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

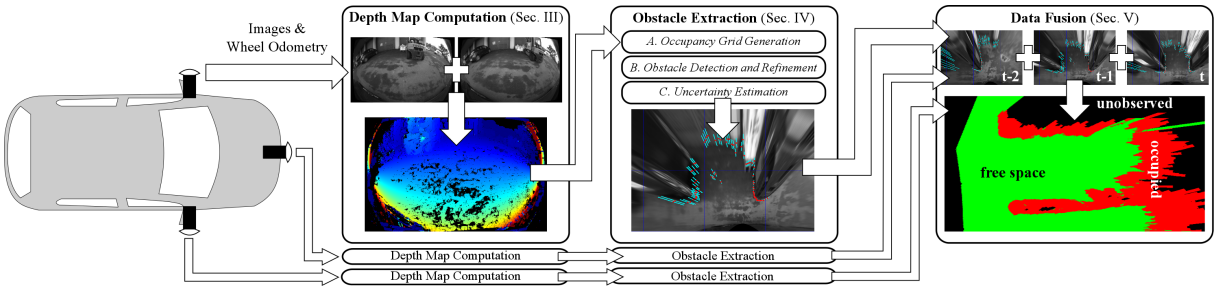

Fig. 1. Overview over the obstacle detection system proposed in this paper.

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

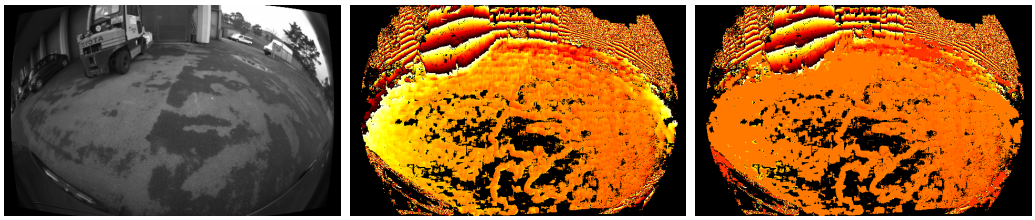

Fig. 2. Results of our depth map computation procedure. (Left) image of the reference view, the depth maps computed (middle) without ground direction plane-sweep and (right) the complete depth map computation procedure including the ground directions. The depth maps are coloured with respect to height 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.

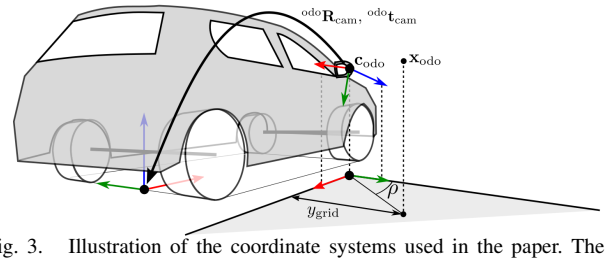

Fig. 3. Illustration of the coordinate systems used in the paper. The x-, y-, and z-axes are colored in red, green, and blue, respectively.

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

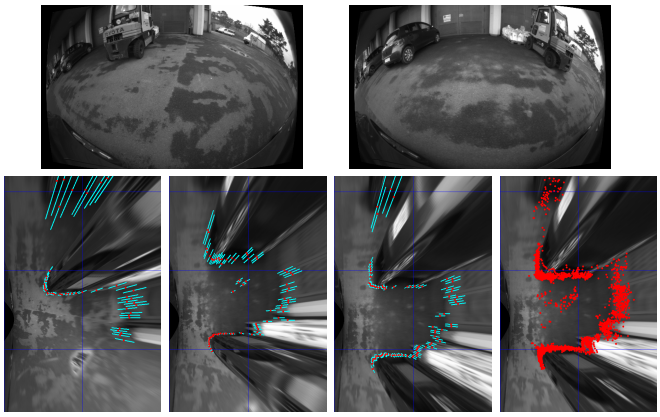

Fig. 4. Results of our obstacle detection approach: (Top row) Two input images. (Bottom row) Obstacles detected in three different frames, projected to the ground plane. The red dots denote the obstacle position and the cyan lines indicate the uncertainty area along the ray. The last image contains all 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

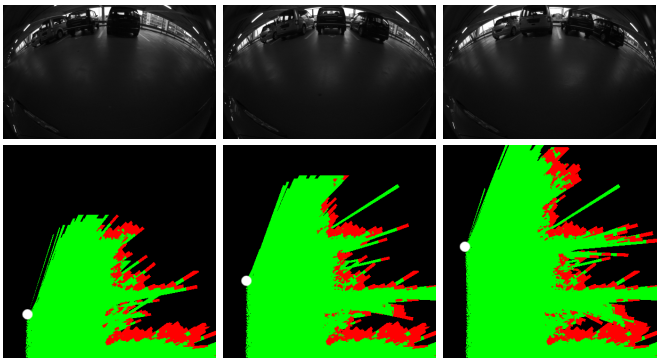

Fig. 6. Fusion results for an indoor sequence. Reflections on the ground cause erroneous detections while backlight complicates precise depth estimation. Still, our system is able to outline the positions of the obstacles.

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

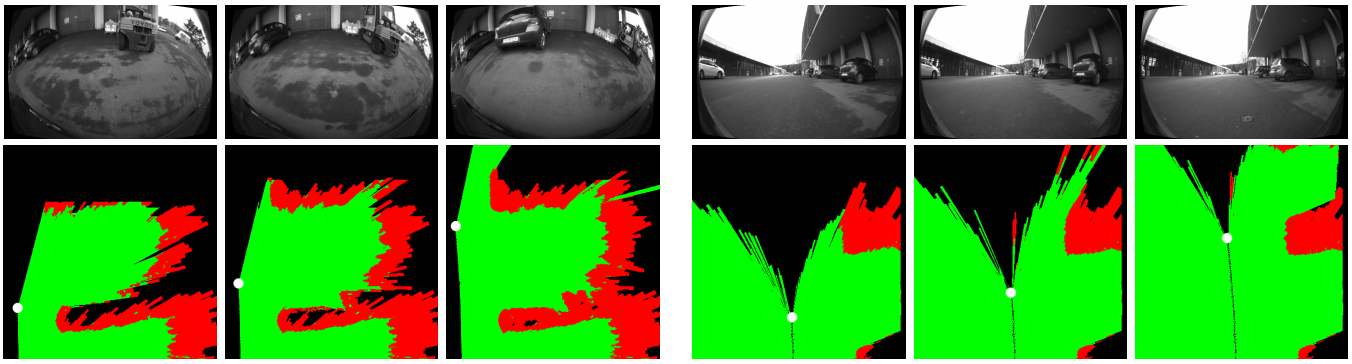

Fig. 5. Obstacle fusion results for two outdoor test sequences. (Top row) input images, (bottom row) fusion results. (Left hand side) right facing camera, (right hand side) front facing camera. Free space is colored green, occupied space in red, and unobserved space in black. The white dot denotes the 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.

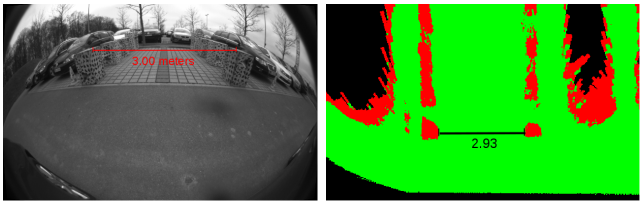

Fig. 7. Left: Ground truth measurment taken with a tape measure. Right: Distance that we determine in the fused results by measuring how wide the minimal gap between the first two foam boxes is (grid resolution 2.5cm).

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.

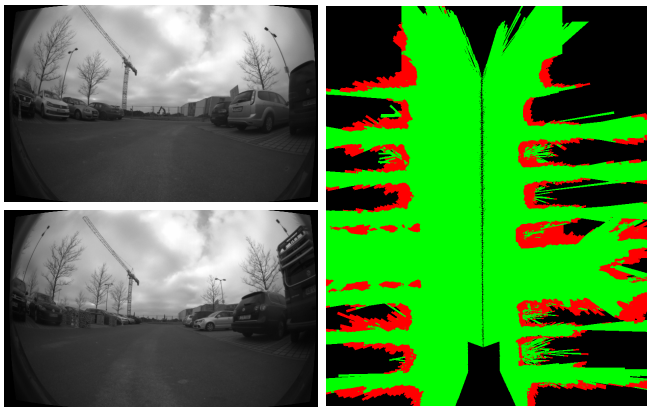

Fig. 8. (Left) Two input images of the front camera. (Right) Result of running our system concurrently on the left, front and right camera. The obstacle frames are entered into the grid with a joint framerate of 37.5Hz.

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.