Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing

Rudolph Triebel Patrick Pfaff Wolfram Burgard
University of Freiburg
Georges-Koehler-Allee 79, 79110 Freiburg, Germany
{*triebel,pfaff,burgard*}*@informatik.uni-freiburg.de*

***Abstract*— To operate outdoors or on non-flat surfaces, mobile robots need appropriate data structures that provide a compact representation of the environment and at the same time support important tasks such as path planning and localization. One such representation that has been frequently used in the past are elevation maps which store in each cell of a discrete grid the height of the surface in the corresponding area. Whereas elevation maps provide a compact representation, they lack the ability to represent vertical structures or even multiple levels. In this paper, we propose a new representation denoted as multi-level surface maps (MLS maps). Our approach allows to store multiple surfaces in each cell of the grid. This enables a mobile robot to model environments with structures like bridges, underpasses, buildings or mines. Additionally, they allow to represent vertical structures. Throughout this paper we present algorithms for updating these maps based on sensory input, to match maps calculated from two different scans, and to solve the loop-closing problem given such maps. Experiments carried out with a real robot in an outdoor environment demonstrate that our approach is well-suited for representing large-scale outdoor environments.**

I. I

The problem of learning maps with mobile robots has been intensively studied in the past. Especially in situations, in which robots are deployed outdoors or in environments with non-flat surfaces, the ability to traverse specific areas of interest needs to be known accurately. In this context, geometric representations have become popular. However, full three-dimensional models typically have high computational demands that prevent them from being directly applicable in large-scale environment.

One popular approach to overcome this problem are elevation maps, which apply a $2\frac{1}{2}$-dimensional representation. An elevation map consists of a two-dimensional grid in which each cell stores the height of the territory. Whereas this approach leads to a substantial reduction of the memory requirements, it can be problematic when a robot has to utilize these maps for navigation or when it has to register two different maps in order to integrate them. Even extensions of elevation maps that allow to handle vertical or overhanging objects [20] can only be applied to environments with single surfaces. For example, a robot that uses extended elevation maps cannot plan a path under and at the same time over a bridge.

In this paper, we propose an extension of elevation maps towards multiple surfaces. These so-called multi-level surface maps (MLS maps) offer the opportunity to model environments with more than one traversable level. Whereas the knowledge about horizontal surfaces is well suited to support traversability analysis and path planning, it provides only weak support for localization of the vehicle or registration of different maps. Modeling only the surfaces means that vertical structures, which are frequently perceived by ground based vehicles cannot be used to support localization and registration. To avoid this problem, our MLS maps additionally represent intervals corresponding to vertical objects in the environment. The advantage of this approach is that they can be compactly stored and at the same time can be used as features that support the data association problem during the alignment of maps.

As a motivating example, consider the three-dimensional data points shown in Figure 1(a). They have been acquired with a mobile robot standing in front of a bridge. The resulting elevation map, which is computed from averaging over all scan points that fall into a cell of a horizontal grid (given a vertical projection), is depicted in Figure 1(b). As can be seen from the figure, the underpass has completely disappeared and the elevation map shows a non-traversable object. Additionally, when the environment contains vertical structures, we typically obtain varying average height values depending on how much of this vertical structure is contained in a scan. When two such elevation maps need to be aligned, such errors can lead to imperfect registrations. The corresponding map calculated with the approach of Pfaff *et al.* [20], which allows to deal with vertical and overhanging objects, is shown in Figure 1(c). Obviously, this approach correctly represents the underpass and allows the robot to move through the tunnel, but it makes it impossible to travel over the bridge. The corresponding MLS map is shown in Figure 1(d). As can be seen, this representation can correctly represent the individual surfaces. It also shows that vertical structures are correctly represented.

This paper is organized as follows. After discussing of related work in the following section, we will describe our approach of MLS maps in Section III. Section IV then introduce our constraint-based pose estimation procedure for calculating consistent maps. Finally, we present experimental results in Section V.

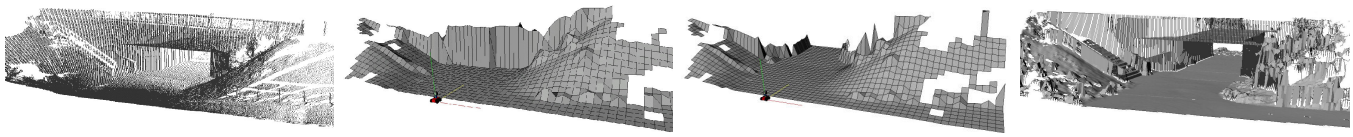

Fig. 1. Scan (point set) of a bridge (a), standard elevation map computed from this data set (b), extended elevation map which correctly represents the underpass under the bridge (c), and multi level surface map that correctly represents the height of the vertical objects (d)

II. R W

The problem of learning three-dimensional representations has been studied intensively in the past. Recently, several techniques for acquiring three-dimensional data with 2d range scanners installed on a mobile robot have been developed. A popular approach is to use multiple scanners that point towards different directions [7], [25], [26]. An alternative is to use pan/tilt devices that sweep the range scanner in an oscillating way [15], [22]. More recently, techniques for rotating 2d range scanners have been developed [10], [30].

Many authors have studied the acquisition of three-dimensional maps from vehicles that are assumed to operate on a flat surface. For example, Thrun *et al.* [25] present an approach that employs two 2d range scanners for constructing volumetric maps. Whereas the first is oriented horizontally and is used for localization, the second points towards the ceiling and is applied for acquiring 3d point clouds. Früh and Zakhor [5] apply a similar idea to the problem of learning large-scale models of outdoor environments. Their approach combines laser, vision, and aerial images. Furthermore, several authors have considered the problem of simultaneous mapping and localization (SLAM) in an outdoor environment [4], [6], [27]. These techniques extract landmarks from range data and calculate the map as well as the pose of the vehicles based on these landmarks. Our approach described in this paper does not rely on the assumption that the surface is flat. It uses m MLS maps to capture the three-dimensional structure of the environment and is able to estimate the pose of the robot in all six degrees of freedom.

One of the most popular representations are raw data points or triangle meshes [1], [12], [22], [28]. Whereas these models are highly accurate and can easily be textured, their disadvantage lies in the huge memory requirement, which grows linearly in the number of scans taken. Accordingly, several authors have studied techniques for simplifying point clouds by piecewise linear approximations. For example, Hähnel *et al.* [7] use a region growing technique to identify planes. Liu *et al.* [13] as well as Martin and Thrun [14] apply the EM algorithm to cluster range scans into planes. Recently, Triebel *et al.* [29] proposed a hierarchical version that takes into account the parallelism of the planes during the clustering procedure. An alternative is to use three-dimensional grids [16] or tree-based representations [23], which only grow linearly in the size of the environment. According to the non-planar structure of natural outdoor environments and the space requirements for large-scale environments, the applicability of these representations and approximations in such environments is limited.

In order to avoid the complexity of full three-dimensional maps, several researchers have considered elevation maps as an attractive alternative. The key idea underlying elevation maps is to store the $2\frac{1}{2}$-dimensional height information of the terrain in a two-dimensional grid. Bares *et al.* [2] as well as Hebert *et al.* [8] use elevation maps to represent the environment of a legged robot. They extract points with high surface curvatures and match these features to align maps constructed from consecutive range scans. Parra *et al.* [21] represent the ground floor by elevation maps and use stereo vision to detect and track objects on the floor. Singh and Kelly [24] extract elevation maps from laser range data and use these maps for navigating an all-terrain vehicle. Ye and Borenstein [31] propose an algorithm to acquire elevation maps with a moving vehicle carrying a tilted laser range scanner. They propose special filtering algorithms to eliminate measurement errors or noise resulting from the scanner and the motions of the vehicle. Lacroix et al. [11] extract elevation maps from stereo images. Hygounenc *et al.* [9] construct elevation maps with an autonomous blimp using 3d stereo vision. They propose an algorithm to track landmarks and to match local elevation maps using these landmarks. Olson [18] describes a probabilistic localization algorithm for a planetary rover that uses elevation maps for terrain modeling. In this paper, we present MLS maps, which can be regarded as an extension to elevation maps. Our approach allows to compactly represent multiple surfaces in the environment as well as vertical structures. This allow a mobile robot to correctly deal with multiple traversable surfaces in the environment as they, for example, occur in the context of bridges. Our approach also represents an extension of the work by Pfaff *et al.* [20]. Whereas this approach allows to deal with vertical and overhanging objects in elevation maps, it lacks the ability to represent multiple surfaces.

Recently, several authors have studied the problem of simultaneous localization and mapping in the context of mobile robots operating on a non-flat surface. For example, Davison *et al.* [3] presented an approach to vision based SLAM with a single camera moving freely through the environment. This approach uses an extended Kalman Filter to simultaneously update the pose of the camera and the 3d feature points extracted from the camera images. More recently, Nüchter *et al.* [17] developed a mobile robot that builds accurate three-dimensional models with a mobile robot. In this approach, loop closing is achieved by uniformly distributing the estimated odometry error over the poses in a loop. In contrast, the work described here employs MLS maps and globally optimizes the

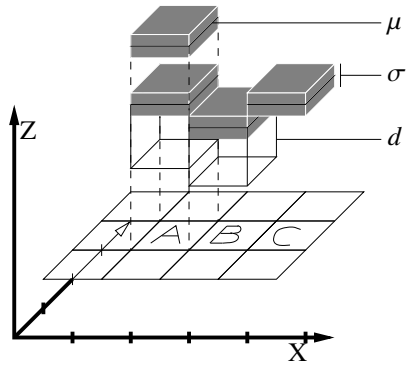

Fig. 2. Example of different cells in an MLS Map. Cells can have many surface patches (cell A), represented by the mean and the variance of the measured height. Each surface patch can have a depth, like the patch in cell B. Flat objects are represented by patches with depth 0, as shown by the patch in cell C.

pose estimates for calculating consistent maps. To achieve this, we efficiently solve the data association problem that occurs, when two maps with different estimates about the surface levels have to be matched or combined.

III. M L S M

Suppose we are given a set of N 3D scan points $C = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ with $\mathbf{p}_i \in \mathbb{R}^3$, and a set of variances $\{\sigma_1^2, \ldots, \sigma_N^2\}$. Here, the variance σ_i^2 expresses the uncertainty in the range measurement from which point $\mathbf{p}_i$ was computed. This uncertainty grows with the measured distance. In the following, we assume that the uncertainty is equal in all three dimensions, in particular the variance in height is assumed to be identical to σ_i^2. Although this assumption is often violated in real environments, this approximation turned out to be viable for our applications. Regarding this, we define a measurement z as a pair $(\mathbf{p}, \sigma^2)$ of a 3D point and a variance.

A. Map Representation

A multi-level surface map (MLS map) consists of a 2D grid of variable size where each cell c_{ij}, $i, j \in \mathbb{Z}$ in the grid stores a list of *surface patches* $P_{ij}^1, \ldots, P_{ij}^K$. A surface patch in this context is represented as the mean μ_{ij}^k and variance σ_{ij}^k of the measured heights at the position of the cell c_{ij} in the map. Each surface patch in a cell reflects the possibility of traversing the 3D environment at the height given by the mean μ_{ij}^k, while the uncertainty of this height is represented by the variance σ_{ij}^k. Throughout this paper, we assume that the error in the height underlies a Gaussian distribution, therefore we will use the terms *surface patch* and *Gaussian in a cell* interchangeably.

In addition to the mean and variance of a surface patch, we also store a *depth value d* for each patch. This depth value reflects the fact that a surface patch can be on top of a vertical object like a building, bridge or ramp. In these cases, the depth is defined by the difference of the height h_{ij}^k of the surface patch and the height $h'^{\,k}_{ij}$ of the lowest measurement that is considered to belong to the vertical object. For flat objects like the floor, the depth is 0. Figure 2 depicts some examples of the map cells in an MLS map.

a) Map Creation: An MLS map can be generated in two different ways: either from a set of 3D measurements, i.e. a point cloud with variances, or by joining two other MLS maps into one. Both ways are equivalent, i.e. if map m_1 is created from point cloud C_1 and map m_2 from cloud C_2, then the map m_3 that results from joining m_1 and m_2 is identical to the map generated by the joined point cloud $C_3 = C_1 \cup C_2$. For a given point cloud C with variances $\sigma_1, \ldots, \sigma_n$ the MLS map is created as follows:

- Each map cell with index (i, j) collects all points $\mathbf{p} = (x, y, z)$, s.t. $si \le x \le s(i+1)$ and $sj \le y \le s(j+1)$ where s denotes the size (edge length) of a map cell.
- In each cell, we calculate a set of *height intervals* from the height values of the stored points. As long as two consecutive height values are closer than a given *gap size* γ, they belong to the same interval. This means that two intervals are at least γ meters away from each other. The gap size should be chosen so that a robot that navigates through the map can still pass the gap, i.e., it should be higher than the robot height. In our implementation we choose $1.0m$.
- The intervals are classified as a *horizontal* or a *vertical* structure. These structures are distinguished according to the height of the interval. If it exceeds a *thickness value* $\tau = 10$cm, it is considered as vertical, otherwise it is horizontal.
- For each interval classified as vertical, we store the mean and variance of the highest measurement in the interval. The intuition behind this is that for traversability only the highest measurement is relevant. Additionally, we store the length of the interval, which is identified with the depth d mentioned above. This value is used when matching two MLS maps together.
- For each horizontal object in a cell, we compute a mean μ and a variance σ from all measurements in the interval. This is done by applying the Kalman update rule to all measurements. The depth d of a horizontal object is set to 0.

After computing the means, variances and depths of the surface patches, we delete the point cloud data. All further calculations are performed only on the map data. This substantially reduces the memory required for an MLS map compared to point clouds and at the same time achieves a highly accurate representation.

b) Map Update: Whenever a new measurement $z = (\mathbf{p}, \sigma)$ is inserted into an MLS map, we first need to know whether the measurement belongs to an object that is already represented in the map, or if it corresponds to a new object. To this end, we first determine the cell c_{ij} in which the measured point falls. Then we find the Gaussian $(\mu_{ij}^k, \sigma_{ij}^k)$ in c_{ij} whose mean μ_{ij}^k is closest to the height of z. If this Gaussian is close enough, we update it with the new measurement z, again using the Kalman update rule. In our implementation, we define a Gaussian to be close to a measurement z if the height value of z is within 3σ of the Gaussian.

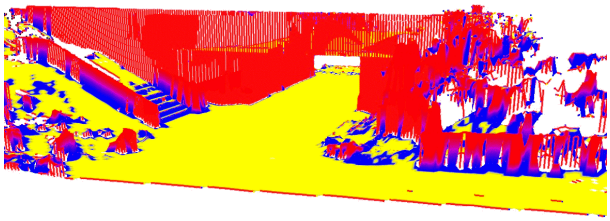

Fig. 3. Classification result for the MLS map depicted in Figure 1(d). The three colors/grey levels indicate the classification result for the individual surface patches into traversable, non-traversable, and vertical ones.

If z is far from the nearest Gaussian, it is still possible that it corresponds to a vertical object. This can be found out by checking whether z is inside the occupancy of one of the Gaussians in the cell. In this case, the measurement z is simply disregarded, because a vertical object with a Gaussian on top already exists. Otherwise, z is introduced as a new Gaussian into the cell.

B. Traversability Analysis and Feature Extraction

In the past, it has often been reported that the data association can seriously be improved by extracting features from the data [17], [20]. In addition to the vertical structures, we therefore additionally classify the horizontal surface patches according to their traversability. To determine whether a surface patch is traversable, we find the nearest Gaussian in each neighboring cell. In our approach, a surface patch can only be traversable if at least 5 of the 8 neighboring cells exist and if the distance in height between the patch and all its neighbors is less than 10cm. Figure 3 shows the classification result for the MLS map depicted in Figure 1(d). The three colors/grey levels indicate the classification result. The yellow/light grey parts of the surfaces represent the traversable surface patches. The blue/dark areas are the non-traversable parts of the surfaces and the red/medium grey structures represent the vertical objects.

C. Map Matching

To calculate the alignments between two local MLS maps calculated from individual scans, we apply the ICP algorithm. The goal of this process is to find a rotation matrix R and a translation vector $\mathbf{t}$ that minimize an appropriate error function. Assuming that the two maps are represented by a set of Gaussians, the algorithm first computes two sets of feature points, $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{N_1}\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_{N_2}\}$. Then in a second step the algorithm computes a set of C index pairs or *correspondences* $(i_1, j_1), \ldots, (i_C, j_C)$ such that the point $\mathbf{x}_{i_c}$ in $\mathcal{X}$ corresponds to the point $\mathbf{y}_{j_c}$ in $\mathcal{Y}$ for $c = 1, \ldots, C$. Then, in a third step, the error function e defined by

$$e(R, \mathbf{t}) := \frac{1}{C} \sum_{c=1}^{C} (\mathbf{x}_{i_c} - (R\mathbf{y}_{j_c} + \mathbf{t}))^T \Sigma^{-1} (\mathbf{x}_{i_c} - (R\mathbf{y}_{j_c} + \mathbf{t})), \quad (1)$$

is minimized. Here, Σ denotes the covariance matrix of the Gaussian corresponding to each pair $(\mathbf{x}_i, \mathbf{y}_i)$. In other words, the error function e is defined by the sum of squared Mahalanobis distances between the points $\mathbf{x}_{i_c}$ and the transformed point $\mathbf{y}_{j_c}$. In the following, we denote this Mahalanobis distance as $d(\mathbf{x}_{i_c}, \mathbf{y}_{j_c})$.

In principle, one could define this function to directly operate on the Gaussians when aligning two different MLS maps. One disadvantage of this approach, however, is that a MLS map of one single scan typically includes a huge number of Gaussians (15,318 on average in the data sets shown in this paper). Accordingly, the nearest neighbor search in the ICP algorithm requires a lot of computational resources. Additionally, we need to take care of the problem that the intervals corresponding to vertical structures vary substantially depending on the view-point. Moreover, the same vertical structure may lead to varying heights in the surface map when sensed from different points. In practical experiments, we observed that this introduces serious errors and often prevents the ICP algorithm from convergence. To overcome this problem, we separate Equation (1) into three components each minimizing the error over the individual classes of points. These three terms correspond to the three individual classes, namely surface patches corresponding to vertical objects, traversable surface patches, and non-traversable surface patches.

Let us assume that $\mathbf{u}_{i_c}$ and $\mathbf{u}'_{j_c}$ are corresponding points, extracted from vertical objects. The number of points sampled from every interval classified as vertical depends on the height of this structure. In our current implementation, we typically uniformly sample four points per meter. The corresponding points $\mathbf{v}_{i_c}$ and $\mathbf{v}'_{j_c}$ are extracted from traversable surface patches, $\mathbf{w}_{i_c}$ and $\mathbf{w}'_{j_c}$ are extracted from not traversable surfaces. The resulting error function then is

$$e(R, \mathbf{t}) = \underbrace{\sum_{c=1}^{C_1} d_v(\mathbf{u}_{i_c}, \mathbf{u}'_{j_c})}_{\text{vertical cells}} + \underbrace{\sum_{c=1}^{C_2} d(\mathbf{v}_{i_c}, \mathbf{v}'_{j_c})}_{\text{traversable}} + \underbrace{\sum_{c=1}^{C_3} d(\mathbf{w}_{i_c}, \mathbf{w}'_{j_c})}_{\text{non-traversable}}. \quad (2)$$

In this equation, the distance function d_v calculates the Mahalanobis distance between the lowest points in the particular cells. To increase the efficiency of the matching process, we only consider a subset of these features by sub-sampling.

IV. L C

The ICP-based scan matching technique described above works well for the registration of single robot poses into one global reference frame. However, the individual scan matching processes result in small residual errors which quickly accumulate over time and usually result in globally inconsistent maps. In practice, this typically becomes apparent when the robot encounters a loop, i.e., when it returns to a previously visited place. Especially in big loops this error grows so large that the resulting inconsistencies make the map useless for navigation. Accordingly, techniques for calculating globally consistent maps are necessary. In the system described here, we apply a technique similar to the one presented in [19] to correct for the accumulated error when closing a loop.

A. Network-based Pose Optimization

Suppose the robot recorded 3D scans at N different poses $P_1, \ldots, P_N$. A pose is defined as a 6-tuple $(x, y, z, \varphi, \vartheta, \psi)$ of location (x, y, z) and orientation $(\varphi, \vartheta, \psi)$. Whenever two local MLS maps m_i and m_j corresponding to the poses P_i and P_j are matched to each other using the map matching technique described above, a constraint is imposed on the poses P_i and P_j. If we represent all these constraints as undirected links in a graph, where the nodes are defined by the robot poses, we obtain a network of robot poses. A global pose optimization that takes all local constraints into account can be performed on such a pose network. In the literature, many different approaches for network-based pose optimization can be found. Recently, a good overview and comparison of different approaches has been presented by Olson *et al.* [19]. They authors also present a new algorithm based on a modified stochastic gradient descent (SGD) that is fast and robust against poor initial pose estimates. For our application, we implemented both the LU decomposition [19] and the SGD approach. Both gave good results, although the LU decomposition appeared to be slightly more robust.

B. Constraints between Robot Poses

As described above, the scan matching between two local MLS maps m_i and m_j is done by minimizing the error function $e(R, \mathbf{t})$ from Equation 2. After convergence of the ICP algorithm we have a set of correspondences between the feature sets $\mathcal{F}_i = (\mathcal{U}_i, \mathcal{V}_i, \mathcal{W}_i)$ and $\mathcal{F}_j = (\mathcal{U}_j, \mathcal{V}_j, \mathcal{W}_j)$ where $\mathcal{U}_i = \{\mathbf{u}_{i_1}, \ldots, \mathbf{u}_{i_{C_1}}\}$ and $\mathcal{U}_j = \{\mathbf{u}_{j_1}, \ldots, \mathbf{u}_{j_{C_1}}\}$. The other feature subsets $\mathcal{V}_i, \mathcal{W}_i, \mathcal{V}_j, \mathcal{W}_j$ are defined analogously. Given these correspondences we define a local constraint between two robot poses P_i and P_j as the rigid-body transformation between features in the local reference frame at P_i and the corresponding features in the local reference frame at P_j. This means, if the scan matcher returns a rotation matrix R_{ij} and a translation vector $\mathbf{t}_{ij}$ between the global coordinates of the features, our local constraint T_{ij} is defined by

$$T_{ij} \quad := \quad P_i^{-1}(R_{ij} P_j(\mathbf{f}_{i_c}) + \mathbf{t}_{ij}) \tag{3}$$

Here we use the notation $P_j(\mathbf{f}_{i_c})$ for the function that transforms a feature $\mathbf{f}_{i_c}$ from the local reference frame at the robot position P_j to the global reference frame. Accordingly, P^{-1} transforms a feature in global coordinates into the local reference frame at position P_j.

C. LU decomposition

Using the above notation we can now formulate the pose optimization problem. For a given set of initial robot poses $P_1, \ldots, P_N$ the scan matcher returns a set of local constraints T_{ij} for $i, j = 1, \ldots, N, i \neq j$ according to Equation 3. These constraints are then encoded in a *goal vector* $\mathbf{g}$ of length $6M$ where M is the number of constraints T_{ij}.

$$\mathbf{g} := (\ldots, x_{ij}, y_{ij}, z_{ij}, \varphi_{ij}, \vartheta_{ij}, \psi_{ij}, \ldots) \tag{4}$$

In other words, $\mathbf{g}$ contains all local constraints expressed as 3D translation and rotation. Next we define the *constraint function* $f : \mathbb{R}^{6N} \rightarrow \mathbb{R}^{6M}$ that maps robot poses to local constraints.

$$f(x_1, \ldots, \psi_1, \ldots, x_N, \ldots, \psi_N) := \begin{pmatrix} \vdots \\ \alpha(P_i^{-1} P_j) \\ \vdots \end{pmatrix} \tag{5}$$

Here we introduced the function α which converts a pose transform P into its 6 parameters $(x, y, z, \varphi, \vartheta, \psi)$. This constraint function f is non-linear due to the rotations. We therefore linearize it according to $f(\mathbf{p}) \approx F|_{\mathbf{p}} + J|_{\mathbf{p}} \Delta\mathbf{p}$ where $\mathbf{p}$ is the vector of all robot poses and $J|_{\mathbf{p}}$ is the Jacobian of the constraint function f at $\mathbf{p}$. At each iteration of the pose optimization $F|_{\mathbf{p}}$ and $J|_{\mathbf{p}}$ are recomputed for the new poses and we will write simply F and J for better readability. The global pose optimization can then be formulated as the minimization of the squared error:

$$\text{argmin}_{\mathbf{d}} \parallel J\mathbf{d} - \mathbf{r} \parallel^2 \tag{6}$$

Here we substituted $\Delta\mathbf{p}$ by $\mathbf{d}$ and $(F - \mathbf{g})$ by $\mathbf{r}$.

By deriving with respect to $\mathbf{d}$ and setting to zero we obtain

$$J^T J\mathbf{d} \quad = \quad J^T \mathbf{r} \tag{7}$$

This is a standard linear algebra problem which can be solved by LU decomposition or by multiplication with the pseudo-inverse of $J^T J$. The overall algorithm can then be described by iterating the following steps:

- Compute F and J from the poses $\mathbf{p}$.
- Minimize (6) according to Equation (7).
- Compute new poses: $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{d}$.

This is repeated as long as the residual $\mathbf{r}$ exceeds a threshold or a maximum number of iterations is reached.

D. Implementation Details

In our experiments it turned out that the described pose optimization technique works well in cases where the number of robot poses was around 70. However, the larger the loops are the higher is the initial pose estimation error from odometry, and it is not enough overlap left for the scan matching algorithm to find correspondences between the first and the last robot poses. Therefore we proceed as follows: Considering that the local scan matches between consecutive robot poses is highly accurate, we match local maps corresponding to 5 consecutive poses into new and bigger local maps. This means that maps $m_1, \ldots, m_5$ are matched into the new map $\bar{m}_1$, maps $m_6, \ldots, m_{10}$ into $\bar{m}_2$ and so on. This way we obtain a set of $\frac{N}{5}$ new local maps, which reduces the pose optimization problem. A further improvement can be achieved by increasing the overlap between consecutive local maps $\bar{m}_i$ and $\bar{m}_{i+1}$. This can be done by adding the last partial map from $\bar{m}_i$ to $\bar{m}_{i+1}$. For example, the local map m_5 is then also a part of the new local map $\bar{m}_2$. In this way, the scan matching error between the joined local maps $\bar{m}_i$ and $\bar{m}_j$ can be reduced and the global pose optimization is more likely to converge to a global optimum.

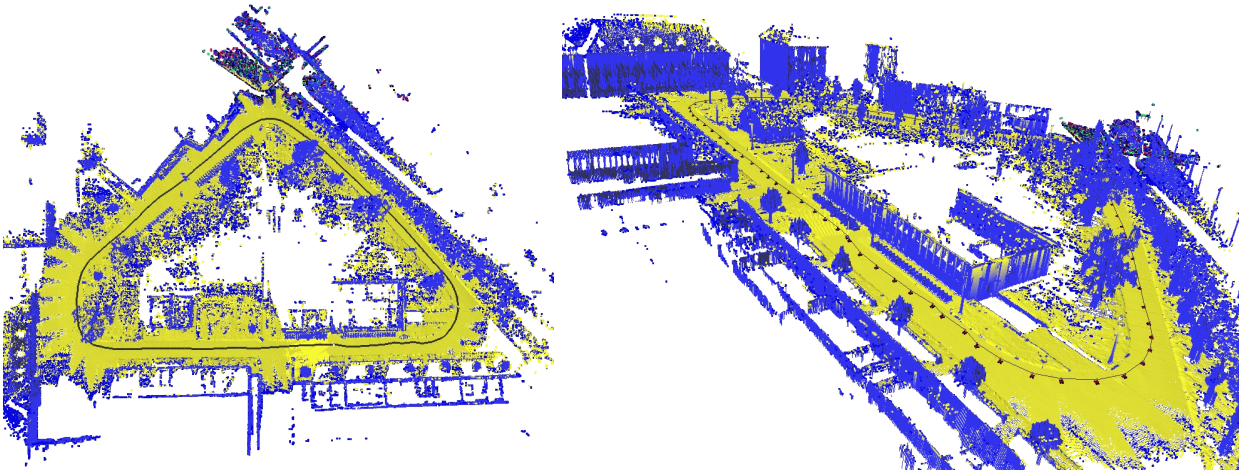

Fig. 4. Two views of the resulting MLS map of the first experiment with a cell size of 10cm x 10cm. The area scanned by the robot spans approximately 195 by 146 meters. During the data acquisition, where the robot collected 77 scans consisting of 20,207,000 data points, the robot traversed a loop with a length of 312m.

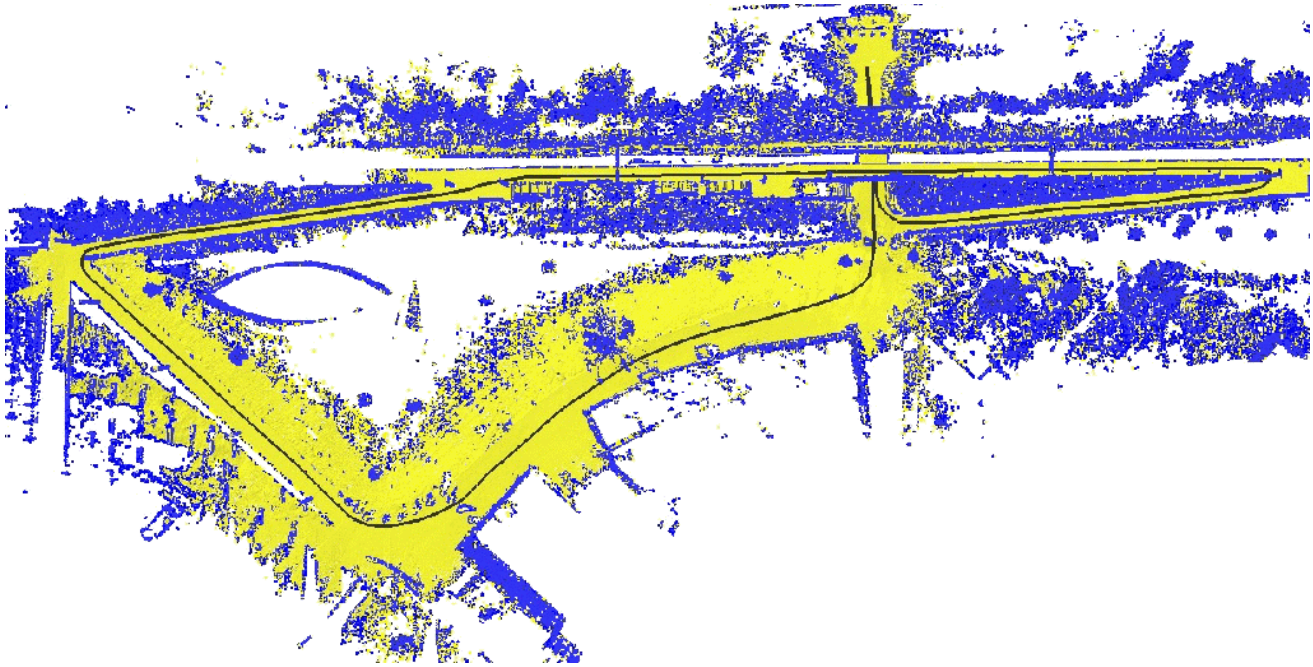

Fig. 5. Resulting MLS map of the second experiment with a cell size of 10cm x 10cm. The area scanned by the robot spans approximately 299 by 147 meters. During the data acquisition, where the robot collected 172 scans consisting of 45,139,000 data points, the robot traversed a loop with a length of 560m.

V. E

Our approach has been implemented and evaluated using our mobile outdoor platform Herbert which is a Pioneer II AT robot equipped with a SICK LMS 291 range sensor mounted on an AMTEC wrist PW70. To acquire the data, we steered the robot over the university campus. On its path, the robot encountered two loops. Additionally, it traversed over a bridge and through the corresponding underpass. The goal of these experiments is to demonstrate that our representation yields a significant reduction of the memory requirements compared to a point cloud representation, while still providing sufficient accuracy. Additionally, they show that our representation is well-suited for global pose estimation and loop closure.

In the first experiment we acquired 77 scans consisting of 20,207,000 data points. The area scanned by the robot spans approximately 195 by 146 meters. During the data acquisition, the robot traversed a loop with a length of 312m. Figure 4 shows two views of the resulting MLS map with a cell size of 10cm x 10cm. The yellow/light grey surface patches are classified as traversable. It requires 57.96 MBytes to store the computed map, where 24% of 2,847,300 cells are occupied.

In the second experiment we acquired 172 scans consisting

of 45,139,000 data points. The area scanned by the robot spans approximately 299 by 147 meters. During the data acquisition, the robot traversed a loop, which has a length of 560m. Figure 5 shows the resulting MLS map with a cell size of 10cm x 10cm. The yellow/light grey surface patches are classified as traversable. It requires 73.33 MBytes to store the whole map, where 20% of 4,395,300 cells are occupied.

VI. C

In this paper, we presented multi-level surface maps as a novel representation for outdoor environments. Compared to elevation maps, multi-level surface maps (MLS maps) store in each cell of a discrete grid a list of surfaces. Additionally, they use intervals to represent vertical structures. We presented algorithms for updating multi-level surface maps based on sensory input, for matching such maps and for solving the loop-closing problem using this representation.

We also described an implementation of multi-level surface maps on a Pioneer II AT platform equipped with a laser range scanner mounted on a pan/tilt unit. We presented large-scale maps learned from the data acquired with this robot. The resulting maps show a high accuracy and at the same time require one order of magnitude less space than the original point data. Additionally, the results demonstrate that multi-level surface maps allow mobile robots to operate in environments with multiple levels. In one of our experiments the robot successfully traveled over a bridge and through the corresponding underpass.

A

This work has partly been supported by the German Research Foundation (DFG) within the Research Training Group 1103 and under contract number SFB/TR-8.

R

[1] P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. Avenue: Automated site modeling in urban environments. In *Proc. of the 3rd Conference on Digital Imaging and Modeling*, pages 357–364, 2001.

[2] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. R. L. Whittaker. Ambler: An autonomous rover for planetary exploration. *IEEE Computer Society Press*, 22(6):18–22, 1989.

[3] A.J. Davison, Y. Gonzalez Cid, and N. Kita. Real-time 3d SLAM with wide-angle vision. In *Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[4] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[5] C. Früh and A. Zakhor. An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision*, 60:5–24, 2004.

[6] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, May 2001. In press.

[7] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[8] M. Hebert, C. Caillas, E. Krotkov, I.S. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 997–1002, 1989.

[9] E. Hygounenc, I.-K. Jung, P. Souères, and S. Lacroix. The autonomous blimp project of laas-cnrs: Achievements in flight control and terrain mapping. *International Journal of Robotics Research*, 23(4-5):473–511, 2004.

[10] P. Kohlhepp, M. Walther, and P. Steinhaus. Schritthaltende 3D-Kartierung und Lokalisierung für mobile inspektionsroboter. In *18. Fachgespräche AMS*, 2003. In German.

[11] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury and; M. Herrb, and R. Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *International Journal of Robotics Research*, 21(10-11):917–942, 2002.

[12] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proc. SIGGRAPH*, pages 131–144, 2000.

[13] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.

[14] C. Martin and S. Thrun. Online acquisition of compact volumetric maps with mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, 2002. ICRA.

[15] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[16] H.P. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. Technical Report CMU-RI-TR-96-34, Carnegie Mellon University, Robotics Institute, 1996.

[17] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *Proc. of the 12th Int. Conference on Advanced Robotics (ICAR)*, pages 242–249, 2005.

[18] C.F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, 2000.

[19] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor estimates. In *ICRA*, 2006. to appear.

[20] Pfaff P. and Burgard W. An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. of the International Conference on Field and Service Robotics (FSR)*, pages 165–176, 2005.

[21] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot. 3-d modelling and robot localization from visual and range data in natural scenes. In *1st International Conference on Computer Vision Systems (ICVS)*, number 1542 in LNCS, pages 450–468, 1999.

[22] K. Pervölz, A. Nüchter, H. Surmann, and J. Hertzberg. Automatic reconstruction of colored 3d models. In *Proc. Robotik 2004*, 2004.

[23] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley Publishing Inc., 1989.

[24] S. Singh and A. Kelly. Robot planning in the space of feasible actions: Two examples. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1996.

[25] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2000.

[26] S. Thrun, D. Hähnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker. A system for volumetric robotic mapping of abandoned mines. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2003.

[27] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7-8):693–704, 2004.

[28] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery Montemerlo, C. Deepayan, and W. Burgard. A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics and Automation*, 20(3):433–442, 2003.

[29] R. Triebel, F. Dellaert, and W. Burgard. Using hierarchical EM to extract planes from 3d range scans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[30] O. Wulf, K-A. Arras, H.I. Christensen, and B. Wagner. 2d mapping of cluttered indoor environments by means of 3d perception. In *ICRA-04*, pages 4204–4209, New Orleans, apr 2004. IEEE.

[31] C. Ye and J. Borenstein. A new terrain mapping method for mobile robot obstacle negotiation. In *Proc. of the UGV Technology Conference at the 2002 SPIE AeroSense Symposium*, 1994.