} := (x,y)$ passes the thresholds on the

dissimilarity cost and the uniqueness ratio for the ground-parallel and fronto-parallel sweeps, respectively. The two depth maps are merged as

$$Z(x,y) = \begin{cases} Z_{\mathrm{g}}(x,y) & \text{if } \alpha_g(x,y) \\ Z_{\mathrm{f}}(x,y) & \text{if } \neg\alpha_g(x,y) \wedge \alpha_f(x,y) \\ -1 & \text{else} \end{cases} , \quad (1)$$

where -1 indicates that no depth is assigned to a pixel. Choosing $T_{C_{\mathrm{f}}} \leq T_{C_{\mathrm{g}}}$ and $T_{U_{\mathrm{f}}} \leq T_{U_{\mathrm{f}}}$, $Z(x,y)$ preferably uses matches on the ground plane. In our experiments, we found this behavior to be desirable as it leads to a smoother reconstruction of the ground which is important to avoid false positive obstacle detections.

For our experiments, we use zero mean normalized cross correlation (ZNCC) scores over a 9×9 pixel window for measuring the image similarity. The matching cost $C.(x,y)$ are computed as the negative ZNCC score normalized to the interval $[0,1]$, i.e., a matching cost of 1 corresponds to a ZNCC score of -1 and a matching cost of 0 corresponds to a ZNCC score of 1. There are 10 planes used for sweeps parallel to the ground and 50 planes for the fronto-parallel direcion. The filtering thresholds are set to $T_{U_{\mathrm{g}}} := 0.9925$, $T_{U_{\mathrm{f}}} := 0.98$, $T_{C_{\mathrm{g}}} := 0.18$ and $T_{C_{\mathrm{f}}} := 0.17$. The depth maps are computed on a resolution of 640×400 pixels and we match two images against the reference image, meaning 3 images contribute to every depth map. Fig 2 shows the results of our depth maps computation procedure. The benefit of the additional ground direction sweeping is mostly visible in the form of a smoother ground plane.

IV. Obstacle Extraction

The output of the previous section is a depth map for each camera. In this section, we describe how to extract obstacles from a single depth map. Following [4], [9], we make the simplifying assumption that the car is moving on a plane, i.e., every object not lying on the plane is a potential obstacle, and perform obstacle detection in 2D.

Obstacle detection consists of three steps. Given a depth map, we first create a 2D occupancy grid that encodes which cells contain depth measurements. From this grid, we then extract obstacles from the occupied cells and refine their positions to avoid discretization artifacts. Finally, we estimate the uncertainty of each detection, which will be used in Sec. V to fuse detections from multiple depth maps.

A. Occupancy Grid Generation

We define the local coordinate frame of the car to coincide with the vehicle odometry frame. It has its origin on the ground plane at the point where the middle of the rear axle orthogonally projects to the ground plane. The x-axis is pointing towards the front of the car, the y-axis to the left and the z-axis is pointing vertically away from the ground (cf. Fig. 3). Due to this choice of coordinate system, we can directly infer the position of the ground for each camera from the known extrinsic calibration of the car's camera system.

For each camera, we define a separate occupancy grid that covers the relevant part of the area observed by the camera.

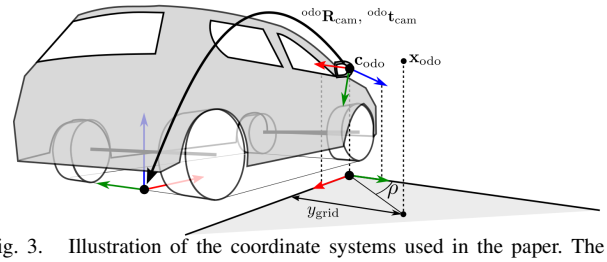


Fig. 3. Illustration of the coordinate systems used in the paper. The x-, y-, and z-axes are colored in red, green, and blue, respectively.

The grid is placed in the ground plane, i.e., coincides with the x-y plane of the vehicle frame. The cameras are mounted such that their x-axes are parallel to the ground. Thus, the x- and y-axes of the occupancy grid are chosen to be parallel to the projections of the x- and z-axes of the camera onto the ground plane. As illustrated in Fig. 3, the origin of the occupancy grid is the projection of the camera center $\mathbf{c}_{\mathrm{odo}}$, expressed in the vehicle frame, to the x-y plane. We express coordinates in the occupancy map by an (angle, disparity) pair (d,ρ), where disparity $d = 1/y_{\mathrm{grid}}$ corresponds to the inverse depth of the measurement.

We use disparities instead of the original y-coordinates since the depth maps are already computed based on disparities by spacing the planes inversely-proportional to the depth. As a consequence, the occupancy grid has a higher resolution very close to the camera, which leads to an unnecessary high grid resolution and thus memory and time consumption. In order to guarantee a minimal cell size in depth direction we virtually shift the grid by y_{shift}. The definition of the final occupancy grid coordinates $(d_{\mathrm{grid}}, \rho_{\mathrm{grid}})$ can now be stated in terms of the Cartesian coordinates in the occupancy grid frame as:

$$\begin{aligned} d_{\mathrm{grid}} &= 1/\left(y_{\mathrm{grid}} + y_{\mathrm{shift}}\right) \\ \rho_{\mathrm{grid}} &= \rho = \arctan\left(x_{\mathrm{grid}}/y_{\mathrm{grid}}\right) \quad . \end{aligned} \quad (2)$$

The occupancy grid coordinates $(d_{\mathrm{grid}}, \rho_{\mathrm{grid}})$ are only defined for Cartesian coordinates with $y_{\mathrm{grid}} > 0$. Notice that the resulting grid has the shape of an isosceles trapezoid, similar to the polar grid in [16].

Let ${}^{\mathrm{odo}}\mathbf{R}_{\mathrm{cam}} \in \mathbb{R}^{3\times3}$ and ${}^{\mathrm{odo}}\mathbf{t}_{\mathrm{cam}} \in \mathbb{R}^{3\times1}$ be the rotation and translation that transform a depth measurement $\mathbf{x}_{\mathrm{cam}}$ from the local camera coordinate system into the vehicle frame, i.e., $\mathbf{x}_{\mathrm{odo}} = {}^{\mathrm{odo}}\mathbf{R}_{\mathrm{cam}}\mathbf{x}_{\mathrm{cam}} + {}^{\mathrm{odo}}\mathbf{t}_{\mathrm{cam}}$. As shown in Fig. 3, the corresponding grid cell is obtained by projecting $\mathbf{x}_{\mathrm{odo}} - \mathbf{c}_{\mathrm{odo}}$ onto the x-y plane of the grid frame, computing the angle and disparity, and performing the shift by y_{shift}.

For each grid cell (d,ρ), we store the number $F(d,\rho)$ of points from the depth map that vote for free space in this specific cell, the number $O(d,\rho)$ of points that vote for occupied space, and additionally the average disparity d_{grid} of the points voting for occupied space within a given cell. If the vehicle frame position $\mathbf{x}_{\mathrm{odo}}$ of a depth measurement is close to (or below in case of errors in the depth map) the ground, it votes for free space in its corresponding cell. If $\mathbf{x}_{\mathrm{odo}}$ lies above the ground plane up to a maximum height, it votes for occupied space in its cell and the corresponding average disparity value is updated. The maximum height

is necessary to handle structures such as overhanging roofs where the car can drive underneath and indoor environments such as underground parking lots.

In our experiments we use a grid resolution of 50×120, depth values up to 30 meters, and an opening angle of $140°$.

B. Obstacle Extraction and Refinement

After filling the occupancy grid, we extract obstacles out of the accumulated data. Each column of the occupancy grid defines a ray corresponding to an angle ρ_{grid}. For each such ray, we want to find the disparity value d_{obst} of the first obstacle along the ray. In order to be robust against noise in the input depth maps, we discard all obstacles that do not pass the following two tests.

A valid obstacle detection should be supported by a minimum amount of depth measurements. We thus compute the strength of an obstacle as

$$S_{\text{obst}}(d,\rho) = \sum_{i=d}^{d+D_{\text{obst}}} O(i,\rho) \ , \tag{3}$$

where we also consider the D_{obst} cells along the ray that follow the first detection. We specify two thresholds $T^{\text{near}}_{\text{strengh}}$ and $T^{\text{far}}_{\text{strengh}}$ to threshold the obstacle strength at the closest and furthest distance of the grid, respectively. This reflects the fact that objects which are further away are represented by fewer image pixels and thus fewer depth measurements. The threshold T^{d}_{strength} for the given cell is then computed through linear interpolation.

Ideally, there should only be votes for free space until we hit the first object and no cell behind this object should contain free space votes. However, this is often not the case close to the camera as sporadic mismatches in the sky and other distant structures, which should be very high above the ground, get reconstructed close to the camera and therefore also close to the ground, resulting in votes for occupied space. The second test is thus designed to reduce the number of erroneously reconstructed obstacles. We define a second obstacle strength measure that rewards free space in front of the obstacle and occupied space behind it as

$$\begin{aligned} S'_{\text{obst}}(d,\rho) &= \sum_{i=0}^{d-1} (F(i,\rho) - O(i,\rho)) \\ &\quad + \sum_{i=d}^{d+D_{\text{obst}}} (O(i,\rho) - F(i,\rho)) \ . \end{aligned} \tag{4}$$

Again, we use a corresponding threshold $T_{\text{strengh}'}$ to discard spurious obstacle detections.

The final extraction procedure is as follows: Beginning at the cell closest to the origin, we compute the obstacle strengths for the cells along each ray. We create an obstacle for the first cell for which both strengths are above the corresponding thresholds. In order to avoid reporting obstacles at discrete positions only, we use the average disparities stored for each grid cell to refine the obstacle position. The final refined position of the obstacle is extracted as the combined average disparity d of the points that where used as votes for the obstacle strength in Equation 3.

In the case that there is no strong enough obstacle along a ray, we extract a special "free space" obstacle that flags the position until which only free space is observed in the grid. This special obstacle represents the fact that the ground plane is visible until this position, which will later be used during the fusion process.

C. Uncertainty Estimation

The positions of the extracted obstacles originate from depth maps that were computed using camera poses provided by the vehicle's wheel odometry, which is not as accurate as vision-based odometry methods. From binocular stereo matching, it is known that the uncertainty of the depth grows quadratically with the depth. Unfortunately, this is not directly applicaple to fisheye lenses [22] and to multi-view matching. However, for a later fusion of the data over time and between cameras, it is important to obtain a (rough) estimate of the uncertainty for each detection. In the remainder of this section, we thus propose a method to estimate the distance uncertainty of the extracted obstacles.

For standard binocular stereo matching using discrete disparities, one can only determine the true displacement of a pixel up to ± 0.5 pixels. Fisheye cameras (approximately) follow an equidistant model [23], i.e., the displacement error in pixels is linearly related to the angle difference $\Delta\alpha$ between the viewing rays through the two pixels. We use this linear relation to estimate the uncertainty of each obstacle detection. Since obstacles are detected in 2D, we also estimate their uncertainty in 2D.

Each depth map is computed from the current and two previous frames. Let $\mathbf{C}_{\text{cur}}$ and $\mathbf{C}_{\text{far}}$ be the 2D positions of the current and the previous camera with the largest distance to the current frame, obtained by projecting the camera centers onto the ground plane. Let $\mathbf{O}$ be the obstacle position and let $\mathbf{C}_{\text{cur}}\mathbf{O}$ and $\mathbf{C}_{\text{far}}\mathbf{O}$ be the rays from the two camera centers to the obstacles. We create two new rays $\mathbf{r}_1$, $\mathbf{r}_2$ originating from $\mathbf{C}_{\text{far}}$ such that the angle between the rays and $\mathbf{C}_{\text{far}}\mathbf{O}$ is $\pm\Delta_{\text{uncertainty}}$ degrees. Intersecting $\mathbf{r}_1$ and $\mathbf{r}_2$ with $\mathbf{C}_{\text{cur}}\mathbf{O}$ then provides an uncertainty interval for the obstacle.

The final result of our obstacle detection approach, 2D obstacle positions and their uncertainties, is depicted in Fig. 4. There is a rather high uncertainty and level of noise in the estimates, as can be seen when obstacles from several frames are simply transformed into a single frame without any proper fusion. In the next section, we thus introduce a simple way of fusing obstacle detections over time. We constantly fuse the available measurements, enabling us to detect and discard incorrect detections such as the outlying measurements observed in the middle of the parking spot shown in Fig. 4. Notice that fusion over time also removes obstacles detected on dynamic objects.

V. Data Fusion

Since we make the reasonable assumption that the car moves on a (locally) planar ground, we again use a 2D grid to fuse the obstacles detected in multiple camera frames. For

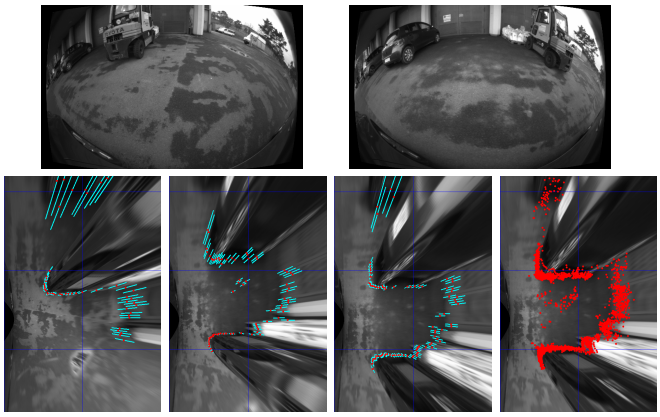


Fig. 4. Results of our obstacle detection approach: (Top row) Two input images. (Bottom row) Obstacles detected in three different frames, projected to the ground plane. The red dots denote the obstacle position and the cyan lines indicate the uncertainty area along the ray. The last image contains all obstacles extracted in an interval of ± 1 seconds around the corresponding frame and shows that simply adding individual detections produces significant noise.

each grid cell, we aim to determine whether it corresponds to free space or is occupied, i.e., contains an obstacle. In robotics, this is often achieved in terms of an occupancy grid using an inverse sensor model [10]. We follow a different approach which originates from the fusion of laser point clouds [24] and has since then been used successfully in many computer vision systems [25], [26], [27]. Its main idea is to follow each ray originating from a camera center until it hits the first obstacle. For all grid cells traversed by the ray, a negative weight is added to denote free space. At the measurement, the ray enters the object and hence the space behind the measurement should be occupied, which is denoted by adding a positive weight. Since we do not know the thickness of each object, we only enter positive weights for a small region behind the obstacle. After adding the measurements for all cameras that should be considered, a cell having a negative accumulated weight is then considered as free space and a cell with a positive weight as occupied space. For cells that have a weight of exactly zero, the space is considered as unobserved.

For real-time processing, we need to limit the size of the grid. We achieve this by only updating a 10×10 meter area which is placed such that it covers the area observed in the current camera frame. Which part of the fusion grid needs to be updated is determined based on the wheel odometry reading. For each cell inside the viewing area of the current camera frame, we determine the obstacle which lies on the ray from the camera center to the grid cell. The weight added to the current grid cell is defined by comparing the distance l_{obst} of the obstacle to the camera with the distance l_{cell} between the cell and the camera. As explained in Sec. IV-B, there are two types of obstacles, conventional obstacles and the "free space" obstacles placed at the end of the definitively visible space along a ray. Consequently, two different types of weights are used. The free space weight w_f is used for both types while the obstacle weight w_o is only employed for the standard obstacles. For both types of obstacles, an

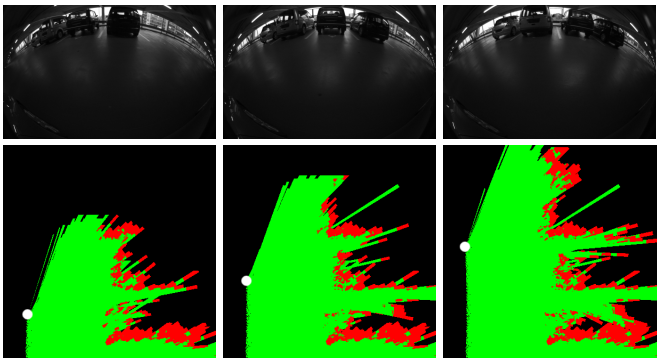


Fig. 6. Fusion results for an indoor sequence. Reflections on the ground cause erroneous detections while backlight complicates precise depth estimation. Still, our system is able to outline the positions of the obstacles.

uncertainty interval $(l_{\text{obst}} - u_1, l_{\text{obst}} + u_2)$ along the ray is given and we distribute the weight of the obstacle along the interval. Thus, obstacles that are measured with low accuracy will only contribute little weight to each grid cell.

The free space weight for the cells along the ray is

$$w_f = \begin{cases} -k & \text{if } l_{\text{cell}} \leq l_{\text{obst}} - u_1 \\ 0 & \text{else} \end{cases}, \tag{5}$$

where the constant $k \geq 0$ is added only to cells in front of the obstacle. The obstacle weight w_o should only be non-zero in the uncertainty regions and is thus defined as

$$w_o = \begin{cases} -\frac{1}{u_1} & \text{if } l_{\text{cell}} \in (l_{\text{obst}} - u_1, l_{\text{obst}}) \\ \frac{1}{u_2} & \text{if } l_{\text{cell}} \in [l_{\text{obst}}, l_{\text{obst}} + u_2] \\ 0 & \text{else} \end{cases}. \tag{6}$$

Additionally, we impose a minimal uncertainty area to make sure that the weight is spread at least into a few cells in the grid. We also discard measurements which have a very large uncertainty interval of more than 4 meters.

All grid cells are initialized with weight zero and are updated whenever new obstacle detections are available by adding the respective weight to the grid cells. The update is constantly performed in a separate compute thread concurrent to depth map generation and obstacle detection. The area of the grid that is updated is set to 400×400 cells, resulting in a 2.5cm resolution for the $10m \times 10m$ area. The constant for free space is set to $k = 4$. Figures 5 and 6 depict results of the fusion approach, where free space is colored green, occupied space is red, and unobserved space is black.

VI. Experiments

In this section, we present the experimental evaluation of our obstacle detection and fusion system. We consider two use cases and evaluate the accuracy in these situations, namely driving past an empty parking spot to measure its width and driving through two objects and measure how far they are away from the car. In the first case, the wide field of view of our fisheye cameras allows us to observe the whole parking spot even at close distance. Similarly, the second case also benefits from the fisheye cameras' ability to detect obstacles close to the car. Both cases are much harder to handle using classical stereo systems due to their limited

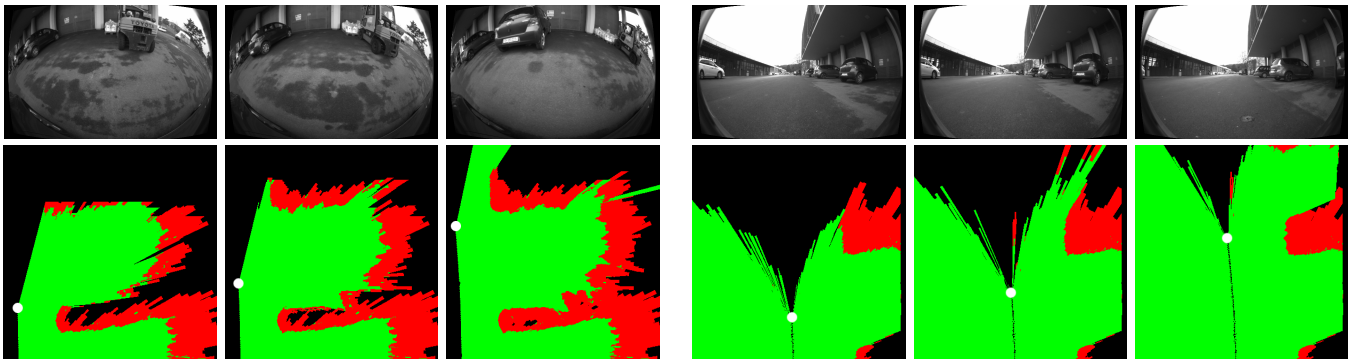


Fig. 5. Obstacle fusion results for two outdoor test sequences. (Top row) input images, (bottom row) fusion results. (Left hand side) right facing camera, (right hand side) front facing camera. Free space is colored green, occupied space in red, and unobserved space in black. The white dot denotes the position of the current camera. Obstacle fusion over time enables our approach to remove outlier detections (cf. Fig. 4) and to recover the structure of the scene.

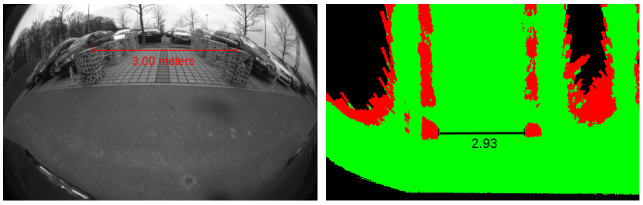


Fig. 7. Left: Ground truth measurment taken with a tape measure. Right: Distance that we determine in the fused results by measuring how wide the minimal gap between the first two foam boxes is (grid resolution 2.5cm).

field of view. We employ foam boxes as obstacles as it is easier to define the ground truth distance between two boxes than two cars, even though these boxes are hard to detect by our motion stereo system.

To enable quantitative evaluation, all the experiments are conducted by replaying sequences recorded by the car on a desktop computer equipped with an Intel Quad Core i7 950 CPU running with a clock rate of 3.07 Ghz and a NVIDIA GeForce GTX 680 GPU. On the car, we are able to run the same software on one of the 6 PCs mounted in the trunk.

A. Measuring the Width of a Parking Spot

The use case of this experiment is that a car is driving past a free parking spot and we would like to measure the width of the parking spot in order to decide whether the car fits into it. For this experiment we naturally use the sidewards facing cameras. We marked the parking spot with foam boxes placed 3 meters apart (measured with a tape measure, cf. Fig. 7). We drove past the parking spot 5 times in each direction and stopped the fusion when the rear axle is at the end of the parking spot. Tab. I reports the mean width of the parking spot measured from the fused results (cf. Fig. 7), together with the standard deviations from 5 measurements from the 5 different runs for each side. As can be seen, the measurements of our system are accurate enough to determine that the car fits into the parking spot. Notice, the slight underestimate in distance is expected as we always use the smallest possible distance between the first two foam boxes to measure the size of the spot. Such a conservative estimate is desired in practice as to avoid initiating a parking maneuver that could potentially lead to a crash.

TABLE I

WIDTH MEASUREMENTS OF AN EMPTY PARKING SPOT

Camera	Mean	Standard Deviation	Ground Truth
Left	2.93m	0.04m	3.00m
Right	2.91m	0.02m	3.00m

TABLE II

DISTANCE IN METERS BETWEEN TWO OBSTACLES

Ground Truth	3.00	3.30	3.60	3.90	4.20	4.50	4.80	5.10	5.40	5.70	6.00
Measured	2.93	3.2	3.48	3.8	4.1	4.4	4.7	5.00	5.3	5.58	5.88

B. Measuring the Distance Between Two Obstacles

While driving between obstacles, it is important to accurately measure the distance of the obstacles to the car. To measure how accurately our system is able to measure the sidewards distance, we drive with the car in between two foam boxes. As a ground truth measurement we measured the distance of the two foam boxes with a tape measure. The fusion is stopped at the point when the two sidewards facing cameras are in between the two foam boxes. Afterwards, the distance is determined by looking at the fusion of the left and right camera and measuring the distance. The foam boxes were set up at different distances from three to six meters. We took one measurement per distance. The results are presented in table II. The results are accurate enough for autonomous driving. Compared to the first experiment, having an accurate calibration is much more important for this use case as the measured distance depends on two cameras. Thus, an error in distance between the two cameras would directly transfer to an error in the measurements. Again, we always took the smallest measurement between the two foam boxes resulting in a slight underestimation of the distance.

C. Runtime Performance

One important aspect of an obstacle sensor running on a self-driving car is its runtime performance. As indicated earlier, we are running the system by replaying a logfile on a desktop computer with a GPU. On the actual car, the same software is running on a similar but slightly less powerful computer. We consider the scenario where a car is driving forward, hence we run the whole system using multi-threading concurrently on the left, front and right camera.

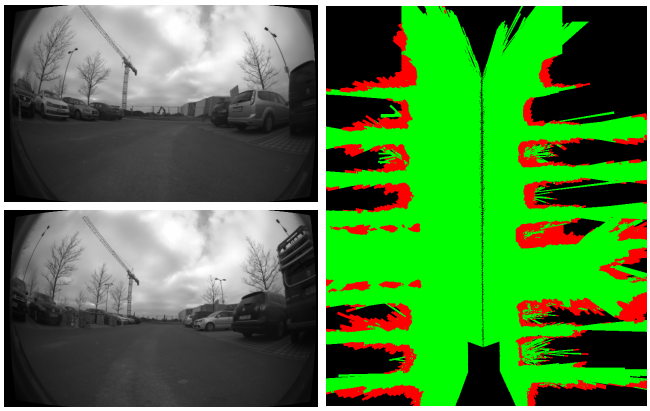


Fig. 8. (Left) Two input images of the front camera. (Right) Result of running our system concurrently on the left, front and right camera. The obstacle frames are entered into the grid with a joint framerate of 37.5Hz.

The images are captured with a framerate of 12.5Hz, therefore our peak performance that we can reach per camera is 12.5Hz. For the result shown in Fig. 8, we are able to process all the frames and hence reach the maximal performance of 12.5Hz, leading to a joint framerate of 37.5Hz. We also confirmed that our system runs in real-time on the car itself. Initial experiments show that even using a NVIDIA Tegra K1 chip is sufficient to process one single camera at lower resolution.

VII. Conclusion and Future Work

In this paper, we have proposed an obstacle detection pipeline for self-driving cars designed to rely purely on monocular cameras and the wheel odometry. Using fisheye cameras allows our system to easily handle situations where classical stereo cameras fail due to their restricted field of view. Our experimental results demonstrate that our method achieves a detection accuracy that is sufficient for practical applications while running in real-time.

The output of our system are a set of obstacles, which we fuse into a consistent grid representation that indicates free and occupied space. This grid can readily be fused with similar results from other sensors. In order to improve the quality, the original obstacle detections could be fused directly instead of fusing the final results. However, properly modeling and fusing the uncertainty characteristics of the different sensor types is a challenging task.

Currently, our system does not detect dynamic obstacles such as other cars. Detecting dynamic objects would require us to associate obstacle detections over time, e.g., based on semantic classification in the images or motion models. This would allow us to link the corresponding obstacle detections and thus prevent the fusion stage from filtering out dynamic obstacles.

References

[1] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1996.

[2] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[3] C. Häne, L. Heng, G. H. Lee, A. Sizov, and M. Pollefeys, "Real-time direct dense matching on fisheye images using plane-sweeping stereo," in *International Conference on 3D Vision (3DV)*, 2014.

[4] H. Badino, U. Franke, and D. Pfeiffer, "The stixel world-a compact medium level representation of the 3d-world," in *Symposium of the German Association for Pattern Recognition (DAGM)*, 2009.

[5] Y.-W. Seo, "Generating omni-directional view of neighboring objects for ensuring safe urban driving," Carnegie Mellon University, Tech. Rep., 2014.

[6] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *Intelligent Vehicles Symposium (IV)*, 2011.

[7] U. Franke, D. Pfeiffer, C.Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. Herrtwich, "Making Bertha see," in *ICCV Workshop Computer Vision for Autounomous Vehicles*, 2013.

[8] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[9] H. Lategahn, W. Derendarz, T. Graf, B. Kitt, and J. Effertz, "Occupancy grid computation from dense stereo and sparse structure and motion points for automotive applications," in *Intelligent Vehicles Symposium (IV)*, 2010.

[10] A. Elfes, "Sonar-based real-world mapping and navigation," *Journal of Robotics and Automation*, 1987.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents. The MIT Press, 2005.

[12] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *Transactions on Vehicular Technology*, 2010.

[13] D. Gallup, M. Pollefeys, and J.-M. Frahm, "3d reconstruction using an n-layer heightmap," in *Symposium of the German Association for Pattern Recognition (DAGM)*, 2010.

[14] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, 2013.

[15] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, and M. Sabbatelli, "Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2014.

[16] H. Badino, U. Franke, and R. Mester, "Free space computation using stochastic occupancy grids and dynamic programming," in *ICCV Workshop on Dynamical Vision*, 2007.

[17] A. Geiger, M. Lauer, and R. Urtasun, "A generative model for 3d urban scene understanding from movable platforms," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[18] P. Furgale *et al.*(34 authors), "Toward automated driving in cities using close-to-market sensors, an overview of the v-charge project," in *Intelligent Vehicles Symposium (IV)*, 2013.

[19] M. Pollefeys *et al.*(19 authors), "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, 2008.

[20] L. Heng, B. Li, and M. Pollefeys, "Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[21] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum, "Symmetric stereo matching for occlusion handling," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[22] H. Ishiguro, M. Yamamoto, and S. Tsuji, "Omni-directional stereo for making global map," in *International Conference on Computer Vision (ICCV)*, 1990.

[23] S. Thirthala and M. Pollefeys, "Radial multi-focal tensors," *International Journal of Computer Vision (IJCV)*, 2012.

[24] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Conference on Computer graphics and interactive techniques*, 1996.

[25] V. Lempitsky and Y. Boykov, "Global optimization for shape fitting," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[26] D. Gallup, J.-m. Frahm, and M. Pollefeys, "A heightmap model for efficient 3d reconstruction from street-level video," in *3D Data Processing, Visualization and Transmission (3DPVT)*, 2010.

[27] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

360° detection and tracking algorithm of both pedestrian and vehicle using fisheye images

Massimo Bertozzi, Luca Castangia, Stefano Cattani, Antonio Prioletti, Pietro Versari

***Abstract*— All-around view is a mandatory element for autonomous vehicles. The European V-Charge project seeks to develop an autonomous vehicle using only low-cost sensors. This paper presents a detection and tracking algorithm that cover all the area around the vehicle using only 4 fisheye cameras. The algorithm is able to detect pedestrians and vehicles and track them, using cylindrical images. The whole pipeline is presented, from the image un-warping to the classification and the tracking algorithms, together with some results.**

I. INTRODUCTION

The EU-funded project Autonomous Valet Parking and Charging (V-Charge) aims to develop an autonomous electric vehicle that can move in parking lots or garages to park and reach re-charging spots. The peculiarity of this project is the use of a low-cost sensor setup, that is close to the one that it is possible to find in a consumer car. This guideline leads to the choice of the sensors suite, that is shown in Fig. 1: just two stereo cameras for long range narrow angle obstacles detection, plus 4 monocular fisheye cameras, aimed to detect moving obstacles all-round the vehicle and ultrasonic bumper sensors.

More details on the project aims can be found in [1]. Cameras are the main sensor of the V-Charge car platform: the presented algorithms employ four fisheye cameras (depicted in blue in Fig. 1) to cover the whole area close to the vehicle.

This paper presents a system to classify and track vehicles and pedestrians all around the autonomous vehicle. As this system is supposed to be used driving at low speeds in parking lots or garages in order to help in different maneuvers (moving forward and backward and making tight curves), the region of interest that must be considered covers the whole area around the vehicle, but it is limited to short distances. Stereo systems, which are not presented in this paper, are employed to detect front and back obstacles only.

The proposed algorithm is a development of the work presented in [2]. As an extension of the previous work, cylindrical images are investigated for classification and multi-camera tracking is developed.

A. Related works

In driving assistance systems, obstacle detection is a crucial component of collision avoidance, especially for dynamic object detection. Many sensors can be used for obstacle detection, such as LIDAR, RADAR and vision sensors. Vision sensors can be divided into pinhole cameras, with a limited field of view, and fisheye cameras [3], [4], [5]. Fisheye cameras can be used for several applications, especially in backup aid and surround view systems [6], [7] because these cameras can frame a wide area close to the vehicle. Feature based object detection algorithms are often used to detect specific kinds of objects such as pedestrians and vehicles [8]. Currently, the major interest relating to the objects classification in the automotive environment is dedicated to pedestrians. Luckily, all the technology developed for this category can be applied to simpler classes of objects like vehicles. The most exploited features in this field are the Haar Features [9], the Local Binary Pattern (LBP) [10], the Histogram of Oriented Gradient (HOG) [11], [12], the Integral Channel Features (ICF) and, recently, Aggregated Channel Feature (ACF) [13], [14]. Features extracted in the image window are processed with an ensemble learning algorithm. Algorithms largely used to detect objects in automotive environments are based on SVM (linear, non-linear, latent) [11] or AdaBoost (and variants) [12]. In the state of the art, both Soft-Cascade techniques and search window pruning (using the hypothesis of flat-terrain) are widespread used [2], [11], [12], [14], to achieve real-time performance. The object detection in fisheye images can be done classifying directly in fisheye images [15], where different classifier are trained on different portion of fisheye image, or applying a classifiers to undistorted image [16].

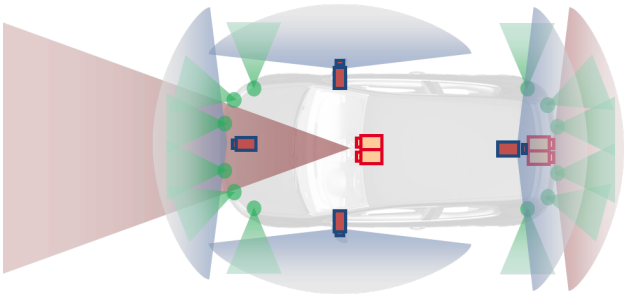


Fig. 1: V-Charge sensors layout: two stereo cameras, on the front and on the back, for long range narrow angle detection; 4 fisheye monocular cameras for all-round view obstacle detection; short range collision avoidance sonars (in green).

II. PROPOSED SOLUTION

The scheme in Fig. 2 represents the main steps of the proposed approach:

- Image un-warping:
  the fisheye images are un-warped in order to both

All the authors are with Department of Information Engineering, Università degli Studi di Parma, Parma, Italy {bertozzi, castangia, cattani, prioletti, versari} @ce.unipr.it

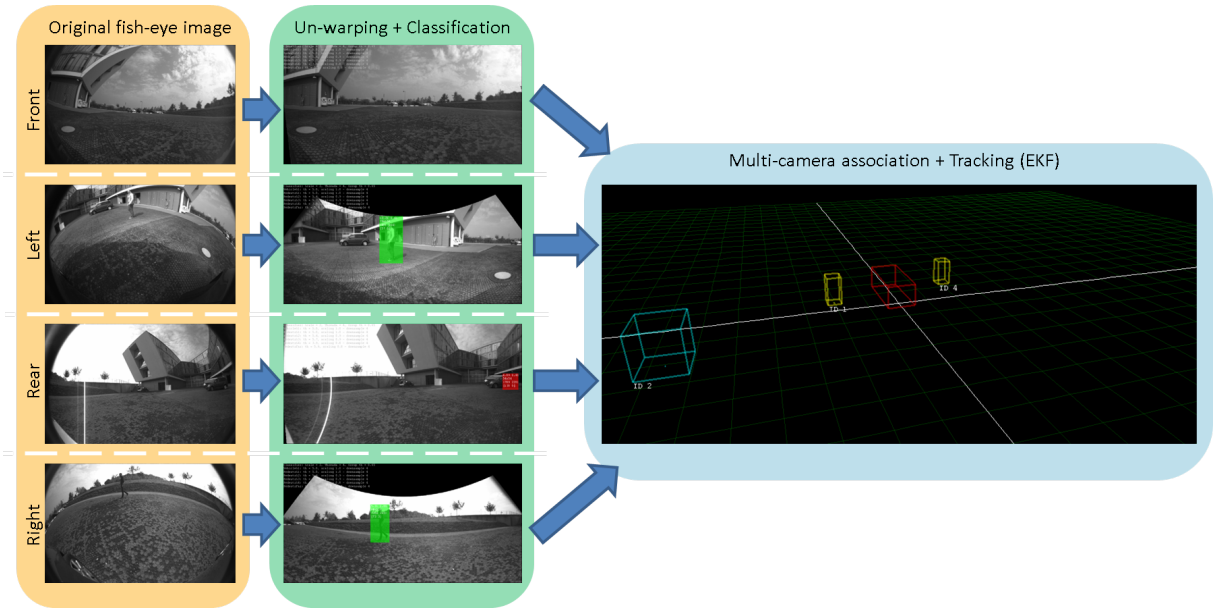


Fig. 2: Overall system design: from the left to the right it is shown the pipeline of the proposed approach. Firstly each fisheye image is un-warped; the chosen model allow to have overlapping field of views between adjacent cameras. Secondly the Soft-Cascade+ICF classifier is used to detect both vehicles (in red) and pedestrians (in green). The outputs from all views are associated each other and finally the Unscented Kalman Filter track all obstacles, allowing to follow an object moving around the vehicle through different field of views.

correct the lens distortion and obtain a wide-angle view without strong aberrations.

- Classification:
  a Soft-Cascade+ICF classifier is run on each camera image to detect vehicles and pedestrians on all the surrounding space.
- Multi-camera tracking:
  in order to perform the 360° tracking, firstly an inter-camera association algorithm is performed then, an Unscented Kalman Filter is used to exploit those information to track and generate an all around description, as an unique high-level sensor with 360° field of view

All of these steps are described in detail in the following paragraphs.

A. *Image un-warping*

Fisheye lenses provide very large wide-angle views (theoretically the entire frontal hemispheric view of 180°), but the produced images suffer from severe distortion since the hemispherical scene gets projected onto a flat surface. A procedure to correct the fisheye lens distortion is necessary unless you use an algorithm that takes into account this effect. A common approach involves the distortion correction by re-projecting the image on a virtual plane in order to obtain a pinhole image [17] but, the wider the resulting pinhole field of view, the stronger aberration you get on image edges.

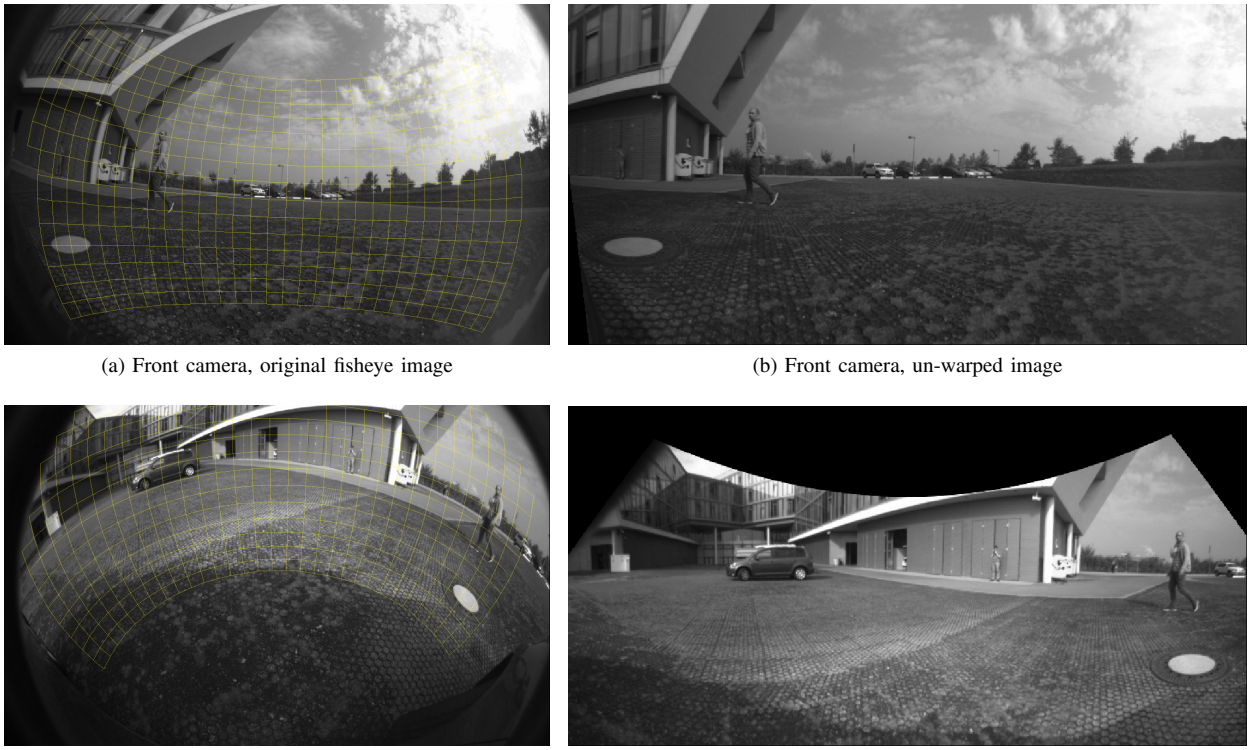
In [2] a virtual views layout has been presented in order to correct the lens distortion and exploit the large fisheye lenses field of view. However that approach has some drawbacks: to find all obstacles virtual views need to overlap; this means more pixels to process and conflicted areas where it is hard to determine what is the right detection. To overcome these shortcomings, it has been decided to increase the number of virtual views and merge them together in a single wide image: the final result is equivalent to re-project the original fisheye image onto a semicylindrical surface, as it is done in [16].

Fisheye cameras manufactured to follow the equidistance mapping function, are designed such that the distance between a projected point and the image optical center is proportional to the projected ray incident angle, scaled only by the equidistance parameter. Exploiting this relationship, the resulted cylindrical model is conveniently described by the following equations:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c_u + \operatorname{atan2}(x, z) \cdot f_\theta \\ c_v + k_v \cdot \frac{y}{\sqrt{x^2+z^2}} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin \frac{u-c_u}{f_\theta} \\ \frac{v-c_v}{k_v} \\ \cos \frac{u-c_u}{f_\theta} \end{bmatrix} \tag{2}$$

where (c_u, c_v) is the un-warped image virtual optical center, k_v is the vertical focal length, f_θ is the equidistance parameter, (x, y, z) is the 3D point being projected to the image and (u, v) is the related point in image domain. The Equation 1 is the projection of a (x, y, z) 3D point from camera reference frame to cylindrical image reference frame, while the Equation 2 is the relation from a (u, v)



(a) Front camera, original fisheye image

(b) Front camera, un-warped image

(c) Left camera, original fisheye image

(d) Left camera, un-warped image

Fig. 3: Examples of fisheye images and the related un-warped images; into the fisheye images is depicted the corresponding semicylindrical surface. The extrinsic parameters have been also used in the un-warping phase and this is considerable especially on side cameras. Note also the overlapping field of view between the two un-warped images.

2D point from cylindrical image reference frame onto a semicylindrical surface in the camera reference frame. In order to preserve the original aspect ratio it is convenient to impose $k_v = f_\theta$.

Exploiting the extrinsic, intrinsic and distortion cameras parameters [18] a proper semicylindrical surface has been defined for each fisheye camera; to facilitate the subsequent algorithms, such the classification phase, those semicylinders have been placed just in front of each camera keeping, their axis perpendicular to the vehicle reference z plane. The Fig. 3 shows some fisheye images and the resulted un-warped images.

These choices imply that each image row correspond to a circumference arc in vehicle reference z plane; this means that objects on the same row in the same image are at the same distance from the related camera: this property is useful for many optimizations in the classification phase. Noteworthy is also that the whole un-warped image does not respect the pinhole camera model but, locally, the difference between a virtual pinhole camera and the un-warped image is hardly visible; a detailed analysis is covered in section III-A.

B. Classification

Pedestrians and vehicles detection is based on a Soft-Cascade+ICF classifier. Due to the greyscale sensors, the detector differs from [19] because it has not been used the LUV color channels. The first phase, regarding the computation of the integral channel, is shared between pedestrians and vehicles detector in order to speed-up the system.

Several classifiers with different pattern sizes have been trained reducing the processing time required in the scaled images computation and improving the detection performances. Five classifiers are used for pedestrians, with dimensions 32×64, 48×96, 58×116, 68×136 and 81×162; two, instead, for vehicles, with dimension 38×38 and 45×45. Two scales for each classifier are computed. The pedestrian classifier with pattern size 32×64 is used to detect only far pedestrian: differently from the other classifier, it is run just at the smallest octave for each scale.

Moreover, the detection is not performed in the regions that do not respect the typical range of vehicle/pedestrian size: for each candidate window, through the world-to-image coordinates transformation, its theoretical area is calculated and is checked if it respects the suitable dimensions for a vehicle or pedestrian. Candidates at the same scale-octave are filtered with a 3×3 non maxima-suppression.

The final training set is composed by 20000 cars images, 40000 pedestrians images and 80000 negatives examples, randomly extracted from the initial 12000 negatives images. Positive samples are enlarged by 8% to include also the information contained in the border. 10 bootstrapping cycle are performed to improve the detection performance and

reduce the false positive rate. At each cycle, the detector is trained on an augmented set composed by the initial negative samples and the hard negative ones obtained by the previous cycle.

C. Multicamera tracking

Tracking is performed jointly through the four cameras allowing to follow an object moving around the vehicle.

1) Association: The "all-around tracking" requires that the association must be performed not only between objects in subsequent frames but also between objects coming from different cameras. The matching in subsequent frames is performed using the bounding boxes overlap. Several comparison metrics, instead, have been analyzed for the multi-cameras one:

- Euclidean distance: objects coming from all the cameras are projected in the world coordinates space and the euclidean distance represents the matching cost. The projection of the vehicle rear axis center points on the ground (tracking point) is used for the projection.
- Polar distance: objects are still projected in the world coordinates but, in this case, the polar distance represents the matching cost.
- Image distance: objects on each camera are projected on the other cameras (if the projection is possible) and the matching cost is represented by the bounding boxes overlap.

Using a tracking algorithm based on image coordinates (as described below), the "Image distance" metric has been selected for the multi-camera matching process.

An association based on features matching between different cameras has been also tested but discarded due to the high computational cost for the features extraction and matching.

Objects of different classes (pedestrians or vehicles) have matching cost zero.

Once defined a comparison metric, the Hungarian algorithm is used to find the best association and maximize the matching cost. After this step, each tracked object may have been associated with one or two elements depending if it has been seen in one or more cameras.

2) Tracking: An Unscented Kalman Filter (UKF) is used for the tracking process. The tracking point, the object width and its relative speed form the filter state. The predicted position is obtained combining the host vehicle movement information (from the inertial sensors) and the object position and speed.

As described in the II-C.1 section, one or two element can be associated to a tracked object after the matching process. In case of a "double" association it is possible to make a higher precision observation, since the object is observed from two different points of view, like a stereoscopic camera. In case of a "single" association, the observation is represented by the tracking point and object width in image coordinates: through the pinhole camera model is possible to switch between image and world coordinates.

In case of a "double" association, instead, the observation is represented by a couple of tracking points and width in pixels from the objects in the two different cameras. In this case, the predicted observation are the projections of the object state in the two image coordinates.

Extra care is required to manage the two different obstacles types, pedestrians and vehicles. This involves different filter parameters, regarding the object movement and, the different extraction of the object bounding box from its state: while vehicles have the same ratio between height and width, the pedestrians height is twice their width.

The bounding box extraction process requires the computation of the top-left and bottom-right points of the object in the world coordinates and, then, the projection of these points in the image coordinates. In case of an object in the front camera, these points are calculated subtracting and adding on the x axis the width value to the tracking point. These procedure, instead, would not be valid for objects in different cameras. Then, a different method has been used for the bounding box extraction: firstly, the tracking point and the top center point of the object are projected in the image coordinates; from these two points it is possible to extract the bounding box height in pixels and use it to find the width and, then, the top-left and bottom-right points in image coordinates starting from the projected tracking point.

III. RESULTS

A. Cylindrical images

The basic assumption of our approach was the classificator compatibility with cylindrical images; since no object detection evaluation dataset with fisheye images was available, we chose to evaluate "differences" between un-warping fisheye images using pinhole model and cylindrical model. For this purpose we defined "difference" d as follow:

$$d(u,v) = \left\| (u,v) - (u',v') \right\| \tag{3}$$

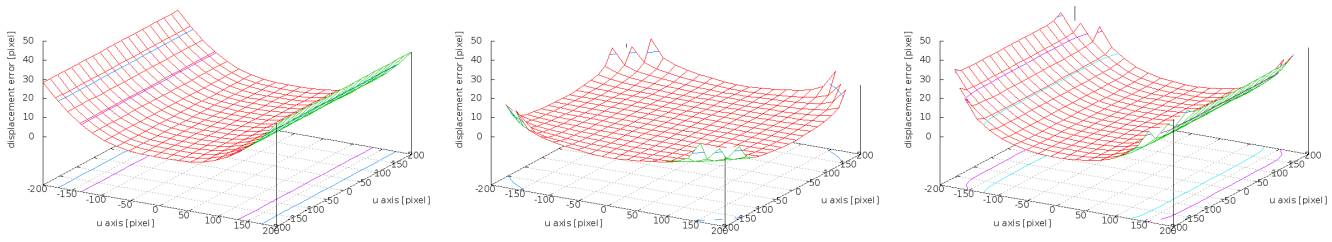
where (u,v) is a point in image domain and (u',v') is the point obtained by re-projecting (u,v) from pinhole model to cylindrical model.

This analysis helps to understand the expected object distortion given its size in the equivalent pinhole model, and consequently it is possible to estimate the maximum acceptable size. In Fig. 4 the results[1] of this analysis are shown: firstly we evaluated the per-axis re-projection displacement errors, then we computed Euclidean norm as described by Equation 3.

Predictably, the main displacement error occurs along u axis: points further than 150 pixels have displacement error u component of approximately 20 pixels. With a similar analysis of "Euclidean norm of displacement errors" graph (Fig. 4 c) we can overestimate the error of objects as wide as 200 pixels in less than 10 pixels, and less than 20 pixels for object as wide as 300 pixels; along the v axis we have less constraints.

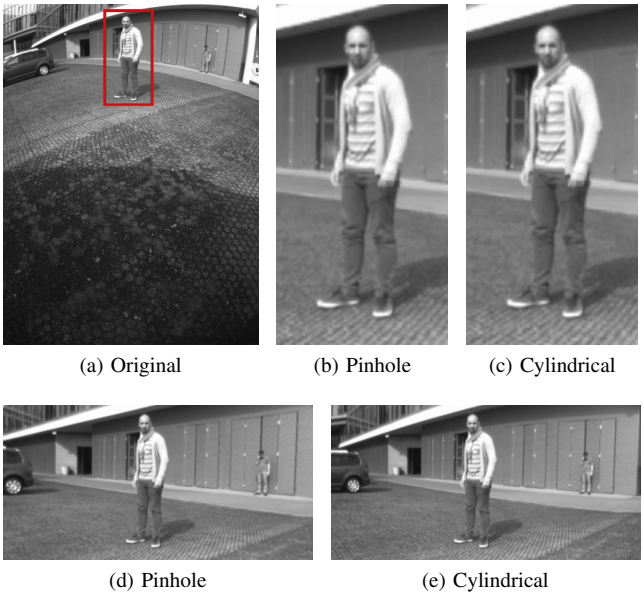
It is important to notice that the further the points are from the pinhole image center, the stronger aberration you

[1]These results are computed using the real intrinsic camera parameter utilized in our system: $f_\theta = k_v \simeq 266.67$ [pixel].



(a) Displacement errors along u axis (b) Displacement errors along v axis (c) Euclidean norm of displacement errors

Fig. 4: Displacement errors between pinhole model and cylindrical model. In (a) and (b) are shown the displacement errors along u and v axes, in (c) is depicted the Euclidean norm of displacement errors as described by Equation 3. The greater the error is, the higher the displacement between pinhole model and cylindrical model is at that coordinate.



(a) Original (b) Pinhole (c) Cylindrical

(d) Pinhole (e) Cylindrical

Fig. 5: Comparison between un-warping the original image (a) by re-project it on a plane (b) or on a semicylindrical surface (c); those images are focused only on a small area in order to highlight the small differences between the two approaches. In (d) and (e) the differences are more evident but still hard to see: with the same number of pixels in (d) we have some aberration on sides, meanwhile in (e) we have a slightly larger field of view.

get: then on the one hand the "difference" as defined by Equation 3 is increasing, but on the other hand the pinhole image we are comparing to begins to be affected by aberration. In Fig. 5 is shown a qualitative comparison between the original fisheye image and pinhole model and cylindrical model.

B. Classification

The classifier, used in the previous version of the system, has been improved on several aspects:

- the pre-processing parts of vehicle and pedestrian classification are merged in a common stage;
- the scales number has been reduced using more classifiers with different pattern size;
- the shrinking factor has been introduced;

These changes lead to reduce the computational time and to achieve real-time performance processing the images coming from all the four cameras.

C. 360° tracking

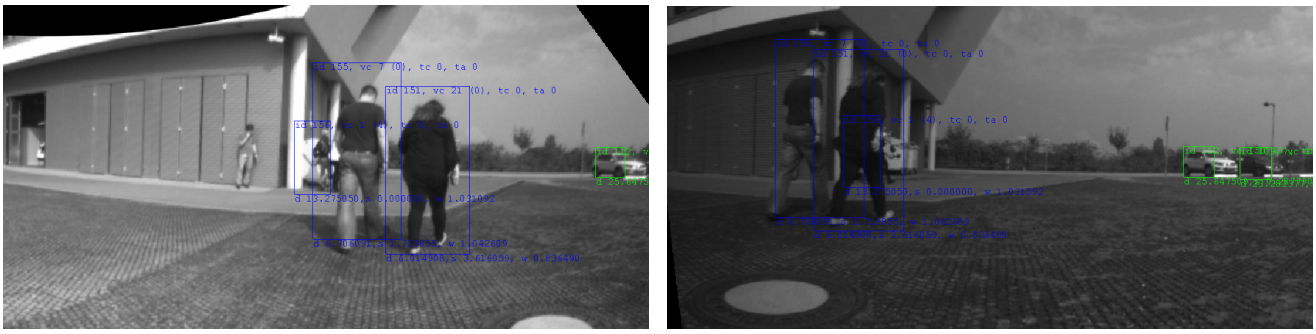
The inter-camera tracking algorithm improves considerably the system performance with respect to perform it individually on each camera. Overlapping fields of view guarantee a higher probability to detect an object than in one camera, both in terms of different points of view from which the object is observed then in terms of possible occlusions in a camera but not in the other one. Moreover, it is possible to exploit the past history of an object passing through the cameras, increasing the tracking accuracy.

In Fig 6 is shown an example where we exploited the two points of view to overcame one occlusion and to improve the detection accuracy.

D. Conclusions and future works

In this paper is presented a method to detect and track pedestrians and vehicles all around the V-Charge prototype. The sensor suite is limited to four fisheye cameras, installed on the front, back, right and left parts of the vehicle. Main contributions of this method are the classification using cylindrical images and the 360° tracking. The obtained results are remarkable in parking slots and garage environments. A great attention has been given to optimize the computational performance, to reach a processing time that allows the managing of different manoeuvres at low speeds, processing four cameras images. Even if the system provides promising results in most situations, when the obstacles are as close to the vehicle to be only partially framed, the classification can provide very noisy results, especially in distance estimation.

The tracking stage still needs further developments: vehicle orientation should be detected in order to improve the



(a) Left cylindrical image

(b) Front cylindrical image

Fig. 6: Example of 360° tracking approach benefits: we exploit the multiple point of views to overcome occlusion (the far pedestrian is visible only in (a)) and to increase the detection accuracy (the two walking pedestrians were detected on both images).

association by different points of view; moreover, tracking of partially visible obstacles should be included. A huge test stage is also mandatory to validate the method in the specific environment considered for the V-Charge project.

IV. ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No 269916, V-Charge.

REFERENCES

[1] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. T. S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart, "Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project," in *IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, Australia, 23–26 June 2013, pp. 809–816.

[2] A. Broggi, E. Cardarelli, S. Cattani, P. Medici, and M. Sabbatelli, "Vehicle detection for autonomous parking using a Soft-Cascade AdaBoost classifier," in *Procs. IEEE Intelligent Vehicles Symposium 2014*, Dearbon, MI, USA, June 2014, pp. 912–917.

[3] B. Zhang, V. Appia, I. Pekkucuksen, A. Batur, P. Shastry, S. Liu, S. Sivasankaran, K. Chitnis, and Y. Liu, "A surround view camera solution for embedded systems," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, June 2014, pp. 676–681.

[4] J. J. Kumler and M. L. Bauer, "Fish-eye lens designs and their relative performance," pp. 360–369, 2000. [Online]. Available: http://dx.doi.org/10.1117/12.405226

[5] T. Pajdla, T. Svoboda, and V. Hlaváč, "Epipolar geometry of central panoramic cameras," in *Panoramic Vision : Sensors, Theory, and Applications*, 1st ed., R. Benosman and S. B. Kang, Eds. Berlin, Germany: Springer Verlag, 2001, pp. 85–114.

[6] F. Nielsen, "Surround video: a multihead camera approach." *The Visual Computer*, vol. 21, no. 1-2, pp. 92–103, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/vc/vc21.html#Nielsen05

[7] C.-C. Lin and M.-S. Wang, "A vision based top-view transformation model for a vehicle parking assistant," *Sensors*, vol. 12, no. 4, pp. 4431–4446, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/4/4431

[8] L. Castangia, P. Grisleri, P. Medici, A. Prioletti, and A. Signifredi, "A coarse-to-fine vehicle detector running in real-time," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, Oct 2014, pp. 691–696.

[9] J. Choi, "Realtime on-road vehicle detection with optical flows and haar-like feature detectors," University of Illinois at Urbana-Champaign," Computer Science Research and Tech Reports, 2012.

[10] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, Sept 2012, pp. 975–982.

[11] M. Gabb, O. Löhlein, R. Wagner, A. Westenberger, M. Fritzsche, and K. Dietmayer, "High-performance on-road vehicle detection in monocular images," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, Oct 2013, pp. 336–341.

[12] J. D. Ortega, M. Nieto, A. Corts, and J. Flrez, "Perspective multi-scale detection of vehicles for real-time forward collision avoidance systems." in *ACIVS*, ser. Lecture Notes in Computer Science, J. Blanc-Talon, A. J. Kasinski, W. Philips, D. C. Popescu, and P. Scheunders, Eds., vol. 8192. Springer, 2013, pp. 645–656.

[13] R. Appel, P. Perona, and S. Belongie, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. PrePrints, p. 1, 2014.

[14] R. Brehar and S. Nedevschi, "Scan window based pedestrian recognition methods improvement by search space and scale reduction," in *Procs. IEEE Intelligent Vehicles Symposium 2014*, Dearborn, MI, USA, June 2014.

[15] K.-Y. Byun, B.-S. Kim, H.-K. Kim, J.-E. Shin, and S.-J. Ko, "An effective pedestrian detection method for driver assistance system," in *Consumer Electronics (ICCE), 2012 IEEE International Conference on*, Jan 2012, pp. 229–230.

[16] G. Cheng and X. Chen, "A vehicle detection approach based on multi-features fusion in the fisheye images," in *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, vol. 4, March 2011, pp. 1–5.

[17] C. Hughes, P. Denny, M. Glavin, and E. Jones, "Equidistant fish-eye calibration and rectification by vanishing point extraction," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 12, pp. 2289–2296, Dec 2010.

[18] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[19] R. Benenson, M. Mathias, R. Timofte, and L. J. V. Gool, "Pedestrian detection at 100 frames per second." in *CVPR*. IEEE, 2012, pp. 2903–2910. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2012.html#BenensonMTG12

Vision-Only Fully Automated Driving in Dynamic Mixed-Traffic Scenarios

Ulrich Schwesinger, Pietro Versari, Alberto Broggi, Roland Siegwart

Abstract: In this work an overview of the motion planning and dynamic perception framework within the V-Charge project is presented. This framework enables the V-Charge car to autonomously navigate in dynamic mixed-traffic scenarios. Other traffic participants are detected, classified and tracked from a combination of stereo and wide-angle monocular cameras. Predictions of their future movements are generated utilizing infrastructure information. Safe motion plans are acquired with a system-compliant sampling-based local motion planner. We show the navigation performance of this vision-only autonomous vehicle in both simulation and real-world experiments.

ACM CCS: Computing methodologies → Artificial intelligence → Control methods → Motion path planning; Computing methodologies → Artificial intelligence → Computer vision → Computer vision tasks → Vision for robotics.

Keywords: automotive, autonomous navigation, motion planning, environment perception

1 Introduction

Autonomous navigation in mixed-traffic scenarios requires a broad spectrum of robotic disciplines like localization, perception, classification, motion planning and control to perform at their peak. The vehicle needs to precisely perceive its local surrounding and compute safe motion plans at a high rate in order to react to unforeseen events. Those components experienced significant advances within the last decade – especially since the impetus of the DARPA Urban Challenge in 2007 [1–4]. Since then, effort from various groups was spent to bring autonomous driving capabilities to public streets. With their BRAiVE platform [5], the Artificial Vision and Intelligent Systems Laboratory (VisLab) successfully demonstrated autonomous navigation in rural roads, highways and urban environments during the PROUD run [6]. BRAiVE's sensor setup consisted of in total ten cameras, five laser scanners and a GPS+IMU solution. Carnegie Mellon's Cadillac SRX completed a 53 km long route on public roads without human intervention [7]. Similar to VisLab's autonomous vehicle, Carnegie Mellon's self driving car utilized a mixture of laser scanners, lidars and cameras. Daimler's "Intelligent Drive" system autonomously drove 103 km on rural roads, small villages and major cities [8] and Google's self driving cars spent remarkable 1 126 500 km on public US streets in autonomous mode up to now [9].



Figure 1: The V-Charge autonomous vehicle platform.

All the above mentioned platforms complemented vision sensors with potentially expensive laser scanners, lidars and GPS+IMU solutions. Within the European project V-Charge (Automated Valet Parking and Charging for e-Mobility) the application of low-speed autonomous driving in restricted parking areas with low-cost

vision sensors only is targeted. Low-cost sensors already available in series cars will lower the consumer price for the additional autonomous navigation functionality and in the end lead to a higher customer acceptance of those systems. As legal issues might prevent the acceptance of autonomous vehicles in public traffic for a foreseeable time, the project aims at the application of autonomous functionality in restricted areas such as parking lots. Here, automated vehicles constitute a convenient solution for approaching vacant parking spots and charging stations (electric vehicles) and provide a valuable drop-off/pick-up service at the entrance of the parking lot.

The V-Charge vehicle features static obstacle avoidance as well as the handling of mixed-traffic scenarios where automated and human-operated cars share a common workspace and have to interact with each other. Navigation has to be accomplished in tight spaces with pedestrians and other vehicles in close proximity to the ego vehicle. These tasks are accomplished with a sensor suite solely comprised of two stereo cameras, four monocular wide angle fisheye cameras, twelve stock ultrasonic sensors and incremental encoders at the vehicle's wheels.

2 System Overview

A high-level overview of the V-Charge system is depicted in Figure 2. From left to right the processing chain is visualized starting with the sensor data and eventually resulting in system inputs to the vehicle. The key sensors to perceive the environment are the front stereo camera as well as the front and side monocular cameras. Specifications and coverage information are illustrated in Figure 3.

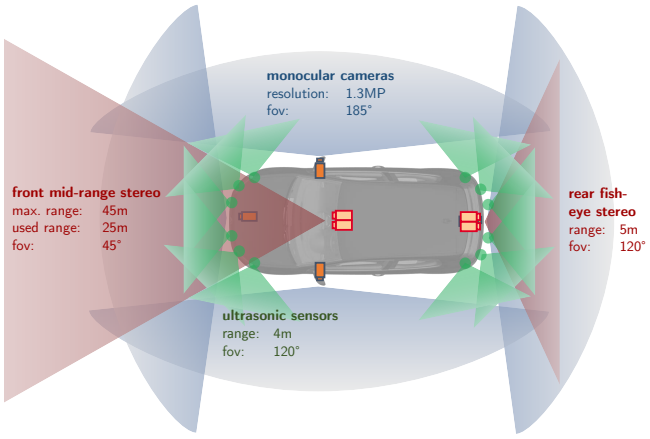


Figure 3: Schematic sensor coverage of the V-Charge setup suite.

The rear fisheye stereo sensor as well as the ultrasonic sensors are used for close-range obstacle detection during parking maneuvers. The ultrasonic sensors additionally serve as the input to an emergency brake module. The 2.5D stixel output of the front stereo sensor is on the one hand used to create a two-dimensional (2D) occupancy grid of the environment, containing static information only. Stixels assigned to dynamic objects are removed from the grid. On the other hand, together with the images from the front monocular camera, it forms the input of the "Stereo Classification" module, which groups stixels to objects and assigns class labels to them based on a database of known object classes. This module is described in detail in Section 3. The classified objects are tracked with an extended Kalman filter, applying suitable motion models for propagation based on the classification result. Once classified and tracked, predictions of the future movement of objects are generated, taking into account the object's dynamic model and further infrastructure information such as the road network stored in a semantic map database. The "Local Motion Planning" module described in Section 4 computes a safe motion plan based on the latest perceptual information and forwards the plan in form of a time-stamped sequence of reference states plus feed-forward system inputs to an underlying "Control" module. The "Control" module takes care of correcting modeling errors from the "Local Motion Planning" module. The system inputs from the control module are fed back to the local motion planner to correctly propagate the system states during the expected cycle time of the planner and to model system dead times.

3 Stereo Object Classification

It is important to distinguish between static obstacles and dynamic objects to better understand the surrounding environment and predict its evolution over time. In order to achieve this capability we chose to classify object hypotheses obtained by a clustering step of stereo stixels. Though the current object database only contains pedestrians and vehicles, adding more classes is a straightforward process.

3.1 Unwarping of Fisheye Images

As the front stereo system has a limited field of view, nearby objects are potentially only partially visible with the stereo sensor, whereas they still appear in the field of view of the front fisheye camera. This can happen for instance if a pedestrian is walking close to the front of the vehicle. The stereo system will still detect the pedestrian, but its upper part is outside the stereo system's field of view. By projecting stixels from stereo hypotheses into the wide-angle images obtained by the front fisheye camera, the stereo object classification is able to utilize the full information of the wide-angle monocular view.

Fisheye lenses provide very large wide-angle views – theoretically the entire front hemispheric view of 180° – but the images suffer from severe distortion since the hemispherical scene gets projected onto a flat surface. To

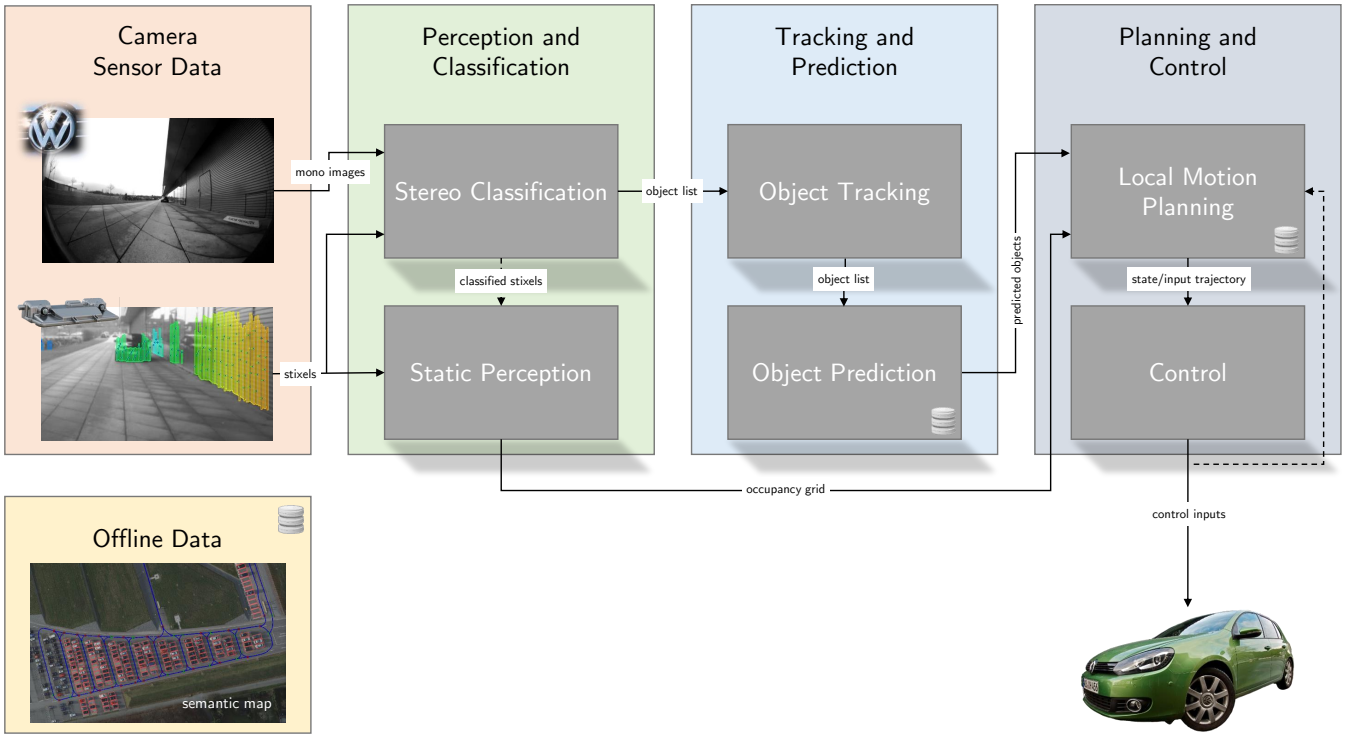


Figure 2: Overview of software modules involved in on-lane navigation. The processing chain reaches from raw sensor data (left) to the system inputs sent to the platform (right).

correct the fisheye lens distortion, the image is reprojected onto a virtual plane in order to obtain a pinhole image, as shown in Figure 4.

3.2 Classification

Each object class stores suitable ranges for the object's size and aspect ratio. Those restrictions are used in a pre-filter step to eliminate hypotheses not coherent with the specific class characteristics. Each object passing the pre-filter step is projected onto the pinhole image and, for each possible object class, a region of interest (ROI) is computed. This process takes into account several factors: the object's distance and its dimension in the image domain, the class aspect ratio and the extrinsic calibration of the camera. An example of the output of this step is shown in the right part of Figure 4.

Every ROI is classified by a Soft-Cascade+ICF classifier to validate or reject the hypotheses. The detector differs from [10] since no LUV color channels are available in the monocular fisheye images. One classifier with a dimension of 38×38 pixels is used for vehicles, whereas a 32×64 pixels and a 48×96 pixels classifier are used for pedestrians. For each classifier four scales are computed in total. We decided to use two classifiers with different dimensions for pedestrians in order to be able to classify small objects and to be accurate in close proximity to the car at the same time. Figure 10 and 11 visualize the result of this module during online navigation, where each classified object is marked with a colored bounding box according to the assigned object class (red stands for vehicle and green for pedestrian).

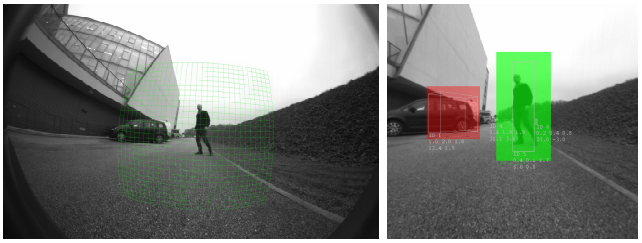


Figure 4: Example of a front fisheye camera image (on the left) and the related un-warped image used in the "Stereo Object Classification" stage (on the right). The left figure depicts the virtual plane used to obtain the pinhole image. For classification, object hypotheses are projected onto the pinhole image in order to define a ROI for the classifier. The ROI for the vehicle class is depicted in red, whereas the ROI for the pedestrian class is marked in green.

4 System-compliant Motion Planning Among Other Traffic Participants

The information from static environment perception, object prediction, together with a high-level plan from a topological planner operating on the roadgraph data structure is processed in a fast on-lane local motion planner, first presented in [11]. The local motion planner handles both static and dynamic mixed-traffic scenarios such as pedestrian avoidance, oncoming traffic and platooning.

4.1 On-lane Motion Planning

The local motion planner operates in a sampling-based manner, generating multiple system-compliant candidate motions along a reference path in a tree-like fashion. The candidate motions are constructed via a forward simulation of a detailed Ackermann vehicle model in conjunction with a simulated controller, regulating the system towards samples of a manifold aligned with the reference path. These candidate motions are drivable by construction and the approach is able to model challenging system characteristics such as dead times and actuator limits easily.

During tree generation, samples $\boldsymbol{m} = (d_{\mathrm{ref}}, v_{\mathrm{ref}}, k_{v,\mathrm{ref}})$ are iteratively drawn from a set M of a discretized state manifold and used as target states for closed-loop system control. The state manifold consists of all states that are heading- and curvature-aligned with the global reference path. A candidate motion is then constructed using a user-supplied vehicle controller $\boldsymbol{g}$, that generates a sequence of system inputs $\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{m})$ to regulate the forward-simulated system model $\boldsymbol{f}$, with

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{f}([\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \ldots], \boldsymbol{u}(t), t, \Delta t) \qquad (1)$$

towards the sample $\boldsymbol{m}$ over a fixed time period, T_{sim}. The computational complexity of the approach increases exponentially with the depth of the tree. In the V-Charge project a tree with depth one is used, which proved to be sufficient for good navigation performance. An exemplary candidate motion set is depicted in Figure 8.

4.2 Collision Detection With Static And Dynamic Scene Elements

Fast collision detection is the bottleneck for many motion planning algorithms.

In order to test candidate motions for collision with static objects a 2D occupancy grid is used. The rectangular shape of the ego vehicle is approximated by a collection of N disc shapes. The quality of the approximation of the rectangular shape improves with the number of discs used. This allows for fast collision tests of ego poses via N comparisons on a distance transform of the gridmap. Candidate motions in collision with a static obstacle at any point within the planning horizon are pruned. For collision-free trajectories, a soft cost term based on the minimum distance of the candidate motion to any static obstacle over time is computed. In our implementation the cost term decreases linearly up to a specified threshold distance.

The collision detection technique based on a 2D map inflation is applicable only for static objects where the time parameter is negligible. However moving objects may occupy the same space at a different point in time. While the majority of previously presented planning frameworks operating in dynamic environments use a naive multiple interference test for collision detection with moving objects where every robot pose is tested against every obstacle and single disk approximations are used for all objects, we found this method to be inappropriate for navigation in fairly densely occupied spaces. The disc approximation is often too coarse and causes an unacceptable loss of space for navigation. Furthermore multiple interference tests with exact collision shapes (e.g. oriented boxes for vehicles) are already too slow for online operation with a fairly small amount of dynamic objects in the scene. In [12] we presented a fast and exact collision detection technique for moving objects based on a bounding volume hierarchy (BVH) data structure in workspace-time space. In our approach we utilize an axis-aligned bounding box tree to efficiently prune the majority of oriented bounding box collision tests. This technique allows us to run the sampling-based motion planner at a frequency of 5 Hz. An exemplary tree is depicted in Figure 5.

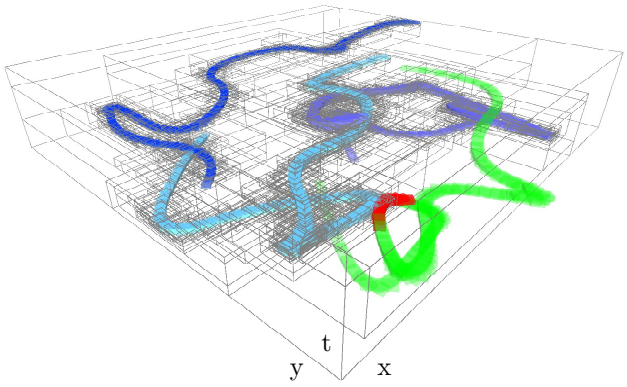


Figure 5: Axis aligned bounding box tree in workspace-time space for fast dynamic collision detection. Three obstacle trajectories are generated by a random walk (blue shades). The ego trajectory (green/red) can be efficiently tested for collision with the tree structure (wireframes).

In contrast to static obstacles, candidate motions in collision with dynamic objects are not pruned from the set. Instead, an additional "soft" cost term based on the estimated time to collision is introduced. This accounts for the fact that motion predictions of other traffic participants are inherently subject to uncertainty. Predictions become more uncertain the farther they are in the future. Therefore a cost term with exponential decay over time, J_{do}, is used (see Table 1).

4.3 Objective Function Design

For each candidate motion a scalar cost value is computed which is comprised of different terms. In the V-Charge project significant effort was spent to design an objective function that performs well in both static and dynamic environments. Each candidate motion is scored with respect to its alignment with the reference path and the lane's reference speed. Furthermore cost terms

Cost term	Effect	Formula
Terminal path and speed offset	lane and target speed keeping, enforces alignment at the end of the solution	$J_{\mathrm{sf}} = \boldsymbol{w}_{\mathrm{f}}^{\mathsf{T}} \begin{bmatrix} d\left(T_{\mathrm{sim}}\right) \\ \left\lvert v\left(T_{\mathrm{sim}}\right) - v_{\mathrm{ref,lane}}\left(c\left(T_{\mathrm{sim}}\right)\right)\right\rvert \end{bmatrix}$
Integrated path offset	lane keeping, the integral avoids short-cutting in sharp turns	$J_{\mathrm{id}} = \int_{t_0}^{T_{\mathrm{sim}}} d\left(t\right)\, \mathrm{d}t$
Integrated speed offset	target speed keeping along the reference speed profile	$J_{\mathrm{iv}} = \int_{t_0}^{T_{\mathrm{sim}}} \left\lvert v\left(t\right) - v_{\mathrm{ref,lane}}\left(c\left(t\right)\right)\right\rvert \, \mathrm{d}t$
Static obstacle distance	Yields safety distance and resilience to perception noise	$J_{\mathrm{so}} = 1.0 - \min\left(\min_{t\in[t_0,T\mathrm{sim}]}\left(d_{\mathrm{obs}}\left(t\right)\right), \epsilon\right) \Big/ \epsilon$
Deviation to last solution	Regularization term resolves nearby minima of the cost function, leads to smooth behaviour over consecutive planning cycles	$J_{\mathrm{reg}} = \boldsymbol{w}_{\mathrm{r}}^{\mathsf{T}}\left(\boldsymbol{m} - \boldsymbol{m}_{k-1}^{*}\right)$
Integrated control effort	Lower energy consumption and smoother behaviour	$J_{\mathrm{u}} = \int_{t_0}^{T_{\mathrm{sim}}} \boldsymbol{u}^{\mathsf{T}}\left(t\right) \boldsymbol{w}_{\mathrm{u}} \boldsymbol{u}\left(t\right)\, \mathrm{d}t$
Dynamic collision	Collision avoidance for dynamic objects, based on time to collision T_{coll}	$J_{\mathrm{do}} = \beta \exp\left(-\gamma \min_{i\in[1,N_{\mathrm{dynobs}}]}\left(T_{\mathrm{coll},i}\right)\right)$

Table 1: Cost terms used in the local planning framework.

are introduced for soft penalization of close distances to static obstacles, integrated steering speed and collisions with dynamic objects. The cost terms and their motivations are summarized in Table 1, where $\boldsymbol{w}_{\mathrm{f}}$, $\boldsymbol{w}_{\mathrm{r}}$ and $\boldsymbol{w}_{\mathrm{u}}$ are user-supplied weight vectors, $c\left(t\right)$ denotes the curvilinear abscissa along the reference path associated with the vehicle pose at time t and $\boldsymbol{m}_{k-1}^{*}$ is the optimal sample in the previous planning cycle. For a more intuitive tuning of the weights, the individual cost terms are normalized to lie within the range $[0, 1]$ (this requires the cost of all candidate motions' cost terms to be available). The final cost of a candidate motion for sample $\boldsymbol{m}_{\mathrm{n}}$ is computed via a weighted sum of the cost terms via

$$J\left(\boldsymbol{m}_{\mathrm{n}}\right) = \boldsymbol{w}_{J}{}^{\mathsf{T}} \begin{bmatrix} \mathrm{norm}\left(J_{\mathrm{sf}}\left(\boldsymbol{m}_{\mathrm{n}}\right)\right) \\ \mathrm{norm}\left(J_{\mathrm{id}}\left(\boldsymbol{m}_{\mathrm{n}}\right)\right) \\ \mathrm{norm}\left(J_{\mathrm{iv}}\left(\boldsymbol{m}_{\mathrm{n}}\right)\right) \\ \mathrm{norm}\left(J_{\mathrm{so}}\left(\boldsymbol{m}_{\mathrm{n}}\right) - J_{\mathrm{so,d}}^{*}\right) \\ \mathrm{norm}\left(J_{\mathrm{reg},n}\right) \\ \mathrm{norm}\left(J_{\mathrm{u},n}\right) \\ G_{d}\left(J_{\mathrm{do},n}\right) \end{bmatrix}. \quad (2)$$

In the following we will discuss certain aspects of a subset of these cost terms in more detail.

Static obstacle distance cost term In equation 2, $J_{\mathrm{so,d}}^{*} = \{J_{\mathrm{so}}\left(\boldsymbol{m}_{\mathrm{n}}\right) \mid \boldsymbol{m} \in M, \boldsymbol{m}_{\mathrm{d}} = \boldsymbol{m}_{\mathrm{n,d}}\}$ is the minimal static obstacle cost of all samples with the same lateral offset d_{ref} as sample $\boldsymbol{m}_{\mathrm{n}}$. This yields a static obstacle cost term of zero for the sample with the largest minimum distance to any obstacle. Without this modification, narrow passages appear as high cost regions. Depending on the ratio between the static obstacle weight and the weights influencing the longitudinal speed terms, the cost function might score candidate motions coming to a full standstill in front of the narrow passage better than motions passing the high cost region induced by the static obstacle cost term. The modification encodes the intention to keep a safety distance to static obstacles if possible, but does not penalize close distances if the navigation space gets tight. This behaviour is depicted in Figure 6 and 7.

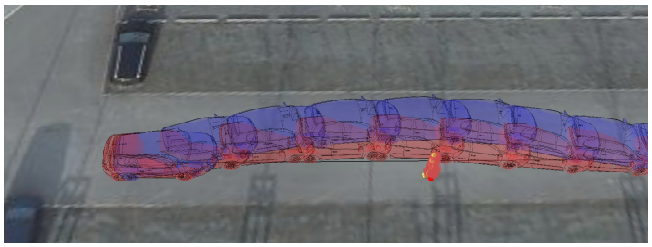


Figure 6: Influence of static obstacle distance cost term. The red track depicts the navigation behaviour without the cost term. Obstacles are passed without safety distance. The blue track shows navigation with active static obstacle distance penalization, resulting in a safe and comfortable avoidance of the object.

Dynamic collision cost term The dynamic collision detection framework returns binary answers to collision queries between candidate motions and predicted trajectories of other traffic participants. This results in an ego motion passing objects with the minimal realizable distance, rendering the planning stage vulnerable to measurement noise and resulting in maneuvers being



Figure 7: Influence of the post-processing step $J_{\mathrm{so}}(\boldsymbol{m}_{\mathrm{n}}) - J^{*}_{\mathrm{so,d}}$ applied to the static obstacle cost term to improve navigation performance in narrow passages. Without the modification, the vehicle denies to pass the narrow passage (red track).

perceived as risky. While the BVH-based collision detection framework provides the possibility for accelerated distance queries between objects' trajectories, this comes with a significant computational overhead compared to binary collision tests. Inspired by [13], we instead use a Gaussian smoothing filter G_d over the dynamic object cost term in the lateral offset dimension d. The filter blends costs from collisions with dynamic objects into neighboring candidate motions which results in the desired side clearance without the need to perform expensive distance queries. The dynamic object cost term is excluded from the normalization procedure due to its safety-critical nature. The resulting navigation behaviour is illustrated in Figure 8 in a simulated run. In the beginning (upper figure), the candidate motions to the left of the blue ego vehicle already collide, however the importance of the collision is low due to the exponential cost decay. The closer the oncoming car approaches (middle figure), the higher the cost of the colliding candidate motions becomes and the ego vehicle initiates the avoidance maneuver. After passing the oncoming vehicle (lower figure), the ego vehicle maneuvers around two static vehicles that entered the field of view of the front stereo camera.

Reference speed The lane's reference speed $v_{\mathrm{ref,lane}}(c)$ at a given curvilinear abscissa c is read from an attribute of the road network description in the semantic map database. Lane width information from the database is used to adapt the lateral offset range of the samples $\boldsymbol{m}$ on-the-fly.

4.4 Situation-Specific Adaptations

The local motion planner operates almost completely without inputs from higher-level situational awareness modules. The primary goal is to encode suitable behaviours in the objective function of the local motion planner and only to fall back on higher-level behaviour modifications if necessary.

For smooth platooning behaviour, the reference speed $v_{\mathrm{ref,lane}}$ is modified to yield a specified time-gap and/or

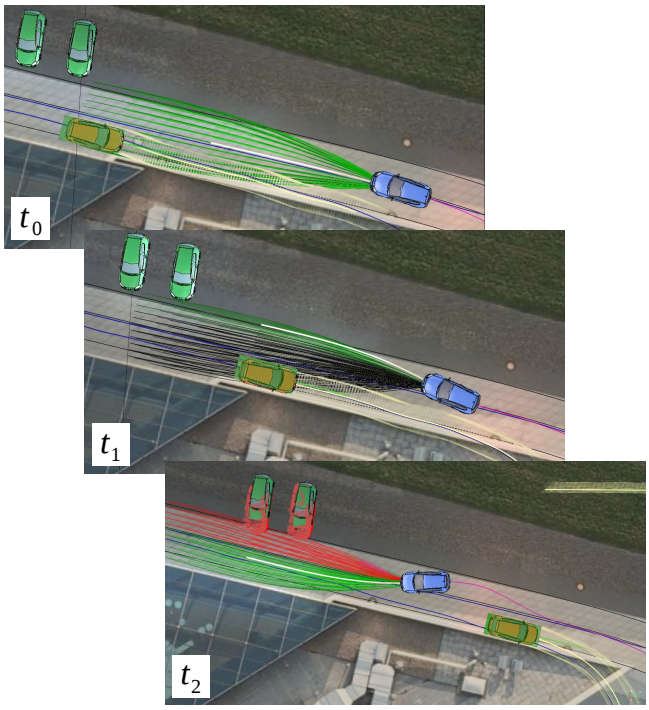


Figure 8: Simulated avoidance maneuver with an oncoming vehicle. Color scheme for candidate motions: green = low cost, black = high cost, red = in collision with static obstacle.

distance between the ego vehicle and the car in front. This solely shifts the minimum of the objective function, leaving the rest of the planning framework unchanged. To determine the relevant object in front, a scoring metric in the lane's Frenet frame is used based on the objects' longitudinal and lateral distances to the ego vehicle along the lane.

5 Movement Predictions For Traffic Participants

Reasoning about future movements of other traffic participants is important for well-behaved navigation in mixed-traffic scenarios. Our motion prediction scheme is based on a forward simulation of different agent motion models. A suitable motion model is chosen online based on the associated object class label determined by the classification module. Our classification module and hence the motion prediction module distinguishes between pedestrians and vehicles. As pedestrians often move in an unconventional way and do not respect their dedicated walkways rigorously (especially in parking lots the V-Charge project is dedicated to), a constant velocity model is applied for pedestrians. In contrast to pedestrians, movements of vehicles are bound much tighter to the infrastructure. Therefore we exploit the lane information of the roadgraph to improve upon the motion predictions of vehicles. In a first step, a kd-tree containing all three-dimensional (3D) poses (2D position x, y and yaw angle θ) of the roadgraph is built. This allows

for fast nearest neighbor queries for any vehicle pose in the 2D workspace in order to find the closest pose on the roadgraph. The heading dimension is included in order to distinguish between nearby lanes with opposite directions. We use a custom distance metric between two poses $\boldsymbol{c_0}$ and $\boldsymbol{c_1}$ in $SE(2)$

$$d^2(\boldsymbol{c_0}, \boldsymbol{c_1}) = (\boldsymbol{c_0}_x - \boldsymbol{c_1}_x)^2 + (\boldsymbol{c_0}_y - \boldsymbol{c_1}_y)^2 + \alpha(\boldsymbol{c_0}_\theta - \boldsymbol{c_1}_\theta)^2 \quad (3)$$

with α being a scaling factor for the angular heading dimension. During forward simulation with a discrete integration scheme three operations are performed in each timestep t_k: First, the closest pose of the simulated vehicle position at t_k to the roadgraph is retrieved from the kd-tree. Second, the lateral offset to the closest roadgraph pose is computed and a reference pose is created by laterally shifting the closest pose by this offset. In the third step, a control law is used to compute a desired steering angle for the simulated Ackermann model. This steering angle regulates the simulated model towards the computed reference state. The speed of the vehicle is kept constant at its measured speed during the simulation time horizon. The result of this motion prediction scheme is illustrated in Figure 9.

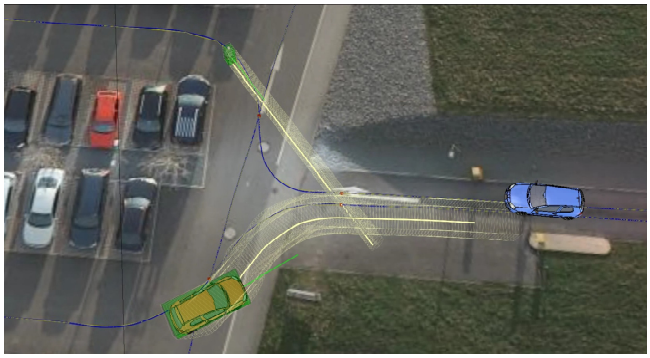


Figure 9: Movement predictions for two different agent types over a time horizon of 10 seconds. Green boxes represent the detected bounding boxes of the objects. The yellow wireframe objects depict the predictions.

6 Experimental Results

In this section we provide real-world data for two scenarios, that frequently appear during autonomous operation on a parking lot.

6.1 Pedestrian Avoidance

Figure 10 depicts the navigation behaviour in the presence of a pedestrian crossing the street. The constant velocity prediction allows the vehicle to infer the future positions of the classified pedestrian agent. A lateral avoidance maneuver is not considered necessary, leading to a short reduction of the ego vehicle's speed in order to give way to the pedestrian. Afterwards the vehicle speeds up again to meet its desired reference speed.

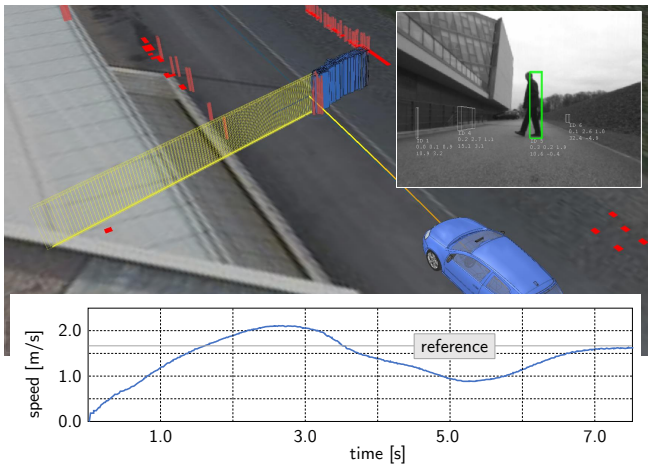


Figure 10: Real-world avoidance of crossing pedestrian. The pedestrian is sensed by the stereo camera (red stixels) and classified as such by the stereo classification module based on the monocular camera images (top right figure). The vehicle slows down to give way to the pedestrian.

6.2 Platooning

Platooning is a situation appearing frequently in traffic. Figure 11 visualizes the navigation performance during such, involving multiple stop-and-go phases. In this run, the car in front is reliably tracked throughout the complete four minutes of the run. The speed profile of the ego vehicle and the target car match with a slight delay induced by delays in the processing chain but mainly due to the low-pass characteristics of the drive-train.

The left figure depicts the first approach to the car in front. The middle figure illustrates the local planner's ability to concurrently adapt its plan (orange trajectory) based on static obstacle information. The static obstacle distance cost term moves the ego vehicle slightly towards the left to keep a safety distance to the pillar on the right. In the rightmost figure the roadgraph-based object prediction utilizing the lane center information (blue path) is visible.

7 Conclusion

The goal of automated driving in mixed-traffic scenarios with a camera-only sensor suite is and remains challenging. The V-Charge automated driving platform is to our knowledge the first platform that does not complement the vision sensor suite with additional laser or lidar scanners. In this paper we reported on the on-lane planning framework and the processing chain necessary to implement smooth interaction with other traffic participants like vehicles and pedestrians. Our future research aims at extending the object detection framework to all

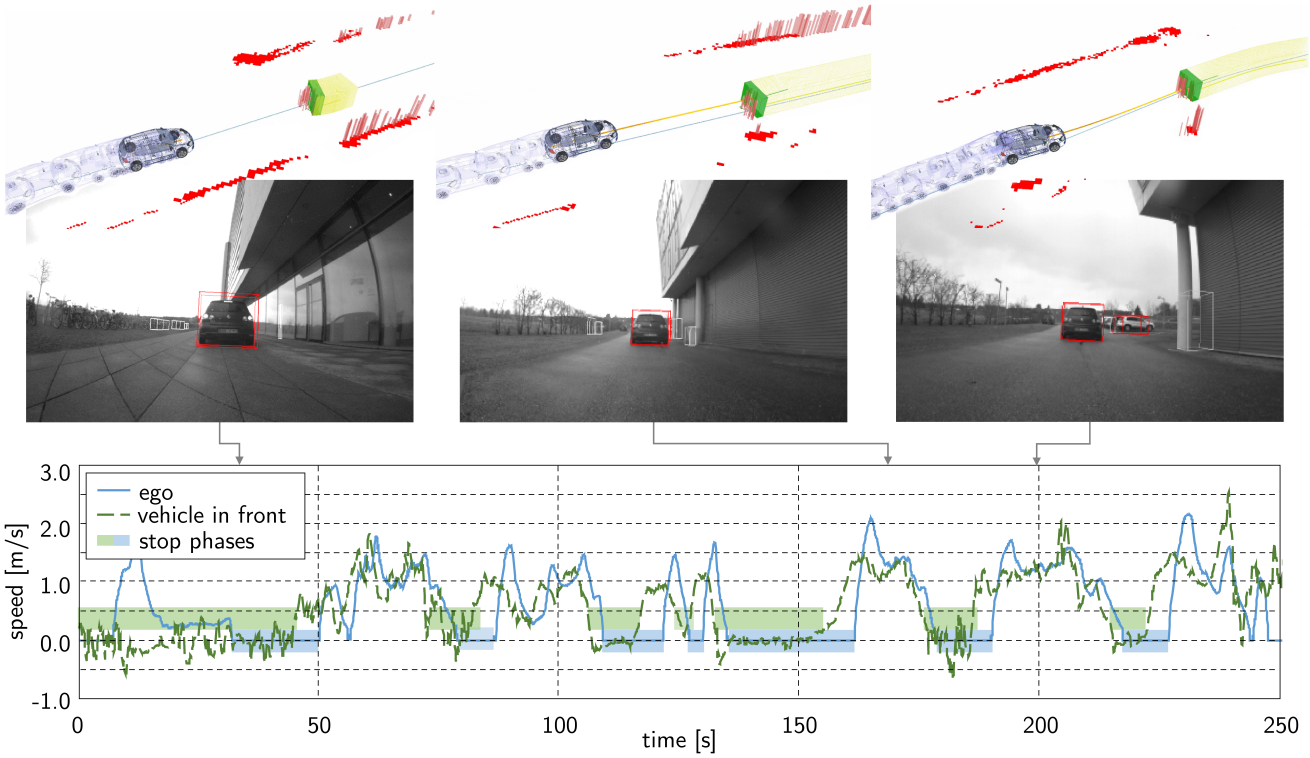


Figure 11: Real-world platooning performance in stop-and-go traffic.

available monocular cameras as well as incorporating interaction-awareness in the object prediction stage by machine learning techniques.

Acknowledgement

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 269916, V-Charge.

Literature

[1] C. Stiller, S. Kammel, B. Pitzer, J. Ziegler, M. Werling, T. Gindele, and D. Jagszent, "Team AnnieWAY's autonomous system," *Lecture Notes in Computer Science*, vol. 4931 LNCS, pp. 248–259, 2008.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y. W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in Urban environments: Boss and the Urban Challenge," *Springer Tracts in Advanced Robotics*, vol. 56, no. 8, pp. 1–59, 2009.

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Springer Tracts in Advanced Robotics*, vol. 56, no. 9, pp. 91–123, 2009.

[4] A. Broggi, A. Cappalunga, C. Caraffi, S. Cattani, S. Ghidoni, P. Grisleri, P. P. Porta, M. Posterli, and P. Zani, "TerraMax Vision at the Urban Challenge 2007," *IEEE Trans. on Intelligent Transportation Systems*, vol. 11, pp. 194–205, Mar. 2010.

[5] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, "Extensive tests of autonomous driving technologies," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, pp. 1403–1415, Sept 2013.

[6] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, M. Panciroli, and A. Prioletti, "PROUD-public road urban driverless test: Architecture and results," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 648–654, 2014.

[7] B. Spice and C. Swaney, "Carnegie mellon creates practical self-driving car using automotive-grade radars and other sensors," Sept. 2013. CMU Press Release, available at http://www.cmu.edu/news/stories/archives/2013/september/sept4_selfdrivingcar.html.

[8] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive-an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, pp. 8–20, Jan. 2014.

[9] C. Urmson, "The latest chapter for the self-driving car: mastering city street driving," May 2014. The Official Google Blog, available at http://googleblog.blogspot.ch/2014/04/the-latest-chapter-for-self-driving-car.html.

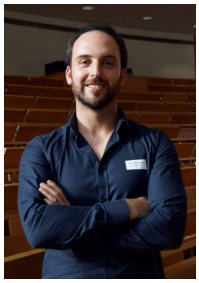
[10] R. Benenson, M. Mathias, R. Timofte, and L. J. V. Gool, "Pedestrian detection at 100 frames per second.," in *CV-*

PR, pp. 2903–2910, IEEE, 2012.

[11] U. Schwesinger, M. Rufli, P. Furgale, and R. Siegwart, "A sampling-based partial motion planning framework for system-compliant navigation along a reference path," in *Proc. of the IEEE Intelligent Vehicles Symposium (IVS)*, pp. 391–396, June 2013.

[12] U. Schwesinger, R. Siegwart, and P. Furgale, "Fast Collision Detection Through Bounding Volume Hierarchies in Workspace-Time Space for Sampling-Based Motion Planners," *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. accepted manuscript.

[13] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, pp. 23–33, Mar 1997.



Ulrich Schwesinger received the Diploma degree in Electrical Engineering and Information Technology from the Karlsruher Institute of Technology (KIT) in May, 2010. During his studies he focused on signal processing, navigation and control. In October 2010, he joined the Autonomous Systems Lab at ETH Zurich under the supervision of Prof. R. Siegwart, where he worked on the locomotion system of ESA's mars rover for the "ExoMars" mission. Since June 2011, his research focuses on motion planning and collision avoidance within the V-Charge project targeting automated navigation and parking for electric vehicles.

Address: Autonomous Systems Lab, Institute of Robotics and Intelligent Systems, ETH Zurich, LEE J301, Leonhardstrasse 21, CH-8092 Zurich, E-Mail: ulrich.schwesinger@mavt.ethz.ch



Pietro Versari received the Master degree in Computer Engineering at the University of Parma in October, 2010. After his Master degree, he joined the Artificial Vision and Intelligent Systems Laboratory (VisLab) under the supervision of Prof. A. Broggi, where he worked on various projects related to the autonomous vehicles fields. Since February 2013, his research focuses on perception and object classification within the V-Charge project.

Address: VisLab, Department of Information Engineering, University of Parma, Parco Area delle Scienze, 181/A, Building 1, IT-43124 Parma, E-Mail: versari@vislab.it



Prof. Alberto Broggi received the Dr. Ing. (Master) degree in Electronic Engineering and the Ph.D. degree in Information Technology both from the University of Parma, Italy, in 1990 and 1994, respectively. He is now Full Professor at the University of Parma and President and CEO of the VisLab spinoff company. He is author of more than 150 publications on international scientific journals, book chapters, refereed conference proceedings. He served as Editor-in-Chief of the IEEE Transactions on Intelligent Transportation Systems for the term 2004-2008; he served the IEEE Intelligent Transportation Systems Society as President for the term 2010-2011.

Address: VisLab, Department of Information Engineering, University of Parma, Parco Area delle Scienze, 181/A, Building 1, IT-43124 Parma, E-Mail: broggi@vislab.it



Prof. Roland Siegwart is a full professor for Autonomous Systems and Vice President Research and Corporate Relations at ETH Zurich since 2006 and 2010 respectively. He has a Master in Mechanical Engineering (1983) and a PhD in Mechatronics (1989) from ETH Zurich. From 1996 to 2006 he was associate and later full professor for Autonomous Microsystems and Robots at the Ecole Polytechnique Fédérale de Lausanne (EPFL). Roland Siegwart is member of the Swiss Academy of Engineering Sciences, IEEE Fellow and officer of the International Federation of Robotics Research (IFRR). He served as Vice President for Technical Activities (2004/05) and was awarded Distinguished Lecturer (2006/07) and is currently an AdCom Member (2007-2011) of the IEEE Robotics and Automation Society. He leads a research group of around 30 people working in the fields of robotics, mechatronics and product design.

Address: Autonomous Systems Lab, Institute of Robotics and Intelligent Systems, ETH Zurich, LEE J205, Leonhardstrasse 21, CH-8092 Zurich, E-Mail: rsiegwart@ethz.ch

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.

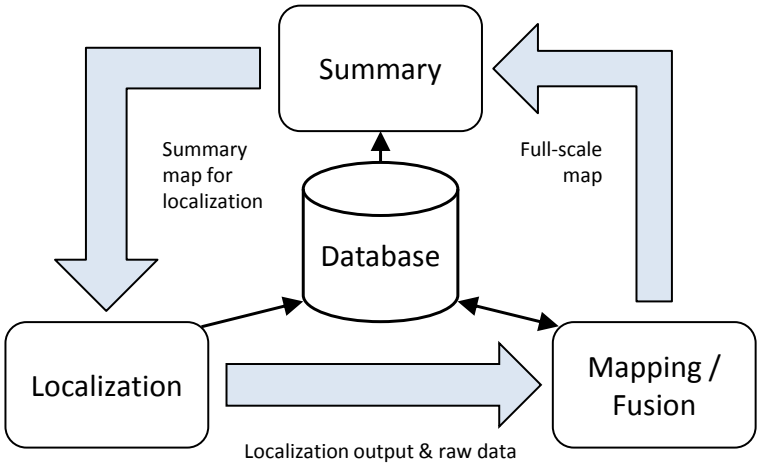


Figure 1: *The life cycle for a map. A database functions as the repository of all information necessary for localization, mapping, and summary of the map.*

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*.

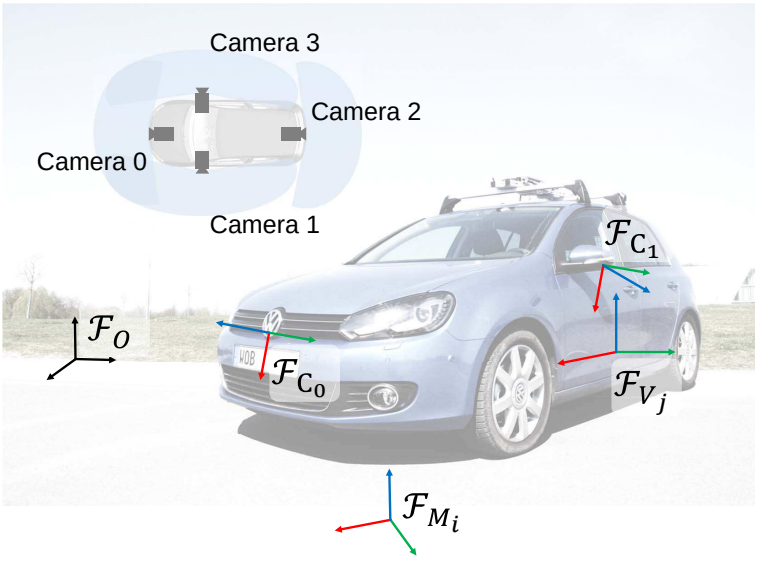


Figure 2: *Our driverless car, showing the location of the involved coordinate frames. Of the four wide-angle fisheye monocular cameras on the car, only the front and left cameras are visible, alongside their associated coordinate frames* $\mathcal{F}_{C_{0,1}}$*. Two additional cameras are arranged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

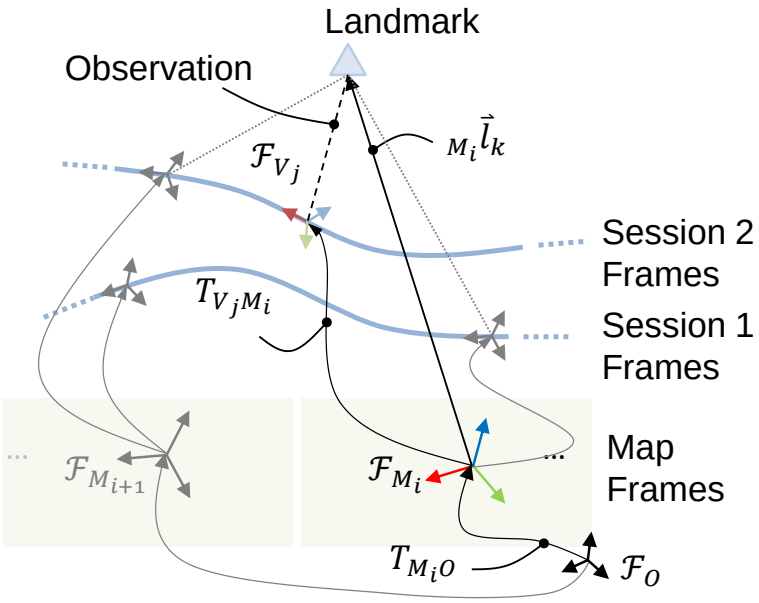


Figure 3: *The data structures and associations involved in the map. A single map and vehicle frame have been highlighted for clarity.*

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.

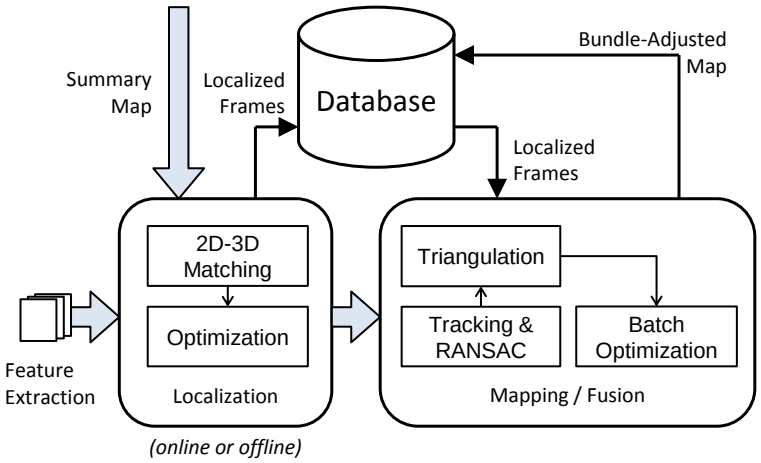


Figure 4: *The pipeline for the localization and mapping steps of the life cycle depicted in Figure 1.*

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.

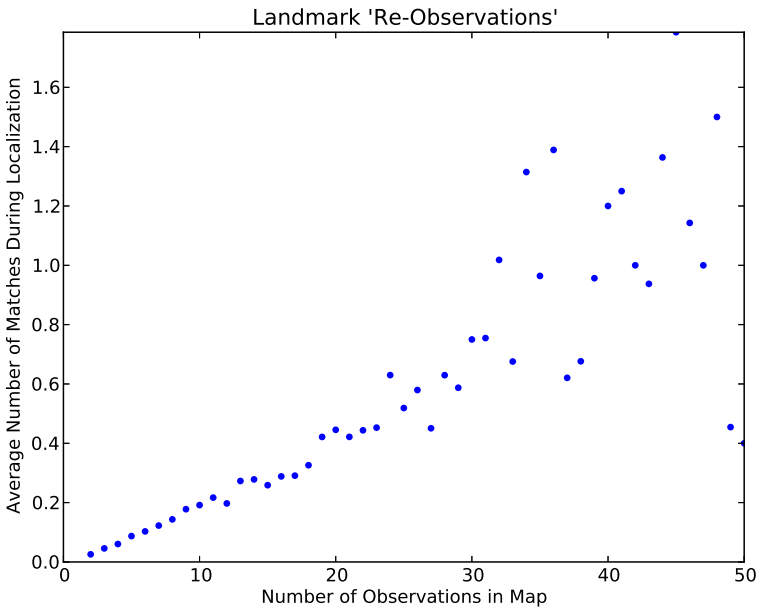


Figure 5: *The number of times a landmark is seen during mapping, versus the average number of times it is matched during localization, at a later point in time. X-Axis is truncated at 50 observations.*

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.

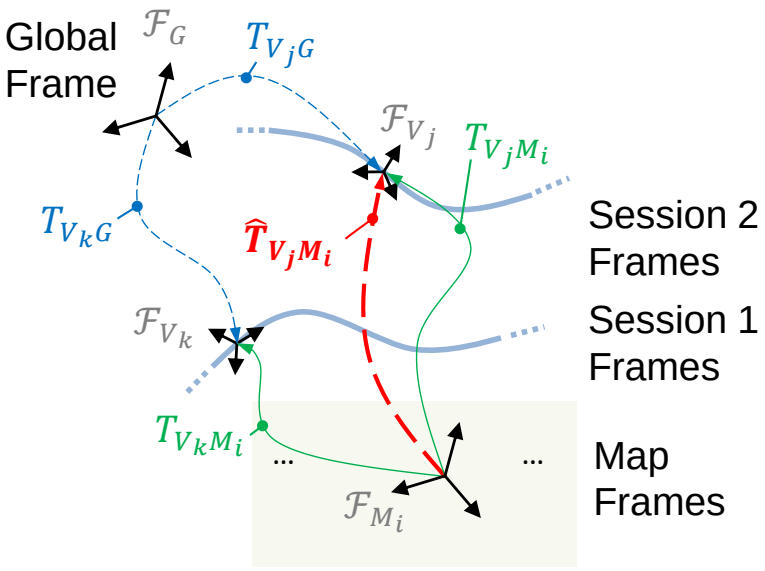


Figure 8: *Coordinate frames and transformations involved in calculating the local metric error. Transformations provided by INS are shown as dashed blue- and vision-based estimates in solid green arrows. The calculated 'reference' transformation* $\hat{\mathbf{T}}_{V_jM_i}$ *is shown in bold-dashed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

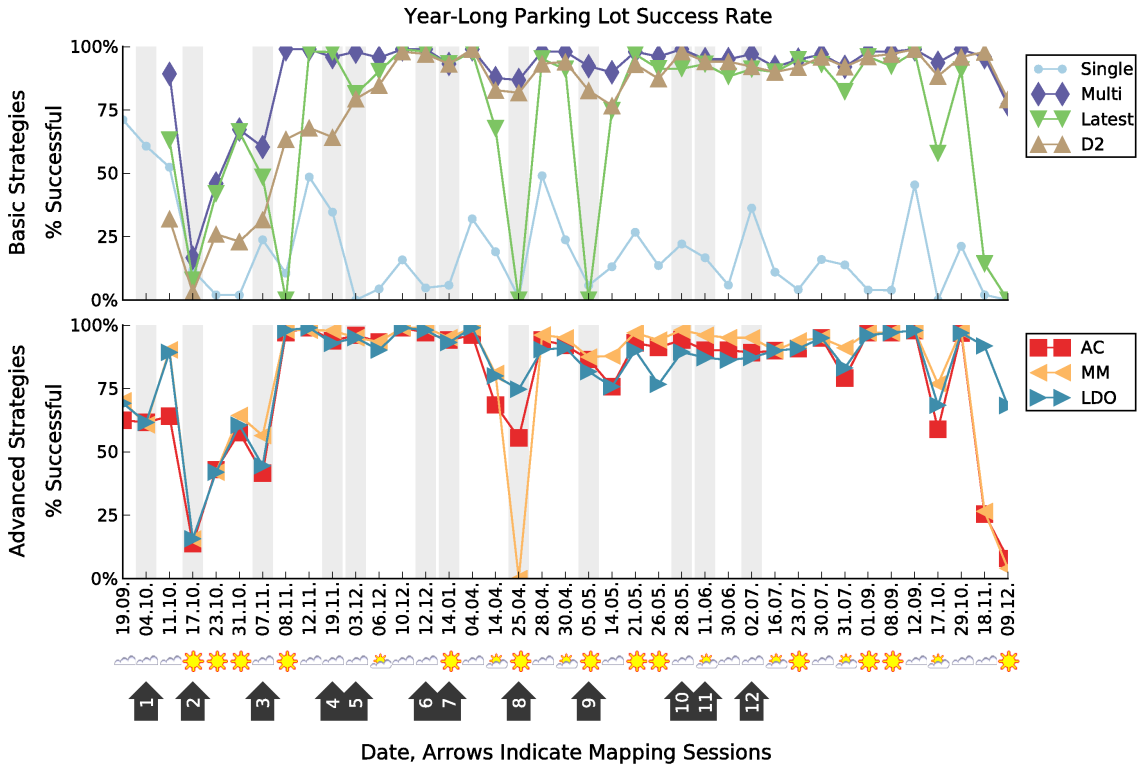
Referring to the first question, we analyze the availability of localization in the proposed framework. To determine if the generated maps are useful, we also analyze whether the metric error of the 6-DOF pose estimates they provide. With regards to the second question, we look at the size of the map, 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.

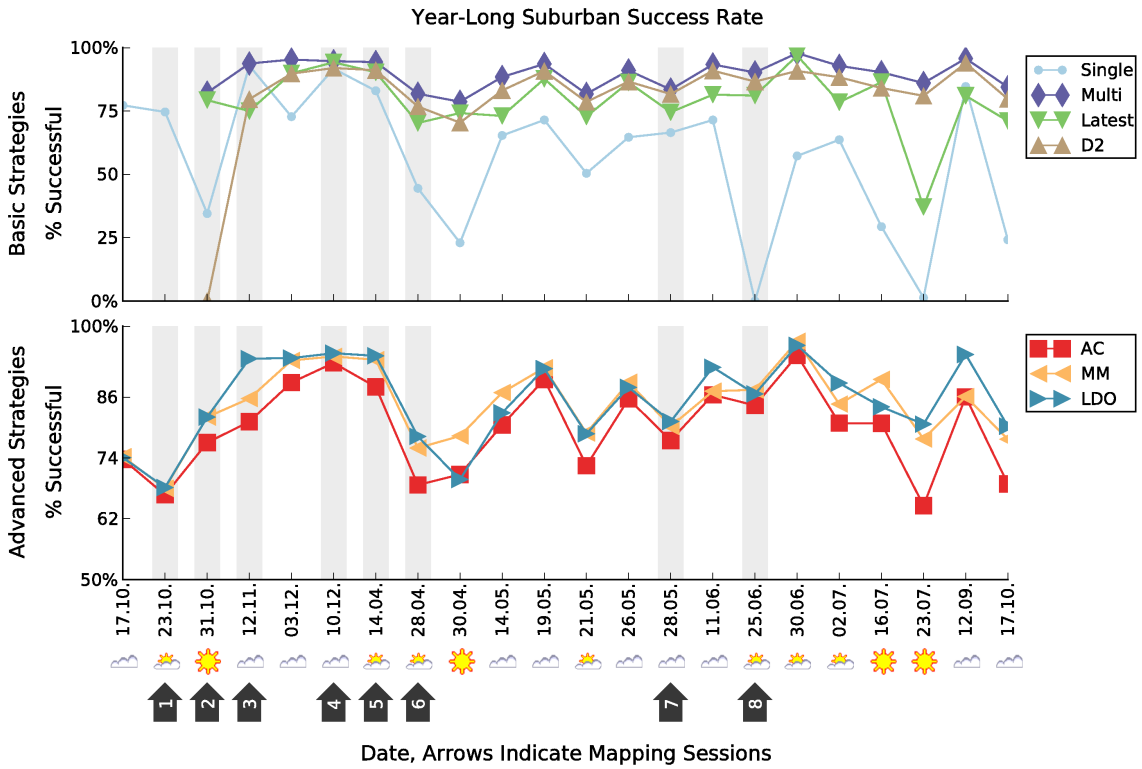
For the Year-Long Parking Lot and Suburban datasets, we used an *incremental* mapping and test procedure. Starting with an initial map base on the first mapping-session for each dataset, we built up the map by fusing all mapping-sessions in chronological order. At each time step, we also generated several 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.*

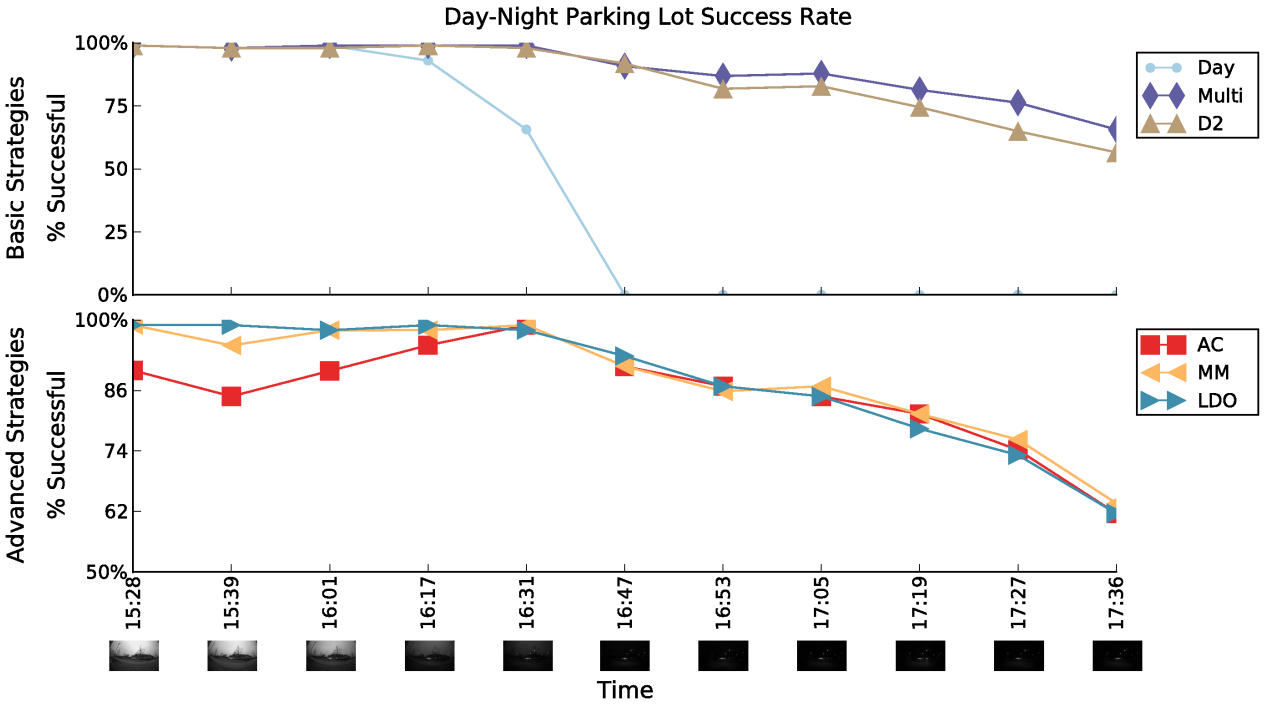


Figure 10: *Availability for the Day-Night dataset, calculated as the number of successful localizations divided by the total of localization attempts (see 4.3.1). Split according between basic and advanced summarization strategies. A downscaled image from the front-camera is shown at the bottom, to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

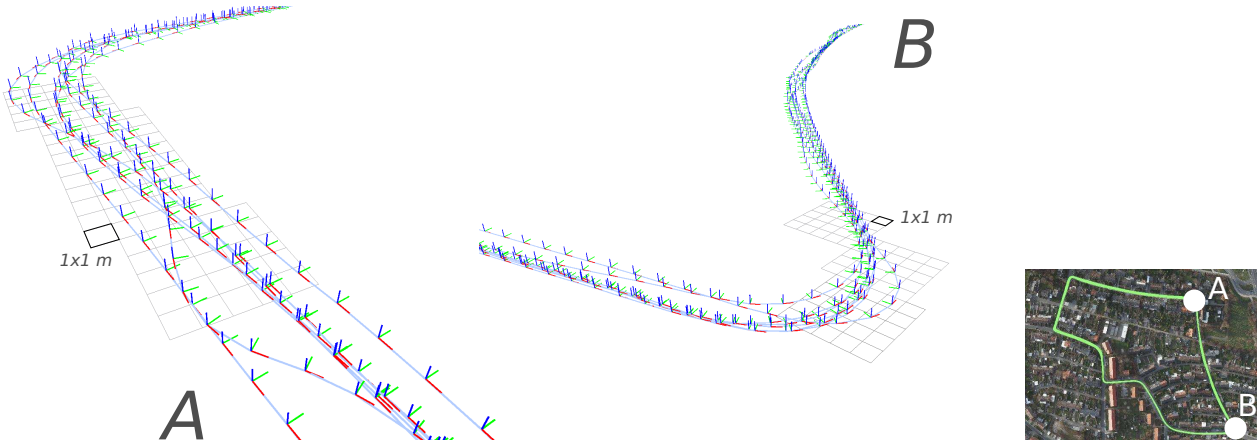


Figure 11: *Metric 6-DOF pose estimates, illustrated as oriented 3D axis-triples in a common reference frame per location. The results for several test runs from the Suburban dataset are shown; the example two locations are identified in the lower right corner. For scale, square grid cells with one meter 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.

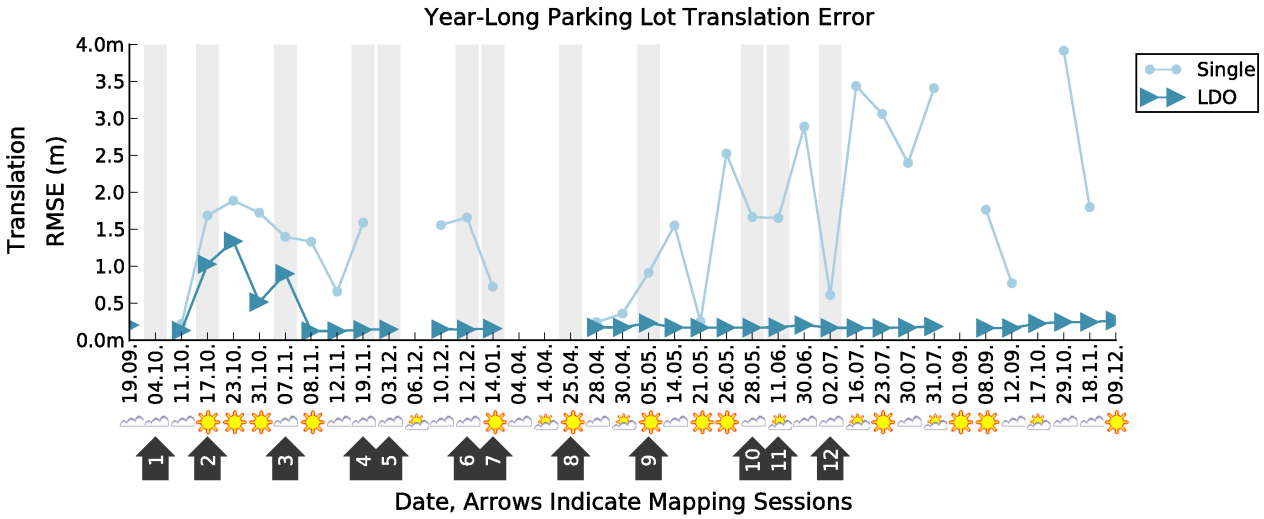
The evaluation of the Day-Night dataset shows us that the proposed mapping procedure can cope with extreme scene changes, in addition to the structural and seasonal variations captured by the long-term datasets. While the performance of the different summary strategies varies, we can nonetheless identify 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

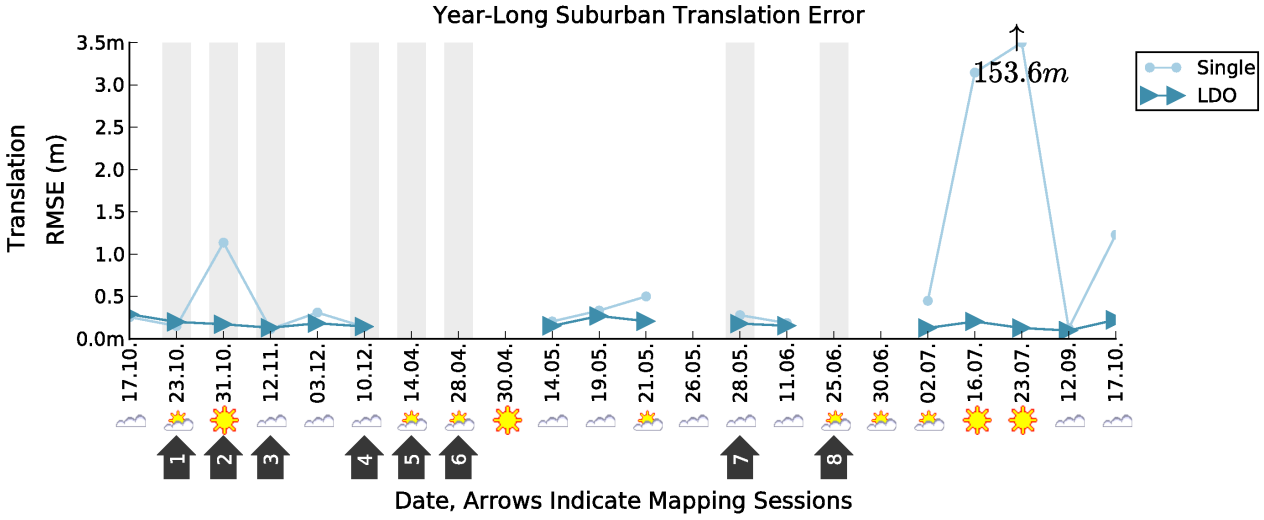
Providing locally accurate metric localization is one of the major goals of the proposed pipeline. Figure 11 illustrates the 6-DOF pose output produced, using the LDO map, at two example locations of the Suburban dataset. The Figure also shows the viewpoint variations in the dataset that result from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

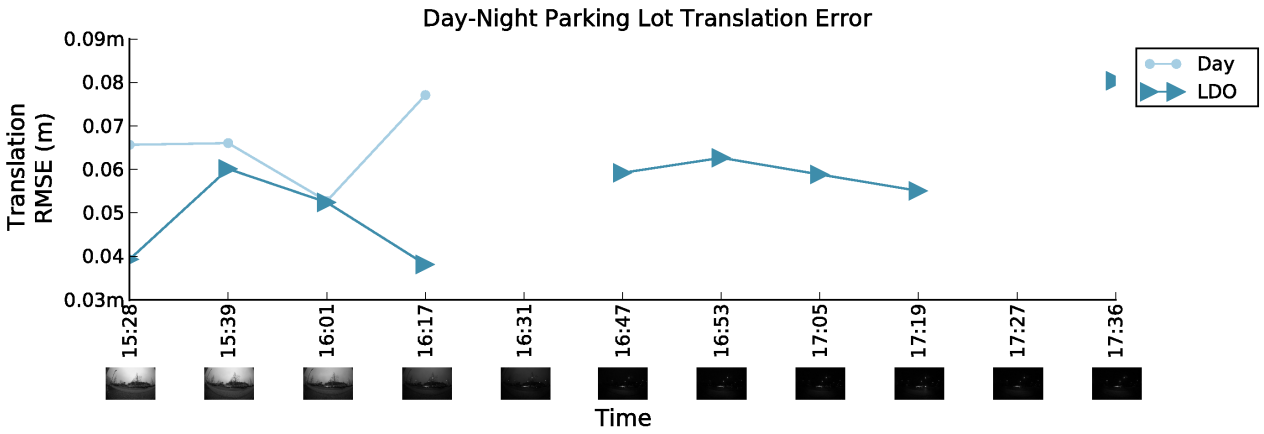


Figure 13: *Evolution of the RMS translational error for all datasets. To demonstrate illumination conditions, a downscaled image is show below each test-time. Note that the DGPS system suffered from intermittent problems at 16:31 and 17:27 – the final error for the LDO map at 17:36 should not be overlooked.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

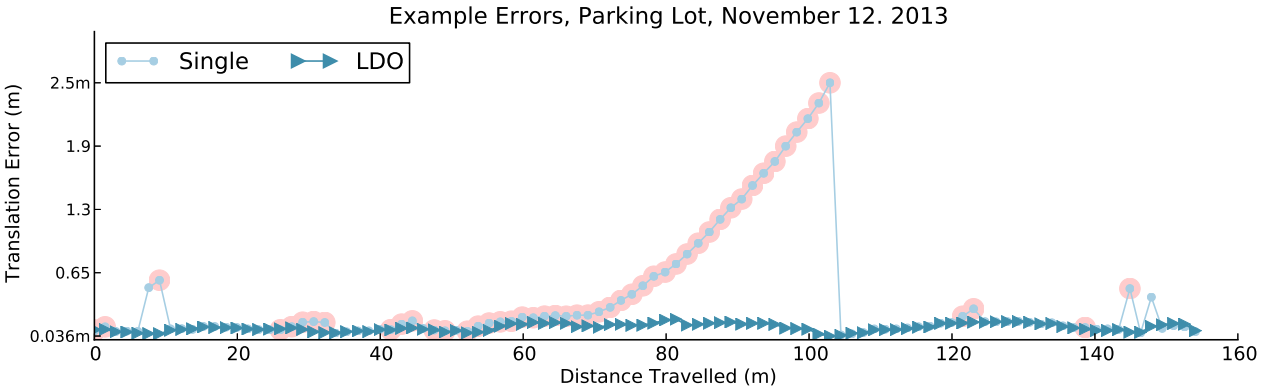


Figure 14: *Translational error, summarized as the norm, $\|t\|$, of the X-, Y-, Z-error vector over distance traveled during a single dataset. Localization failures (see* Availability*, Section 4.3.1) are marked with a faint-red background. This demonstrates how estimates drift when localization fails.*

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

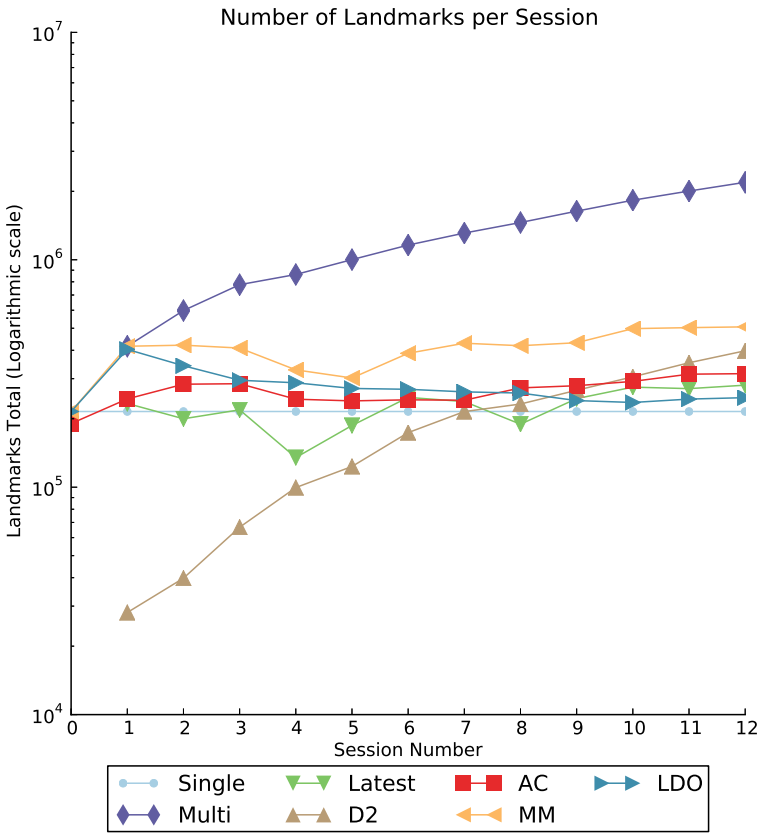


Figure 15: *The landmark total on a logarithmic scale, per session added to the map. Behaviour roughly falls into two groups: maps with a bounded number of landmarks, and maps with unbounded size that grow linearly. The special convergence behaviour of the Local-Ranking maps can be explained as a quantization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

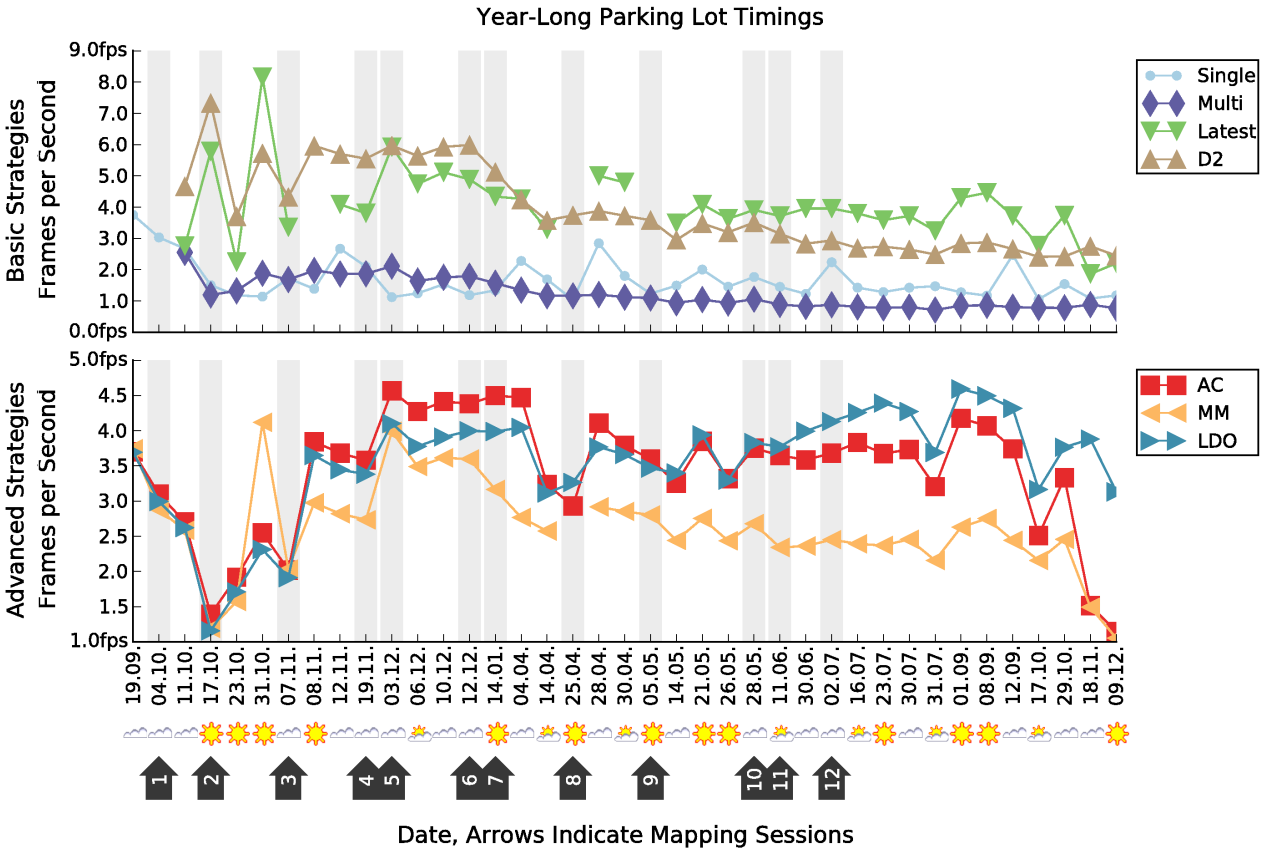


Figure 16: *The framerate of the full localization pipeline without feature extraction, for the various map types, over all datasets. Grey vertical lines indicate when new data was added to the map. The framerate of the unbounded maps steadily drops as new sessions are added, while the advanced strategies 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.