Building Local Terrain Maps Using Spatio–Temporal Classification for Semantic Robot Localization

Stefan Laible[1] and Andreas Zell[1]

Abstract— **The correct classification of the surrounding terrain is an important ability of a mobile robot that drives in outdoor environments. Our robot uses a 3D LIDAR and a camera to classify terrain as either asphalt, cobblestones, grass, or gravel. We build on previous work where we modeled the terrain as a Conditional random field to account for spatial dependencies, which improved results substantially. We now show how to speed up the spatial classification by defining a new energy term for neighborhood relations. Moreover, we now also consider temporal dependencies as the robot moves. This not only further improves the results, but makes it possible to build local terrain maps of the environment. We describe how to efficiently integrate the classification results of each time step into the map in a probabilistic manner. By also detecting obstacles with the LIDAR, the robot can build combined terrain and elevation maps. We show that these maps can be used for semantic robot localization.**

I. INTRODUCTION

Autonomous navigation in outdoor environments calls for different requirements than navigation in indoor scenarios such as office buildings and factories. Characteristic of man–made environments is the geometric structure, which can be used for a very precise localization of the robot. Outdoors there is often a lack of structure. So to enable a safe and efficient navigation in such environments a comprehensive semantic perception of the environment is essential. Of particular interest is the surrounding terrain. The knowledge about it is crucial for a safe and efficient path planning on the one hand, but it is also valuable for robot localization, as an addition or alternative when GPS is too unreliable or not available at all, e.g. near buildings, under trees, or in general when there is no clear line of sight to the satellites. The accuracy requirements of such a localization depend on whether the robot is to follow, for example, a field boundary or a road, or whether it should travel long distances over open field.

In previous work [LKZ13] we presented a method for classifying the terrain in front of the robot using a 3D LIDAR and a camera. Therefore, the terrain is divided into a grid, and at first, every grid cell is classified individually using the sensor measurements. By modeling the terrain grid as a Conditional random field (CRF) we additionally consider spatial dependencies between the cells, which improves results substantially. In this work, we also take into account temporal dependencies in order to get a spatio–temporal terrain classification. This improves the quality of the classification further, and enables the robot to map its traveled path; we call this building local terrain maps. We show that these maps can then be used to localize the robot by applying Monte Carlo localization, where we are not interested in a precise but a semantic localization.

[1]S. Laible and A. Zell are with the Chair of Cognitive Systems, Computer Science Departement, University of Tübingen, Sand 1, 72076 Tübingen, Germany {stefan.laible, andreas.zell}@uni-tuebingen.de

II. RELATED WORK

There are several related works about terrain classification that consider spatial or temporal dependencies. In [WSFB05] a 2D LIDAR is used for terrain mapping. The terrain is classified in navigable and non–navigable regions using Hidden Markov models. Small classification errors in the map are removed with a Markov random field. In [KZ10] vibration data acquired by an inertial measurement unit is used to segment different types of terrain with an unsupervised learning approach. The clustering is based on a Markov random field to consider temporal dependencies between consecutive measurements. [HAW$^+$13] also use Markov random fields, for considering spatial dependencies between cells of a terrain grid. They use a high–resolution 3D LIDAR and several color cameras to classify the terrain into the classes road, rough and obstacle. We showed, however, in previous work [LKZ13] that CRFs are better suited for our task than Markov random fields. A 2D LIDAR is used in [WKK$^+$13] to distinguish between vegetation and asphalt by looking at the intensity values of laser measurement points. They show that it is also possible to distinguish between a dark street and light tiles. Results from consecutive measurements are combined probabilistically. In our work, we distinguish not only between navigable and non–navigable terrain, or between vegetation and non–vegetation, but use multiple terrain classes.

Semantic perception and classification of the environment for mapping and localization is used, for example, in [WB10] in the domain of agricultural robotics. They present an alternative to GPS–based navigation for a robot driving through a maize field. The robot can localize itself by detecting states like being in the middle of a row, at a row gap, or at the end of the row. In [NH08] semantic maps are generated by labeling coarse scene features like walls and floors, and more complex objects detected by a classifier. Such maps that contain annotations of known objects in addition to spatial information can be very beneficial for localization tasks.

III. SPATIAL CLASSIFICATION

At the core of our approach is the robot's ability to classify its surrounding terrain. In each time step the terrain in front of the robot is divided into a grid and each cell of this grid is classified as either asphalt, cobblestones, grass, or gravel. For this task our robot is equipped with an AVT Marlin F-046 C Color Camera and a Nippon Signal FX6 3D LIDAR. The camera has a VGA resolution and takes color images, however, we only use gray–scale images. The 3D LIDAR has a very low resolution with only 59×29 data points and provides for each measurement point in addition to the distance an intensity value, which indicates the proportion of the emitted light that arrives back at the sensor. This sensor is largely illumination–independent and works with ambient light of up to 100,000 Lux. The camera is only used for terrain classification, while the LIDAR is also used for detecting the ground plane and obstacles.

Once the transformation between the LIDAR and the camera coordinate system is computed using a calibration method, for each cell of the above mentioned terrain grid the corresponding features for the laser measurement points and image pixels can be extracted. We use roughness and intensity histograms for the laser scans as described in [LKBZ12]. Especially the distribution of the intensity values of the scan points provides characteristic features, since it results from the reflection properties of the different terrain types. To extract features from the image patches we use Local ternary patterns (LTP) [TT10]. LTPs are basically histograms based on intensity differences in neighboring pixels. Since only differences are considered, a certain independence from changes in illumination is achieved. Using labeled training data, a terrain model can then be learned for each of the two sensors. For this purpose we use Random forests [Bre01]. Random forests are an ensemble learning method that use multiple decision trees for classification. Each tree gives a vote for one class and the majority of votes determines the most probable class. In this way we get a probability for each class as the proportion of trees that voted for this class, which is very important for our approach, as we shall see. For example, in grid cells where data from both sensors are available, the two classification results can be fused by simply combining the corresponding probability values through a weighted sum.

To increase the classification rate we also considered spatial dependencies between the individual grid cells using CRFs. We will briefly describe this in Sec. III-A. For a more detailed description we refer the reader to [LKZ13]. In Sec. III-B we show our modification to this approach, which yields similar results with a much lower computational cost.

A. Classification Considering Spatial Dependencies

Terrain appears in contiguous areas. This is an important information that is ignored when classifying each grid cell individually. There, we get the most likely label $y^* = \arg\max_{\mathbf{y}} p(y|x)$ for a cell directly from the Random forest given the extracted features x. It may then happen that the terrain labels assigned to the grid cells change often even in small areas, which rarely occurs in those outdoor environments in which our robot normally drives. There, rather large coherent areas of the same terrain type occur, with clear boundaries between areas of different terrain. We now want to keep the feature–dependent classification, but also consider the spatial context in which a grid cell appears. To take these spatial dependencies into account, we model the terrain grid as a CRF. CRFs are in essence a family of conditional probability distributions $p(\mathbf{y}|\mathbf{x})$ represented by means of an undirected graph. In order to limit the complexity of that model we define a neighborhood $\mathcal{N}$ that determines for each grid cell which other cells do have a direct influence on that cell. This neighborhood can consist, for example, of the four direct neighbors as in Fig. 1, or as we define it, of the 8–neighborhood, that is, the eight surrounding cells.

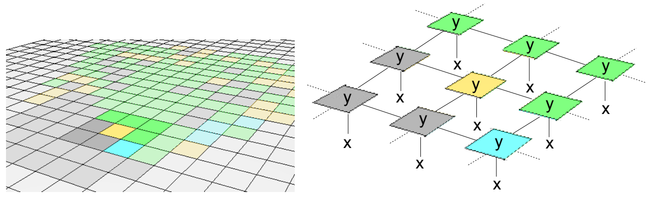


Fig. 1. The terrain label y of a grid cell depends on the measured features x, but also on the labels of its neighboring grid cells.

We now have a two–stage classification process. In the first stage, for each cell, the features x_i are extracted and the probabilities $p(y_i|x_i)$ are obtained by the Random forests. In the second stage, the optimal label configuration $\mathbf{y}^*$, which maximizes the conditional probability $p(\mathbf{y}|\mathbf{x})$, is to be found. Here, also spatial dependencies between grid cells are taken into account by modelling the terrain grid as a CRF. Since it is not feasible to consider all the exponentially many possible configurations of terrain labels in finding the optimal solution $\mathbf{y}^*$ for a grid, we use an approximate inference method, namely a Gibbs sampler in a Simulated–annealing scheme [GG84]. In this scheme, the label configuration is changed iteratively until a convergence criterion is reached. These changes can be very large at the beginning, but with the decreasing of a temperature factor will be less. So it is possible to find the global maximum in the presence of many local maxima. In general, this maximum, which is the maximum probability $p(\mathbf{y}|\mathbf{x})$ in our case, is found by bringing a system in a state of minimum energy. We must therefore formulate our problem as an energy–minimization problem [NL11]:

$$y^* = \arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) \tag{1}$$

$$= \arg\min_{\mathbf{y}} E(\mathbf{y}, \mathbf{x}) \tag{2}$$

The energy E consists of two components corresponding to the two components of the CRF: a feature–dependent component E_f, and a component $E_{\mathcal{N}}$ that describes neighborhood relations between cells.

$$E(\mathbf{y}, \mathbf{x}) = \sum_i E_{f_i}(y_i, x_i) + E_{\mathcal{N}_i}(y_i, x_i, \{y_j, x_j : j \in \mathcal{N}_i\}) \tag{3}$$

The feature energy E_{f_i} only depends on the label y_i and the features x_i of the cell, whereas the neighborhood energy additionally depends on the labels y_j and features x_j of the neighboring cells. In [LKZ13] we used the following energy terms:

$$E_{f_i} = -\lambda \log\left(p(y_i|x_i)\right) \tag{4}$$

$$E_{\mathcal{N}_i} = \sum_{j\in\mathcal{N}_i} \mathbf{1}_{\{y_i\neq y_j\}} \exp\left(-\beta(x_i-x_j)^2\right) \tag{5}$$

The influence of the two components can be adjusted by the weighting factors λ and β. E_{f_i} is simply the energy equivalent of the conditional probability computed in the first stage. The key idea behind the above formulation of $E_{\mathcal{N}_i}$ is that it is very likely that two adjacent cells belong to the same type of terrain, but if they do not, i.e. $y_i \neq y_j$, their appearance must also differ greatly, with this difference expressed by the squared difference $(x_i - x_j)^2$ of the feature vectors. The time required for calculating this difference depends on the length of the feature vectors. Since the energy terms have to be computed many times for different label configurations during the simulated annealing, this computational cost is a crucial factor. On the other hand, it is also questionable whether this vector difference reflects the differences in appearance adequately. We present an alternative definition of the neighborhood–dependent energy term $E_{\mathcal{N}_i}$ in the following section.

B. Neighborhood Energy

We set two requirements for the new energy term $\hat{E}_{\mathcal{N}_i}$. First, the differences in appearance are not to be described by the difference of two feature vectors, but also by probabilities. And second, the distinction should not simply be made between neighbors of the same terrain type and those of different terrain types. Instead, the actual types should be considered, since different types are adjacent to each other with a different probability.

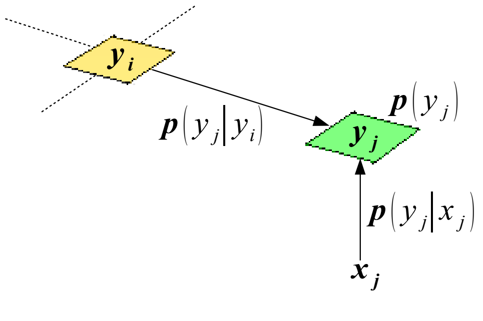


Fig. 2. Both the feature vector x_j and the label y_i of the adjacent grid cell have an influence on the probability of the label y_j.

We meet these requirements by defining the neighborhood energy $\hat{E}_{\mathcal{N}_i}$ as the energy equivalent of the average probability that a neighboring cell is of its assigned type y_j, given the observed features x_j of that cell and the label y_i of the centered cell (see Fig. 2), with a weighting factor β:

$$\hat{E}_{\mathcal{N}_i} = -\beta|\mathcal{N}_i|^{-1} \sum_{j\in\mathcal{N}_i} \log(p(y_j|x_j,y_i)) \tag{6}$$

Using Bayes' theorem and the observation that x_j and y_i are conditionally independent, it follows that:

$$\begin{aligned} p(y_j|x_j,y_i) &= \frac{p(x_j|y_j)p(y_j|y_i)}{p(x_j)} \\ &= \frac{p(y_j|x_j)p(y_j|y_i)}{p(y_j)} \end{aligned} \tag{7}$$

The probabilities $p(y_j|y_i)$ and $p(y_j)$ can be set as required or be learned from training data, by simply counting occurrences. On our campus, for example, asphalt and grass are often adjacent to each other, while asphalt and gravel are not. As Fig. 3 illustrates, having a bad inital classification based on Random forests, using the energy term $E_{\mathcal{N}_i}$ of Eq. 5 can make things even worse, whereas the energy term $\hat{E}_{\mathcal{N}_i}$ of Eq. 6 improves the result significantly.

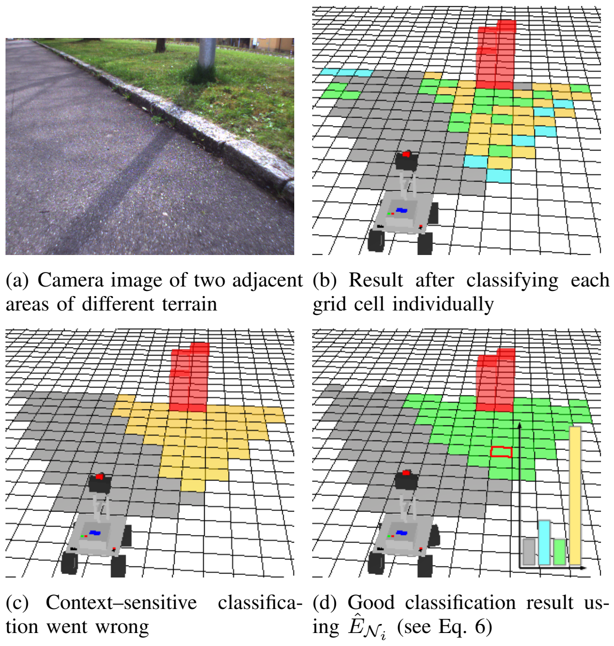

(a) Camera image of two adjacent areas of different terrain
(b) Result after classifying each grid cell individually
(c) Context–sensitive classification went wrong
(d) Good classification result using $\hat{E}_{\mathcal{N}_i}$ (see Eq. 6)

Fig. 3. The classification of the terrain in the camera image (a) can give poor results when the grid cells are classified only individually (b). Since most cells were incorrectly classified as gravel, the results get even worse when considering spatial context (c). However, when integrating the observation that asphalt and gravel areas are rarely adjacent in the specific environment in which the robot operates, the classification result gets very satisfactory (d). (Gray: asphalt, blue: cobblestones, green: grass, yellow: gravel, red: obstacles)

The histogram in Fig. 3(d) shows the probability distribution of the four terrain classes for the red–bordered cell. This is an example that illustrates that a cell that would be wrongly labeled as gravel when only considering the measured features, is now correctly assigned to the class grass by also considering the spatial context of that cell, and the characteristics of the specific environment.

The energy term defined in Eq. 6 thus has some advantages. It is independent of the actual feature descriptor used, and is instead probabilistically motivated. As we will see in Sec. VI, it is also much faster to compute without a loss of quality.

IV. TEMPORAL CLASSIFICATION

Now that we have classified the terrain in front of the robot at each time step, taking into account spatial dependencies, we want to update a local terrain map with these classification results as the robot moves. We call these maps *local*, as we use only odometry to estimate the robot's position and we do not use techniques like scan matching and loop closure detection to reduce or eliminate the inevitable drift in odometry. So our maps are only locally consistent, but this is quite sufficient for many applications.

A. Projection of Grid Cells Onto the Terrain Map

Each cell of the terrain map contains, like the grid cells, the probabilities of the terrain classes. Thus, updating the terrain map with the current classification result means updating the terrain probabilities of the relevant map cells. Therefore, the grid G is projected onto the map M (see Fig. 4). Note the differentiation between the terrain grid G in front of the robot and the terrain map M relative to which the robot moves.

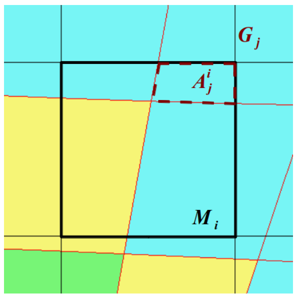


Fig. 4. Projection of grid G (red lines) onto map M (black lines). A_j^i outlines the sectional area between grid cell G_j and map cell M_i.

To compute the probabilities for map cell M_i that correspond to the current measurement, the probabilities of all grid cells G_j that overlap with M_i need to be considered. The influence of each grid cell G_j is proportional to the sectional area A_j^i of M_i and G_j. So for all terrain classes y the probability $p(Y_i^M = y)$ that map cell i has label y is calculated as:

$$p(Y_i^M = y) = \eta \sum_j \frac{A_j^i}{A_i} p(Y_j^G = y) \tag{8}$$

$$\eta = \left(\sum_y \sum_j \frac{A_j^i}{A_i} p(Y_j^G = y) \right)^{-1} \tag{9}$$

Here A_i is the area of the map cell M_i, which is usually constant throughout the map. To compute the sectional area A_j^i efficiently we use the Sutherland–Hodgman algorithm [SH74], originally invented for fast polygon clipping in the field of Computer graphics, which computes the intersection points $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$. The actual surface area A_j^i can then be calculated as follows [Bey84]:

$$A_j^i = \frac{1}{2} \sum_{k=1}^{n} (a_k b_{k+1} - a_{k+1} b_k) \tag{10}$$

B. Temporal Updating of Terrain Probabilities

We just showed how to associate the probabilities of the grid cells with the cells of the map. The values in a map cell represent the terrain probabilities given all previous measurements $\mathbf{x}_{1:t-1}$. To update the map with the current measurement $\mathbf{x}_t$ as the robot moves, we use the same formula as in [WKK+13] where they map two terrain classes, vegetation and non–vegetation, and which was originally used for Occupancy grid maps [Mor89]:

$$p(y_i|\mathbf{x}_{1:t}) \propto \left(1 + \frac{1 - p(y_i|\mathbf{x}_t)}{p(y_i|\mathbf{x}_t)} \frac{1 - p(y_i|\mathbf{x}_{1:t-1})}{p(y_i|\mathbf{x}_{1:t-1})} \frac{p(y_i)}{1 - p(y_i)}\right)^{-1} \tag{11}$$

Since we not only have two classes, such as occupied and unoccupied, or vegetation and non–vegetation, we have to normalize the probabilities by dividing by $\sum_i p(y_i|\mathbf{x}_{1:t})$.

C. Recomputation of Terrain Probabilities

The temporal updating of the map is therefore solely based on probabilities, which means in particular that the actual label configuration $\mathbf{y}$ is not considered. On the other hand, only labels and no probabilities are changed in the second stage of the spatial classification described in Sec. III. In order to take into account the result of the inference method, namely the optimal configuration $\mathbf{y}^*$ (see Eq. 1), we need to incorporate this configuration in the terrain probabilities of the grid cells. The idea now is similar to that for the definition of the neighborhood energy in Eq. 6, namely to involve the neighborhood of a cell for computing the terrain probability (see Fig. 5).

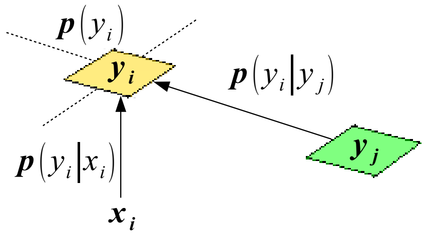


Fig. 5. The updated terrain probabilities $p(y_i|x_i, \{y_j : j \in \mathcal{N}_i\})$ stored in the map not only depend on the features x_i, but also on the labels $\{y_j : j \in \mathcal{N}_i\}$ of the neighboring cells.

Using analogous considerations as in the derivation of Eq. 7, and considering the whole neighborhood $\mathcal{N}_i$, it follows that:

$$p(y_i|x_i, \{y_j : j \in \mathcal{N}_i\}) = p(y_i|x_i) \prod_{j \in \mathcal{N}_i} \frac{p(y_i|y_j)}{p(y_i)} \tag{12}$$

The recomputation of the terrain probabilities makes in fact a big difference in the final classification results as can be seen in Fig. 6(d). Another possibility would be to model the whole map as a CRF, but this would no longer be computable in real time.

D. Elevation Mapping

In addition to the probabilities of the terrain classes each cell of our map also stores a height value h and the corresponding variance σ_h^2. We use a Kalman–filter based approach for elevation mapping as described in [KD07] where the height of a cell is estimated given all previous height observations.

After detecting the ground–plane in the point cloud of the 3D LIDAR and transforming the cloud so that the ground plane equals the xy plane with the z-axis pointing upwards, the current height observation of a cell is simply the maximal z-value z_{max} of the scan points belonging to that cell. The estimated variance σ_z^2 of this observed height is computed by the plane–detection module and is updated at each time step. With z_{max} and σ_z^2 we can now update h and σ_h^2 by applying a Kalman filter. Since the laser scanner is tilted it can measure very different heights for the same cell when driving towards vertical obstacles like walls. To account for this, no Kalman update is performed when $|h - z_{max}| > \sigma_h$, but h is set to $\max\{h, z_{max}\}$ instead. With this exception rule the estimated height h can grow by leaps and bounds, but not shrink in the same manner. The latter is, however, desirable in case of dynamic obstacles or incorrectly measured observations. We change the rule in [KD07] thus slightly and update yet again if the observed height z is below a given threshold, that is, almost zero.

V. LOCALIZATION

With the spatio–temporal classification method described in Sec. III and IV the robot is now able to build locally consistent maps of the environment, containing terrain and obstacles. Detecting and classifying obstacles and terrain by itself is an important ability of a mobile outdoor robot for planning safe and efficient driving maneuvers. Moreover, as we will show now, such maps can also be used for robot localization. Our goal here is not a precise localization of a robot working in a small area, but a semantic localization when traversing long distances. We use a particle filter for localization, since this method has proven to be very effective for our purposes.

A. Particle Filter

The particle filter, also known as Monte Carlo localizer [DFBT99], estimates the pose of the robot, that is, both the position and the orientation. Here, each particle is a candidate for the actual pose. In our case the motion prediction for the particles is based solely on the odometry. Measurements are used to set importance weights for the particles, with these weights being proportional to the resampling probability. These two steps, movement and measurement, let the particles converge to the true pose of the robot. In our case the measurements are, on the one hand, the grid cells with assigned terrain labels, and on the other side, the height values of the obstacles. We will now show how to compute importance weights $w_t, w_h \in [0, 1]$ from terrain and height measurements, respectively. Since often several thousand particles are used, the calculation of these weights must be very fast.

Let I_t be the set of indices of those grid cells to which a terrain label was assigned, with that label being denoted as y_j for $j \in I_t$. Let further be $i(j)$ the corresponding map cell index obtained by projecting the grid cell center onto the map. Then, the measurement weight $w_t \in [0, 1]$ for the terrain labels is defined as:

$$w_t = |I_t|^{-1} \sum_{j \in I_t} p_{i(j)}(y_j) \tag{13}$$

We call a cell occupied if its height value h exceeds a threshold t. As a measure of whether the observation is consistent with the map, we define for every grid cell with index $j \in I_h$, where I_h is the set of indices of those grid cells that are occupied:

$$H_j = \begin{cases} 1 & \text{if } h_{i(j)} > t \\ 0 & \text{else} \end{cases} \tag{14}$$

We then set the weight $w_h \in [0, 1]$ as follows:

$$w_h = |I_h|^{-1} \sum_{j \in I_h} H_j \tag{15}$$

The final importance weight $w = \kappa w_t + (1 - \kappa) w_h$ of a particle is a combination of w_t and w_h, with $\kappa \in [0, 1]$. The pose of the robot is then estimated using the weighted mean of all particles within a certain radius of the particle with the maximum weight.

VI. EXPERIMENTS AND RESULTS

In our experiments the terrain types asphalt, cobblestones, grass, and gravel were considered. The LIDAR is mounted on the robot at a height of approximately 50 cm and at an angle of about 25° to the horizontal, with the camera mounted on top of it. The grid and the map resolution are both set to 20 cm. We only classify grid cells with at least ten laser measurement points in it, or with at least 200 pixels and a minimum side length of ten pixels for the corresponding image patches. These limitations lead to a lower detection range, the LIDAR can detect terrain in a range of about 1.5 m in front of the robot, whereas the camera has a detection range of about 3 m; but the classification provides better results, which pays off in the end in temporal classification. In order to compare the terrain classification results a ground–truth data set was created. This data set consists of 200 terrain grids, which were labeled by hand. The classification rates were determined using a 10–fold cross–validation and are shown in Tab. I.

Classifying each grid cell individually using Random forests only yields a classification rate of 82.0%. In our experiments we use Random forests with 100 trees each. Considering spatial dependencies in addition brings a significant better rate of 96.8%. For our dataset we get the same results, on average, regardless of whether we are using the old or the new energy term. However, in some environments as we have seen in Sec. III-B, using the new energy term $\hat{E}_{\mathcal{N}_i}$ along with the additional information about the terrain

TABLE I

RESULTS OF TERRAIN CLASSIFICATION

Classification method	Classification rate in %
Cell–wise classification	82.0
CRF with $E_{\mathcal{N}_i}$ (Eq. 5)	96.8
CRF with $\hat{E}_{\mathcal{N}_i}$ (Eq. 6)	96.8
Temp. classification w/o recomp.	≈92.2
Temp. classification w/ recomp. (Eq. 12)	≈98.4

Classification rates using 10–fold cross–validation for the cell–wise classification, the spatial classification with the old and new energy terms $E_{\mathcal{N}_i}$ and $\hat{E}_{\mathcal{N}_i}$, respectively, and the temporal classification without and with the recomputation of the terrain probabilities.

can improve results significantly. Moreover, the computation is also much faster as shown in Tab. II.

For the evaluation of the temporal classification five terrain maps like the one shown in Fig. 6(c) were labeled by hand for having ground truth. Since the ground–truth data generated for such large maps is less precise than for the individual terrain grids, the classification rates are only approximate values, but with a tendency towards underestimation. Only using temporal classification, that is, without considering spatial dependencies, a classification rate of about 92.2% is achieved. This is a better result then classifying cells individually, but it is worse than the spatial classification results. And finally, using spatio–temporal classification with recomputed probabilities as described in Sec. IV-C yields an almost perfect classification with over 98.4%. An example of what a difference the recomputation of terrain probabilities makes can be seen in Fig. 6(d). Here, the information of the left map cannot be used by the robot to drive along the grass border.

TABLE II

AVERAGE RUNTIMES OF THE MAIN PARTS OF THE ALGORITHM

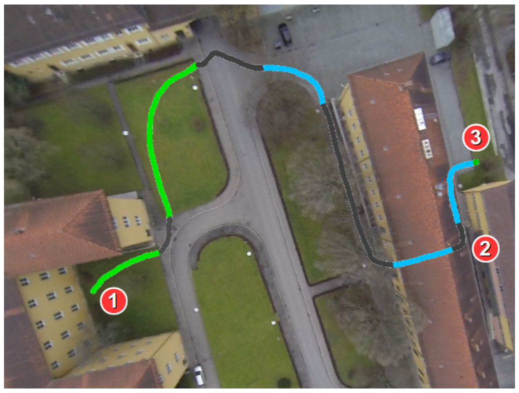
	Average time [ms]	Std. dev. [ms]
LIDAR–feature extraction	0.4	0.1
Image–feature extraction	9.5	0.4
Cell–wise classification	11.2	1.5
Simulated annealing with $E_{\mathcal{N}_i}$ (Eq. 5)	8.9	6.7
Simulated annealing with $\hat{E}_{\mathcal{N}_i}$ (Eq. 6)	1.7	1.0
Spatial classification	**22.8**	3.0
Temporal classification	1.0	0.2
Elevation mapping	0.1	0.4
Spatio–temporal classification	**23.9**	3.6
Particle filter w/ 2000 particles	29.9	0.4

Tab. II shows the average runtimes of the main parts of the classification method (using a CPU with 3.20 GHz and, in contrast to [LKZ13], compiler optimization). The spatial classification consists of the LIDAR– and image–feature extraction, the initial cell–wise classification, and the simulated annealing used for finding the optimal label configuration considering spatial dependencies. In simulated annealing the new energy term $\hat{E}_{\mathcal{N}_i}$ is used because of its efficient computation. Together with temporal updating of the terrain map, that is, updating terrain probabilities and height values, we get an average runtime for the spatio–temporal classification of 23.9 ms, which corresponds to about 41.8 Hz. This shows that our classification method is real–time capable.

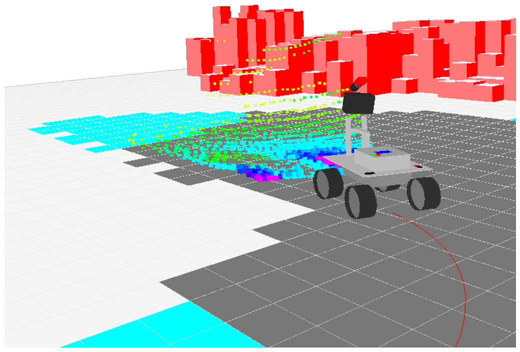
Finally, to show that these terrain and elevation maps can also be used for robot localization, we recorded five log files of our robot driving along the test track seen in Fig. 6(a), with an approximate length of 120 m. Since we are not interested in a precise but a semantic localization, we define a successful localization as one where the robot can localize itself from the beginning (at spot ①) to the end (at spot ③) of the track, and where it knows at any time in which exact terrain segment of the track it is located; otherwise, the whole localization test is regarded as not successful. For this purpose, from each of the log files a map was built, while localization was tested with the other four. The whole procedure was repeated five times for a total of 80 localization tests. Using a particle filter with 2000 particles and $\kappa = 0.5$, of these 80 tests 66 were successful, which is 82.5%. The localization uncertainty grows as the robot drives over open field (see Fig. 6(c)), but decreases abruptly as a different terrain type or a wall is seen.

VII. CONCLUSIONS AND FUTURE WORK

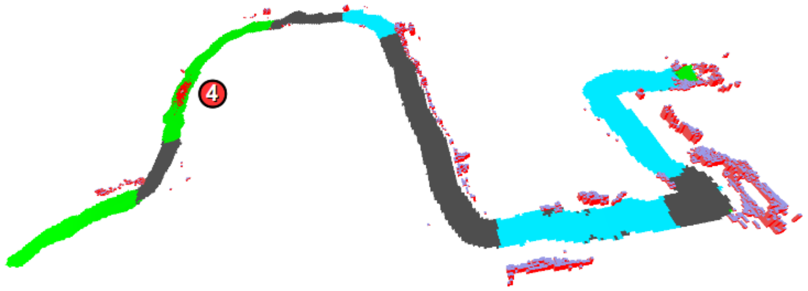
We presented a method for building local terrain and elevation maps with spatio–temporal terrain classification. The surrounding terrain of a driving robot could be classified with a classification rate of 98.4% using four terrain classes. We exploit the fact that terrain occurs in contiguous areas and therefore has a high spatial and temporal coherence. Considering this dependencies in a probabilistic manner provides very high classification rates. The presented method is hereby not limited to only terrain classification, but can be used for all grid–based classification problems. We also showed how the computation time of the spatial classification can be reduced, and how its results can be used for temporal classification by updating the terrain probabilities accordingly and by integrating the terrain grids into the map efficiently. The whole classification and mapping process runs at 41.8 Hz and is thus real–time capable. The basis of the classification is the model of the terrain generated by the Random forests in a supervised fashion. As also previously shown in [LKBZ12] and [LKZ13], these models can cope with terrain that looks very different from the training data in terms of lighting, texture, or even leaves on the ground. In order to make the method even more robust for all kinds of environments, we think about a semi–supervised learning method, which constantly updates the model as new terrain is seen. Finally, we also showed that the local terrain maps can be used for semantic robot localization using a particle filter, where the robot localizes itself based on terrain and obstacles. Ongoing work deals with object recognition, both using 3D scans and camera images. This also provides valuable information for localization and can be easily integrated into our existing framework.



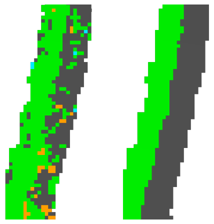
(a) Aerial view of the test track (only used for illustrative purposes). On the right side, the route leads through a covered passage.



(b) The robot building a map as it drives past spot ②



(c) The final terrain and elevation map. At spot ④ the particles of the particle filter are seen as red dots.



(d) Left: Mapping without recomputation of terrain probabilities, right: with recomputation as described in Sec. IV-C

Fig. 6. An aerial view of the test track with manually drawn route is shown in (a). (b) shows a 3D view of the mapping process with visible laser measurement points. The final terrain and elevation map can be seen in (c), and (d) shows what a difference the consideration of spatial dependencies can make. (Gray: asphalt, blue: cobblestones, green: grass, yellow: gravel, red:obstacles)

REFERENCES

[Bey84] William H. Beyer. *C.R.C. Standard Mathematical Tables*. CRC Press, 1984.

[Bre01] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[DFBT99] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, November 1984.

[HAW$^+$13] Marcel Häselich, Marc Arends, Nicolai Wojke, Frank Neuhaus, and Dietrich Paulus. Probabilistic terrain classification in unstructured environments. *Robotics and Autonomous Systems*, 61(10):1051 – 1059, 2013.

[KD07] Alexander Kleiner and Christian Dornhege. Real-time localization and elevation mapping within urban search and rescue scenarios. *Journal of Field Robotics*, 24(8-9):723–745, 2007.

[KZ10] Philippe Komma and Andreas Zell. Markov random field-based clustering of vibration data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, pages 1902–1908, Taipei, Taiwan, October 2010.

[LKBZ12] Stefan Laible, Yasir Niaz Khan, Karsten Bohlmann, and Andreas Zell. 3d lidar- and camera-based terrain classification under different lighting conditions. In *Autonomous Mobile Systems 2012*, Informatik aktuell, pages 21–29. Springer Berlin Heidelberg, 2012.

[LKZ13] Stefan Laible, Yasir Niaz Khan, and Andreas Zell. Terrain classification with conditional random fields on fused 3d lidar and camera data. In *European Conference on Mobile Robots (ECMR 2013)*, Barcelona, Catalonia, Spain, September 2013.

[Mor89] Hans Moravec. Certainty grids for sensor fusion in mobile robots. *Sensor Devices and Systems for Robotics*, pages 243–276, 1989.

[NH08] Andreas Nüchter and Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[NL11] Sebastian Nowozin and Christoph H. Lampert. *Structured Learning and Prediction in Computer Vision*. Foundations and Trends in Computer Graphics and Vision. Now Publishers, 2011.

[SH74] Ivan E. Sutherland and Gary W. Hodgman. Reentrant polygon clipping. *Commun. ACM*, 17(1):32–42, January 1974.

[TT10] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Trans. Img. Proc.*, 19(6):1635–1650, June 2010.

[WB10] Ulrich Weiss and Peter Biber. Semantic place classification and mapping for autonomous agricultural robots. In *Proceeding of IROS Workshop on Semantic Mapping and Autonomous, Knowledge Acquisition*, 2010.

[WKK$^+$13] Kai M. Wurm, Henrik Kretzschmar, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. Identifying vegetation from laser data in structured outdoor environments. *Robotics and Autonomous Systems*, 2013.

[WSFB05] Denis F. Wolf, Gaurav S. Sukhatme, Dieter Fox, and Wolfram Burgard. Autonomous terrain mapping and classification using hidden markov models. In *International Conference on Robotics and Automation*, pages 2038–2043, Apr 2005.