Terrain-Adaptive Obstacle Detection

Benjamin Suger Bastian Steder Wolfram Burgard

***Abstract*— Reliable detection and avoidance of obstacles is a crucial prerequisite for autonomously navigating robots as both guarantee safety and mobility. To ensure safe mobility, the obstacle detection needs to run online, thereby taking limited resources of autonomous systems into account. At the same time, robust obstacle detection is highly important. Here, a too conservative approach might restrict the mobility of the robot, while a more reckless one might harm the robot or the environment it is operating in. In this paper, we present a terrain-adaptive approach to obstacle detection that relies on 3D-Lidar data and combines computationally cheap and fast geometric features, like step height and steepness, which are updated with the frequency of the lidar sensor, with semantic terrain information, which is updated with at lower frequency. We provide experiments in which we evaluate our approach on a real robot on an autonomous run over several kilometers containing different terrain types. The experiments demonstrate that our approach is suitable for autonomous systems that have to navigate reliable on different terrain types including concrete, dirt roads and grass.**

I. Introduction

Obstacle detection is a crucial ingredient for autonomously navigating robots. A sophisticated trade-off between safety and mobility constitutes the core of the problem. The former one has to guarantee that the robot does not harm itself or the environment, including humans and animals. The latter is not of less importance, because it is in charge to provide the robot its full capabilities for path planning and execution, and is therefore responsible whether or not a robot is able to successfully finish a task. Besides these considerations, the obstacle detection needs to be updated fast, since the robot should be able to react as fast as possible to safety-relevant information. Especially when going from flat indoor environments to challenging outdoor environments additional aspects regarding the safe navigation become relevant. In this work we set the focus on outdoor environments where the robot needs to navigate on different types of terrain, like ordinary streets or walkways, grass or meadows and forest roads or dirt paths. This setting is particularly challenging, as the question of what corresponds to an obstacle and what not also depends on the type of the terrain. It is relevant for different applications of robots, for example in agriculture or forestry.

Fig. 1 gives an intuition about the difficulties in mixed scenarios. The main problem is that the flatness assumption of man-made environments does not hold. Grass of various heights may easily appear as fake obstacles, because observations of the same height differences on a street may constitute a hazardous situation. This means that in mixed terrain settings there is no fixed threshold or heuristic that applies to decide about what is an obstacle, because free space on one type of terrain may look like an obstacle for another terrain type and vice versa. In this work we tackle this problem by adding semantic information about the terrain type to the obstacle detection, while we will rely only on a 3D-Lidar scanner as perceptual sensor. The choice for the lidar is that it is more robust to different lighting conditions, while, *e.g.*, a camera in a forest is highly affected by the varying lighting conditions. The semantic information allows for a differentiated interpretation of geometric measures like step height and steepness, which we calculate directly from the lidar data as it is delivered.

Fig. 1. Our robot navigating autonomous in a forest environment. We need to cope with grass of various heights, small trees and bushes. The corresponding point clouds need to be dealt with differently depending on the type of terrain the robot moves on.

Our 3D-Lidar sensor is a Velodyne HDL 64, which provides a full 360° horizontal field of view, returning ∼1.3 million points per second with distance and remission information that need to be processed. To cope with the resulting computational challenges, our approach takes advantage of the special structure of the Velodyne scans. The approach we present is able to detect traversable regions on medium high grass, dirt roads and regular streets using two threads on a quad core CPU i7@3.50GHz with a workload of ∼100% (of 400%) leaving enough computational power for other modules, like, *e.g.*, path planning and control, the robot may need.

II. Related Work

A comprehensive survey of traversability analysis methods for unmanned ground vehicles was recently published by

All authors are with the University of Freiburg, Institute for Computer Science, 79110 Freiburg, Germany. This work has been partly supported by the European Commission under the grant numbers ERC-AG-PE7-267686-LifeNav and FP7-610603-EUROPA2

Fig. 2. The trajectory of the outdoor experiment on an aerial image. The trajectory contains regular streets, grass and forest roads.

Papadakis [12]. He categorizes, non-exclusively, the approaches into proprioceptive and exteroceptive sensor data interpretation, which can be either geometry- or appearance-based. Our approach relies purely on exteroceptive sensor data, perceived with a 3D-Lidar, and we interpret the data geometry- and appearance-based. Moreover, he points out that the 2D-digital-elevation-maps [5, 11] are the most common choice when a 3D-Lidar is employed. The elevation maps were later used for mapping [13] and were extended to multilevel surface maps [17] to capture more complex environments. In this work we also employ a 2D-grid based map to organize and interpret the sensor data, where each cell stores features and information about the terrain type as well as about the geometry.

Most approaches concentrate on a certain terrain scenario, either structured outdoor urban environments or unstructured rough off-road terrain, since both environments typically underlie very different assumptions. In urban environments it is more important to detect and track dynamic objects, like cars, pedestrians and cyclists, and doing reasoning about the street lane or sidewalks [1, 4, 9]. For rough terrain analysis estimating the load bearing surface seems promising as done by [18, 19], but those approaches typically have a high computational burden, *e.g.*, 3D-ray-tracing, and also rely on different types of sensors. Another way is the estimation of the step height of cells, calculating the difference in the z-coordinate of points that fall into the cell, like in [3] where range-dependent thresholds were applied to determine obstacles. A probabilistic formulation was utilized by Thrun et al.. They eimployed five 2D-Lidar sensors, mounted in different angles, and perfomed statistical reasoning about the step height taking also time-differences into account. Manduchi et al. utilize RGB-D data from stereo-cameras and similar geometrical considerations like we do [8]. Instead of a grid-map based approach they reason about the obstacle natures of points by checking truncated cones that are placed on ground plane points. Terrain classification is deduced from the RGB-D information using a mixture of Gaussians. Moreover they rely on a 2D-lidar sensor for terrain classification analyzing histograms of distances. In contrast to this, our approach relies purely on 3D-Lidar data, which is less sensitive to lighting conditions and typically provides more accurate data.

Adding semantic information about the terrain type can be very useful, since different terrain types may follow different rules for different mobile robot platforms. Assuming a pure forest environment, [10] classifies the ground plane and trees, estimating the tree center using a least-squares fit on circles for candidate points, using 3D-Lidar data. A segmentation of 3D point clouds, classifying the categories *scatter*, to represent porous volumes, *linear*, to capture, *e.g.*, wires and tree branches, and *surface* to capture solid objects like ground surfaces, was presented by Lalonde et al. [7]. A Gaussian Mixture Model is trained with an expectation maximization algorithm, and segments are found using connected component analysis. Besides an online time-of-flight camera based basic obstacle detection, Santamaria-Navarro et al. perform an offline terrain classification using a Gaussian process classification approach [14], while the classification is only for traversable and not traversable. Wurm et al. use a tilted 2D-Lidar scanner to distinguish low grass vegetation from street in structured outdoor environments. The system utilizes self-supervised classification learning by employing a vibration classifier (proprioceptive sensor information) to train a support vector machine [20]. The features applied for classification are range, incident angle and remission. Laible et al. employ camera and lidar to distinguish asphalt, big tiles, grass, gravel and small tiles. In our work, we utilize distance and remission values from the 3D-Lidar to compute various features that we employ to determine the terrain class, where we apply a random forest classifier [2] to distinguish between regular street, dirt roads and grass. Finally we combine the semantic terrain class estimate with geometrical measures like step height and steepness, calculated from points of single scans, exploring the special structure of the sensor for efficient computations.

III. Online Obstacle Detection

In this section, we describe how we calculate the geometric measures from the 3D-Lidar data. First, we give a short technical overview of the sensors characteristics. Then we explain how we exploit the special structure of the data to efficiently calculate the geometrical measures.

A. Velodyne Intrinsic

The Velodyne HDL sensors are very popular for autonomous robots. Until recently, there were two versions with 32 and 64 individual laser beams available. A new scanner with 16 beams was introduced lately but is not used in this work. The individual lasers are mounted in the sensor with different pitch angles, setting the vertical field of view which ranges from $-20°$ to $20°$ for the HDL 32 and $-24.8°$ to $2°$ for the HDL 64. The data of the sensor is delivered in spherical coordinates, providing the current azimuth and elevation angle, the distance, the remission and the id of the laser. This leads to the typical ring structure of the Velodyne scans (as, e.g., visible in Fig. 3).

B. Geometric considerations

Utilizing this structure, we can directly arrange the 3D-points as a matrix P_{ij}, where the azimuth determines i and the elevation j. For every point P_{ij} we search in each

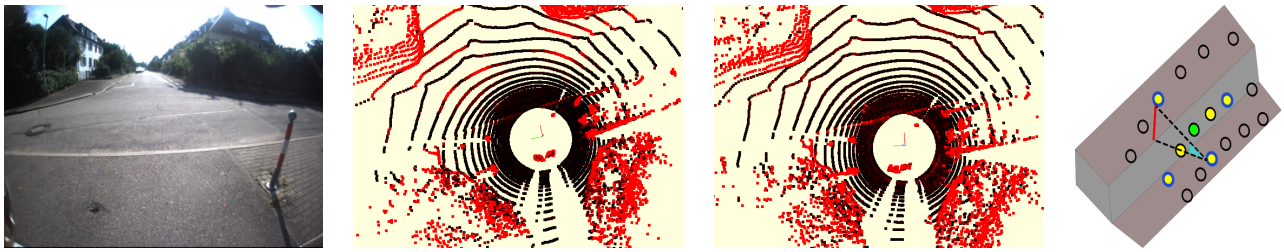

Fig. 3. Example for the basic traversability analysis of the Velodyne data. From left to right: Photo of the scene, visualization of $stepHeight$ (black are low values, red are high values), visualization of $incline$ (same coloring) and an example of a measurement to explain the calculation of the values. In this example, the green point is the point that is currently considered and the yellow points mark the local neighborhood for the computations, they are along the ring and perpendicular to it. The points with blue strokes satisfy our requirements of a minimum distance and are used for the computations. The two points connected with the dashed line are those with the largest difference in the z-coordinate, whereat the $stepHeight$ is indicated by the red line of the triangle. Accordingly we take the angle marked in cyan as $incline$.

direction on the same ring $P_{(i\pm k)j}$ and perpendicular to it $P_{i(j\pm l)}$, with $0 \leq k \leq K$ and $0 \leq l \leq L$, for the first point that has an Euclidean distance larger than x in 3D. This procedure returns at most four neighbors with a maximum of $2(K+L)+1$ comparisons. From these points we calculate the step height ($stepHeight$), as the maximum absolute difference of the z-coordinates, as well as the inclination angle ($incline$) of the line that connects the two points and the xy-plane, which corresponds to the steepness between the points. For a visualization and a simple example of $stepHeight$ and $incline$ see Fig. 3. Due to the finite number of comparisons and the constant time look-up for the points, the overall computation of $stepHeight$ and $incline$ for one point in the scan can be done in constant time. In our implementation we use $K = 5$, $L = 2$ and $x = 0.05m$, which is more than 2σ of the typical sensor noise of the Velodyne.

Please note that this procedure requires knowledge about the robot's current pitch and roll angle, as, e.g., provided by an IMU on the robot (or in case of the Velodyne HDL 32 its internal IMU).

As a pre-processing step, in order to reduce the influence of sensor measurement noise, we average close-by neighboring points along the ring, $P_{(i\pm k)j}$ with $0 \leq k \leq K$, thereby smoothing the surface structure. We compute this average efficiently using a sliding window so that the pre-processing does not break the constant time cost stated above.

In our current C++ implementation we can calculate $stepHeight$ and $incline$ for every point provided by the Velodyne with approximately 50% CPU load on a single core of an Intel i7@3.5GHz PC.

C. Basic Obstacle Detection

In our implementation, we maintain a rolling window, fixed resolution, grid map that is updated whenever a 3D-Lidar measurement arrives. The measures explained above in Sec. III-B are typically sufficient to perform a traversability analysis for environments like offices, fabric halls, or urban roads. For such environments one can easily threshold the geometric measures. Therefore, we choose an appropriate upper bound for the geometric measures, $maxStepHeight$ and $maxIncline$, taking the capabilities of the robot and the environment into account. For such environments we choose $maxStepHeight = 0.05$ and $maxIncline = 20.5\,\text{deg}$. When we look at Fig. 3, we see on the middle left image that $stepHeight$ is a strong indicator for traversability in the dense areas of the scan (e.g., it shows the lowered sidewalk but only as a low value) but there are false positives in the sparser, further away regions of the scan. The $incline$ values, on the middle right image, behave the other way around (note, e.g., that the lowered sidewalk has a strong response). Accordingly, a cell C is occupied by an obstacle if and only if $stepHeight(C) > maxStepHeight$ **and** $incline(C) > maxIncline$. However, in our context, the thresholds largely depend on the type of terrain the robot operates on. Accordingly, our goal is to dynamically adapt them.

IV. Terrain Analysis

When the environment is more scattered in its geometry, like a meadow or forest environments, it is not possible to rely purely on the geometrical measures, since *e.g.*, grass may appear as fake obstacles with step heights even larger than the ground clearance of the robot. On the other hand, a robot that should not drive on grass may consider a mowed meadow as traversable ground and drive over it. To avoid those situations, we use a terrain classification based on features, extracted from points that are registered into a fixed resolution rolling window grid-map and feed it to a random forest classifier to determine the terrain class. We opted for a random forest classifier since it is fast and not affected by the extend of individual dimensions of the feature vector. Yet, in general every multiclass classifier could be employed. Since we are only interested in the local vicinity of the robot, we utilize the raw odometry aided by gyroscope data for the registration of the points. Points from the lidar are mapped with this pose estimates until the robot has covered a distance of d meters. Then we compute one feature for each cell of the map, if more than a predefined value of $minPoints$ are registered in that cell. Using the inverse sensor model from the random-forest classifier we can maintain a probability distribution over the terrain class for each cell.

A. Features

For the feature-based representation of the sensor data in the terrain classification task, we need to find measures that represent the relevant aspects of the environment to get

meaningful results. We chose a mixture of the point geometry as well as additional information, *i.e.*, the remission values of the laser scanner. More detailed, our five-dimensional feature consists of the following measures:

- Maximum remission
- Mean remission
- Standard deviation of the intensities
- Roughness
- Slope

The first three dimensions are computed by standard statistical operations while the latter two are computed using eigen-analysis of the covariance matrix of the points. As roughness of a cell we use the smallest eigenvalue and the slope of a cell is the incident angle of the normal, which is estimated by the eigen-vector that belongs to the smallest eigenvalue, and the xy-plane. The first three dimensions represent the perceptive or appearance-based information that we get from the intensity values of the 3D-Lidar scanner and the last two represent characteristic geometric information of the environment.

B. Classification

In our approach, we apply a random-forest classifier, as introduced by Breiman [2] with the gini-index as split criterion for the nodes of the trees. We hand-labeled data from several scenes that include different environments like a street, a meadow and some dirt- and forest roads to train the classifier. For the labeling we apply a custom tool to determine areas of the corresponding terrain type. All the labeled data was assigned to one of the following classes: *Street, Grass, Dirt* or *Other*. The first three classes build the terrain we consider as potentially traversable, the last class is ideally the union of all kind of possible obstacles. We train the random forest to model the inverse sensor model, *i.e.,* it returns a probability estimate $P(c \mid f)$ for the terrain class c given the feature f.

C. Terrain-Class-Map

The terrain class map is a fixed resolution rolling window grid-map, where we temporarily store the raw points until we compute a feature. Then we calculate a probability distribution over the terrain class based on the probabilities provided from the classifier. We assume that the cells in the terrain map are independent from each other and that the state of a cell is static. Therefore we can maintain the probability distribution over the terrain class given the registered features, $P\left(c \mid f_1, \ldots, f_n\right)$, for each cell independent. Since we want to update the information incrementally as soon as they are available, we apply Bayes rule twice with the assumptions stated above to derive

$$P(c \mid f_1, \ldots, f_n) = \eta \frac{P(c \mid f_n) P(c \mid f_1, \ldots, f_{n-1})}{P(c)}. \quad (1)$$

The term η is a normalizing constant that summarizes terms that do not depend on the class c. In our implementation we use a uniform prior for the class and the inverse sensor model from the random-forest classifier to compute the

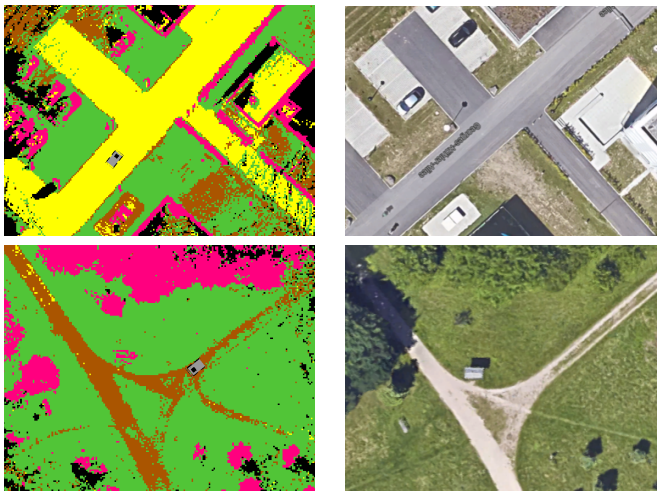

Fig. 4. Left: Results of our terrain classification. Pink is other, yellow street, green is grass and brown is dirt. Right: Aerial image of the scene. Our terrain classification accurately classifies different terrain types in urban and outer urban environments.

class probabilities of individual cells. To save computational resources, we apply the assumption of the static terrain class, setting a cell as classified if it has been seen more than 10 times and if the probability of the most likely terrain class exceeds 0.9. Classified cells are not considered for updates of the terrain class but keep their state. In order to counteract cases where the state of the cells may change from time to time, *e.g.*, a parking car that may start driving, we reset classified cells every t seconds. The main idea of our combined approach is to choose proper values for the thresholds $maxStepHeight$ and $maxIncline$, see Sec. III.

V. Terrain-Adaptive Obstacle Detection

Our obstacle detection module combines the methods explained in Sec. III and Sec. IV. In our implementation we use two asynchronous threads, one that computes the geometric measures and one that does the terrain analysis. To circumvent race conditions we maintain two independent maps, one for the terrain class and one that fuses the information of our combined approach, which is then given to a planner. Therefore, the system has critical information available as soon as it arrives but can still gain from the slower updates of the terrain class model. This summarizes the description of our terrain-adaptive approach for collision avoidance, which utilizing the information over the terrain type for the correct interpretation of geometric measures in the range scans. The individual thresholds for the different terrain types are specified in Tab. I. The $maxIncline$ values are the same for all terrain classes, since this measure corresponds to the capabilities of the robot and is therefore independent of the terrain type. The $maxStepHeight$ for *Street* is obviously the same value as in Sec. III-C, while the values for *Grass* and *Dirt* are much higher, corresponding to our expectation of a reasonable risk for those terrain types.

VI. Experiments

We designed our experiments to highlight the different aspects of the proposed approach. We provide qualitative and

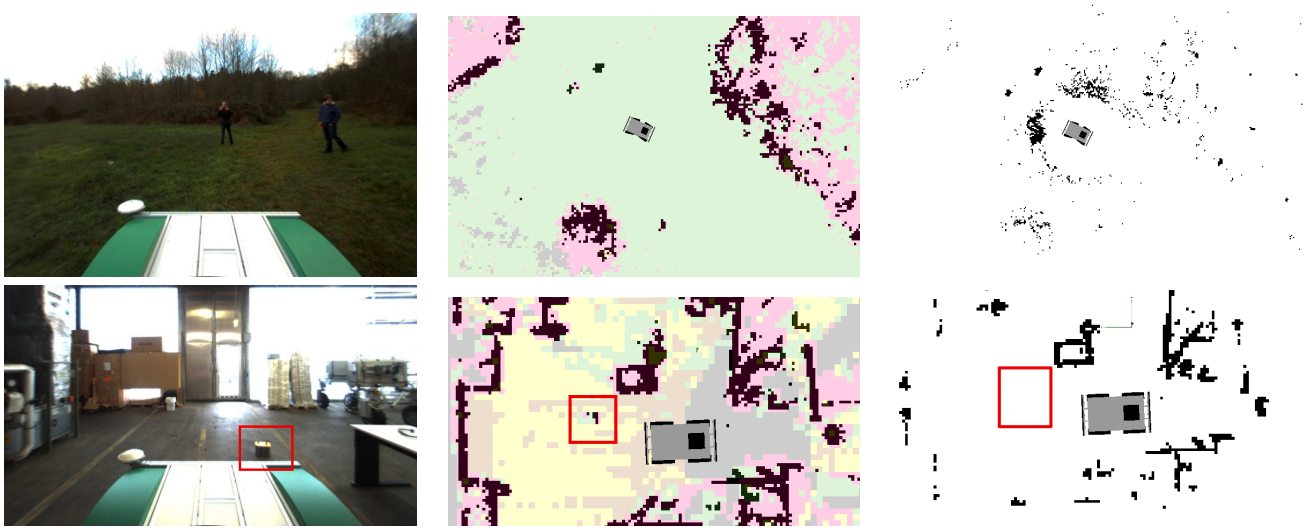

Fig. 5. Left: An on-board view of the scene. Middle: The resulting obstacle map blended with the terrain classification of our approach. Right: Resulting obstacle map of the basic approach. In the top row example our approach guarantees full mobility while the basic approach is trapped by many fake obstacles induced from the grass. In the lower row example we see what happens if we set the threshold of the basic approach to the same as for grass in our combined approach. Since our approach can use adaptive thresholds and correctly find the obstacle, the basic approach misses the massive battery in front of the robot, as marked with the red rectangle.

TABLE I

TERRAIN CLASS DEPENDENT THRESHOLDS

Terrain Class	$maxStepHeight$ [m]	$maxIncline$ [°]
Street	0.05	20.5
Grass	0.50	20.5
Dirt	0.25	20.5
Other	0.05	20.5

quantitative results about the performance of our approach. In addition, we present a real world experiment of an autonomous four km run through challenging terrain. Furthermore we provide a detailed analysis of the computational requirements, illustrating the economical use of computational resources by our approach. For all experiments we used the same parameters and the same random forest, which consists of six trees. The trees were grown without depth limit, each split considered two randomly chosen variables, while the minimal number of features to split a node was 100 and the split criterion was based on the gini-index. The terrain analysis grid had a resolution of $0.2m$, a size of 300×300 and d was set to $0.5m$. The map that was given to the planner, which is updated by our combined approach, had a resolution of $0.1m$ and a size of 600×600. The maximum speed of our robot was set to $1.2m/s$. In order to circumvent inconsistent lidar data we applied the calibration approach by Steder et al. [15] for our Velodyne HDL64.

A. Possible – Impossible

If no additional semantic information is available we can only make a decision based on the values of $maxStepHeight$ and $maxIncline$. Therefore we consider the street thresholds in Tab. I, where we easily run into problems if we are actually not on a street. In Fig. 5, we show two examples of situations where the naive approach, without terrain class considerations, is either blocked by fake obstacles like grass or misses the detection of real obstacles if a more reckless threshold is utilized. In the first row of Fig. 5 we use the street thresholds from Tab. I, the rightmost image shows the result of the obstacle detection, where the robot is trapped by grass showing up as fake obstacles all around the robot. A simple but dangerous fix for the naive approach would be to raise the thresholds, e.g., to the grass thresholds of Tab. I, but then, as seen in the second row of Fig. 5, we may encounter hazardous situations, since on the rightmost image the battery is not recognized as an obstacle anymore. In contrast, with our terrain-adaptive classification, we can classify both scenes correctly, as can be seen in the images in the middle column of Fig. 5.

B. Terrain Classification Accuracy

To evaluate the performance of the terrain classification we evaluated scenes like shown in Fig. 4. We used partially labeled scenes from urban environments taken from our campus as well as from data in the forest. All in all $\sim$100K labeled cells were evaluated, since only cells for which we could doubtless determine the class were labeled. The resulting confusion matrix is shown in Tab. II, which presents the results of our cell-wise integrated classification, see Eq. (1), as well as the most likely class per feature, which is given in brackets. Our classifier can reliably distinguish the four classes from each other, but there are some confusions with street and dirt as well as with dirt and grass. One reasons for this is probably, as we observed, that often the streets are covered with a little bit of dirt from construction site machines or agricultural machines, which has a high influence on the remissions of the lidar. Also the remissions on dirt seem very similar to those of grass, depending on its composition. It can also be seen in Fig. 4 on the top left image, where grass and dirt are mixed in some regions, but the aerial image on the right indicates that it may not be totally clear what the correct labeling should be. Neverthe-

TABLE II

TERRAIN CLASSIFICATION CONFUSION MATRIX

Class	Street	Grass	Dirt	Other
Street	0.79 (0.64)	0.06 (0.15)	0.14 (0.19)	0.01 (0.004)
Grass	0.02(0.005)	0.91 (0.86)	0.05 (0.13)	0.02 (0.002)
Dirt	0.02 (0.10)	0.18 (0.18)	0.78 (0.72)	0.03 (0.002)
Other	0.04 (0.06)	0.10 (0.10)	0.10 (0.03)	0.76 (0.82)

less, classifying grass reliable is important since most fake obstacles appear in grass and the false classifications of grass did no harm as far as we could observe, since they are rather isolated.

C. Real World with Computational Analysis

In this experiment we evaluate the capabilities and the computational economy of our approach in a real world scenario. The trajectory was four km from our campus to a nearby forest, see the GPS-track in Fig. 2. Furthermore, during this experiment we measured the time the different tasks took. Our algorithm was evaluated using a Intel(R) Core(TM) i7-2700K @ 3.50GHz with four physical cores. We used a global planner that gives local goalpoints to a local planner that planned a path based on the obstacle map of our approach. The robot was driving autonomous during the experiment. In Fig. 6 we visualized the CPU percentages of the components of our approach in a pie-chart. The whole run took about $4400s$ which in return results in an average speed of $0.9m/s$, while the maximum speed of the robot was limited to $1.2m/s$. The overall processor clock time was $7760s$, where also path planning, collision checking, motion control, a GUI and the message management was included, which leads to an overall average processor usage of about 180%. For the terrain classification, including the feature computation, the system used on average about 33% of its processing power, while for the high frequent geometric measures, including the obstacle grid map update, it needed about 68%. This underlines the economic of our approach, since on our quadcore CPU more than half of the processing power was available when our navigation software was running. Moreover, we showed that our approach can be applied in real-world and online on an autonomous robot for safe navigation in forested and outer urban environments.

VII. CONCLUSION

In this paper, we presented an approach for online obstacle detection adapted according to semantic terrain information. Our method combines low frequency terrain analysis and high frequency basic obstacle detection and provides a reliable obstacle classification with a moderate usage of computational power. In our experiments we identified situations where a naive approach, without the terrain adaptation, would yield suboptimal results for safety or mobility reasons. Moreover, we demonstrated that our approach is economical with computational power, fast enough to run onboard and capable to control an autonomous robot along a challenging four km track including different terrain types like grass, dirt roads and regular streets.

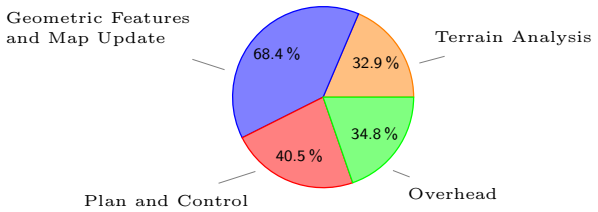

Fig. 6. CPU usage of our approach during the real world experiment.

In future work, we would like to extend our method to a probabilistic approach that can also take the uncertainty about the terrain classification estimates into account.

REFERENCES

[1] P. Babahajiani, L. Fan, and M. Gabbouj. Object recognition in 3d point cloud of urban street scene. In *Computer Vision-ACCV 2014 Workshops*, 2014.

[2] L. Breiman. Random forests. *Machine learning*, 2001.

[3] T. Chang, T. Hong, S. Legowik, and M. Abrams. Concealment and obstacle detection for autonomous driving. In *Proc. of the Intl. Association of Science and Technology for Development-Robotics and Application*, 1999.

[4] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 2014. doi: 10.1002/rob.21534.

[5] I. Kweon and T. Kanade. High resolution terrain map from multiple sensor data. In *IROS*, 1990.

[6] S. Laible, Y. N. Khan, K. Bohlmann, and A. Zell. 3d lidar-and camera-based terrain classification under different lighting conditions. In *Autonomous Mobile Systems 2012*. 2012.

[7] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 2006.

[8] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous robots*, 2005.

[9] J. Maye, R. Kaestner, and R. Siegwart. Curb detection for a pedestrian robot in urban environments. In *IEEE Int. Conf. on Rob. & Aut.*, 2012.

[10] M. W. McDaniel, T. Nishihata, C. A. Brooks, P. Salesses, and K. Iagnemma. Terrain classification and identification of tree stems using ground-based lidar. *Journal of Field Robotics*, 2012.

[11] K. E. Olin and D. Y. Tseng. Autonomous cross-country navigation: an integrated perception and planning system. *IEEE expert*, 1991.

[12] P. Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 2013.

[13] P. Pfaff, R. Triebel, and W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *Int. Jour. of Rob. Res.*, 2007.

[14] À. Santamaria-Navarro, E. H. Teniente, M. Morta, and J. Andrade-Cetto. Terrain classification in complex three-dimensional outdoor environments. *Journal of Field Robotics*, 2015.

[15] B. Steder, M. Ruhnke, R. Kümmerle, and W. Burgard. Maximum likelihood remission calibration for groups of heterogeneous laser scanners. In *IEEE Int. Conf. on Rob. & Aut.*, 2015.

[16] S. Thrun, M. Montemerlo, and A. Aron. Probabilistic terrain analysis for high-speed desert driving. In *Robotics: Science and Systems*, 2006.

[17] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2006.

[18] C. Wellington and A. Stentz. Online adaptive rough-terrain navigation vegetation. In *IEEE Int. Conf. on Rob. & Aut.*, 2004.

[19] C. Wellington and A. Stentz. Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetation. In *Field and Service Robotics*, 2006.

[20] K. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *IEEE/RSJ Int. Conf. on Intel. Rob. and Sys. (IROS)*, 2009.