Robotics and Autonomous Systems 56 (2008) 915–926

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Towards semantic maps for mobile robots

Andreas Nüchter [*], Joachim Hertzberg

University of Osnabrück, Institute of Computer Science, Knowledge-Based Systems Research Group, Albrechtstr. 28, D-49069 Osnabrück, Germany

A R T I C L E I N F O

Article history:
Available online 15 August 2008

Keywords:
3D mapping
6D SLAM
Scene interpretation
Object detection
Semantic mapping

A B S T R A C T

Intelligent autonomous action in ordinary environments calls for maps. 3D geometry is generally required for avoiding collision with complex obstacles and to self-localize in six degrees of freedom (6 DoF) (x, y, z positions, roll, yaw, and pitch angles). Meaning, in addition to geometry, becomes inevitable if the robot is supposed to interact with its environment in a goal-directed way. A semantic stance enables the robot to reason about objects; it helps disambiguate or round off sensor data; and the robot knowledge becomes reviewable and communicable.

The paper describes an approach and an integrated robot system for semantic mapping. The prime sensor is a 3D laser scanner. Individual scans are registered into a coherent 3D geometry map by 6D SLAM. Coarse scene features (e.g., walls, floors in a building) are determined by semantic labeling. More delicate objects are then detected by a trained classifier and localized. In the end, the semantic maps can be visualized for human inspection. We sketch the overall architecture of the approach, explain the respective steps and their underlying algorithms, give examples based on a working robot implementation, and discuss the findings.

1. Introduction

1.1. What is a semantic map?

If it is agreed that semantic knowledge can help an autonomous robot act goal-directedly, then, consequently, part of this knowledge has to be about objects, functionalities, events, or relations in the robot's environment. The data structure holding the space-related information about this environment is the *map*. Typical state-of-the-art robot maps represent the environment geometry – often in 2D, sometimes in 3D, sometimes topologically – and, maybe, additional sensor-relevant information such as specific features, or texture [5]. This typical map content is in harmony with today's typical purpose of maps for mobile robots, namely, navigation. A *semantic map* augments that by information about entities, i.e., objects, functionalities, or events, that are located in space.

We assume that the main purpose, or family of purposes, for a semantic stance in map contents is some type of reasoning based on individual entities in the map and/or their classes; examples for such reasoning are planning, explanation, prediction, and sensor data interpretation. To enable this reasoning, some background knowledge about entities is required, an informal example being a rule like *A chair typically rests on the floor*. The knowledge may come in any suitable knowledge representation format [4], as needed for the type or types of reasoning to be associated with the entities in the map. Given that such knowledge is typically independent of space, it is not strictly part of the map; however, we require that it exists for entities represented in the semantic map. In brief, then:

> A *semantic map* for a mobile robot is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine.

In the technical part of this paper, we will use special instances of sensor configuration, map type, and reasoning. Note, however, that we understand semantic maps as being a more general concept than what we have experimented with and that we will describe below. We will get back to this in the discussion part of this paper.

1.2. System and paper architecture

Our approach uses 3D laser range and reflectance data for environment mapping and for perceiving 3D objects on an autonomous mobile robot. Starting from an empty map, multiple 3D scans, acquired by the robot in a stop-scan-go fashion, are registered consistently by 6D SLAM, i.e., by a version of *Simultaneous Localization and Mapping* that allows for using 6DoF robot poses (x, y, z positions; yaw, pitch and roll angles). Then, the coarse structure of the resulting 3D scene is interpreted using plane

[*] Corresponding author. Tel.: +49 541 969 2623.
E-mail address: andreas.nuechter@uos.de (A. Nüchter).

0921-8890/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2008.08.001

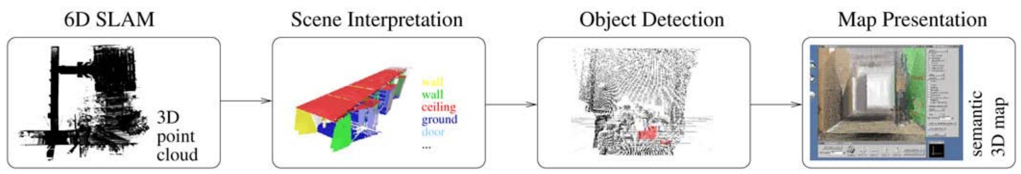

Fig. 1. System overview: From left to right: 6D SLAM acquires and registers a point cloud consisting of multiple 3D scans; scene interpretation labels the basic elements in the scene; object detection locates previously learned objects in 6 DoF; finally, the semantic map is presented to the user.

extraction and labeling, thereby exploiting background knowledge represented in a constraint network [34]. After that, the 3D range and reflectance data are transformed into 2D images by off-screen rendering, and they are used in this form for detecting and localizing objects by two alternative approaches [35,42]; the object localization is then transformed back into the 3D data. Finally, the semantic map is presented using tools from computer graphics. Fig. 1 gives a system overview. Our system contains the classical architecture to derive symbols from sensor data [22]. While these building blocks in isolation have been described in previous publications, we present in this paper for the first time how these components have been put together to build semantic maps according to our definition. Note that the simple cascade-style architecture just described is only an initial point for building semantic maps: In general, one would use feed-back, e.g., from object detection to scene interpretation. The matter will be discussed below.

The paper presents our work in terms of the building blocks in Fig. 1. We emphasize scene interpretation and object detection, and the feed-back control loop beyond the simple cascade in Fig. 1. So after finishing this introduction by remarks on related work, Section 2 summarizes some technical background concerning the 6D SLAM algorithm used. Section 3 describes the process of bottom-up interpretation of gross scene features. These are augmented by recognized objects, as presented in Section 4. Section 5 wraps up the process and provides example results. Section 6 discusses our own results and puts them into perspective with semantic mapping in general. We conclude the paper in Section 7.

1.3. Related work

Robotic mapping. In the considerable body of literature about robot maps and mapping, maps are metrical in most cases, and less frequently, topological. As in regular language, a map contains space-related information about the environment, i.e., not all that a robot may know or learn about its world need go into the map. Metric maps are supposed to represent the environment geometry quantitatively correctly, up to discretization errors. We will use the term *geometry map* henceforth to refer to maps that represent (metrically more or less truthfully) the environment geometry. [41, Ch. 5] gives a general introduction into the topics of maps and mapping; [46] covers probabilistic approaches in particular. Both textbooks also give introductions to SLAM, i.e., the process of building a map based on imprecise sensor data and on the imprecise robot motion model.

Most robot maps in the literature are given in 2D, usually upright projections of the scene. Since the early 2000s, some groups have been using pitching or rotating laser scanners for acquiring 3D data, e.g.,[43,48,50]. As these data are much richer in information than the 2D scans mostly used in 2D mapping, slightly different algorithms are used for 3D. Based on consistent 3D scans of the environment, scan matching variants are often applied for constructing a 3D map [13,18,26,30,36,40,44]. [33] summarizes the state of the art in 3D mapping.

Only few groups in robotics have been working on variants of semantic mapping. [21] presents a robot control architecture that fuses mapping and object detection, resulting in a labeled map. [20] presents a mapping system that reconstructs 3D models assuming 3 DoF, i.e., planar, robot motion in RoboCup Rescue. In the same context, [9] uses labeled maps for automatic behavior activation. [1] and [3] also present mapping approaches that include object detection. They repeatedly map environments and identify changing occupancy of grid cells using difference maps, focusing on representing uncertain object knowledge in such occupancy object maps. [25] describes a probabilistic approach for inserting in the map hierarchical environment structures and spatial relations, all based on 2D data. [12] is a study, also based on 2D data, to combine metric, topological, and semantic aspects in a map. It uses the semantic level for reasoning (“This room contains no sink, it cannot be the kitchen!”).

Scene understanding. To understand *understanding* has been a topic in AI from its early days on. The problem could be described as [39, pg. 791]

> We are given a set of ambiguous inputs, and from them we have to work backwards to decide what state of the world could have created these inputs.

Prominent lines of research in AI include language/speech understanding, image understanding, and scene understanding — all in the sense just quoted. More recent AI research mostly avoids the term due to its generality, imprecision, and metaphorical overloadedness. Yet, it describes nicely what is needed for building semantic maps. Recent work in computer vision uses the term *Cognitive Vision*, cf. [7]. We will come back to the approach by Neumann and Möller [29]: They use a description logic domain theory and a representation of perceived environment objects and processes for aggregating bottom-up scene information from camera images and for hypothesizing top-down features to look for in the given image stream. That is clearly an important ingredient of building semantic maps. A point is typically lacking in scene understanding work that semantic mapping in closed-loop robot control should include: Physical robot action in sensor data acquisition, as by changing the pose or even physical interaction with the environment.

Symbol grounding. Object anchoring [8] is a line of robotics-related research that aims at building up and maintaining the links between symbolic representations of objects (as in a logic-based knowledge representation formalism) and their images in the sensor data stream. This is clearly related to semantic mapping; it is also more ambitious than the latter, as anchoring assumes projecting the development of anchored objects into the time-space future, which semantic mapping, as considered here, does not necessarily involve. On the same line, a semantic map is related to solving the symbol grounding problem [16]. Note, however, that semantic mapping deals only with a small fraction of symbol grounding in general.

2. Technical prolegomena: 6D SLAM

For building a semantic 3D map, we start with building some version of geometric 3D map of the environment first. In the cascade architecture of Fig. 1, this prior map even needs to have

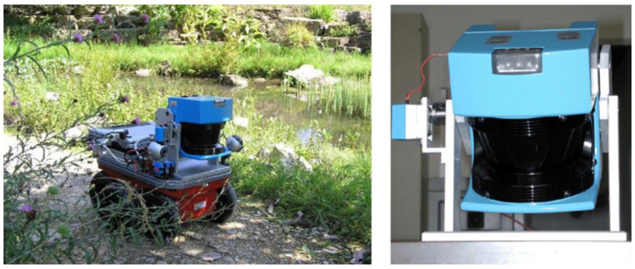
Fig. 2. *Left*: The mobile robot Kurt3D. *Right*: The 3D laser scanner.

some degree of completeness, as the interpretation processes coming later do not add or correct data. In our work, we have used a 6D SLAM method and software that we have described previously (e.g., [33]). The way how the initial 3D geometry map of the environment is obtained from original sensor data, is unimportant for the subsequent interpretation process. (However, note that subsequent steps do require that the data consist of surface points in space, as delivered by laser scan data!) Therefore, we include a sketch of the 6D SLAM part in this paper to make it self-contained; but we keep it short, and refer to [33] for details.

To start with, the robot used here is a Kurt3D (Fig. 2). The 3D laser scanner is built on the basis of a 2D SICK scanner, extended by a mount and a standard servo motor for controlled pitch motion [43].

The 6D SLAM method is supposed to take 3D scans of the complete environment and register them into a globally consistent and correct 3D map. Registration has to compensate the fact that every single scan pose is given in 6 DoF, i.e., registration has to consider three translation and three rotation dimensions, too. In our algorithm, individual scans are registered by the Iterative Closest Points (ICP) algorithm [2]. ICP calculates point correspondences iteratively. In each iteration, it selects the closest points as corresponding and calculates the transformation $(\mathbf{R}, \mathbf{t})$ for minimizing the equation

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left\| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t}) \right\|^2, \qquad (1)$$

where N_m and N_d, are the numbers of 3D points in the model set $\mathcal{M}$ and the data set $\mathcal{D}$, respectively. $w_{i,j}$ are the weights for a point match. They are assigned like: $w_{i,j} = 1$, if $\mathbf{m}_i$ is closest to $\mathbf{d}_j$ within a close limit; $w_{i,j} = 0$ otherwise. The transformation is calculated by the quaternion-based method by Horn [19]. The point correspondences are assumed correct when $E(\mathbf{R}, \mathbf{t})$ has fallen below a given threshold.

However, ICP alone is not enough for practical 6D SLAM, since errors do accumulate. We have extended it in the following ways for enhancing flexibility and speed [33]:

(1) Extrapolate the odometry to all six DoF of the robot pose.
(2) Calculate heuristic initial estimations for ICP scan matching based on this extrapolation.
(3) Register the 3D scans into a common coordinate system using ICP.
(4) If applicable (a previous scan of the same region like the most recent one exists), close the loop and distribute the error.
(5) After all scans are taken, refine the model by global relaxation.

These five steps come at different computational costs. In our experiments, we would usually take 3D scans with 20,000 up to 300,000 3D points. While the first step is computed instantaneously, the heuristic in step two, which is based on an octree reduction of the scans, would need up to two seconds, if applied naively, for calculating the two octrees and then roughly aligning the scans. Since octrees are usually computed fast, the influence of larger data sets is negligible here. The loop closing step is of similar cost, since we use the octree heuristic again [33]. The most computation time is needed in the scan matching step and in the model refinement step five. While the model can easily be refined offline, i.e., after the robot has finished the data acquisition, the scan matching is an essential part of the online mapping procedure. We have a number of methods to reduce significantly the run time, namely, point reduction, k-d trees, approximate k-d trees, and cached k-d trees [33]. By these means, an online, on-board variant of the ICP algorithm is constructed and has been demonstrated to work reliably in a large suite of experiments.

3. Scene interpretation

We now assume a 3D geometry model of the scene is given — in our case, in terms of a cloud of surface points. Note that, depending on scan resolution and scene size, the model would normally include millions of points. Each and every one of them is subject to slight measurement and registration errors.

By *Scene Interpretation*, we refer to the process of labeling large meaningful structures in the 3D geometry model. Such structures would typically be represented by points in the model, and of a large number of them at that; examples are walls, the floor, and the ceiling inside a building. Walls, e.g., are normally characterized by a shape (flat) and an orientation (perpendicular). This need not be true for all interesting structures, however. In an outdoor scenario, say, a park, a sand path to follow is often neither flat nor horizontal, yet it is utterly relevant for a mobile robot and should be identified. Interesting structures may not be defined by data points exclusively; the *absence* of 3D points may also be of interest. Take as an example the structure *navigable surface*, which could be defined by sufficiently smooth ground and free volume the size of the robot bounding box directly above.

Scene interpretation differs notably from object recognition, which is described later (Section 4). We assume objects are compact and segmentable. The structures to be identified in scene interpretation, in contrast, need have no clear or perceivable boundaries (where exactly does a sand path end?), and are typically spread out over a larger area of the scene. Interpretable structures may be defined directly or locally on the sensor data (the 3D points in our case), or on features previously detected in these data. We will give examples for both.

We describe next our approach to plane (i.e., feature) extraction and labeling from point clouds. This instance of scene interpretation is of obvious interest for indoor robot applications. In the final part of this section, we describe a different algorithmic method for identifying drivable surface based directly on the data points, which may even run online on single 3D scans; for this approach, we present data from an outdoor area.

3.1. Plane extraction and labeling

Given a 3D point model of a scene, identifying planes is first and foremost an algorithmic problem, for which quite a number of solutions are known. RANSAC (Random Sample Consensus) [38] is a well known algorithm for plane extraction from point sets. It is used in [6]. RANSAC is suited for fitting models robustly in the presence of many data outliers. It first selects N data items randomly and uses them to estimate the parameters of the plane. Next, the number of data points fitting the model is computed, regarding a user-given tolerance. RANSAC accepts the fit if the computed number of points exceeds a certain limit. Otherwise it iterates with another random sample [38].

In our work, a combination of RANSAC and ICP has performed best, providing fast plane extraction for a point cloud. No prior meshing algorithms need be applied. A plane p is defined by three

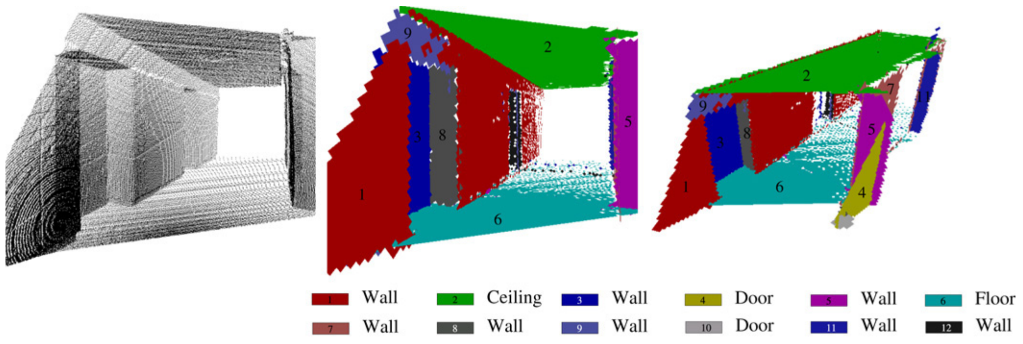

Fig. 3. *Left*: Point cloud. *Middle, right*: Extracted planes with interpretations.

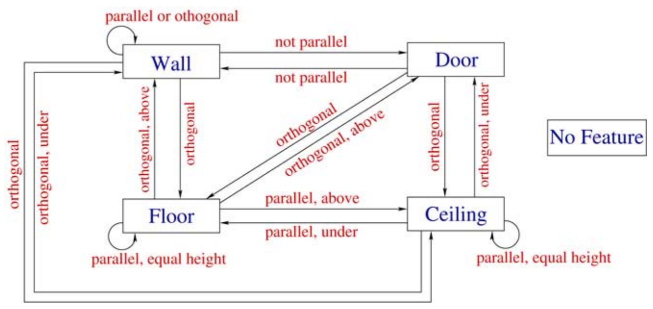

Fig. 4. Constraint network for coarse scene interpretation.

3D points ($\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in R^3$) or by one 3D point and the surface normal ($\mathbf{p}_1, \mathbf{n}$ with $\|\mathbf{n}\| = 1, \mathbf{p}_1, \mathbf{n} \in R^3$). To detect a surface, the algorithm randomly selects a point and estimates a plane through two neighboring data points. Now those data points $\mathbf{x} \in R^3$ are calculated which fulfill:

$$|(\mathbf{x} - \mathbf{p}_1) \cdot \mathbf{n}| < \epsilon. \tag{2}$$

If this set of points exceeds a limit, e.g., 50 points, an ICP-based optimization is started. All data points satisfying Eq. (2) form the model set M and are projected on the plane to form the data set D for each iteration of ICP. It takes only a few iterations to minimize the ICP error function (1) by transforming the plane with this point-to-plane metric. The time-consuming search is replaced by direct calculation of the closest point and the transformation $(\mathbf{R}, \mathbf{t})$ is calculated efficiently [19]. Given the best fit, all plane points are marked and subtracted from the original data set. The algorithm terminates after all points have been tested according to Eq. (2).

The extracted 3D planes are unbounded in size. Surfaces are finally extracted from the points by mapping them onto the planes. A quadtree-based method generates the surfaces. Fig. 3 shows an example with 7 extracted planes of a single 3D scan containing 58,680 range data points.

We now turn to labeling, i.e., interpreting the planes. To this end, we use a bit of common-sense knowledge about buildings. A generic model of an indoor scene is implemented as a constraint network based on [15], which is also used in [6].

Nodes of a constraint network here represent different plane types in a building. Relations among them are encoded using different connections. Possible labels of the nodes are $L = \{$`Wall`, `Floor`, `Ceiling`, `Door`, `No Feature`$\}$. `Door` represents a door *ajar* (i.e., not parallel to the wall that surrounds it). $R = \{$`parallel`, `orthogonal`, `equalheight`$\}$ are the inter-feature relations. The spatial relations `above` and `under` relate to their planes and are therefore not commutative. Fig. 4 shows the entities and their relations. The constraint network can easily be extended, but additional entities would have to be accompanied by matching feature detectors.

Prolog is used to implement the constraint network solver, encoding the network by definite clauses. The nodes of the network are arguments, and the arcs define relations on the nodes. Next, all facts for the relation `parallel` are shown:

```
parallel(floor,floor).
 parallel(ceiling,floor).
 parallel(ceiling,ceiling).
 parallel(floor,ceiling).
 parallel(wall,wall).
 parallel(X,_) :- X == nofeature.
 parallel(_,X) :- X == nofeature.
```

Similarly, the other relations are encoded:

```
orthogonal(ceiling,door).
 orthogonal(ceiling,wall).
 ...
 equalheight(floor,floor).
 equalheight(ceiling,ceiling).
 equalheight(door,_).
 ...
```

For encoding the label `nofeature`, a condition is used. This prevents Prolog's unification algorithm from assigning planes with the label `nofeature`, as long as different assignments appear possible. In addition to the representation of the constraint network, a clause of the following form is compiled from the analysis of the planes: Each plane of the set of planes found by the extraction is represented by a variable `P0`, `P1`, etc. Then the relations between the individual planes are computed from the scan data in a data-driven fashion using thresholds. Finally, Prolog clauses are generated automatically, describing the consisting labels for all planes found; assuming 4 planes were found, we would generate, e.g.:

```
labeling(P0,P1,P2,P3) :-
        parallel(P0,P1),under(P0,P1),
        orthogonal(P0,P2),under(P0,P2),
        orthogonal(P0,P3),under(P0,P3), ...
```

Prolog's unification and backtracking are used to label the planes consistently.

```
consistent_labeling(P0,P1,P2,P3) :-
          labeling(P0,P1,P2,P3).
```

The label `nofeature` is considered, iff the unification fails. In this case, an additional Horn clause is used to generate a consistent labeling that unifies exactly one variable with `nofeature`. All combinations are computed to unify the variable:

```
consistent_labeling(P0,P1,P2,P3) :-
          comb([P0,P1,P2,P3],[nofeature]),
          labeling(P0,P1,P2,P3).
```

The `comb` predicate generates all combinations of the planes `Pi` with one of them assigned `nofeature`. In case of failure of `consistent_labeling` with one plane assigned `nofeature`, the process is continued with assigning to two `nofeature` values, and

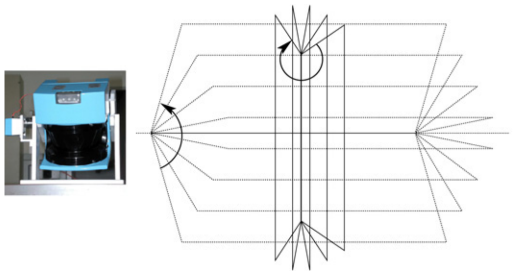
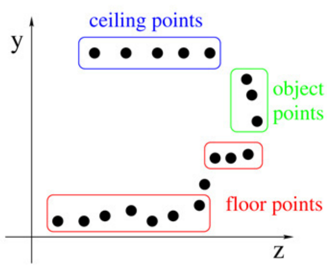

Fig. 5. *Left*: Scan points are transformed into a vertical cylindrical coordinate system and swept vertically. *Right*: Interpretation schema in one vertical sweep plane.

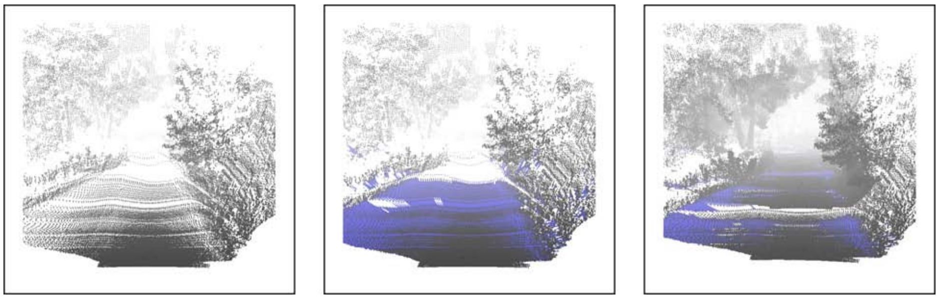

Fig. 6. *Left*: A single outdoor 3D scan of a gravel path in Osnabrück's Botanical Garden (cf. Fig. 2, left). Note that the path is uneven. *Middle*: Areas (triangles) between neighboring surface points all labeled drivable are grouped into a solid surface (triangulated). Note that there are some disconnected patches of surface points in and behind the path shoulder. *Right*: View into the model from the same virtual view point as before, but with the next scans along the path registered. Sufficiently large areas sufficiently dense with drivable surface points are filled up.

so on, until success is achieved. Finally the following query is submitted:

```
?- consistent_labeling(P0,P1,P2,P3).
```

and the automatic generated Prolog program starts, computing the solution. This simple algorithm is of course exponential in terms of planes to label. Yet, we did not consider that worth optimizing, as we had run times that are negligible for an offline process (e.g., 314 ms for labeling 13 planes, Pentium IV-2400, SWI-Prolog [45]). As a footnote for the historically minded, the method used here for consistent labeling is local constraint propagation like classical Waltz labeling [49] used in one of the earliest approaches to scene understanding.

3.2. Labeling points drivable

Scene interpretation as a part of semantic mapping may come in many forms. To show that other approaches than feature-based labeling as just described make sense, we sketch a method for labeling areas drivable directly based on the point data. Similar approaches have been presented, e.g., in [37]. To keep it short, we describe this for single 3D scans; the transfer to registered point maps is straightforward. For details, cf. [31].

Our algorithm uses the approach in [50], which allows "flatness" of the ground to be determined efficiently by the local gradient between data points in a *vertical* cylindrical coordinate system (cf. Fig. 5 middle, vertical system). Let $\mathbf{p}_{i,j} = (\phi_i, y_{i,j}, z_{i,j})$ be the jth data point in the vertical sweep plane at the angle ϕ_i. (Assume the point cloud is swept vertically from bottom to top at a cylinder angle $\phi_i \pm \epsilon$.) The gradient between point j and its kth neighbor in sweep order is then given by

$$\alpha_{i,j} = \arctan\left(\frac{z_{i,j} - z_{i,j-k}}{y_{i,j} - y_{i,j-k}}\right)$$

with $-\frac{1}{2}\pi \le \alpha_{i,j} < \frac{3}{2}\pi$. In comparison with a fixed threshold τ (here: $\tau = 20°$), each 3D point is assigned to one of the following three groups, which has proved to be robust against uneven and tilting ground:

1. $\alpha_{i,j} < \tau$: $\mathbf{p}_{i,j}$ is a ground point
2. $\tau \le \alpha_{i,j} \le \pi - \tau$: $\mathbf{p}_{i,j}$ is an object point
3. $\pi - \tau < \alpha_{i,j}$: $\mathbf{p}_{i,j}$ is a ceiling point.

A result of the ground segmentation is displayed in Fig. 6. Note that the classification of a scan point as "ground" is based on its neighborhood, rather than on comparing estimated absolute elevation. This irons out uncertainty in the absolute pitch and roll angles in the robot pose. Absolute height values of points do come into play when nearby drivable surface points are aggregated into one large drivable area. Our point classification scheme leaves the possibility open that two nearby points are both correctly labeled ground, but differ significantly in absolute height, such as on the two horizontal sides of a sharp pothole edge. This needs to be checked when neighboring ground points are aggregated into drivable surfaces.

4. Object detection and interpretation in 3D data

Following the sensor data flow bottom-up (cf. Fig. 1), the next stage in semantic mapping is *object detection*. We have divided this block into five steps (Fig. 7), which we will address in turn. We will present two different methods for classification, to give another example for the potential variety of approaches that we envision at work behind semantic mapping.

Detecting objects of known rigid shapes in 3D scene models could be seen as a problem of 3D geometry matching. We have chosen a different route, which allows methods and algorithms from 2D image interpretation to be reused: We generate object hypotheses from 2D renderings of our 3D data, use the projection of a detected 2D object image back into the 3D data for narrowing down the possible position of the 3D object, and then attempt 3D model matching of the object class hypothesis only in this narrow 3D data subset.

So after scanning, the 3D data points are first projected by an off-screen OpenGL-based rendering module onto a 2D image plane. The virtual camera for this projection is located in the scanner origin. The following subsections include various example images.

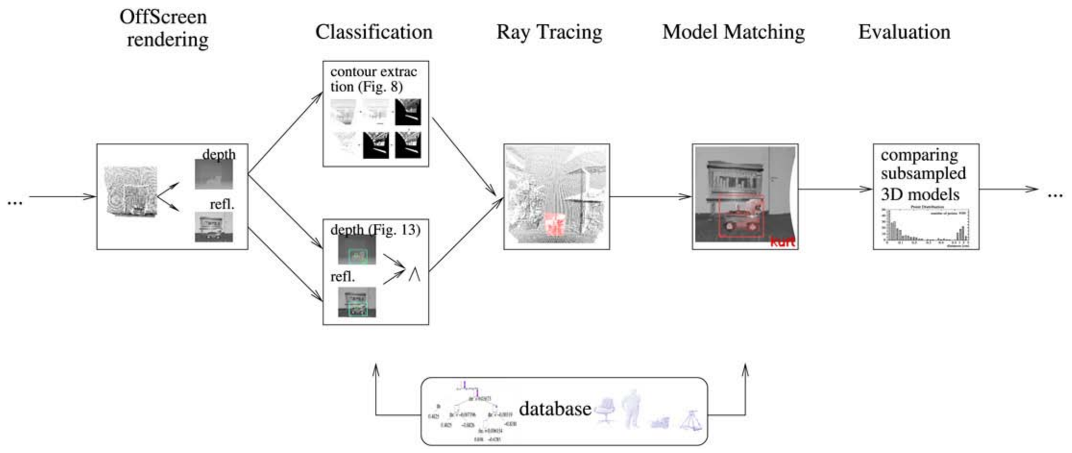

Fig. 7. After acquiring 3D scans, depth and reflection images are generated. In these images, objects are detected using a learned representation from a database. Ray tracing selects the points corresponding to the 2D projection of the object. A 3D model is matched into these points, followed by an evaluation step.

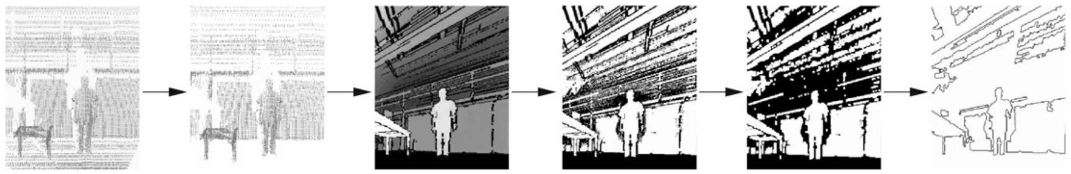

Fig. 8. Cascade for contour extraction. From left to right: (1) Scanned scene as point cloud. (2) Point cloud with removed floor points. (3) Generated range image without interpolation at jump edges. (4) Binarized image using adaptive thresholding. (5) Morphological opening of the image. (6) Final contour representation.

4.1. Classification using contour data

Our first approach to object classification works on the 2D rendering of distance data, with distances encoded by grey values. It is based on recognizing known contours in this representation. Fig. 8 summarizes the steps to getting from the laser data to the contour representation.

Imagine segmenting objects in a 2D range image. The feet of a human standing on the floor, say, will equal exactly the floor area around in grey value, as they have the same distance: The feet form only a crease edge with the floor, no jump edge. To handle this, we use the scene interpretation results from the previous step: scan points belonging to interpreted scene structures are taken off the 3D point model prior to rendering. So the human standing on the floor will "float in nothingness" (represented as black) in the corresponding 2D image. Fig. 9 shows an example for removing the floor.

This solves object segmentation against the scene structures detected earlier. To enhance segmentation of objects against the rest of their background, we simply do not interpolate the range image, if the range difference between two neighboring points exceeds a fixed threshold. This method yields without further effort a range image with each object sufficiently distant from the rest of the scene enclosed by a black contour. Fig. 9 (right) shows the result.

The actual contour extraction is done after applying a binarization filter with an adaptive threshold and convolving the result with a non-linear filter for "morphological opening", cf. Fig. 8(4, 5). The details are out of the scope of this paper, so we refer to [42].

Finally, contours are extracted from the binarized image using a contour following algorithm, see Fig. 8(6). We use the Eigen-CSS feature extraction method to describe a contour [42], improving the original CSS method by Mokhtarian [10,28]. In particular, our method is robust against contour rotation, as inspired by the work by Drew, Lee and Rova [10].

We have used Support Vector Machines (SVM) for classifying objects based on their contour representations. In the training phase, test scans are taken from each object and three range images generated from each scan under different views. Then every range image is segmented, and stored as a positive example in case of correct segmentation. Each classifier was trained using 200 positive and 700 negative examples. Fig. 10 presents results of the complete classification system. In the experiments on test data (different from the training data), we have achieved an accuracy of 0.989 and an AUC value of 0.986 [42].

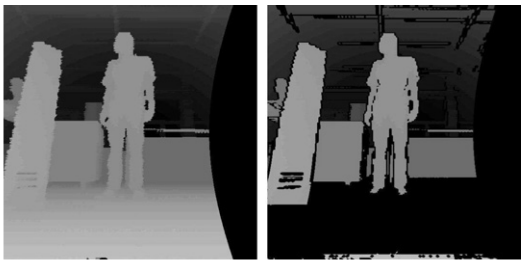

Fig. 9. *Left*: Range image generated with all 3D scan points. *Right*: Range image with removed ground points and without grey-value interpolation at range jumps.

4.2. Object detection using range and reflectance data

Our second approach to object classification works with features extracted from rendered depth and reflectance images. Feature-based systems operate usually much faster than pixel-based systems [47]. The features used here are also used in [24,47]. Fig. 11 shows the eleven basis features, i.e., edge, line, diagonal, and center surround features. Since they are compositions of rectangles, they are computed with several look-ups in an integral image or rotated integral image and subtractions weighted with the area of the black and white rectangles. An integral image I is an intermediate representation for the image N containing the sum of grey-scale pixel values $I(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} N(x', y')$. The integral image is computed recursively in linear time [47]. Rotated feature values can be computed in rotated integral images [24].

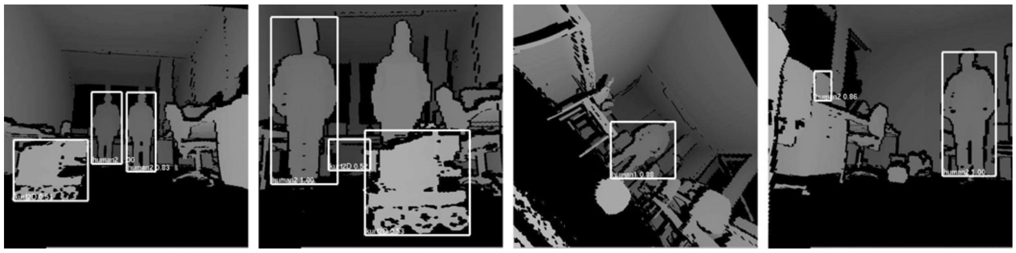

Fig. 10. Example scenes with detected objects (human & robot Kurt3D). The 4th picture shows a false detection of a wrongly recognized human above the table.

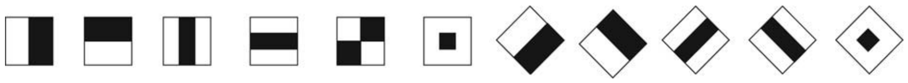

Fig. 11. Edge, line, diagonal and center surround features are used for classification.

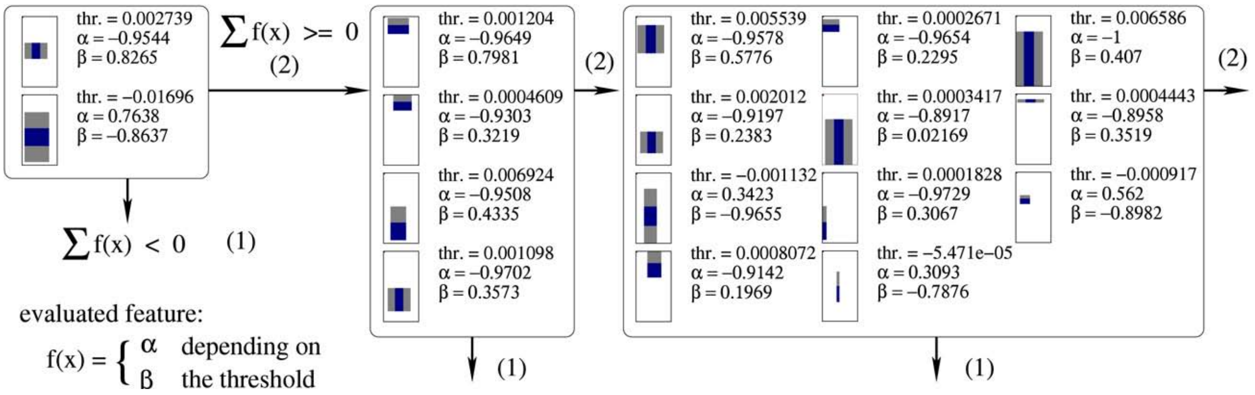

Fig. 12. The first three stages of a cascade of classifiers to detect an *office chair* in depth data. Every stage contains several simple classifiers that use Haar-like features with a threshold and return values α, β.

The base resolution of the object detector is, e.g., 30×30 pixels, thus, the set of possible features in this area is very large (642,592 features, see [24] for calculation details). The set of rectangle features is not minimal.

To detect a feature over an image area, a threshold is required for comparing the sum of pixel values of the classifier's feature regions. This threshold is determined automatically during a fitting process, such that a minimal number of examples are misclassified. Furthermore, the return values (α, β) of the feature are determined, such that the error on the examples is minimized. The examples are given in a set of images to be classified as positive or negative. The set is also used in the learning phase that is sketched next.

Learning a classifier. The Gentle Adaptive Boost Algorithm (Ada Boost) is a variant of the powerful boosting learning technique [11]. It is used to select a set of simple features to achieve a given detection and error rate. According to [23], it is the most successful learning procedure tested for face detection applications. The result of AdaBoost learning is a single classifier that consists of an accumulation of feature classifiers, with a given detection rate.

The performance of a single classifier is not suitable for object classification, since it produces a high hit rate, e.g., 0.999, but also a high error rate, e.g., 0.5. Nevertheless, the hit rate is significantly higher than the error rate. To construct an overall good classifier, several classifiers are arranged in a cascade, i.e., a degenerated decision tree. In every stage of the cascade, a decision is made whether the image contains the object or not. The number of feature classifiers used in each classifier may increase with additional stages. Fig. 12 shows part of a learned cascade for the object *office chair* in depth data.

Applying the cascades. The detection starts with the smallest classifier size, e.g., 16×40 pixels for the human classifier, 23×30 for the robot classifier. The image is searched from top left to bottom right by applications of the cascade. To detect objects on larger scales, the detector is rescaled. An advantage of the used features is that they are easily scalable. Each feature requires only a fixed number of look-ups in the integral image, independent of the scale. Time-consuming picture scales are not necessary to achieve scale invariance. Fig. 15 shows examples of the detection.

To decrease the false detection rate, we combine the cascades of the depth and reflectance images. There are two possible ways for combining: Either the two cascades run interleaved or serial and represent a logical "and" [32,35]. The joint cascade decreases the false detection rate close to zero. To avoid the reduction of the hit rate, several different off-screen rendered images are used, where the virtual camera is rotated and the apex angle is changed [35].

Efficient classifier learning. Learning needs multiple positive and negative examples to construct a classifier. Negative examples are supplied by using arbitrary grey-scale images. To make training efficient, we developed a strategy to generate many positive training images. After the object that has to be learned was scanned once, the 3D data points belonging to the object are manually extracted. Afterwards, different backgrounds, i.e., grey-scale images, are placed behind the object using a 3D rendering tool. Finally, the object is virtually rotated in 3D and views are saved using off-screen drawing mode. Fig. 13 shows the object *printer* in front of two backgrounds. Using just one 3D scan to learn an object does not erode the quality of object detection.

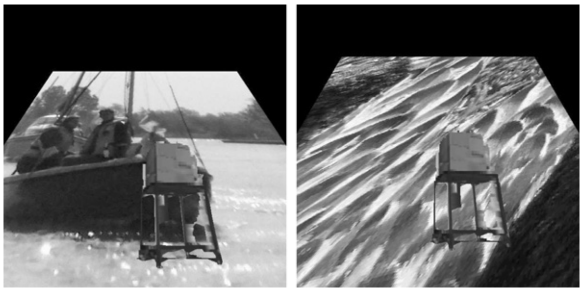
Fig. 13. The object *printer* in front of two different backgrounds.

4.3. Object point estimation

After detecting an object, or rather, an object hypothesis, in the 2D projection, we next retrieve the corresponding 3D points using ray tracing. All 3D points that have been projected into the classified 2D area are back-projected into 3D using a special OpenGL projection matrix. Fig. 14 (right) shows a rendering of ray traced 3D points.

4.4. Model matching and evaluation

After the 3D points (set $\mathcal{D}$) that contain the object is found, a given 3D model from the object database is matched into the point cloud. The model $\mathcal{M}$ is contained in the object database as a 3D point cloud. ICP (cf. Section 2, Eq. (1)) is used again for matching. After that, the matching result is evaluated using normalized subsampling. Fig. 15 shows results of the 3D object detection.

5. Semantic 3D maps

In this section, we integrate the previously described algorithms into a complete system for semantic 3D mapping. Prior to the mapping, the object database needs to be initialized and filled with object descriptions both for the 2D and 3D representations, in our case. Positive examples for the respective training procedures mentioned earlier are generated by scanning the objects (in the extreme case, scanning each of it just once), and generating many different views from them, as far as needed for working with 2D renderings. Moreover, the knowledge base or bases coming together with the semantic map have to be set up. In the current state of our work, this part is sparsely covered, basically consisting of the constraint network for coarse scene interpretation (Fig. 4), which gets used in plane interpretation, as described. We will get back to this issue in the discussion.

We start this section with recapitulating the bottom-up data processing cascade that is depicted in Fig. 1 and that we have used for describing our approach until here in this paper. After that, we will briefly describe an example of top-down influence among the processing modules of Fig. 1.

5.1. Bottom-up processing of 3D scan data

Semantic mapping in our system, as described until here, is done by the following steps (cf. Fig. 1):

(1) *6D SLAM.* 3D scans of the environment are acquired (in teleoperation). They are registered globally in 6DoF.
(2) *Scene interpretation.* 3D surfaces are extracted and labeled using their orientation.
(3) *Object detection.* Instances of known object classes are identified in each single 3D scan based on 2D renderings (possibly with interpreted surfaces removed). The 6D object poses are estimated.
(4) *Visualization* of the semantic map.

Due to the 3D nature of the semantic map, it needs to be rendered for visualization. In the full 3D map, the following information is available:

- Object information, i.e., the label of the selected object is visualized. Note: In the current implementation, labels are attached in 3D to 3D objects, so that a label might be viewed from the back, e.g., Fig. 16, bottom row. In addition, the 6D object pose is available as $(x, y, z, \theta_x, \theta_y, \theta_z)$.
- The global 3D coordinates, i.e., $(x, y, z) \in R^3$ of all scan points.
- For each scan point, the information whether it is part of some scene structure or detected object, and, if so, which one.
- Distances between all pairs of 3D points and to the current view pose.
- Information about the robot poses at which a 3D scan was acquired and (estimated) trajectory data.

With the following data set, we demonstrate by example, how the previously described method yields a semantic map. The data set was taken in our institute building. A corridor and some offices have been mapped. In the initialization phase, we added 12 objects (different chairs, printers, plants, and a human standing upright) to the database. After scanning the area, the 3D model contained 32 3D scans with a total of about 2.5 million points. The total map postprocessing time was 4.5 minutes. 82% of all 3D points were labeled and since we had one object detection failure, namely, one false negative of a fire extinguisher, ~0.1% of the labels were wrong. Fig. 16 visualizes the results. An animation of the scene is available at www.informatik.uni-osnabrueck.de/nuechter/videos.html. The complete scan data set (distance and remission values) is available at http://kos.informatik.uni-osnabrueck.de/3Dscans/ (Univ. Osnabrück data set).

5.2. Top-down data flow in semantic mapping

One of the intended uses of the semantic stance in semantic maps is to help in the mapping process itself by providing in a top-down direction hypotheses or constraints for processing the sensor data on lower processing levels. This would add feed-back loops to the straightforward processing cascade in Fig. 1 Our current work, as exemplified on the office building data set includes a case of that

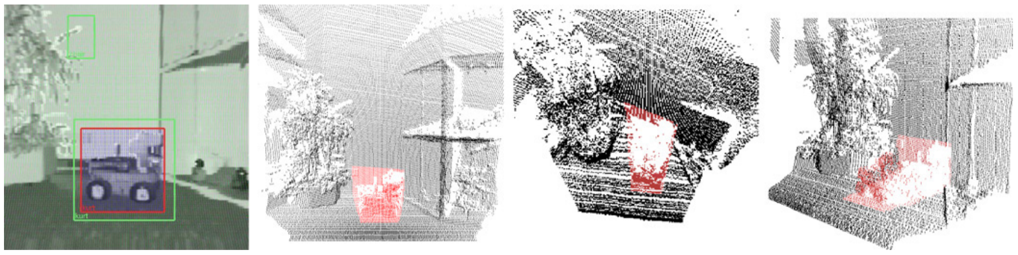
Fig. 14. Object point estimation by ray tracing. *Left*: All points inside a detection area are extracted. *Next*: 3D views of the scan points taken from different virtual view poses. 3D points inside the detector area (viewing cone) are drawn in lighter shade (red).

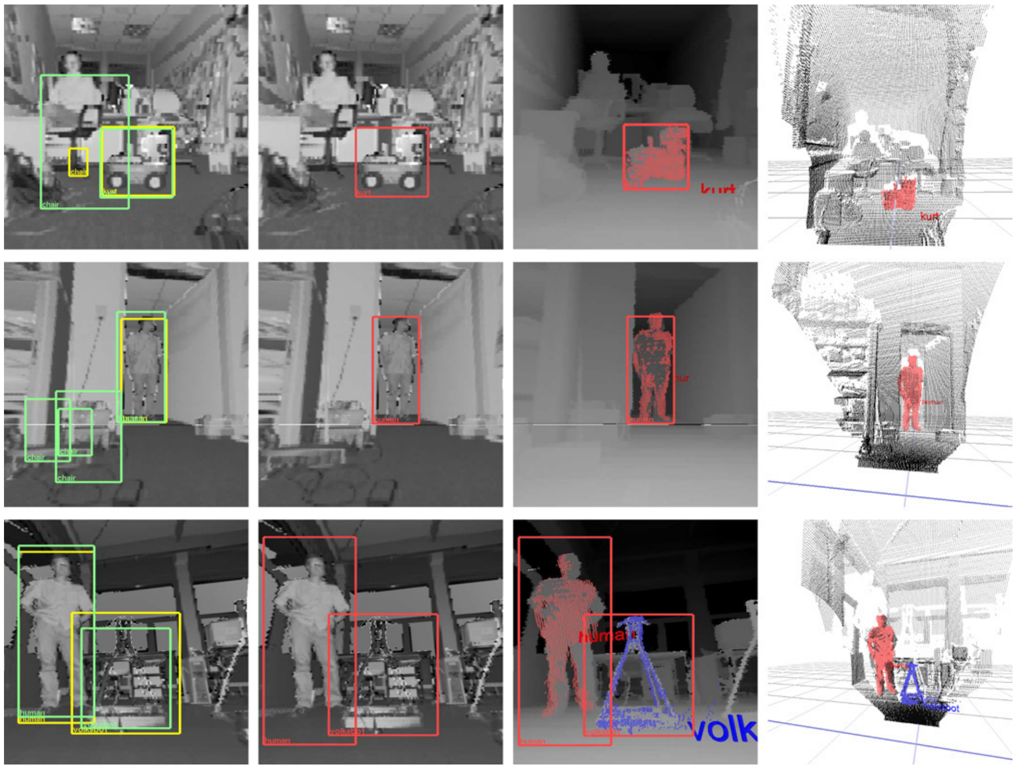

Fig. 15. Examples of object detection and localization. Top: Kurt robot; middle: human; bottom: human and Volksbot robot. Columns from left to right: (1) Detection using a single cascade of classifiers in reflection and depth images. (2) Detection using the combined cascade. (3) The respective matched 3D models are superimposed to the depth images. (4) Detected objects in the raw scanner data, i.e., in the point representation.

kind, which we will present here. The issue in more generality will be discussed in the following Section 6.

Our current physical 3D scanner (Fig. 2) would produce a more or less systematic measuring error, due to a synchronization problem in controlling the tilt servo: Each single scan of a deep area, like a corridor, will map the true environment into the 3D point cloud with a slight warp. For example, a scan straight into a long corridor, visualized from the side, will generate a slightly banana shape. Fig. 17, top right, gives an impression of this effect, presenting a side view of registered scans. Registering several banana scans of a straight corridor all with a like curvature will produce a banana model of the corridor, if registered optimally using ICP, as described in Section 2: The complete model will in all likelihood show the average warp of the single scans, like in Fig. 17, top right.

Fig. 17, bottom right presents a fix of that problem, based on exactly the same individual scans. Here is how it was done: The warp in every single scan is sufficiently little so that the floor plane is correctly detected and interpreted, using the procedure in Section 3.1 So when registering two subsequent scans, the individual floor planes of the two scans can be forced to match. The model from constrained registration in Fig. 17 has defined the floor plane of the first (leftmost) scan as the global one; all others are registered by fixing their individual floor planes on this one, and by registering, after that, all other points are according to ICP optimally, subject to the floor plane constraint. The bottom of the registered 3D map, as shown in Fig. 17, is still uneven, because the measured scan points of every single scan are unchanged, so the floor points of every single scan keep their positions around (rather than on) the floor plane.

So why is this interesting in terms of semantic mapping? We see an example here of top-down influence of higher-level information (coarse scene interpretation) to lower-level processing (scan registration). Normally in semantic mapping one would expect such a feed-back to work aided by reasoning in the background knowledge — that is, supported by some suitable extension of the constraint network for plane labeling, in our example. Our current implementation has in fact achieved the effect in a different way, namely, by a targeted addition to the registration module. This leads to discussing how far our own current version of semantic mapping has advanced, and what this tells about semantic mapping in general.

6. Discussion and outlook

To quote from the introduction, we have stated:

> A *semantic map* for a mobile robot is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine.

Having presented our results, let us discuss how far we have advanced towards semantic maps, what remains to be done, and what are plausible ideas from the literature about how to proceed from here.

We have clearly specialized semantic maps in the sense that the sensor for mapped data is a 3D laser scanner. There are good reasons for using this sensor: it yields precise, rich, and even bimodal (distance, reflection) data in a fused form. However, there is no claim that using a 3D scanner is required for semantic mapping. Regular 2D laser distance scans alone may be of help; on the other hand, high-resolution color texture information as by a camera would be of obvious value, too. The concept of semantic mapping in itself is independent of the sensor configuration used. In fact, our future plans include working with different such configurations.

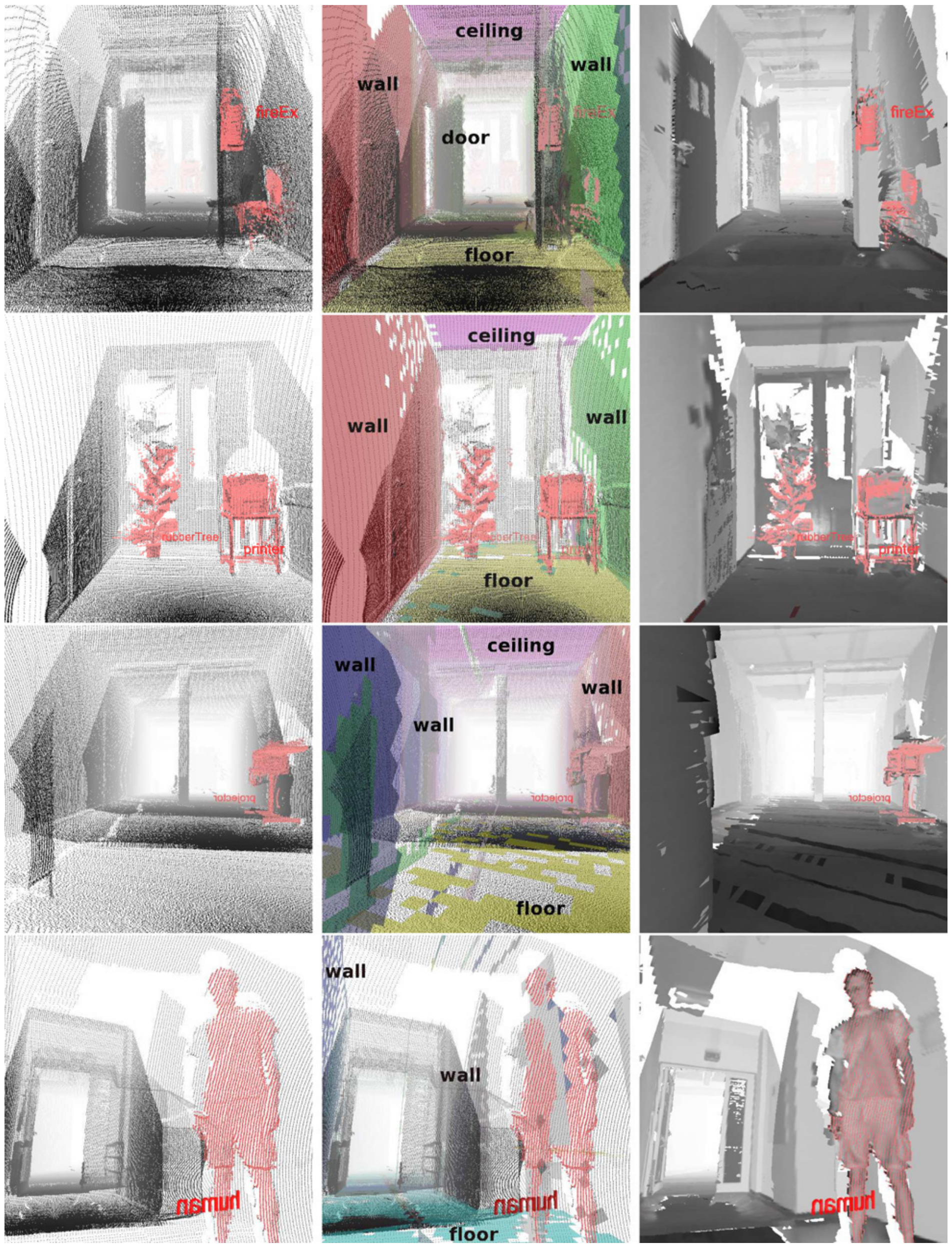

Fig. 16. Renderings of a 3D semantic map of an office building (AVZ, University of Osnabrück). Labels derived by coarse scene interpretation are black, whereas lighter labels result from object localization. *Left*: 3D point cloud with just objects labeled. *Middle*: Coarse scene interpretation. *Right*: Reflectance values.

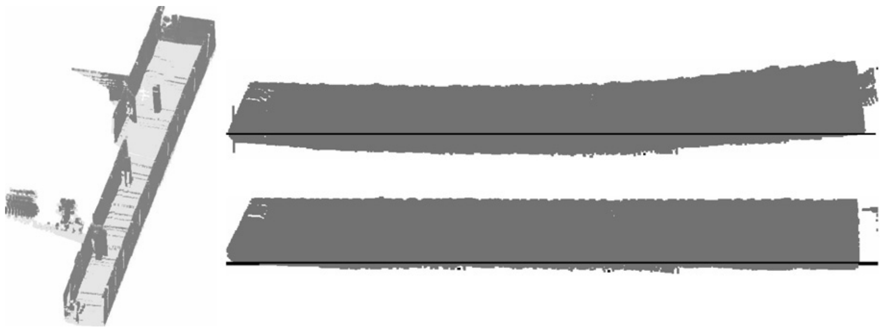

Fig. 17. *Left*: Map, top view. *Right, top*: The unconstrained 3D mapping shows a banana-shaped form. *Right, bottom*: The horizontal justification and the constrained mapping lead to qualitatively correct maps.

To have a semantic map rather than a map including features labeled with tags, the informal definition from the introduction requires that the map be connected with background knowledge that can be used for reasoning about objects or object classes present in the map. In this respect, we have only scratched the surface here. As described, we have used the constraint network for interpreting planes in the scene; the constraint solver is implemented by Prolog backtracking, and the representation of concrete planes in a scene in terms of Prolog facts is generated automatically from the scan data. So this is clearly a form of inference in terms of a KR language (constraints), but it is specialized and limited. For the future, we are planning to use two more formalisms and corresponding inference formats in the context of semantic maps, namely, Description Logics [4, Ch. 9] and HTN plans [14, Ch. 11].

Our intended use of DL in reasoning about semantic maps is inspired by [29]. The point is to reason about aggregates of basic objects based on sensor data taken in the scene, which may be noisy and incomplete; the example in [29] is to discover and classify covers on a table, based on camera data. The issue in that type of reasoning is to match two strands of information, namely: (1) bottom-up classifications of data from the scene like detected pieces of silverware, which come from a processing cascade in analogy to the one used in this paper, and (2) top-level expectations about the scene, which are generated from interpreting detected basic objects as components of complex aggregates, from which the presence of other, yet to be detected components is hypothesized.

The intended purpose of HTN planning is to provide a plan-based layer in robot control [27]. A semantic map is required, as generating HTN plans online needs information in symbolic form about recent facts. We have proposed a way to extract this efficiently from a DL domain model [17], thereby allowing DL to be the sole KR formalism to accompany the semantic map. Details are out of scope here. However, a point to note is that the KR and reasoning aspect of the work presented here need improving so that the semantic information contained in our maps can be brought to bear. This is another topic for future work.

The DL-based reasoning in [29] gives another example how bottom-up sensor data processing can profit from top-down information — in this case by using top-down hypotheses about basic objects and their positions in the scene to expect. We have given an example how top-down reasoning can help correct erroneous sensor data; the expectation-driven interpretation in [29] strongly suggests that correcting other mundane sensor data deficits like occlusion could also profit from using top-down semantic information.

This leads to the last issue to mention here: uses of semantic maps. We take for granted that many applications of mobile robots could benefit from a semantic stance in their maps, seen from an application or user perspective; in particular robot planning on a symbolic level is impossible without such a map. We want to emphasize that, in addition to that, the map building and domain model maintenance processes would profit from the availability of a semantic map, and they would profit in particular from the possibility of including top-down information in their work. Seen from the perspective of robot control design, the first and foremost profiteer from the availability of a semantic map on a mobile robot appears to be the robot itself.

7. Coda

A semantic map integrates spatial information about some environment and the locations of structures and objects of known classes. If an autonomous robot is supposed to reason on a symbolic level about its environment and about its own action, then it needs some form of semantic map. The robotics mapping literature has mostly dealt with metric and geometric information — for the perfect reason that geometry comes first for a mobile robot supposed to avoid collision and localize itself. The robotics literature includes a few examples of a semantic stance in mapping. Most of them work bottom-up, i.e., part of the given sensor data (mostly camera and/or laser scanner) are interpreted in semantic terms. Only a small fraction of these approaches includes top-down aspects, i.e., a change or completion of the very sensor data as triggered by reasoning on the semantic level. We have argued that only the combination of both directions would bring the full potential of semantic mapping to bear.

We have presented a comprehensive study in bottom-up semantic mapping based on 3D laser scan data. We have left unaddressed quite a number of issues, such as the integration of camera data, the role and potential of learning, and formal and technical details concerning each and every one of the modules described. Some of these issues (definitely technical details) had to be skipped here — more about it is available in the respective references. Our purpose here has been to present comprehensively semantic mapping and the sort of semantic maps that can be built.

The top-down direction in semantic mapping is mostly unexplored. So in addition to presenting one implemented example, we could only sketch our understanding of its purpose and potential. Extending our study by integrating selected top-down control strands will be a main direction of the work ahead.

References

[1] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, S. Thrun, Learning hierarchical object maps of non-stationary environments with mobile robots, in: Proc. 18th Conf. Uncertainty in AI, UAI '02, Edmonton, Alberta, Canada, August 2002.

[2] P. Besl, N. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14 (2) (1992) 239–256.

[3] R. Biswas, B. Limketkai, S. Sanner, S. Thrun, Towards object mapping in dynamic environments with mobile robots, in: Proc. Conf. Intelligent Robots and Systems, IROS '02, Lausanne, Switzerland, September 2002.

[4] R. Brachman, H. Levesque, Knowledge Representation and Reasoning, Morgan Kaufmann, 2004.

[5] W. Burgard, M. Hebert, World modeling, in: Springer Handbook of Robotics, Springer, 2008.

[6] H. Cantzler, R.B. Fisher, M. Devy, Quality enhancement of reconstructed 3D models using coplanarity and constraints, in: Proc. annual Symp. for Pattern Recognition, DAGM '02, Zürich, Switzerland, September 2002, pp. 34–41.

[7] H. Christensen, H.-H. Nagel (Eds.), Cognitive Vision Systems – Sampling the Spectrum of Approaches, in: Lecture Notes in Computer Science (LNCS), vol. 3948, Springer Verlag, Berlin, 2006.

[8] S. Coradeschi, A. Saffiotti, An introduction to the anchoring problem, in: Perceptual Anchoring, Robotics and Autonomous Systems 43 (2–3) (2003) 85–96. (Special issue). Online at http://www.aass.oru.se/Agora/RAS02/.

[9] C. Dornhege, A. Kleiner, Behavior maps for online planning of obstacle negotiation and climbing on rough terrain, in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS '07, San Diego, CA, USA, 2007.

[10] M. Drew, T. Lee, A. Rova, Shape Retrieval with Eigen-CSS Search. Technical report, School of Computing Science, Simon Fraser University, Vancouver, B.C., Canada V5A 1S6, 2 2005.

[11] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Machine Learning: Proc. 13th Intl. Conf., 1996, pp. 148–156.

[12] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernandez-Madrigal, J. Gonzalez, Multi-hierarchical semantic maps for mobile robotics, in: Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems, IROS '05, Edmonton, Canada, 2005, pp. 3492–3497.

[13] A. Georgiev, P.K. Allen, Localization methods for a mobile robot in urban environments, IEEE Transactions on Robotics and Automation (TRO) 20 (5) (2004) 851–864.

[14] M. Ghallab, D. Nau, P. Traverso, Automated Planning: Theory and Practice, Morgan Kaufmann, 2004.

[15] O. Grau, A scene analysis system for the generation of 3-D models, in: Proceedings IEEE Intl. Conf. Recent Advances in 3D Digital Imaging and Modeling, 3DIM '97, Ottawa, Canada, May 1997, pp. 221–228.

[16] S. Harnad, The symbol grounding problem, Physica D 42 (1990) 335–346.

[17] R. Hartanto, J. Hertzberg, Fusing DL reasoning with HTN planning, in: KI 2008: Advances in Artificial Intelligence. 31st Annual German Conference on AI, Proceedings, 2008, September, Kaiserslautern, Germany, in: LNAI, vol. 5243, Springer, pp. 62–69.

[18] M. Hebert, M. Deans, D. Huber, B. Nabbe, N. Vandapel, Progress in 3-D mapping and localization, in: Proc. 9th Intl. Symp. Intelligent Robotic Systems, SIRS '01, Toulouse, France, July 2001.

[19] B.K.P. Horn, Closed-form solution of absolute orientation using unit quaternions, Journal of Optical Society of America A 4 (4) (1987) 629–642.

[20] L. Iocchi, S. Pellegrini, Building 3d maps with semantic elements integrating 2d laser, stereo vision and imu on a mobile robot, in: Proc. 2nd ISPRS Intl. Workshop 3D-ARCH 2007: 3D Virtual Reconstruction and Visualization of Complex Architectures, Zurich, Switzerland, July 2007.
[21] H.A. Kestler, S. Sablatnög, S. Simon, S. Enderle, A. Baune, G.K. Kraetzschmar, F. Schwenker, G. Palm, Concurrent object identification and localization for a mobile robot, KI – Künstliche Intelligenz (2000) 23–29.
[22] B. Kuipers, Y. Byun, A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations, in: Toward Learning Robots, 1993.
[23] R. Lienhart, A. Kuranov, V. Pisarevsky, Empirical analysis of detection cascades of boosted classifiers for rapid object detection, in: Proc. German 25th Pattern Recognition Symp., DAGM '03, Magdeburg, Germany, September 2003.
[24] R. Lienhart, J. Maydt, An extended set of haar-like features for rapid object detection, in: Proc. IEEE Conf. Image Processing, ICIP '02, New York, USA, September 2002, pp. 155–162.
[25] B. Limketkai, L. Liao, D. Fox, Relational object maps for mobile robots, in: Proc. 19th Intl. J. Conf. Artificial Intelligence, IJCAI '05, Edinburgh, Scotland, 2005.
[26] M. Magnusson, T. Ducket, A Comparison of 3D Registration Algorithms for Autonomous Underground Mining Vehicles, in: Proc. Second Eur. Conf. Mobile Robotics, ECMR '05, Ancona, Italy, September 2005, pp. 86–91.
[27] D. McDermott, Robot planning, AI Magazine 13 (2) (1992) 55–79.
[28] F. Mokhtarian, Silhouette-based isolated object recognition through curvature scale space, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 17 (1995) 539–544.
[29] B. Neumann, R. Möller, On scene interpretation with description logics, in: H. Christensen, H.-H. Nagel (Eds.), Cognitive Vision Systems – Sampling the Spectrum of Approaches, in: Lecture Notes in Computer Science (LNCS), vol. 3948, Springer Verlag, 2006.
[30] P. Newman, D. Cole, K. Ho, Outdoor SLAM using visual appearance and laser ranging, in: Proc. IEEE Intl. Conf. Robotics and Automation, ICRA '06, Florida, 2006.
[31] A. Nüchter, K. Lingemann, J. Hertzberg, Extracting drivable surfaces in outdoor 6D SLAM, in: Proc. 37rd Intl. Symp. on Robotics, ISR '06 and Robotik 2006, Munich, Germany, May 2006.
[32] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, Accurate object localization in 3D laser range scans, in: Proc. 12th Intl. Conf. Advanced Robotics, ICAR '05, July 2005, pp. 665–672.
[33] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6D SLAM - 3D Mapping Outdoor Environments, in: Quantitative Performance Evaluation of Robotic and Intelligent Systems, Journal of Field Robotics (JFR) 24 (8/9) (2007) 699–722. (Special Issue).
[34] A. Nüchter, H. Surmann, J. Hertzberg, Automatic model refinement for 3D reconstruction with mobile robots, in: Proc. 4th IEEE Intl. Conf. Recent Advances in 3D Digital Imaging and Modeling, 3DIM '03, Banff, Canada, October 2003, pp. 394–401.
[35] A. Nüchter, H. Surmann, J. Hertzberg, Automatic classification of objects in 3d laser range scans, in: Proc. 8th Conf. Intelligent Autonomous Systems, IAS '04, Amsterdam, The Netherlands, March 2004, pp. 963–970.
[36] P. Pfaff, R. Triebel, W. Burgard, An efficient extension to elevation maps for outdoor terrain mapping and loop closing, International Journal of Robotics Research (IJRR) (February) (2007).
[37] I. Posner, D. Schroeter, P. Newman, Describing composite urban workspaces, in: Proc. IEEE Intl. Conf. Robotics and Automation, ICRA '06, Rome, May 2007, pp. 4962–4968.
[38] The RANSAC (random sample consensus) algorithm, 2003 http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/FISHER/RANSAC/.
[39] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall, Englewood Cliffs, NJ, 2003.
[40] V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, D. Hogg, Automated 3D reconstruction of interiors with multiple scan–views, in: Proc. SPIE, Electronic Imaging '99, SPIE's 11th Annual Symp., San Jose, CA, January 1999.
[41] R. Siegwart, I. Nourbakhsh, Introduction to Autonomous Mobile Robots, MIT Press, Cambridge, MA, 2004.
[42] S. Stiene, K. Lingemann, A. Nüchter, J. Hertzberg, Contour-based object detection in range images, in: Proc. 3rd IEEE Intl. Symp. on 3D Data Processing, Visualization and Transmission, 3DPVT '06, June 2006.
[43] H. Surmann, K. Lingemann, A. Nüchter, J. Hertzberg, A 3D laser range finder for autonomous mobile robots, in: Proc. 32nd Intl. Symp. Robotics, ISR '01, Seoul, April 2001, pp. 153–158.
[44] H. Surmann, A. Nüchter, K. Lingemann, J. Hertzberg, 6D SLAM A preliminary report on closing the loop in six dimensions, in: Proc. 5th IFAC Symp. on Intelligent Autonomous Vehicles, IAV '04, Lisbon, July 2004.
[45] SWI Prolog, 2003. http://www.swi-prolog.org/.
[46] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
[47] P. Viola, M.J. Jones, Robust real-time face detection, International Journal of Computer Vision 57 (2) (2004) 137–154.
[48] A. Walthelm, R. Kluthe, A. Mamlouk, Ein 3D Weltmodell zur teilaktiven Positionsverfolgung in komplexen dynamischen Umgebungen, in: Proceedings AMS, 1999, pp. 272–281.
[49] D.L. Waltz, Understanding Line Drawings of Scenes with Shadows, McGraw-Hill, New York, U.S.A, 1975.
[50] O. Wulf, K.O. Arras, H.I. Christensen, B.A. Wagner, 2D mapping of cluttered indoor environments by means of 3D perception, in: Proc. IEEE Intl. Conf. Robotics and Automation, ICRA '04, New Orleans, USA, April 2004, pp. 4204–4209.

Andreas Nüchter is a research associate at the University of Osnabrueck. His past affiliations were with the Fraunhofer Institute for Autonomous Intelligent Systems (AIS, Sankt Augustin) and University of Bonn, from where he received the diploma degree in computer science in 2002 (best paper award by the German society of informatics (GI) for his thesis). He holds a doctorate degree (Dr. rer. nat) from the University of Bonn. His research interests include reliable robot control, 3D environment mapping, 3D vision, and laser scanning technologies, resulting in fast 3D scan matching algorithms that enable robots to map their environment in 3D using 6 degrees of freedom poses. The capabilities of these robotic SLAM approaches were demonstrated at RoboCup Rescue competitions, ELROB and several other events. He is a member of the GI and the IEEE.

Joachim Hertzberg is a full professor for computer science at the University of Osnabrück, where he is heading the Knowledge-Based Systems lab and is currently the Dean of the school of Mathematics/Computer Science. He has graduated in Computer Science (U. Bonn, 1982; Dr.rer.nat. 1986, U. Bonn; habilitation 1995, U. Hamburg). His former affiliations were with GMD and with Fraunhofer AIS in Sankt Augustin. His areas of scientific interest are Artificial Intelligence and Mobile Robotics, where he has contributed to action planning, robot localization and mapping, plan-based robot control, active sensing, robot control architectures, temporal reasoning, logical reasoning about action and change, constraint-based reasoning, and applications of these. In these areas, he has written or edited 6 books and published over 90 refereed or invited papers in books, journals or conferences.