Journal of Intelligent & Robotic Systems (2019) 93:723–743
https://doi.org/10.1007/s10846-018-0865-x

Employing Natural Terrain Semantics in Motion Planning for a Multi-Legged Robot

Dominik Belter[1] · **Jan Wietrzykowski**[1] · **Piotr Skrzypczyński**[1]

Received: 2 November 2017 / Accepted: 2 May 2018 / Published online: 22 May 2018

Abstract

This paper considers motion planning for a six-legged walking robot in rough terrain, considering both the geometry of the terrain and its semantic labeling. The semantic labels allow the robot to distinguish between different types of surfaces it can walk on, and identify areas that cannot be negotiated due to their physical nature. The proposed environment map provides to the planner information about the shape of the terrain, and the terrain class labels. Such labels as "wall" and "plant" denote areas that have to be avoided, whereas other labels, "grass", "sand", "concrete", etc. represent negotiable areas of different properties. We test popular classification algorithms: Support Vector Machine and Random Trees in the task of producing proper terrain labeling from RGB-D data acquired by the robot. The motion planner uses the A* algorithm to guide the RRT-Connect method, which yields detailed motion plans for the multi-d.o.f. legged robot. As the A* planner takes into account the terrain semantic labels, the robot avoids areas which are potentially risky and chooses paths crossing mostly the preferred terrain types. We report experimental results that show the ability of the new approach to avoid areas that are considered risky for legged locomotion.

Keywords Walking robot · Mapping · Terrain classification · Motion planning

1 Introduction

In the recent decade walking robots made a great progress toward autonomy. The legged locomotion modality makes these robots capable of traversing diversified terrain types and negotiating various obstacles, both natural and man-made. These capabilities are particularly pronounced in multi-legged robots, which are often slower than their four-legged or bipedal counterparts but can explore a larger number of ground contact points to move safely over challenging terrain. A walking robot can carry a variety of sensors for environment monitoring. The perception-based motion planning was shown to be more efficient in rough terrain [5] than the popular behavior-based control paradigm [30, 36]. Behavior-based motion forces the robot to try the given movement of the legs many times, unless stable and safe footholds are reached.

In the earlier publications, we presented an integrated approach to perception-based motion planning for hexapod robots. The main highlights of this approach are the two-tier hierarchical motion planner named guided-RRT, and a matching two-level environment model in the form of coupled elevation grids. However, the planner from [5] considered only the geometry of the terrain. In contrast, a person walking in rough terrain or among structured obstacles considers not only the shape of the terrain (and objects) but also the semantics of the perceived environment. Owing to our experience we can associate particular semantic labels, such as "grass", "mud", "concrete", with particular physical properties, and a proper motion behavior. Finally, we can avoid some areas altogether if the imposed semantics tells us that stepping over this area could be dangerous. The motion planner

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-018-0865-x) contains supplementary material, which is available to authorized users.

✉ Dominik Belter
Dominik.Belter@put.poznan.pl

Jan Wietrzykowski
Jan.Wietrzykowski@put.poznan.pl

Piotr Skrzypczyński
Piotr.Skrzypczynski@put.poznan.pl

[1] Institute of Control, Robotics and Information Engineering, Poznan University of Technology, Poznan, Poland

described in [5] can handle this only to a very limited extent, as the relation between the semantics (which we want to know) and the geometry (which is encoded in the map) is at least ambiguous.

Therefore, we propose to include the semantic labels directly into the terrain map. We assume that data used for classification are produced by an RGB-D sensor. The availability of photometric information enables our classification method to distinguish between classes that are similar with respect to the geometric description. Then, the class labels are attached to cells of the elevation grids used in the system. The motion planner can utilize information about the semantic labels in various ways. Our research employs this information in the guiding (upper-tier) path planner utilizing the A* algorithm. The semantic labels allow the planner to prefer some terrain classes because they provide more stable and safer support to the robot's feet. Whereas concrete or asphalt-covered roads provide trustable footholds (Fig. 1a), on a surface covered by fallen leaves or in a lawn area (Fig. 1b) stable footholds are available below the level suggested by the geometry of the elevation map. Eventually, the rough guiding path computed by A* can make a detour from the path that is the shortest one considering only the terrain geometry. Thus, the full motion plan computed by the lower-tier RRT-Connect algorithm following the rough path deals only with the areas of preferred physical properties, which makes the execution safer and faster. Note that in this paper we do not consider the use of the semantic information in the RRT-based planner. Although it is possible and could be beneficial for safer foothold selection, we consider avoiding or preferring whole areas as a sufficient strategy, which conserves time in the lower-tier planner.

The main contribution of this research is the new environment model, which combines the geometric and the semantic description of the terrain. Unlike other recent works on pixel-wise semantic image segmentation for natural terrain description, we directly fuse all the perceived information in the 3-D volumetric environment model. We infer the semantic labels from the 3-D map, which allows us to take advantage from the geometric structure of this model. Moreover, we add to the guided-RRT algorithm the ability to consider semantic labels of terrain patches. An additional contribution is the experimental evaluation of the Kinect v2 sensor for outdoor perception, localization, and 3-D environment model generation. A data set from these experiments, including semantically labeled sequences of RGB-D images, is made publicly available. A proof-of-concept version of our approach to terrain semantics-aware motion planning was initially evaluated on mockups in a laboratory, and presented in the workshop paper [2]. The journal paper describes a full implementation of this approach, which integrates robot localization and improved terrain classification, and reports results of extended tests in a real outdoor environment (Fig. 1).

2 Related Work

This section addresses the most notable related work within three aspects of our system: environment mapping, terrain classification, and planning algorithms. We highlight the advancements made in this research with respect to current state of the art.

2.1 Environment Maps

The most popular terrain representation in motion planning systems for robots moving on uneven terrain is the elevation grid [25]. This simple concept was modified and extended in numerous works, for example the variant described in [35] makes it possible to model scenes with bridges, tunnels, and similar structures. Ye and Borenstein introduced heuristic rules to integrate 2-D laser range measurements into an elevation grid [50]. We have modified this approach using Kalman filtering to handle the spatial uncertainty of range measurements [4]. Then, this method was further refined in [3] including uncertainty modeling for RGB-D sensors. In contrast to laser scanners, the RGB-D sensors provide dense depth data and compatible RGB images yielding richer data for mapping. The popular Kinect/Xtion structured-light RGB-D sensors already proved to be useful for indoor elevation mapping on walking robots [3]. However,

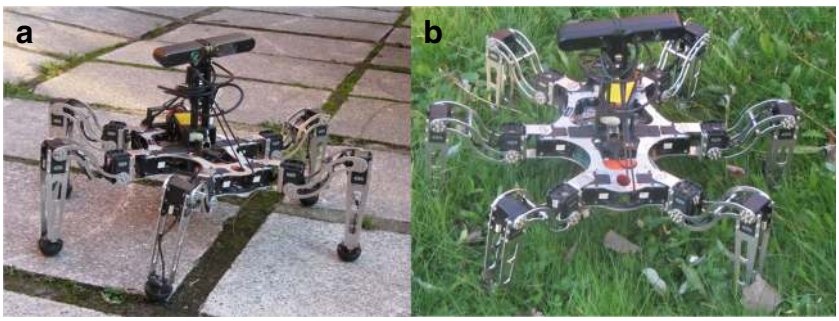

Fig. 1 Messor II robot in natural outdoor environment: on a pavement (**a**), and in a vegetation-covered area (**b**). These examples show differences between the terrain types that have to be handled in motion planning

for outdoor experiments we use the recent technology Kinect v2 time-of-flight RGB-D sensor. The Kinect v2 was successfully tested on a legged robot by Fankhauser et al. [16], and proved to be superior to the structured-light RGB-D technology in robot navigation [24].

Whereas elevation grids are a popular representation due to the inherent simplicity of their data structure and the ability to quickly update and query the map, they have significant drawbacks if the representation of more complex environments is considered. Therefore, volumetric map representations are considered for 3-D terrain mapping. The voxel-based map building algorithm presented in [15] used the notion of positive and negative occupancy mass and provided efficiency with respect to the map size. The OctoMap library [21] employs the octree data structure to implement a multi-resolution voxel-based map. OctoMap efficiently handles noisy measurements and the uncertainty in robot poses, while the octree structure results in a memory-efficient representation of the map. Considering these advantages we have decided to employ the OctoMap as an intermediate data structure used to register together with the point clouds we obtain from the RGB-D sensor. Other types of 3-D maps are also used to represent outdoor or mixed outdoor/indoor environments. A notable example is the work of Droeschel et al. [14] that employs variable-resolution grid maps and surfel-based representation in navigation system.

2.2 Environment Classification

In some environments, the local shape of the terrain surface does not allow for sufficient traversability assessment. In order to fully consider the terrain properties, the robot has to know the underlying semantics of the areas the robot should traverse. Hoepflinger et al. [20] proposed to recognize the type of terrain from haptic data using a force sensor mounted on a leg performing the specific probing motion. A more practical variant of this idea was considered by Walas [45], who used force/torque sensors on the forelegs of a hexapod, enabling the robot to label the terrain type while walking. As adding force or force/torque sensors to the legs of a walking robot increases its cost and complicates the mechanics, some researchers propose to use information readily available from the servo drives of a legged robot for terrain classification [8, 30]. Similarly, internal sensors like the inertial measurement unit (IMU) and motor encoders were successfully employed for terrain classification in wheeled robots [33]. Unfortunately, to measure parameters of the interactions with the terrain using either tactile, force/torque or IMU data, the legged robot has to step into the given area. Thus, terrain labels acquired this way are mainly used by reactive controllers, rather than by path planners. For example, the amphibious walking robot described in [17] differentiated sand from shallow water in order to switch between the walking and swimming gaits. More recently, a behavior-based robot proposed by Stejskal et al. [40] had to walk on the off-road area to realize the mistake and to go back to the desired path.

In contrast, terrain classification based on visual and/or range sensing gives the robot a potential to take the right decision in advance, before any physical contact with the undesirable area occurs. Visual terrain classification was researched extensively for wheeled ground robots [12, 18] and self-driving cars [41]. The cited works used stereo or monocular cameras and laser scanners, respectively, but terrain classification for navigation of ground robots employs also RGB-D sensors, and multi-sensor setups. Information from a passive camera can be complemented by laser scanner data [27, 47]. The laser scanner provides intensity values of the reflected light that help distinguish vegetation [49].

The visual terrain classification systems known from the literature extract various features from the RGB and/or depth images, and apply a broad range of classifiers. The most popular seems to be the Support Vector Machine (SVM) [9], but the Random Trees classifiers [10] were also applied successfully [47], as well as regression-based classifiers [18] and neural networks [33]. In the last few years, the Convolutional Neural Network (CNN) architecture and the deep learning paradigm significantly improved the state of the art in semantics segmentation of images. Following this success, deep learning is nowadays also the most researched approach in terrain classification using visual data. A notable recent example is the work of Maturana et al. [29] employing a custom CNN for segmentation of images into regions belonging to several classes relevant for terrain traversability. The semantic information is then projected onto a 2.5D elevation map obtained from laser range data and used to plan the motion of a wheeled vehicle. Deep learning can be also applied in semantics segmentation of multi-sensory data, as demonstrated in [43], where RGB, depth, and near-infra-red images are processed together to obtain pixel-wise terrain labels. In spite of their good performance, the CNN-based approaches have some drawbacks. They require a long process of learning on a large data set, which has to labeled. In contrary, our Random Tress classifier was successfully learned on a sequence of 60 frames. Moreover, in the CNN solutions taken from computer vision, the images are directly fed to a CNN to generate a pixel-wise labeling. Our solution performs classification on the voxel-based 3-D map, which allows us to use features that depend on the local geometry of the map. A terrain classification system can be enhanced by performing probabilistic inference after the main classification stage. Laible et al. [27] use

conditional random fields (CRF) in the inference stage to describe dependencies between image segments labeling in a way that encourages similar neighboring segments to have the same labels. This concept was also applied in our recent work [47] resulting in significantly fewer misclassified segments in the traversable areas. Therefore, in this research we also applied CRF to post-process the voxel-based map labeled by using the Random Trees classifier.

2.3 Motion Planning

Motion planning for a multi-legged robot is a challenging task because the algorithm has to avoid collisions with obstacles, find feasible contact points, and preserve the stability of the robot. Larger legged robots traversing moderately rough terrain can adopt well-known 2-D path planning algorithms, such as A* [48] or D*-Lite [13]. The information from a terrain map available to the robot may be used to adjust parameters of the cyclic gait, as in the LS3 robot [1]. However, an accurate terrain map can be used directly to define constraints imposed on the robot motion, as in the RRT-based planners, or can be used to define a cost map that guides the planner along low-cost paths. The latter approach, characteristic of most of the classic planning algorithms requires defining the traversability cost upon some perceivable features of the terrain [34]. Our coarse path planner, based on the A* algorithm also utilizes a cost map computed using the spherical variance [38], which captures in a compact form the local roughness of the terrain. It should be noted that classic planning algorithms, such as A*, can be used with topological maps that represent recognizable areas in the space [51].

Some motion planning systems for legged robots divide the planning problem into two separate stages, considering at first the main body path, and then footholds and feet trajectories [23]. This can lead to suboptimal results, as some feasible movements are not considered by the 2-D planner. Therefore, we plan the path of the whole robot and the motion of the legs (i.e. the contact points) at the same time. In such an approach it is essential to avoid combinatorial explosion due to the high-dimensional search space of a multi-legged robot. Thus, sampling-based motion planners, as the Probabilistic Roadmap Method (PRM) [22], and the Rapidly-exploring Random Tree (RRT) [28], were found to be an efficient way to handle the problem of high-dimensionality [19]. The idea of RRT spawned many variants of this algorithm and made sampling-based motion planning popular for high-dimensional problems in robotics. The RRT* algorithm was used recently on the StarlETH quadruped to avoid obstacles [46]. Planning motion of statically stable robots, such as hexapods or crawling quadrupeds allows to neglect the dynamics, and use more elaborated kinematic models instead [39]. The RRT-Connect variant [26], which is used in our approach, grows two Random Trees that expand towards each other to increase the chance of finding a feasible solution. Our two-tier motion planner, which combines RRT-Connect and A* [5] was to some extent inspired by the work of Vonasek et al. [44] that employs an auxiliary path to guide the growth of the tree through the environment, increasing the chance to quickly explore narrow passages of the configuration space.

Achieving the ability to plan and execute feasible motion of a walking robot on uneven terrain in real-time is naturally related to the terrain mapping and localization capabilities of the robot. Earlier motion planning systems, e.g. those developed for the LittleDog robot within the Learning Locomotion Programme [23, 52] relied mostly on accurate terrain maps known in advance and external motion capture systems for localization. On the other hand, the perception-based motion planning systems for multi-legged robots described so far in the literature were often limited to coarse path planning, employing some form of cyclic gaits and reflexes to control the motion of the legs [1, 37, 48]. In [5] we have shown that the adaptive, two-tier motion planner can work autonomously with real-time terrain perception and mapping using on-board sensors, and integrated localization that utilized the same sensory data. Recently, real-time motion planning capabilities with on-line mapping have been also demonstrated by others, e.g. for a quadruped robot [46], and a hybrid legged-wheeled mobile manipulator [14].

3 Perception and Environment Model

3.1 Perception System

The approach to motion planning presented in this paper is tightly coupled with a dedicated terrain mapping system, which in turn is tailored to the requirements of this planner, taking also into account properties of the legged robot perception system.

The perception system is designed for the Messor II hexapod [7] (Fig. 1). The body of the robot is 299 mm long and 205 mm wide, and the mass of the whole machine is 2.5 kg. The maximal clearance between the body and the ground is 290 mm. The robot has 18 active degrees of freedom actuated by Robotis Dynamixel RX-28 servos. Though the robot is small, the torques produced by its servos (2.5 Nm in each joint) are sufficient to carry external sensors attached to the upper deck of the body. For most of the experiments, Messor II is equipped with an RGB-D sensor, usually the compact Asus Xtion PRO Live. Although using larger and heavier sensors is possible, they limit the motion capabilities, causing excessive slippage of the feet

and occasional overheating of the servos. While the robot has an IMU module and touch sensors in the feet, they are currently not integrated within the mapping framework and used only by the reactive control functions. Therefore, the RGB-D sensor is the sole source of data for environment mapping and yields both information about the observed surfaces, and the pose of the robot with respect to a global reference frame.

The localization function is accomplished by a program that is external to our perception and mapping framework – the ORB-SLAM2 [31]. This is a real-time SLAM system that can work with passive camera images in the monocular mode, or in the stereo mode, using either pairs of images or RGB-D data with depth information. The stereo/RGB-D mode enables to avoid the annoying map initialization procedure [31] and provides more reliable pose estimates without a scale drift. Thus, the ORB-SLAM2 was used with the Kinect v2 RGB-D frames to localize the robot in the outdoor scenes. However, for the early indoor experiments, the poses of the RGB-D sensor were obtained off-line by manually stitching the local point clouds [2].

3.2 Environment Model Architecture

The mapping procedure involves two main data structures: the 3-D, voxel-based octree map, and the 2.5-D, grid-based elevation map. The octree voxel map implemented through the OctoMap library [21] is used as an intermediate environment representation, which allows our system to capture all aspects of the environment that are perceived by the RGB-D sensor. Then, we generate from the voxel-based map a simpler, grid-based map, which provides quick access to the terrain elevation values for both planning algorithms in the two-tier motion planner. Also, the OctoMap is used at two different spatial resolutions. This is motivated by the requirements of the semantic classification procedure. The terrain classification process is integrated within the mapping framework. The semantic categories are determined for individual voxels of the OctoMap and then transferred to the elevation grid. However, a sufficient number of RGB-D measurements have to accumulate in a voxel to enable efficient classification. Thus, the OctoMap structure is also used at two different resolutions. The RGB-D measurements are integrated into a fine granularity OctoMap, having $1.5\times1.5\times1.5$ cm voxels, which describes all geometric aspects of the scene. The granularity of the OctoMap is compatible with the resolution of the elevation grid, which in turn is chosen considering the size of the robot foot. In contrast, the classification process uses much larger, $18\times18\times18$ cm blocks that are imposed on the main OctoMap structure. The architecture of the whole mapping system is depicted in Fig. 2.

3.3 Data Registration and Octree Map

Environment mapping starts with the registration of the RGB-D point clouds obtained from a sensor into an octree representation with $1.5 \times 1.5 \times 1.5$ cm voxels. To obtain information about geometry of the whole scene each point cloud measured by the RGB-D camera $\mathbf{p}^C$, and expressed in the camera frame C is transformed into the global map coordinate system M using homogeneous transformation (Fig. 3):

$$\mathbf{p}^M = \mathbf{M}^{-1} \cdot \mathbf{C} \cdot \mathbf{p}^C, \tag{1}$$

where $\mathbf{M}$ and $\mathbf{C}$ are the transformation from the global frame to the map and camera coordinate frames, respectively. To find the coordinates of a point cloud obtained from the particular vantage point we have to find the pose of the RGB-D camera $\mathbf{C}$, or in other words, to localize the robot. Whereas as demonstrated in [2] a simple procedure using the Umeyama algorithm [42] can be applied to register few point clouds during indoor experiments, for the outdoor missions, we applied the ORB-SLAM2 to estimate the sensor poses directly from the sequence of RGB-D frames.

Comparing to the original OctoMap from [21], each voxel is augmented by information that is then needed for terrain classification. As it is inefficient to keep all the registered RGB-D measurements in the octree structure, the additional information is updated sequentially when the measurements are integrated into the voxels. These values are the average RGB color of the voxel $\mathbf{k}^v_{n+1}$, and its average elevation h^v_{n+1}:

$$h^v_{n+1} = h^v_n + \frac{1}{n+1}(h^v_{\text{new}} - h^v_n), \tag{2}$$

$$\mathbf{k}^v_{n+1} = \mathbf{k}^v_n + \frac{1}{n+1}(\mathbf{k}^v_{\text{new}} - \mathbf{k}^v_n), \tag{3}$$

where h^v_n is the previous value of the voxel elevation, h^v_{new} is the measured height of the next integrated RGB-D point, $\mathbf{k}^v_n$ is the previous value of the average color, $\mathbf{k}^v_{\text{new}}$ is color of the next integrated point, and n is the number of the voxel updates. Note that $\mathbf{k}^v$ is a vector, as it holds three color parameters, while h^v is a scalar value. Then, the actual features used in classification are defined upon these basic values. These features are computed only whenever they are required by the classifier to conserve memory. Once the terrain gets classified, the voxels are also augmented with the computed semantic labels. The example set of aligned point clouds and corresponding octree representation of the environment is presented in Fig. 4a and b, respectively.

3.4 Elevation Map

We use two motion planning algorithms, the RRT-Connect, and the A* that require grid maps of different resolution.

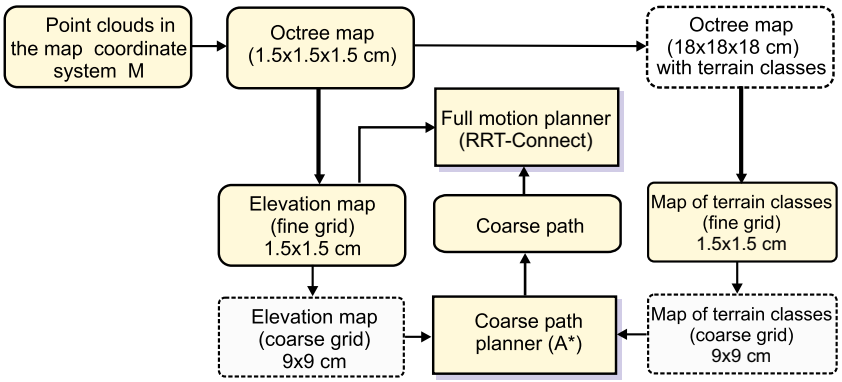

Fig. 2 General scheme of the terrain modeling system and its relation to motion planning which shows how the colored point cloud is used to create a geometric terrain model enhanced by the information about terrain type. Rectangles denote processing blocks, while rounded rectangles are data structures with the virtual data structures marked by the dashed contour. Note that this scheme focuses on data structures and shows only main processing blocks of the planner

The RRT-Connect algorithm involves foothold selection, thus it requires a fine grid with the cell size smaller than the footprint of the robot's leg. Using this fine elevation grid for coarse path planning with the A* algorithm would be inefficient, as the A* computation time depends on the number of states it has to visit (i.e. the number of cells). On the other hand, the traversability cost computation procedure used by this planner does not require a fine terrain model. In result, we implemented a grid data structure with small cells (1.5×1.5 cm), but the coarse path planner reads from this grid much bigger virtual cells (9×9 cm) that suit this algorithm. The size of the larger cells is chosen considering the dimensions of the walking robot's body, as it is assumed that the coarse path planner should prefer terrain where the whole robot can get through. Possible more difficult passages are handled by the RRT-Connect planner that considers individual footholds and controls the robot posture. The elevation grid is generated directly from the OctoMap structure of matching voxel size (1.5×1.5×1.5 cm) with respect to the global coordinate system. Hence, we avoid the data filtering and integration procedures in the elevation grid that we have used in our earlier implementations [3, 4].

The elevation grid is stored in a two-dimensional array which provides constant and short access to each cell value. This property is very important regarding the sampling-based motion planning method used in this research. We compute the elevation map directly from the voxel-based octree representation. We take into account the stack of n voxels $v^{[i,j,0]} \dots v^{[i,j,n]}$, where i and j are the indices of the structure in the horizontal plane, located above the $c^{[i,j]}$ cell of the elevation grid. From all voxels in that stack the one that has the highest elevation $v^{[i,j,k]}{}_{h_{\max}}$ $(k \in 0 \dots n)$ is used to set the elevation of the considered cell $c^{[i,j]}$ in the 2.5-D grid. OctoMap voxels that are located higher than the maximal height of the robot (including the sensors on top) with respect to the local ground plane level are not considered for elevation grid update. The input OctoMap and the obtained elevation map are presented in Figs. 4b and 4c, respectively. To make the semantic information available to the path and motion planning algorithms, we augment the elevation grid with terrain class labels. To this end, the information contained in the coarse 18×18×18 cm blocks of voxels has to be projected onto the elevation grid. For each voxel of the octree map belonging to a coarse block that got classified as a particular terrain type, we read the class label held there. This 3-D semantic information has to be projected onto the planar grid structure. We use the class label that is on top of the OctoMap stack of voxels to augment the corresponding cell in the elevation grid. Only labels of the voxels that were used for elevation update are considered. This choice is justified by the observation that voxels placed higher carry more useful information about the obstacles, and in general, they are classified better, as seen as in Fig. 5. In result, each cell of the elevation grid contains information about the terrain type. The map of terrain type semantic labels is shown in Fig. 4d.

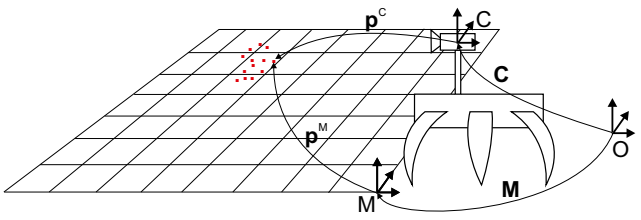

Fig. 3 Geometrical relations between the map and camera coordinate frames used to update the map from the RGB-D point cloud

Although the terrain class labels are transferred to the fine-grained elevation grid, the semantic information is used by the coarse path planner that requires a map with 9×9

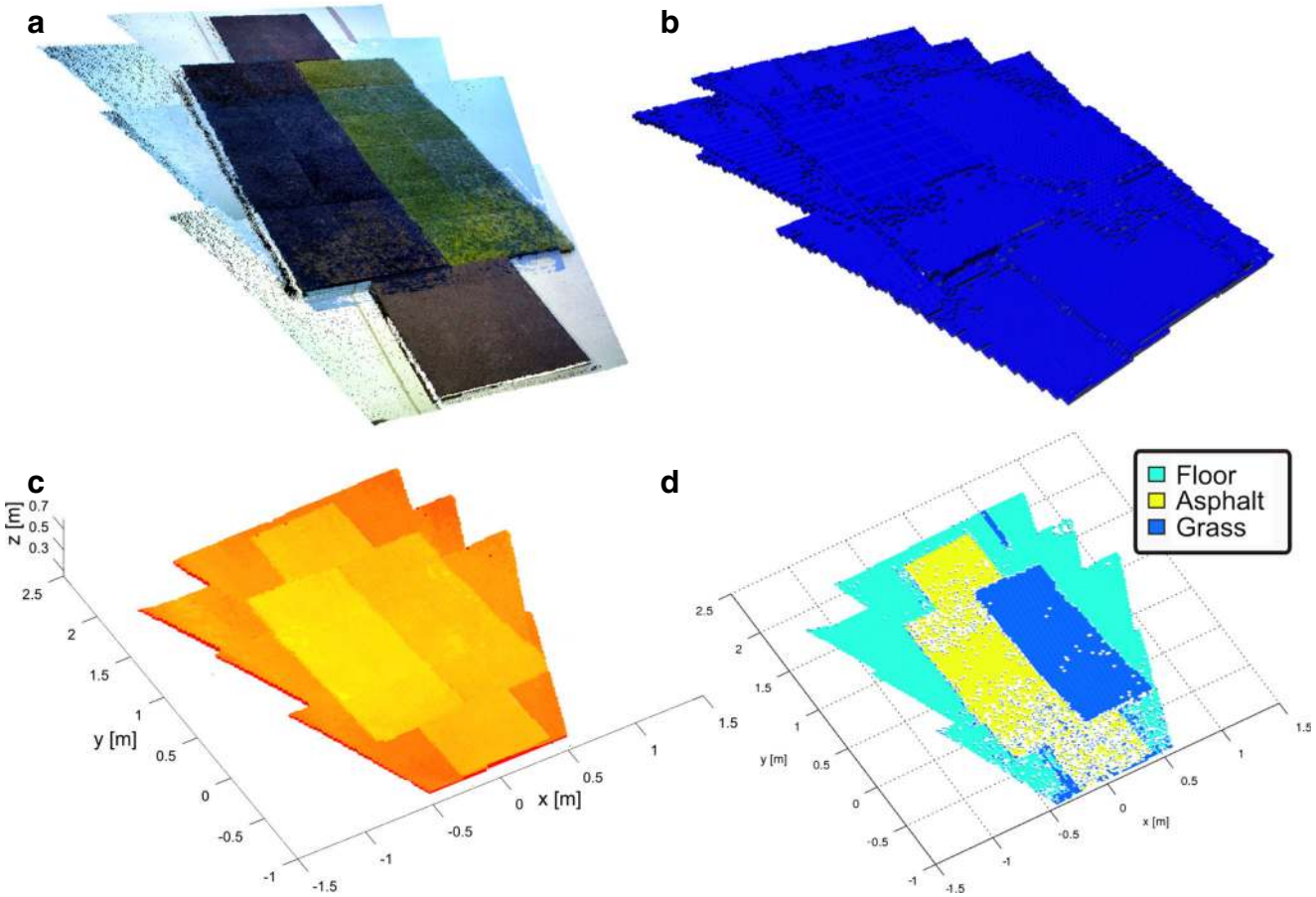

Fig. 4 The representations of the environment used by the walking robot. The aligned point clouds (**a**) are used to update OctoMap (**b**). The elevation map (**c**) is obtained from the octree representation together with the map of classes (**d**)

cm cells. Hence, we downsample the fine grid to obtain the coarse grid. Taking into account the memory requirements, the coarse grid is a temporary data structure only (virtual local map). We do not create the whole coarse grid, but we compute the elevation values for the queried cells whenever they are needed. However, for the computational efficiency, we store information about updated cells of the virtual coarse grid. To compute the elevation of the coarse grid cell, the maximal elevation of the fine grid cells covered by the larger cell is selected. However, the downsampled coarse

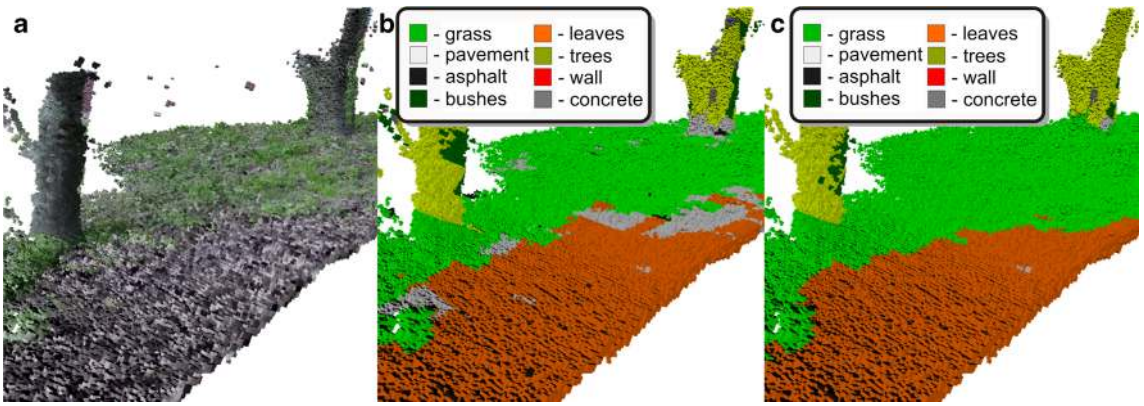

Fig. 5 Close-up view on a scene represented as colored voxels in the octree map (**a**), and classification results for RT (**b**) and RT+CRF (**c**) methods. To update the map of terrain classes we use a label of the top voxel. It is justified by the fact that those voxels are usually better classified in contrast to those placed lower in the stack

cell does not get a unique terrain class label transferred from the fine grid. We use the larger cells in the coarse grid to establish a simple probabilistic representation of the terrain semantics, and we compute a probability that the specified region (larger cell) belongs to the n-th class $P(s_n)$:

$$P(s_n) = \frac{1}{i_{\max} \cdot j_{\max}} \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} \begin{cases} 1 & \text{if } c_{i,j} \text{ is } s_n \\ 0 & \text{if } c_{i,j} \text{ is not } s_n, \end{cases} \tag{4}$$

where $i_{\max}$ and $j_{\max}$ are the number of rows and columns, respectively, in the patch of fine grid covered by a single cell of the coarse grid ($i_{\max}$=6, $j_{\max}$=6 in the implementation), s_n is the specified class label, and $s_{i,j}$ is the class label stored in the $[i, j]$ cell of the fine grid. Finally, the cells of the coarse grid contain information about the height of the terrain and probability of occurrence of the specified terrain type. In Eq. 4 we can use probabilities given by the Random Trees algorithm. Instead, we use a voting scheme, which assumes that the fine grid cells are labeled according to the decision taken by the majority of trees in the classifier. Thus, the probability that the larger cell belongs to the given class depends only on the number of the small cells belonging to that class located within the larger one. This scheme avoids situations when a few fine grid cells classified with high certainty outweigh a much larger number of cells that have however more uncertain labels. As a result, we obtain larger differences between class probabilities $P(s_n)$ stored in the coarse grid. Searching for the path in such a grid is faster because the differences between preferred and undesirable terrain types are larger.

4 Terrain Classification

Whereas the motion planner relies mainly on the geometry of the terrain surface encoded by the elevation map, the coarse path planning algorithm A* can benefit from the recognized terrain classes by taking more informed decisions as to the nature of the areas it suggests to traverse. The robot not only should distinguish different terrain surfaces, such as pavement and grass, but should be aware of such obstacles as tree trunks and man-made structures. Therefore, terrain classification is performed on the augmented octree structure, rather than on the elevation grid. This allows our system to collect and use more information about the environment semantics, particularly about the vertically extended structures: trees, bushes, and walls. The classification process needs larger voxels that capture a much bigger number of measured points, because a sufficient statistic has to be gathered for each voxel. Thus, all operations related to classification are performed on blocks of the octree voxels that are 18×18×18 cm in size, i.e. they contain 12 × 12 voxels.

A schematic overview of the whole terrain classification module is depicted in Fig. 6. The whole process starts with an RGB-D point cloud acquired by the robot. Points from different views are transformed to a common frame of reference using the known RGB-D sensor poses. The point cloud is used to build the octree map, and to estimate the ground plane in the scene area. The estimated ground plane is required to localize the octree voxels with respect to the global scene coordinate system, to know how high above the ground each voxel is located. The octree map is then classified and the results are refined using Conditional Random Fields (CRF) to perform a probabilistic inference.

The coarse octree map holds in each voxel the average color $\mathbf{k}^v$, median elevation h^v, and histograms of H and S components from the HSV space, denoted $\mathbf{f}_H^{\text{hist}}$ and $\mathbf{f}_S^{\text{hist}}$, respectively. The classifiers we apply require features that enable to distinguish between terrain types and these quantities are used to recognize the basic semantics of the observed scene. The features should also generalize the description of voxels belonging to the same terrain type to account for appearance variations. For each octree voxel, separate features were concatenated to form a features vector and then fed to the classifier. The choice of a proper features vector is crucial for the classification results. In our system, the features can be chosen depending on the characteristics of the environment, and the expectations as to the discriminative power of the particular feature. Thus, in our initial indoor experiments, we used extremely simple features to facilitate the computing speed. These features were the average color components in the voxel. To minimize the influence of the varying lighting, the Hue-Saturation-Value (HSV) representation of the average color was used, resulting in the vector of features: $\mathbf{f}^{\text{ind}} = \left[f_{\bar{H}}, f_{\bar{S}}\right]$, where $f_{\bar{H}}$ and $f_{\bar{S}}$ stand for the average hue and saturation components, respectively. The value component is not used, as it depends too much on the external lighting conditions.

The information about the color properties of the voxel turned out to be sufficient to distinguish between few classes of the flat surfaces the robot was confronted with in the lab experiments [2]. However, we had to extend the set of features for the outdoor operation. Both, the natural and the man-made objects encountered outdoors come in many different shades of their basic colors and their surfaces are often characterized by the strong local variation of the color. Therefore, the set of features was modified by replacing mean color values with the color histograms computed in each voxel. Specifically, histograms of H and S components $\mathbf{f}_H^{\text{hist}}$ and $\mathbf{f}_S^{\text{hist}}$, respectively, were used with 32 bins for each one. These histograms represent well the scattering of the color on each surface type. Moreover, the median elevation f_h of the voxel was added as the feature, because some objects having surfaces of similar

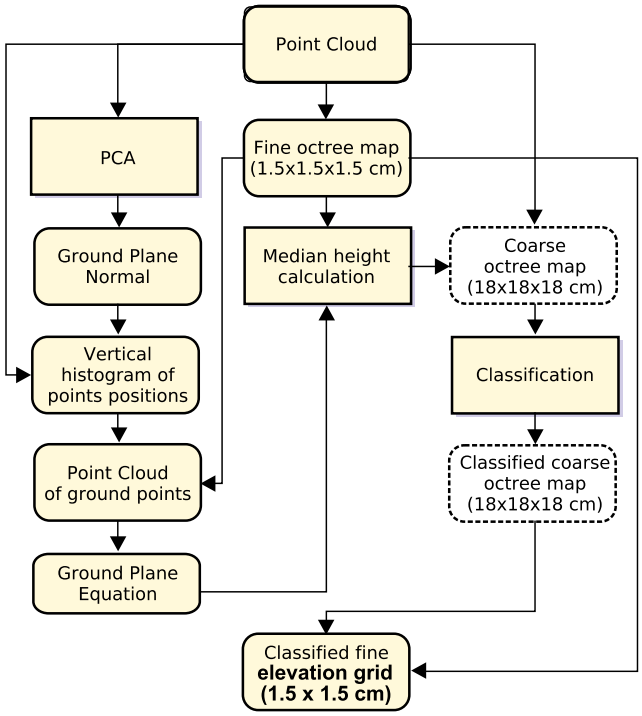

Fig. 6 Block scheme of the data flow inside the terrain classification module that finally produces a fine elevation grid augmented with terrain semantics. Virtual data structures are marked by dashed contours

photometric characteristics, e.g. pavement and concrete walls, can be distinguished by their elevation above the ground plane. The resulting vector of features had three elements: $\mathbf{f}^{\text{out}} = [f_h, \mathbf{f}_H^{\text{hist}}, \mathbf{f}_S^{\text{hist}}]$. Although we had to use a more complex vector of features for the outdoor settings, we tried to keep it as simple and compact as possible to facilitate real-time operation of the terrain classification system. We also took into account the properties of the sensor mounted on the legged robot. Our earlier results in terrain classification for wheeled robots [47] suggest that we should avoid the use of image texture that is easily destroyed by motion blur. As our other results, [32] show that RGB-D frames acquired in-motion on a legged robot are usually corrupted by motion blur, we used features based on the color and global geometry of the voxel, but not on texture.

The estimation of the ground plane is based on a detection of dominant directions in which the registered points, forming the point cloud $\mathcal{P}$, are scattered. By means of the principal component analysis (PCA), we find the principal component with the lowest variance, and we assume that it represents the coarse normal vector $\mathbf{n}_c$ of the ground plane. If we denote by $\mathbf{C}$ the covariance matrix of points positions, the normal vector is the eigenvector that corresponds to the lowest eigenvalue. Having the normal vector, we compute a histogram $\mathbf{w}$ of points positions along the direction of the normal, that is positions of points $\mathbf{p}_i$ projected onto this vector. The position of the histogram bin with the largest number of points determines the distance d^* from the origin to the ground plane. The values of the histogram bins are computed as follows:

$$w_j = \left|\left\{\mathbf{p}_i : |\mathbf{p}_i \cdot \mathbf{n} - d_j| \leq 0.5\tau_g\right\}\right|, \tag{5}$$

where w_j is the value of the j-th bin, d_j is the center of this bin, and τ_g is the width of the bins. The maximum value of w_j is then used to choose d^*:

$$d^* = \arg\max_{d_j \in \mathcal{D}} w_j, \tag{6}$$

where $\mathcal{D}$ is a set of bins centers. To increase the accuracy of the ground plane model, we further refine the obtained estimate, calculating coefficients of the final ground plane equation π using all inlier points, that is, points that are closer than τ_g to the initial ground plane. To ensure that points are evenly scattered across the whole area, we use only one, central point per the fine octree map voxel (denoted as $\mathcal{O}_f$). The pseudocode of the algorithm is presented as Algorithm 1, while the example results are shown in Fig. 7.

Algorithm 1 Ground plane estimation: $\mathcal{P}, \mathcal{O}_f \rightarrow \boldsymbol{\pi}$

```
// Compute coarse plane normal and centroid
[n_c, c_c] = PCA(P) ;
// Initialize histogram with zeros
w = [0, ..., 0] ;
// Build histogram of vertical positions of points
for p ∈ P do
    j = ComputeBinIndex(p, n_c) ;
    w_j = w_j + 1 ;
end
// Find the bin with the largest number of points
d* = arg max_{d_j ∈ D} w_j ;
// Initialize a set of ground points as an empty set
P_g = {} ;
// Select points from fine octree map that belong to the ground surface
for p ∈ O_f do
    // If distance to the origin is within bin's range
    if | Distance(p, n_c) − d*| < 0.5τ_g then
        P_g = P ∪ p ;
end
// Compute refined plane normal and centroid
[n_f, c_f] = PCA(P_g) ;
// Compute plane equation using refined normal and centroid
π = PlaneEquation(n_f, c_f) ;
return π ;
```

Besides a proper choice of features, a crucial decision that has to be made while designing a classification system

is the classifier itself. From the implementation point of view, the most convenient decision was to use the Support Vector Machine algorithm, which is available in the libSVM software [11]. The SVM is a widely known classifier [9], used in many state-of-the-art solutions to terrain recognition. In the system presented in this paper, SVM was used with a Radial Basis Function (RBF) kernel. This solution was tested for the indoor experiments using only the average color features [2]. The classifier was trained on series of raw RGB images. Pixels in the training images were labeled manually. Five surface classes were used: grass, floor, asphalt, sandstone, and wood. The sufficient number of training examples for the SVM classifier has been verified experimentally [2]. Considering these results, each separate class was trained using 1600 images.

Though the SVM classifier working with the simple color features enabled the robot to properly distinguish between the five classes of flat artificial surfaces in the indoor experiments [2], it performed much worse in the outdoor setting, even fed with the extended features vector $\mathbf{f}^{\text{out}}$ (Fig. 8). We attributed this problem to the varying importance of particular features for different classes. The semantic classes defined outdoors were: grass, bushes, trees, leaves, walls, concrete, pavement, and asphalt. Some of these categories could get a common label from the point of view of the terrain traversability for a walking robot, such as trees and bushes that both are non-traversable vegetation. However, distinguishing a larger number of categories increases the reliability of classification, decreasing the intra-class appearance diversity, as only really similar objects belong to the same class. Moreover, the larger number of classes enables to construct more flexible policies for path planning.

Whereas the average color was important when distinguishing between grass and asphalt or wall, it was less discriminative when making the distinction between grass and pavement, as some pavement tiles got a quite greenish shade over the years. A similar observation can be made for the pavement and the concrete, where the former label was used to describe such structures as outer parts of the manholes protruding from the ground. For these two classes, the median elevation is much more important than the color. Therefore, for the outdoor experiments, we have replaced SVM by the Random Trees (RT) classifier, which has the ability to rank the importance of the features in a natural way [10]. Random Trees is based on a collection of decision trees T_i, that is also called a forest. Given the input features $\mathbf{f}^{\text{out}}$, classification is performed separately on every tree. The result is a probability distribution of the class labels s for the considered block. The probability values are computed as the ratio of the number of trees that voted for the class to the number of all trees:

$$r_l = \frac{\left|\left\{T_i : T_i(\mathbf{f}^{\text{out}}) = l\right\}\right|}{|\mathcal{T}|}, \tag{7}$$

where $T_i(\mathbf{f}^{\text{out}})$ denotes an output from tree T_i, and $\mathcal{T}$ is the set of all trees. In the presented system, we chose RT classifier to be a trade-off between complexity and discriminative abilities, hence there were 200 trees of maximal depth equal to 10. Because other features than the pixel color had to be taken into account, the RT classifier was trained on data from the octree, not directly on RGB images. This is a feature distinguishing our approach from the state-of-the-art methods, including the recent solutions based on deep learning.

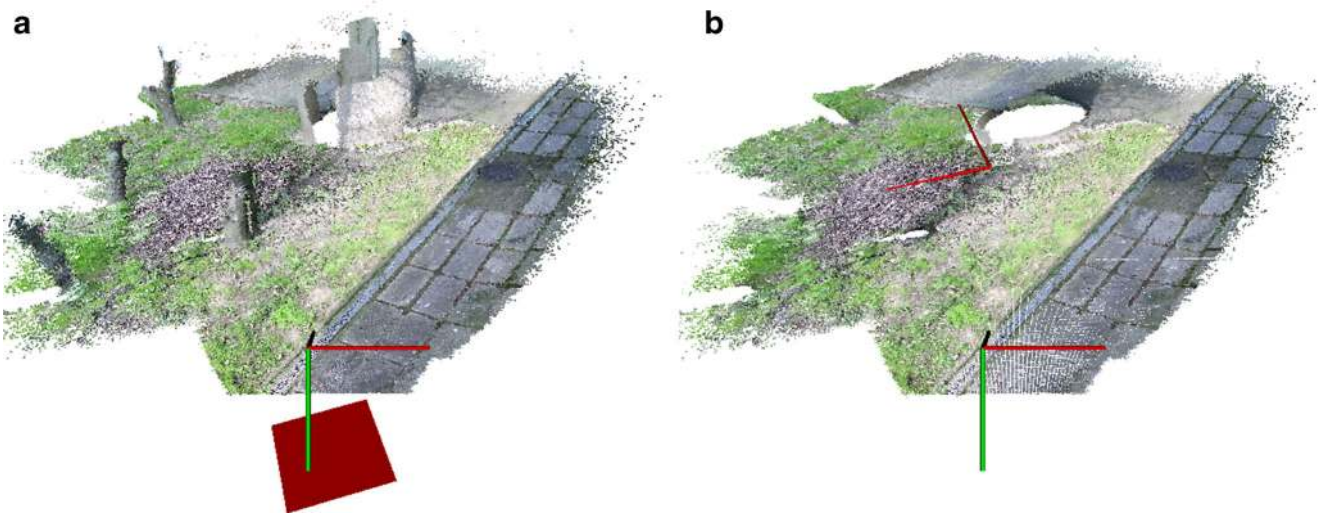

Fig. 7 A visualization of ground plane estimation process showing **a** the full point cloud and a ground plane segment (red square) and **b** points recognized as belonging to the ground plane and principal components scaled by their eigenvalues (red arrows)

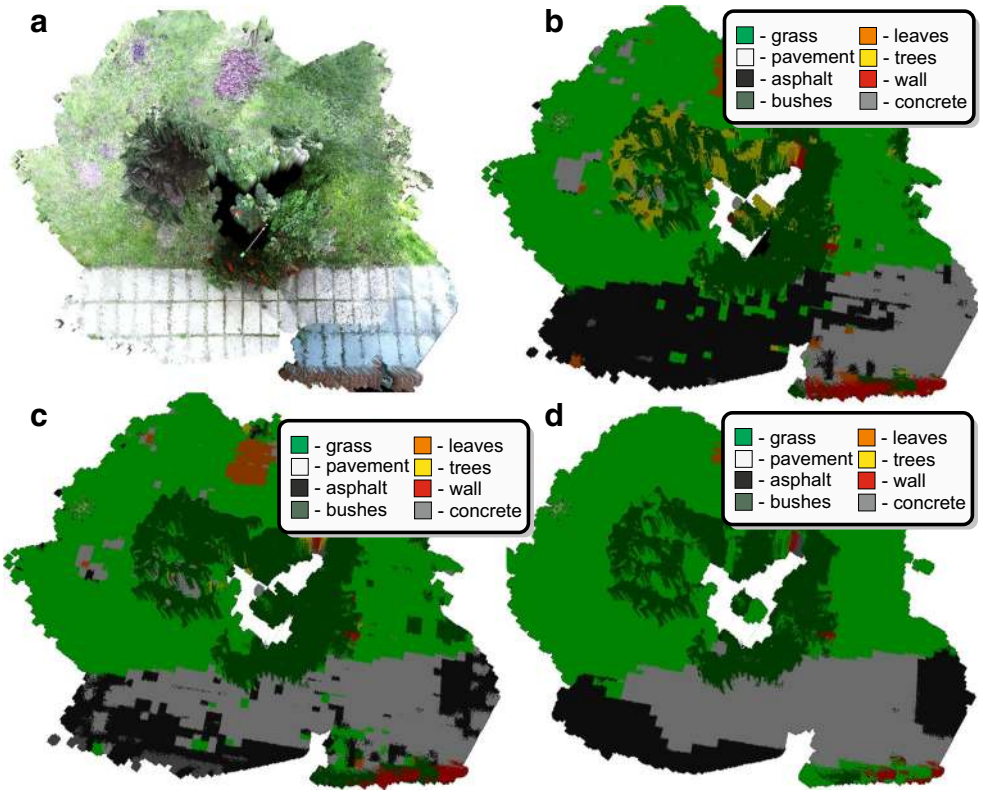

Fig. 8 Comparison of classification methods: registered colored point clouds (**a**), SVM classifier (**b**), RT classifier (**c**), and RT classifier + CRF results (**d**)

To further improve the results of the classification, we perform a probabilistic inference using Conditional Random Fields (CRF). CRFs enable to build a probabilistic model $\mathcal{M}$ that describes dependencies between neighboring blocks, thus making it possible to compute a consistent set of labels that maximizes the joint probability, not only marginal probabilities for each block individually. The model exploits the classification results as a prior knowledge and additionally harnesses the knowledge of adjacency between blocks. For each pair of adjacent blocks, their relation is conditioned on how similar are features describing them. Those features, denoted as $f^E_{i,k}$ for node i and feature k, are different from features used during classification to avoid overcomplicated models. We used mean RGB values and median height as the inference features, because computing differences in RGB space is straight-forward in opposition to HSV space. The model was described with the following formula for the joint probability:

$$p(\mathbf{s}|\mathbf{f}) = \frac{1}{Z(\mathbf{f})} \exp\left\{ \sum_{i\in\mathcal{N}} \sum_{l\in\mathcal{L}} \theta^N_l g_l(s_i, \mathbf{f}_i) + \sum_{(i,j)\in\mathcal{E}} \sum_{k\in\mathcal{F}} \theta^E_k h(s_i, s_j, f^E_{i,k}, f^E_{j,k}) \right\}, \tag{8}$$

$$g(s_i, \mathbf{f}_i) = -\log(p(s_i|\mathbf{f}_i)), \tag{9}$$

$$h(s_i, s_j, f^E_{i,k}, f^E_{j,k}) = \mathbf{1}_{\{s_i \neq s_j\}} \exp(-\beta(f^E_{i,k} - f^E_{j,k})^2), \tag{10}$$

where $\mathbf{s}$ is a set of labels for all blocks, $Z(\mathbf{f})$ is a normalizing factor, $\mathcal{N}$ is a set of blocks, $\mathcal{L}$ is a set of labels, θ^N_l are parameters controlling trust in classification results, $\mathcal{E}$ is

Algorithm 2 Terrain classification: $\mathcal{O}_c, \mathcal{T} \rightarrow \mathcal{O}_{cc}$

```
// Initialize classified coarse octree map with the given coarse octree map
O_cc = O_c ;
// For each cell in the map
for B ∈ O_cc do
    f = ExtractClassFeatures(B) ;
    r = Classify(T, f) ;
    AddClassResults(O_cc, r) ;
end
// Build CRF model using structure of the map and inference features
M = BuildCRFModel(O_cc) ;
Inference(M) ;
// For each cell in the map
for B ∈ O_cc do
    // Extract the label that with other labels gives the the highest joint probablility
    s= ExtractInferenceResults(M, B) ;
    AddInferenceResults(O_cc, s) ;
end
return O_cc ;
```

a set of edges, $\mathcal{F}$ is a set of inference features, θ_k^E are parameters controlling the level of agreement depending on inference feature k, $p(s_i|\mathbf{f}_i)$ are probablities from the classification step and β is a parameter controlling how fast grows the tendency to neighboring blocks having the same label when their features approach the same value.

As an inference mechanism, we used Loopy Belief Propagation algorithm with Tree Reparametrization scheduling. The parameters θ were computed with log-likelihood maximization procedure that used a set of labeled examples. The parameter β had to be tuned manually and a value 0.05 was chosen. Figure 9 contains a comparison between classification results without and with the inference step.

A data set consisting of 2944 RGB-D frames from three sequences, recorded with the Kinect v2 sensor in the outdoor experimental site was used. A global octree map representation for each sequence was built from these RGB-D frames using the same procedure which was used during classification. Every 50-th RGB image in those sequences was manually labeled indicating the image areas belonging to the considered classes. Some small areas that didn't fit to any of the defined classes were left unlabeled. Then, the labels from the 2-D areas in the images were projected onto the octree voxels, using the known position of the Kinect v2 sensor. Finally, we had 3 instances of the labeled octree maps that were used for training both, the classifier and the probabilistic model.

5 Motion Planning

To find the path between initial and goal pose of the robot we applied the guided-RRT algorithm [5]. The algorithm utilizes two motion planning algorithms: A* and RRT-Connect. The block diagram of the algorithm is presented in Fig. 10 and the formal description is given in Algorithm 3. The goal position of the robot $q_{\text{goal}}^{\text{3D}}$ is defined in 3-D space by a human operator. It is more practical than defining the full state of the robot at the end of the trajectory. Instead, our algorithm computes footholds for the goal position of the robot (x, y, yaw) and optimizes posture to find the optimal inclination of the robot's body and distance to the ground $(q_{\text{goal}}^{\text{24D}})$. The algorithm returns a sequence of robot poses defined in 24-dimensional space between the initial pose of the robot $q_{\text{curr}}^{\text{24D}}$ and goal position $q_{\text{goal}}^{\text{3D}}$.

In the first step, the A* planner uses coarse grid and the map of classes to find a coarse path between current and goal position of the robot (Fig. 10). The heuristic cost estimation from the current node is computed using the Euclidean distance. The adjacent cost between considered and neighboring nodes of the graph is computed using Euclidean distance and spherical variance ω on the elevation map [5]. We also use probabilistic predictor based on the Kernel Density Estimation to estimate if the transition between considered nodes is possible as it was proposed in [5]. However, in contrast to [5], we compute the final cost of the transition taking into account the terrain type. The transition cost for the robot c_{final} is computed as follows:

$$c_{\text{final}} = k_1 \cdot (d + \omega) + k_2 \cdot \sum_{i=0}^{5} (w_i \cdot P(c_i)), \tag{11}$$

where ω is the normalized spherical variance, w_i is the cost assigned to terrain class c_i (safety factor), d is the distance between neighboring nodes, k_1 and k_2 are coefficients which scale the cost to 1. The spherical variance ω is computed for the region of the map which has the same size as the robot's body.

Similarly, to take into account the size of the robot we compute the distance d between neighboring nodes considering the maximal height of the cells below the robot's body. The weight w_i is assigned to each class by the human operator. The weight allows to intuitively determine the preferences for the robot. We divide terrain types according to their physical properties as safe for the robot (asphalt, pavement, concrete), moderately safe (leaves, grass) and objects that should be avoided by the robot (bushes, wall, trees). The A* planner guides the robot through regions which are potentially less risky. The safety factor is also related to the efficiency of walking. Walking on a grass requires more energy than walking on pavement or asphalt due to slippages and a higher risk of robot's fall.

Algorithm 3 Guided-RRT: $q_{\text{curr}}^{\text{24D}}, q_{\text{goal}}^{\text{3D}} \rightarrow \{q_{\text{curr}}^{\text{24D}}, ..., {}_{\text{goal}}^{\text{24D}}\}$

```
while q_R^24D != q_goal^3D do
    A_path = AstarFind(q_curr^24D, q_goal^3D);
    if A_path is NULL then
        return NO PATH;
    end
    while not RRT_SUCCESS do
        q_temp^3D = CreateTempGoal(A_path, rrt_distance);
        RRT_path = RRTConnect({q_curr^24D, q_temp^3D});
        if RRT_path is not NULL then
            RRT_SUCCESS = true;
        else if q_temp^3D = q_goal^3D then
            return NO PATH;
        else
            rrt_distance = rrt_distance + 0.2;
        end
    end
    Execute path from q_curr^24D to q_m^24D;
    q_R^24D = q_m^24D;
end
return {q_curr^24D, ..., q_goal^24D};
```

Fig. 9 Confusion matrices for the SVM (**a**), RT (**b**), and RT+CRF (**c**) classification methods. The overall classification accuracy is 71.5%, 80.0%, and 87,6% for SVM, RT, and RT+CRF, respectively

If the A* planner does not return a path to the goal position we assume that the feasible path for the robot does not exist. In the other case, the algorithm uses the A* path A_{path} to determine the temporary goal q^{3D}_{temp} for the RRT-based planner. The temporal goal is located on the A* path A_{path}. The minimal distance between the current position of the robot and temporary goal should be bigger than rrt_{distance} (in our case 1.2 m). Then, the precise RRT planner determines the full state of the robot in the 24-dimensional space. First, the algorithm computes footholds for the temporal 3-D position q^{3D}_{temp} of the robot. Then, the planner finds the posture of the robot which is statically stable and maximizes the kinematic margin [5].

The path between the current position of the robot q^{24D}_{curr} and temporary goal q^{24D}_{temp} is found by the modified version of the RRT-Connect algorithm [26]. The planner creates two trees. The root of the first tree is located in the current position of the robot. The root of the second tree is located in the position determined in the previous step. The trees are alternately extended to the random direction. If the tree is extended successfully, the algorithm tries to extend the second tree in the direction to the previously added node. The algorithm ends with success when two trees are connected. The algorithm is general and does not assume any specific shape of obstacles. The obtained motion path allows to efficiently climb over obstacles.

We adopted the EXTEND procedure of the RRT-Connect algorithm to find the full state of the robot. The EXTEND procedure checks if the robot can reach a stable position in the given direction. At the beginning of the procedure, the new footholds for the robot are selected using the algorithm from [6]. Then, the posture of the robot is optimized to maximize kinematic range, preserve stability and avoid collisions. We check collisions taking into account full 3D mesh model of the robot. The minimal clearance between the robot's body and the terrain is set to 0.02 m. Then, the algorithm plans the motion of the legs above the obstacles. The planner avoids collisions and keeps the feet inside the workspace of the robot's legs. If the planner successfully finds the transition between the current and the new node, the new node is added to the tree. The new node contains information about the full state of the robot (inclination, distance to the ground and footholds) and transition path for body and legs from the current node to the neighboring node. The details of the RRT-based planner for the six-legged robot are given in [5].

The RRT planner returns the path between the current and temporal position of the robot q^{24D}_{temp}. The robot executes

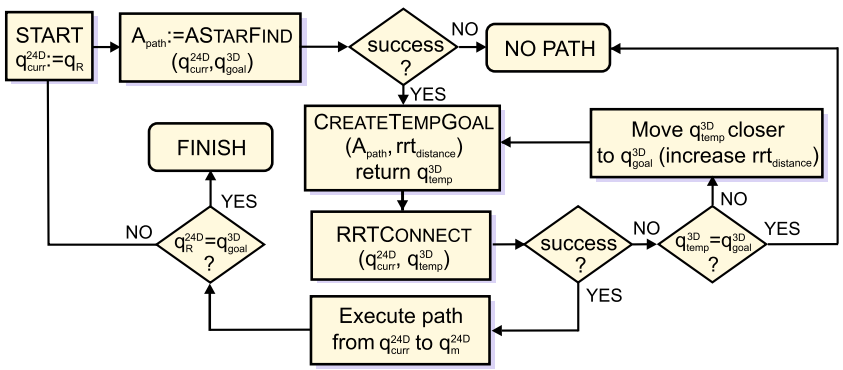

Fig. 10 Block diagram of the guided-RRT algorithm. Note that for the sake of clarity data structures exchanged by the processing modules are not shown explicitly – see text for details

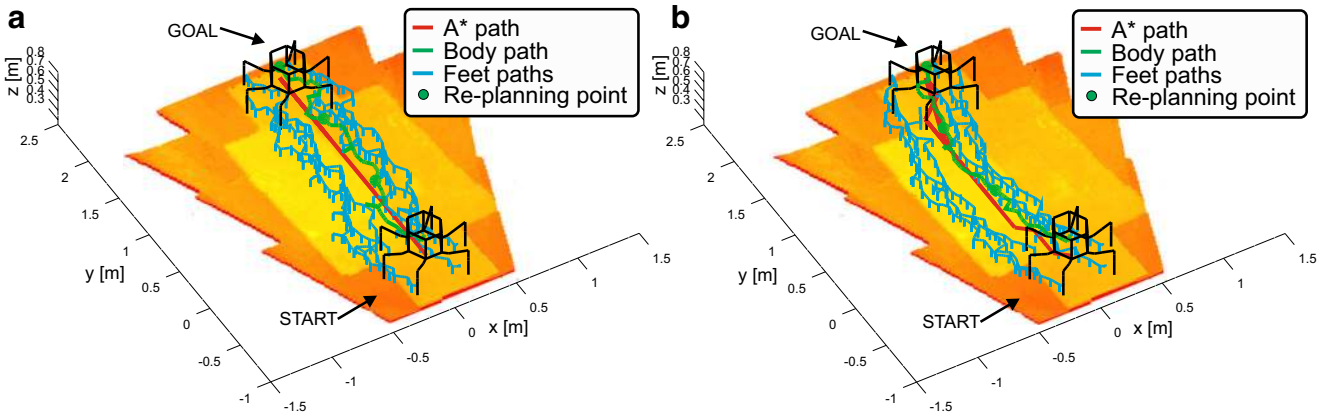

Fig. 11 Results of the comparison experiment with the original version of the guided-RRT algorithm and the proposed version which uses information about terrain type to plan the motion of the robot

the planned path using selected gait (if not mentioned the tripod gait is used in experiments presented in the paper). The robot executes only part of the planned path which is within the area measured by the precise range sensor. In our case, the robot executes 0.8 m of the planned path. If the RRT planner reaches the maximal number of iterations and can't find the path the temporary goal $q_{\text{temp}}^{\text{3D}}$ is moved further along A* path A_{path} from the current position of the robot. In this case, the rrt_{distance} is increased. The planning procedure is repeated until the robot reaches the goal position. The computation of RRT path to the goal position which is 1 m from current position takes about 10 s on flat terrain. The computation time increases with the roughness of the terrain.

6 Results

6.1 Indoor Experiments

First, we performed a set of experiments in the laboratory to show the selected features of the proposed motion planning algorithm.[1] We simulated outdoor environment using five various terrain patches (floor, grass, asphalt, wood, sandstone). The global map of the environment is obtained off-line, before the experiment. We align point clouds obtained from the RGB-D camera (Asus Xtion) moving above the terrain mockup and create the proposed environment model. The elevation maps enhanced by the information about the terrain type is used to plan precise motion of the robot to the goal position. The highest cost of transition (weight w_i) for the A* planner is set to sand-rocks and wood. The cost of transition decreases gradually for grass, floor, and asphalt. We expect that with this configuration the robot will avoid sand-rocks and prefer routes over asphalt. Small detours to grass and floor are also acceptable by the planner.

The first experiment is performed in the environment presented in Fig. 4a. The obtained paths are shown in Fig. 11. We use the environment presented in Fig. 4a to compare the original version of the guided-RRT with the algorithm presented in this article. In Fig. 11a we show the path found by the original version of the guided-RRT algorithm. A* planner in the guided-RRT algorithm takes into account geometrical properties of the terrain. The cost of the transition depends on the roughness of the terrain and distance to the goal position. Because the considered terrain is almost flat the A* planner returns straight path to the goal position. The precise RRT planner follows this path and returns almost straight path. For the same environment, the proposed planner returns path which avoids grass (the cost of transition for the grass is higher than for the asphalt). The robot takes a detour from the shortest path and selects longer but more secure path to the goal (Fig. 11b). In this case, the robot avoids grass and prefers walking over the asphalt.

The second experimental set is presented in Fig. 12. The wooden box between the initial and goal pose of the robot prevents taking the straight path. The robot has to take a longer detour from the straight path to reach the goal position. Two ways are possible: on the asphalt (left of the wooden box) and on the on the floor (right of the wooden box). The robot chooses the path on the asphalt because the cost of the transition for the asphalt is smaller than for the floor. From the beginning of the experiment, the A* planner guides the RRT algorithm above the asphalt (Fig. 12c). We also use this experiment to show that the planner is robust to classification results or non-uniform structure of the terrain (Fig. 12b). For each cell of the map used by the A* planner, we compute the probability of each class. The final cost of the transition depends on the occurrence probability of

[1] Video is available at http://lrm.cie.put.poznan.pl/terrainPlan.mp4

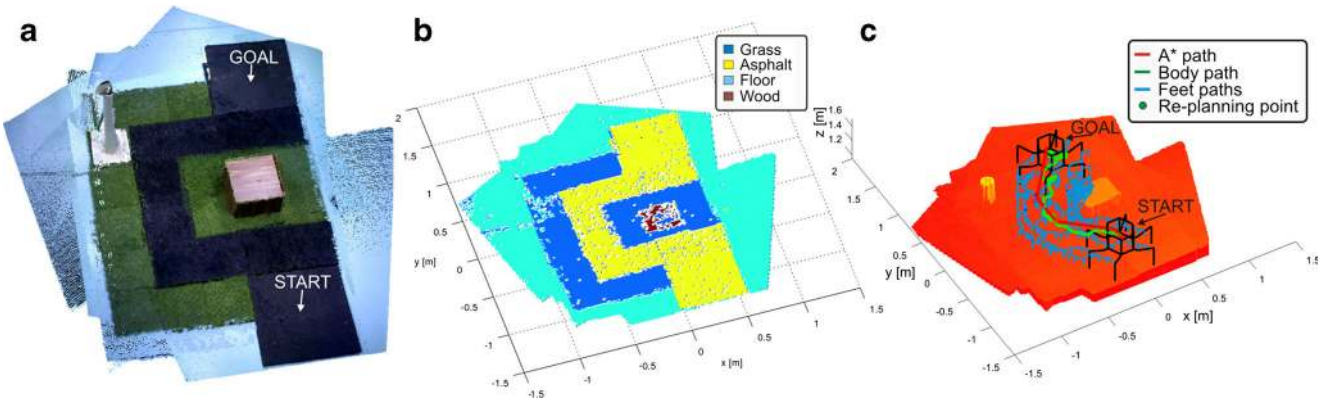

Fig. 12 Second experiment on the terrain mockup: aligned color point cloud (**a**), corresponding class map (**b**) and the obtained path of the robot on the elevation map (**c**)

the terrain type below the robot (11). Thus, even incorrect classification result for some voxels of the OctoMap does not influence significantly the proposed motion planner.

In the third experiment, we added a rough terrain mockup between the initial and goal pose of the robot. The robot chooses the path along the asphalt (Fig. 13) despite the fact that it can climb the mockup [5]. Note that the information about the terrain type is used by the A* planner only. The RRT planner, which is guided by the A* planner, searches for the acceptable path on the regions of the map which are less risky and safe for the robot. The robot still can place its feet on the less preferred terrain types (grass, floor) or move above sand-stone. This approach allows the robot to find stable footholds or avoid collisions with the obstacles (Fig. 13c). Again, despite some misclassified cells (Fig. 13b) the robot plans its path through preferred regions of the map (asphalt).

6.2 Outdoor Experiments

The maps for the outdoor experiments are obtained using Kinect v2 sensor. The sensor is moved freely above the ground and localized using ORB-SLAM2 [31]. The initial pose of the sensor in ORB-SLAM2 is set to be identical to the global reference frame. The proposed model of the environment is obtained using procedures presented in chapter 3.3. Finally, we plan the motion from the initial to the goal position of the robot using the terrain-aware guided-RRT algorithm. We recognize eight terrain categories of terrain. The lowest transition cost is set to asphalt and pavement and higher cost is set for grass and fallen leaves laying on the ground. The highest transition cost, which forces the robot to avoid this type of obstacles, is set for bushes, trees, walls and concrete structures.

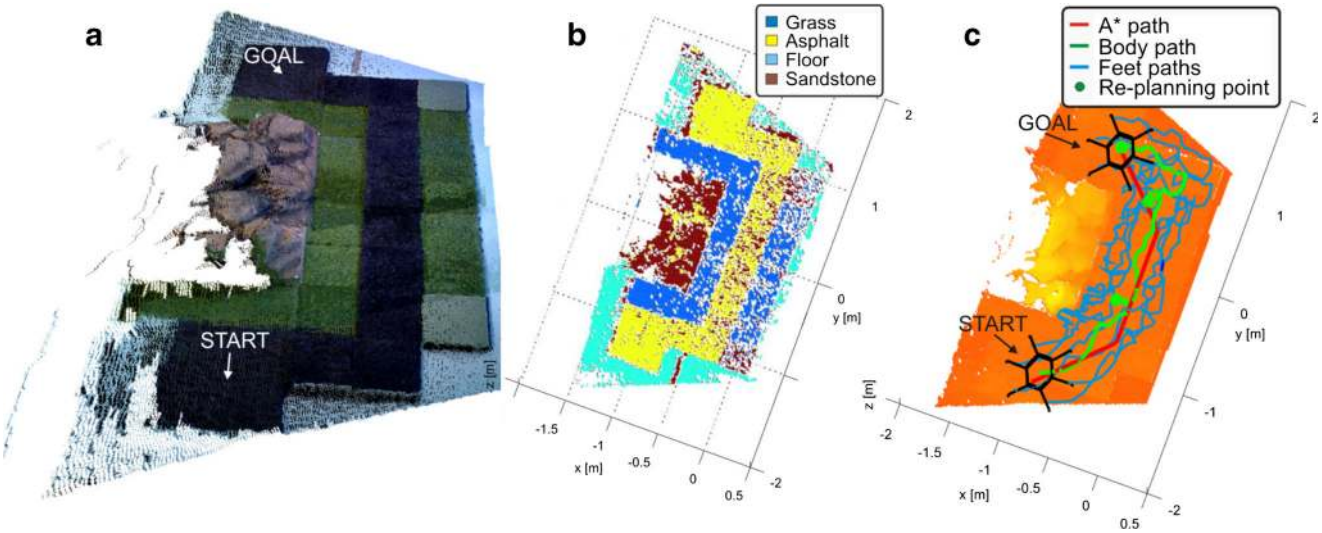

Fig. 13 Third experiment on the terrain mockup: aligned color point clouds (**a**), corresponding class map (**b**) and the obtained path of the robot (**c**)

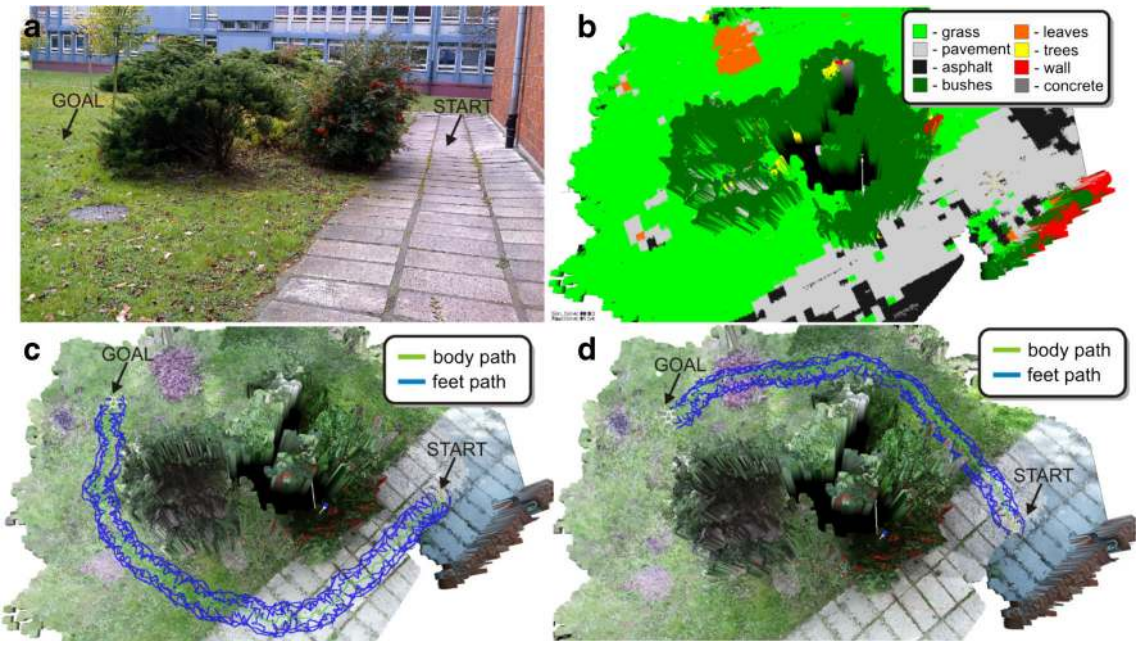

Fig. 14 Experimental set (**a**), classification results on the elevation map (**b**) and motion planning results when the robot prefers pavement over the grass (**c**) and when the robot uses geometric properties of the environment only to plan its motion (**d**)

The results of the first experiment are presented in Fig. 14. The experimental set is presented in Fig. 14a. The classification results are shown in Fig. 14b. we don't have the ground truth map for the classification (the map contains hundreds of thousands of cells that should be labeled manually) so we can assess visually the classification results. The classifier can easily distinguish between grass, bushes, and leaves. Some cells of the pavement are incorrectly classified as asphalt because they are visually similar. This is not a problem for the planner because it uses the probability of the terrain type in the considered cell and because the pavement and asphalt have similar physical properties. The robot can safely walk on both terrain types.

The final path obtained from guided-RRT planner is presented in Fig. 14c. The close-up view on the planned trajectory at the start and goal pose of the robot is presented in Fig. 15. The planner returns positions of the supporting feet on the ground, the trajectory of feet above obstacles during swing phase, the inclination of the platform and distance of the body to the ground. Each position of the robot is collision-free and statically stable. The robot walks on the pavement as long as possible. The planner minimizes the length of the path which goes above the grass because it is a less preferable type of the terrain for the robot. We compare the computed path to the path obtained from the original version of the guided-RRT algorithm [5]. The results of the original guided-RRT algorithm, which takes into account geometrical properties of the environment, are presented in Fig. 14d. The length of the obtained path is 11.84 m (Fig. 14d) while the length of the path obtained from the proposed planner is 14.34 m (Fig. 14c). The proposed planner returns longer but much safer path for the robot. The robot voluntarily takes the longer path on the pavement and avoids more risky grass and leaves. The planning time for the path presented in Fig. 14c is 212 s, and for the path presented in Fig. 14d is 223 s.

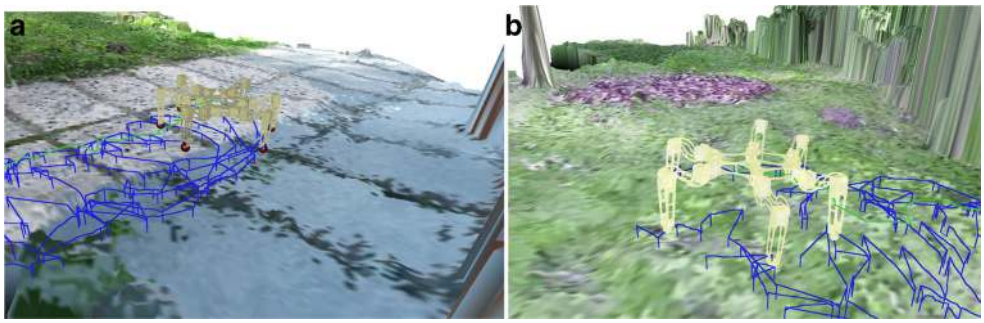

Fig. 15 Close-up view on the planned trajectory at the start (**a**) and goal pose of the robot (**b**)

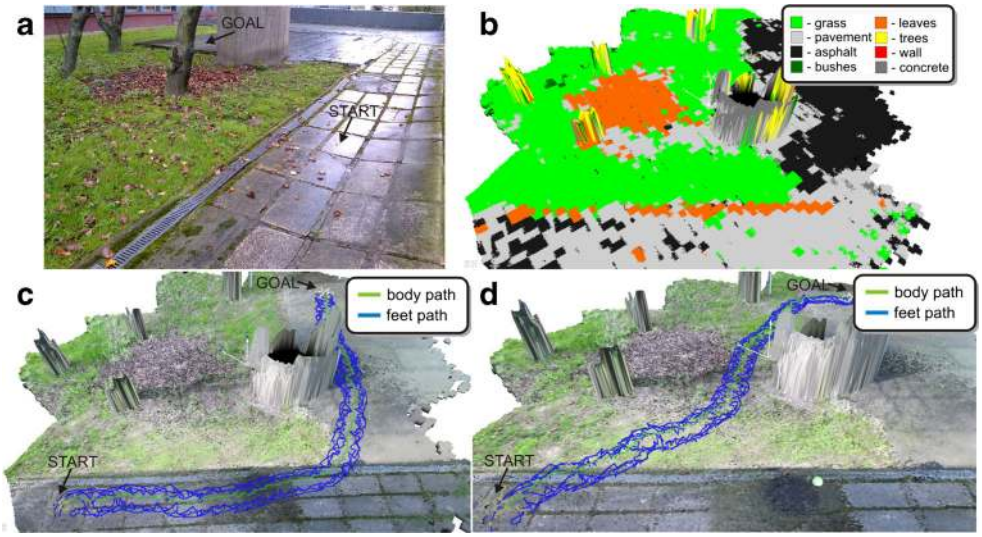

Fig. 16 Experimental set (**a**), classification results on the elevation map (**b**) and motion planning results when the robot prefers pavement over the grass (**c**) and when the robot prefers grass over the pavement and asphalt (**d**)

The second experiment was performed on the terrain presented in Fig. 16a. The proposed classifier can properly identify pavement, asphalt, and grass (Fig. 16b). Also, the robot is aware of trees, leaves and concrete construction. The path found by the planner is presented in Fig. 16c. The robot walks on the preferred terrain types: pavement and asphalt and avoids grass and non-traversable obstacles. The precise RRT based-planner, which is computationally more demanding than A* planner, does no explore risky areas (leaves, trees, concrete construction). The A* planner guides the RRT method to the areas which are potentially safer for the robot. The length of the obtained path is 15.23 m. We also modified the weights related to the terrain types to verify the output from the planner. The results are presented in Fig. 16d. In this case, the robot has a different strategy. The weight related to the grass is the smallest. We also increased weight related to the pavement and asphalt. The modified strategy gives shorter path (13.45 m) but potentially more risky for the robot. The planning time for the path presented in Fig. 16c is 379 s, and for the path presented in Fig. 16d is 202 s.

The experimental set for the last outdoor experiment is presented in Fig. 17a. The robot can identify the obstacles and terrain types in the environment (Fig. 17b) and plan its path to avoid contact with unwanted terrain types (Fig. 17c). We use this experimental set to show the flexibility of the proposed algorithm. The results are presented in Fig. 17d. We modified the weights related to each class so the A*

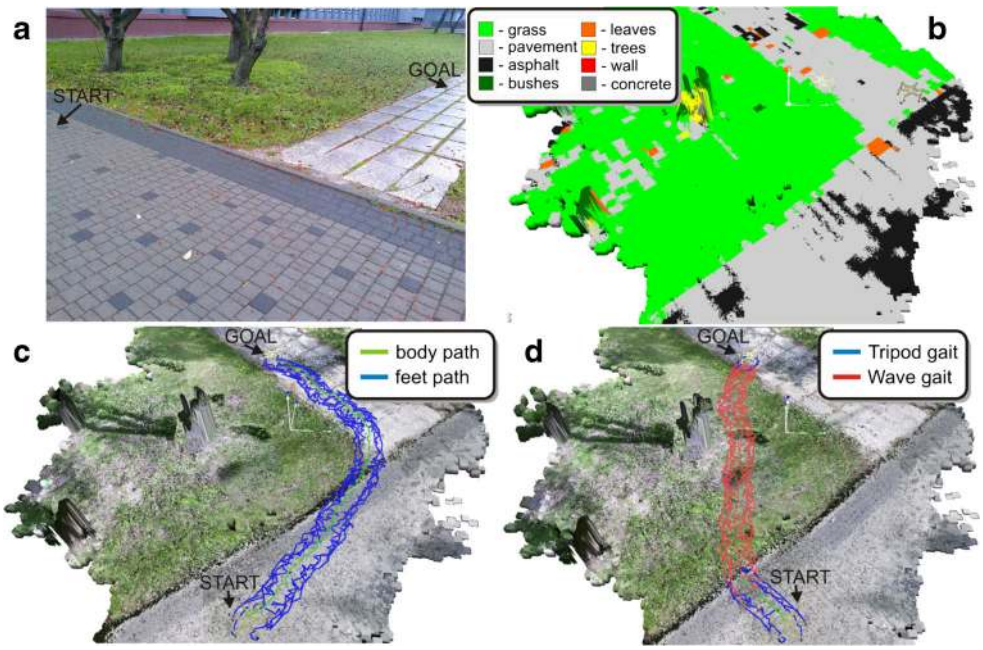

Fig. 17 Experimental set (**a**), classification results on the elevation map (**b**) and motion planning results when the robot prefers pavement over the grass (**c**) and when the robot prefers grass and modifies gait type according to the terrain type (**d**)

Table 1 Comparison between various motion planning strategies: A – proposed method, B – method from [5], C – proposed method with modified planning preferrences (robot preferrs grass over pavement and asphalt). Bold values correspond to the terrain type preferred by the robot with method A, B, and C

	Exp. 1		Exp. 2			Exp. 3		
Method	A	B	A	B	C	A	B	C
Length [m]	14.34	11.84	15.23	13.17	13.45	10.89	9.69	9.29
Planning time [s]	212	223	379	418	202	99	108	106
Sph. var. ω	2.5e-7	6.3e-8	1.1e-8	1.4e-8	1.0e-8	1.5e-8	1.4e-8	1.4e-8
P(grass)	0.54	**0.78**	0.04	**0.59**	**0.69**	0.01	**0.78**	**0.79**
P(pavement)	**0.41**	0.12	**0.58**	0.26	0.11	**0.89**	0.17	0.17
P(asphalt)	0.05	0.0	0.35	0.11	0.15	0.07	0.04	0.02

planner prefers grass. However, in this case, the planner is aware of the higher cost of transition of the grass in comparison to the pavement and modifies the walking pattern. On the pavement, the robot uses tripod gait which is the fastest statically stable gait for a hexapod robot, but less stable. When the robot walks on the grass it changes the gait to wave gait to increase the support polygon and stability margin. With this strategy, the robot can switch between various gaits when walking on various terrain types. Moreover, the robot can ignore the foothold selection and precise path planning on compliant terrain types, e.g. high grass or leaves. When walking on this type of the terrain the geometrical properties are hardly related to the stable support for robot's leg. In this case, the robot should use behavioral approach but plan this strategy in advance. The planning time for the path presented in Fig. 17c is 99 s, and for the path presented in Fig. 17d is 106 s.

The results from the outdoor experiments are summarized in Table 1. We compare the proposed method (column A in Table 1) with the guided-RRT planner which uses geometric information only [5] (column B in Table 1). Additionally, we show results for the planner with modified terrain preferences (robot prefers grass over pavement and asphalt – column C in Table 1). For each experiment, we show the length of the obtained path, the planning time, average spherical variance ω and the average probability of three selected terrain classes (grass, pavement, asphalt) along the obtained path.

When we compare the average spherical variance (which is related to the roughness of the terrain) for each experiment we can note that the value for the path along the grass and for the pavement does not differ significantly. For some experiments, the roughness of the pavement is even larger than for the lawn area. This is related to the properties of the perception system. The uncertainty of the depth measurements from the RGB-D sensor does not allow to properly distinguish between some terrain types on the basis of their geometry. The semantic information which we added to the elevation map allows the planner to choose a path over the preferred terrain, in spite of the fact that the roughness measure provides a vague support for the computation of the cost function. The statistics of the terrain roughness shows that the qualitative semantic information helps to cope with the unavoidable uncertainty of the geometric terrain map. In Table 1 we also compare the average probability of the specified classes. For the method presented in the article, the probability of pavement and asphalt is maximized. In experiments 2 and 3 the robot spends more than 90% of the time on the pavement and asphalt. In the first experiment, the average probability of the pavement along the grass is 0.41. The probability decreases to 0.12 when we modify the preferences of the planner. When the robot uses geometric features for planning the probability of grass increases from 0.54 to 0.78.

7 Conclusions

Extensive experiments in both a controlled lab environment and real outdoor settings demonstrate that it is possible to determine the basic semantics of a natural environment, and then to use the semantic labeling to enhance motion planning capabilities of a legged robot. Specifically, the ability to distinguish between a sufficient number of classes (eight labels in the outdoor experiments) to describe the semantics of a typical urban outdoor scene was demonstrated. Then, the influence of the semantic interpretation of the particular areas perceived by the robot on the coarse path planning results was shown. The robot was forced to consider different terrain classes as preferable, which resulted in different paths with the same elevation map. We consider this as a first step toward terrain perception that is similar to the way animals or even human beings perceive a natural environment.

Although the main building blocks of the presented system have been already used in our previous research [3, 5, 47] or are known from the literature and available to the community, as the OctoMap [21], we demonstrate

here how they can be used to obtain a complete architecture for perception-based motion planning. In particular, we contribute the following new elements:

- The use of the octree map structure for semantic labeling of natural environments from RGB-D data. Making an informed choice of the features and the classifier (motivated by our previous experience) we are able to classify the voxels on relatively large scenes obtaining results that are consistent with the meaning a human being would ascribe to particular areas of these scenes. Unlike many terrain classification systems that use raw images adopting semantic image segmentation methods well known in computer vision, we implement classification directly on the octree map structure. This approach makes possible to consider also geometry-related features in classification.
- The new terrain mapping architecture, which combines the OctoMap and the elevation grid. The octree structure is used for efficient 3-D data registration, ensuring proper object representation and sufficient statistics for classification. The elevation map serves as a projection of those aspects of the 3-D model that are essential to the guided-RRT planner, ensuring a compact data structure and quick access to the information.
- The coarse path planner, based on the A* algorithm, but extended by the ability to take into account the semantic labels in the elevation grid. This method is engineered to be maximally flexible – we ascribe weights to the semantic categories that result in different behaviors of the coarse planner (cautious or aggressive terrain exploration). Moreover, we compute the probability of semantic labeling for the coarse grid cells used by the path planner, which results in an increased tolerance to isolated, wrongly classifies voxels. Owing to this concept we obtain satisfying results even without the use of the CRF-based reasoning, which improves classification results, but slows down the whole classification process. To the best of our knowledge, we are the first who demonstrate full motion planning for a legged robot that considers 3-D semantic labeling in a natural terrain in contrast to reactive controllers [8, 40] that employ semantic labels only to avoid specific terrain types.

Obviously, the presented system can be improved and extended in a number of directions. In the OctoMap we have terrain classes detected at the granulation of the 1.5×1.5 cm voxels. This information is in fact not used in the current implementation, but may be useful for an enhanced foothold selection procedure, and can assist in the implementation of a gait supported by reflexive behaviors. Also, tighter integration of the semantic information within the RRT-Connect planner is possible, as the "meaning" of the given area may influence the probability of sampling the robot configurations in that area. Whereas the presented experiments relied on RGB-D data, we expect similar performance with a good quality stereo camera, which may be more suitable for outdoor operations.

Acknowledgements This research is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780883. We thank Szymon Bartoszyk and Patryk Kasprzak who worked on the environment model for the indoor experiments and provided the initial version of the terrain classifier.

References

1. Bajracharya, M., Ma, J., Malchano, M., Perkins, A., Rizzi, A., Matthies, L.: High fidelity day/night stereo mapping with vegetation and negative obstacle detection for vision-in-the-loop walking. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3663–3670. Tokyo (2013)
2. Bartoszyk, S., Kasprzak, P., Belter, D.: Terrain-aware motion planning for a walking robot. In: Proceedings of International Workshop on Robot Motion and Control, pp. 29–34. Wasowo (2017)
3. Belter, D., Łabęcki, P., Fankhauser, P., Siegwart, R.: RGB-D, terrain perception and dense mapping for legged robots. Int. J. Appl. Math. Comput. Sci. **26**(1), 81–97 (2016)
4. Belter, D., Łabęcki, P., Skrzypczyński, P.: Estimating terrain elevation maps from sparse and uncertain multi-sensor data. In: Proceedings IEEE International Conference on Robotics and Biomimetics, pp. 715–722. Guangzhou (2012)
5. Belter, D., Łabęcki, P., Skrzypczyński, P.: Adaptive motion planning for autonomous rough terrain traversal with a walking robot. J. Field Robot. **33**(3), 337–370 (2016)
6. Belter, D., Skrzypczyński, P.: Rough terrain mapping and classification for foothold selection in a walking robot. J. Field Robot. **28**(4), 497–528 (2011)
7. Belter, D., Walas, K.: A compact walking robot – flexible research and development platform. In: Szewczyk, R., et al. (eds.) Recent Advances in Automation, Robotics and Measuring Techniques, AISC, vol. 267, pp. 343–352 (2014)
8. Best, G., Moghadam, P., Kottege, N., Kleeman, L.: Terrain classification using a hexapod robot. In: Proceedings of the Australasian Conference on Robotics and Automation (2013)
9. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Fifth Annual Workshop on Computational Learning Theory, COLT '92, pp. 144–152 (1992)
10. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)

11. Chang, C.C., Lin, C.J.: libSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(27), 1–27 (2011)
12. Chetan, J., Krishna, M., Jawahar, C.V.: Fast and spatially-smooth terrain classification using monocular camera. In: International Conference on Pattern Recognition, pp. 4060–4063. Istanbul (2010)
13. Chilian, A., Hirschmüller, H.: Stereo camera based navigation of mobile robots on rough terrain. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4571–4576 (2009)
14. Droeschel, D., Schwarz, M., Behnke, S.: Continuous mapping and localization for autonomous navigation in rough terrain using a 3D laser scanner. Robot. Auton. Syst. **88**, 104–115 (2017)
15. Dryanovski, I., Morris, W., Xiao, J.: Multi-volume occupancy grids: An efficient probabilistic 3D mapping model for micro aerial vehicles. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1553–1559 (2010)
16. Fankhauser, P., Bloesch, M., Rodriguez, D., Kaestner, R., Hutter, M., Siegwart, R.: Kinect v2 for mobile robot navigation: Evaluation and modeling. In: Proceedings of International Conference on Advanced Robotics, pp. 388–394. Istanbul (2015)
17. Giguére, P., Dudek, G., Saunderson, S., Prahacs, C.: Environment identification for a running robot using inertial and actuator cues. In: Robotics: Science and Systems (2006)
18. Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., LeCun, Y.: Learning long-range vision for autonomous off-road driving. J. Field Robot. **26**(2), 120–144 (2009)
19. Hauser, K., Bretl, T., Latombe, J.C., Harada, K., Wilcox, B.: Motion planning for legged robots on varied terrain. Int. J. Robot. Res. **27**(11-12), 1325–1349 (2008)
20. Hoepflinger, M.A., Remy, C.D., Hutter, M., Spinello, L., Siegwart, R.: Haptic terrain classification for legged robots. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 2828–2833 (2010)
21. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Auton. Robot. **34**(3), 189–206 (2013)
22. Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
23. Kolter, J.Z., Rodgers, M.P., Ng, A.Y.: A control architecture for quadruped locomotion over rough terrain. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 811–818 (2008)
24. Kraft, M., Nowicki, M., Schmidt, A., Fularz, M., Skrzypczyński, P.: Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect. Mach. Vis. Appl. **28**(1), 61–74 (2017)
25. Krotkov, E., Hoffman, R.: Terrain mapping for a walking planetary rover. IEEE Trans. Robot. Autom. **10**(6), 728–739 (1994)
26. Kuffner, J., LaValle, S.: RRT-Connect: An efficient approach to single-query path planning. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 995–1001 (2000)
27. Laible, S., Khan, Y., Zell, A.: Terrain classification with conditional random ields on fused 3D lidar and camera data. In: Proceedings of European Conference on Mobile Robots, pp. 172–177. Barcelona (2013)
28. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: Progress and prospects. In: Donald, B.R. (ed.) Algorithmic and Computational Robotics: New Directions, pp. 293–308. A. K. Peters/CRC Press (2001)
29. Maturana, D., Chou, P., Uenoyama, M., Scherer, S.: Real-time semantic mapping for autonomous off-road navigation. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics, SPAR, vol. 5, pp. 335–350. Springer (2018)
30. Mrva, J., Faigl, J.: Tactile sensing with servo drives feedback only for blind hexapod walking robot. In: Proceedings of International Workshop on Robot Motion and Control, pp. 240–245. Poznań (2015)
31. Mur-Artal, R., Tardos, J.D.: ORB-SLAM2: an opensource SLAM system for monocular, stereo and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)
32. Nowicki, M., Belter, D., Kostusiak, A., Cízek, P., Faigl, J., Skrzypczyński, P.: An experimental study on feature-based SLAM for multi-legged robots with RGB-D sensors. Indus. Robot: Int. J. **44**(4), 428–441 (2017)
33. Ojeda, L., Borenstein, J., Witus, G., Karlsen, R.: Terrain characterization and classification with a mobile robot. J. Field Robot. **23**(2), 103–122 (2006)
34. Papadakis, P.: Terrain traversability analysis methods for unmanned ground vehicles: a survey. Eng. Appl. Artif. Intel. **26**(4), 1373–1385 (2013)
35. Pfaff, P., Triebel, R., Burgard, W.: An efficient extension to elevation maps for outdoor terrain mapping and loop closing. Int. J. Robot. Res. **26**(2), 217–230 (2007)
36. Roennau, A., Kerscher, T., Ziegenmeyer, M., Marius, J., Zölner, J.M., Dillmann, R.: Adaptation of a six-legged walking robot to its local environment. In: Kozłowski, K. (ed.) Robot Motion and Control 2009, LNCIS, vol. 396, pp. 155–164. Springer (2009)
37. Rusu, R.B., Sundaresan, A., Morisset, B., Hauser, K., Agrawal, M., Latombe, J.C., Beetz, M.: Leaving flatland: efficient real-time 3D perception and motion planning. J. Field Robot. **26**(10), 841–862 (2009)
38. Sanctis, L., Garrido, S., Moreno, L., Blanco, D.: Outdoor motion planning using fast marching. In: Tosun, O. et al. (eds.) Mobile Robotics: Solutions and Challenges, pp. 1071–1080. World Scientific, Singapore (2009)
39. Satzinger, B., Lau, C., Byl, M., Byl, K.: Tractable locomotion planning for RoboSimian. Int. J. Robot. Res. **34**(13), 1541–1558 (2015)
40. Stejskal, M., Mrva, J., Faigl, J.: Road following with blind crawling robot. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 3612–3617 (2016)
41. Thrun, S., Montemerlo, M., Aron, A.: Probabilistic terrain analysis for high-speed desert driving. In: Robotics: Science and Systems. Philadelphia (2006)
42. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4), 376–380 (1991)
43. Valada, A., Oliveira, G., Brox, T., Burgard, W.: Towards robust semantic segmentation using deep fusion. In: RSS Workshop on Limits and Potentials of Deep Learning in Robotics. Ann Arbor (2016)
44. Vonasek, V., Faigl, J., Krajnik, T., Preucil, L.: RRT-path – a guided rapidly exploring random tree. In: Kozłowski, K. (ed.) Robot Motion and Control 2009, LNCIS, vol. 396, pp. 307–316. Springer (2009)
45. Walas, K.: Terrain classification and negotiation with a walking robot. J. Intell. Robot. Syst. **78**(3), 401–423 (2015)
46. Wermelinger, M., Fankhauser, P., Diethelm, R., Krüsi, P.A., Siegwart, R., Hutter, M.: Navigation planning for legged robots in challenging terrain. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1184–1189 (2016)
47. Wietrzykowski, J., Skrzypczyński, P.: Terrain classification for autonomous navigation in public urban areas. In: Silva, M. et al. (eds.) Human-Centric Robotics, pp. 319–326. World-Scientific (2017)
48. Wooden, D., Malchano, M., Blankespoor, K., Howardy, A., Rizzi, A., Raibert, M.: Autonomous navigation for BigDog. In:

Proceedings of IEEE International Conference on Robotics and Automation, pp. 4736–4741 (2010)
49. Wurm, K.M., Stachniss, C., Kümmerle, R., Burgard, W.: Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1217–1222. St. Louis (2009)
50. Ye, C., Borenstein, J.: A novel filter for terrain mapping with laser rangefinders. IEEE Trans. Robot. Autom. **20**(5), 913–921 (2004)
51. Zhong, C., Liu, S., Zhang, B., Lu, Q., Wang, J., Wu, Q., Gao, F.: A fast on-line global path planning algorithm based on regionalized roadmap for robot navigation. IFAC-PapersOnLine **50**(1), 319–324 (2017). 20th IFAC World Congress
52. Zucker, M., Ratliff, N., Stolle, M., Chestnutt, J., Bagnell, J.A., Atkeson, C.G., Kuffner, J.: Optimization and learning for rough terrain legged locomotion. Int. J. Robot. Res. **30**(2), 175–191 (2011)

Dominik Belter graduated from Poznan University of Technology (2007). He received PhD degree in robotics from the same University in 2012. Since 2012, he has been an Assistant Professor at the Institute of Control and Information Engineering of the Poznan University of Technology. Dominik Belter is the author or co-author of over 60 technical papers in the fields of robotics and computer science. His research interests include walking robots, machine learning, vision, robot manipulation and soft computing.

Jan Wietrzykowski graduated from Poznan University of Technology in 2015. He received BSc and MSc in Automatic Control and Robotics from the same university in 2014 and 2015, respectively. Since 2015 he is a PhD student at the Faculty of Electrical Engineering. In 2016 he became a research assistant at the Institute of Control and Information Engineering. His is author or coauthor of 13 technical papers in the area of robotics and machine learning. His current research interests include robotic global localization, machine learning, and simultaneous localization and mapping.

Piotr Skrzypczyński received the Ph.D. and D.Sc. degrees in Robotics from Poznań University of Technology (PUT) in 1997 and 2007, respectively. Since 2010 he is an associate professor at the Institute of Control, Robotics and Information Engineering (ICRIE) of PUT, and head of the Mobile Robotics Laboratory at ICRIE. He is author or co-author of over 160 technical papers in robotics and computer science. His current research interests include: autonomous mobile robots, simultaneous localization and mapping, multisensor fusion, machine learning, and computational intelligence in robotics.