Reasoning About Geometry: An Approach Using Spatial-Descriptive Ontologies

Markus Eich and **Frank Kirchner**[1]

Abstract. Autonomous robots perceive their environment usually through exteroceptive sensors which generate features in the spatial domain. Finding the corresponding semantic or symbolic description can be referred to as the *anchoring* or *grounding* problem. Some of the latest research in robotics is dedicated to the generation of semantic maps. This includes labeling of metric maps which are provided by 3D point clouds. Usually, a model database has to be generated beforehand in order to classify objects in the spatial domain. In our approach, we propose a semantic classification based on object primitives and their spatial 3D relationship. We introduce spatial feature descriptors which can be mapped directly to a symbolic level. By looking at the relationships in the spatial domain, we are able to describe and classify known objects without model learning. The spatial entities can be defined directly using domain knowledge and ontologies. We apply our approach to a mobile dual-manipulator robot with application to logistic scenarios. We propose an ontology-based description of an indoor environment and a probabilistic reasoning approach based on spatial feature descriptions. The paper will give an overview of our current work on applying AI methods to logistics scenarios.

1 INTRODUCTION

Mobile robots are usually moving in and interacting with a 3D environment, so 3D perception is mandatory for such systems. Besides path-planning and map-building approaches, which have been thoroughly investigated over the last decade, robots need an understanding about their environment.

As more processing power and more sophisticated 3D range sensors become available, an increasing number of approaches is dealing with the generation of coherent, metric 3D models. Nevertheless, it becomes clear that simple metric information about the environment is not sufficient to establish real autonomy in terms of interacting with and reasoning about the robot's environment. For intelligent behavior, it is preferable to send the robot high-level commands, like "Move to the table in the office!" or "Take the box from the table in the office!" instead of sending the robot to pure metric coordinates.

The problem is always to anchor the spatially detectable features to a semantic level. A typical indoor environment is shown in Figure 1 which was generated by our robot in a logistic setup. The task of the robot is to reach the shelf and grab a certain object. While in our experiments the object detection is based on vision and SIFT-features, the shelf itself is detected using a tilting 3D laser scanner. Learning a geometric model of a shelf would include learning different types of shelves, with different numbers of boards and with different sizes.

[1] Robotics Innovation Center, DFKI Bremen, Germany, email: first_name.last_name@dfki.de

Our key idea is to separate the model from the physical appearance and include a divide and conquer strategy. On a semantic level a typical shelf can be described by the following description:

- All rectangular shapes have an elongation of less than two meters.
- The number of rectangular shapes is more than one.
- All rectangular shapes covered an area less than two square meters.
- All rectangular shapes are parallel to each other and to the floor.
- All rectangular shapes have the same size.
- All rectangular shapes have the same vertical alignment.

This intuitive semantic description will hold for most of the shelves which are used in our lab for experiments. An important feature of the description above is that all semantic relationships and features are measurable in the spatial domain. This means, by analyzing the extracted shape information and their relationship to each other, the bridging between the spatial and semantic representation can be achieved for some shape primitives. In the remainder of this work we will describe how objects can be detected using spatial/semantic feature descriptors. The process of analyzing the environment using 3D

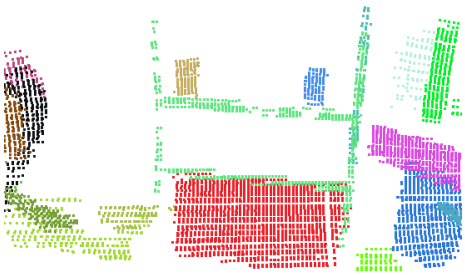

Figure 1. A point cloud taken by the robot in our lab. It shows a shelf and some objects which are contained in the shelf.

laser range finders is basically performed in two consecutive steps. In a first step, laser data is acquired using a tilting laser setup or 3D LIDAR (light detection and ranging) system and matched to an existing point cloud model. In a second step of the scene-recovery process, geometric information is extracted from the merged point cloud data. This can be achieved by using 2D plane extraction [16] or the direct extraction of 3D primitives [13]. Some common surface reconstruction methods include the ball pivoting algorithm [1] and

the Delaunay triangulation [8].
Most of the described algorithms are aimed at reconstructing accurate surfaces and finding their application in reverse engineering of 3D structures. The accuracy of these algorithms results in high computational costs. They are thus not suited for robotic applications because the surface reconstruction can take up to several minutes.
The plane extraction algorithm described in [16] works well on noisy sensor data and uses an iterative region-growing algorithm. The drawback of this approach is that it needs organized point clouds (i.e. the neighbor of each point is known). This is common for 3D LIDAR systems but not true for merged or registered scans. The approach we present in this paper will allow scene recovery from unorganized point sets and will also extract features from the spatial domain. The approach described in [9] provides a fuzzy clustering algorithm in order to segment a 3D scenery into cluster subsets without model learning. An approach of iterative triangulation of unorganized point clouds is described in [10]. All the described algorithms above are dealing with the spatial domain and are usable for identifying regions in LIDAR generated 3D point cloud data.
Coming from the semantic side, [6] and [7] describe how semantic maps are used for high-level planning and spatial reasoning. The authors describe in their work the bridging between the spatial domain and the semantic domain which they call S-Box (spatial box) and T-Box (taxonomy box). The semantic interpretation of physical objects is carried out by optical marker identification but not directly on spatial interpretation of point cloud data.
In the work described in [11], a constraint network is chosen in order to identify spatial entities such as wall, floor, and doors. That work shows how an environment can be described efficiently by using only the two constraints "parallel to" and "orthogonal to". We will extend this idea by adding additional spatial features which can be directly extracted using shape recovery on segmented regions.

2 SHAPE RECONSTRUCTION

3D data of the scenery is generated using a tilting laser scanner. From this, we generate a data set Ψ with vector elements $\psi_i = (x_i, y_i, z_i)^T$, representing points in space. We process the raw data in two consecutive steps. First, we apply a fast region-growing algorithm to segment the point cloud data into regions which belong to a coherent plane. In a second step, the geometric data of the detected planes are extracted. The shape of the planes are of major interest. Therefore, the segmented point regions are polygonized using alpha shapes. Our region-growing approach follows partly the algorithm described in [12], with the extension that our approach is able to process unorganized point clouds by efficiently integrating K-nearest neighbor (KNN) search into the growing process. Computationally, the most expensive function in the algorithm is the KNN search which can be approximated with a runtime of $\mathcal{O}(n \log(n))$ [15]. Due to the requirement of being able to process unorganized point clouds which occur in registered scans, composed by using ICP [3], we have to optimize the KNN search during the region-growing process.
Our key idea is to perform the cost-intensive KNN search at the beginning and store each point separately with a pointer to its KNNs. The growing plane keeps track of its growing frontier, i.e. each plane has its assigned nearest neighbors which are the next candidate for the growing process. During the region-growing process, the KNNs of each point, which is grown into the plane, are merged with the region frontier. We briefly summarize Rabbani's approach in Algorithm 1 and state our extension to this algorithm.

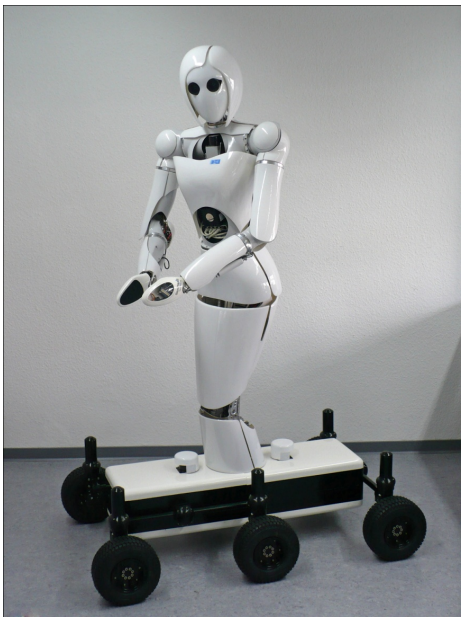

Figure 2. The semi-humanoid robot Aila, which was developed for dual manipulation tasks in a logistic scenario. The robot uses a tilting 3D LIDAR system for precise point cloud generation in 3D space. A second PMD LIDAR is used for obstacle avoidance.

The problem concerning runtime in the base algorithm is that the nearest neighbor search has to be performed each time a new seed point is selected. Searching the nearest neighbor can be done in constant time if a single scan line is taken from a tilting laser or a 3D LIDAR system. This does not hold for merged, registered or filtered scans because the information of the nearest neighbor is lost.

Algorithm 1 Region Growing Algorithm

```
Input : Pointset Ψ
Output : Subsets R_1, ..., R_n of regions belonging to one plane
while Ψ not empty do
    Select random seed point ψ from Ψ
    Find nearest neighbor of ψ
    if MeanSquareError(R_i ∪ ψ) < δ and
    ||BestFittingPlane(R_i) − ψ||_⊥ < γ then
        Add new point ψ to R_i
    end if
end while
```

In order to extend Rabbani's algorithm to be efficient for unorganized point clouds, we modify the algorithm by performing a KNN search only once at the beginning.
Each KNN of each point is then stored using a priority queue, ordered according to the distance to the seed points. Whenever a new point is added to the current region, the priority queue of the new added point is merged into the current region-growing front. By using pointer arithmetic we reduce memory usage.
After the regions have been segmented using Algorithm 1, the shape of each segmented point region is extracted. Because the appearance of each shape is important for later scene analysis, we extract the concave hull, i.e. the polygon which approximates the shape of the original object.
We choose the alpha shape approach, which is described more detailed in [14] and [2], in order to extract the concave polygon of the

detected regions. Figure 3 gives an idea of how alpha shapes are computed. For the analytic description of shapes the reader is pointed to [14]. The alpha shapes are a special case of a Delaunay triangulation

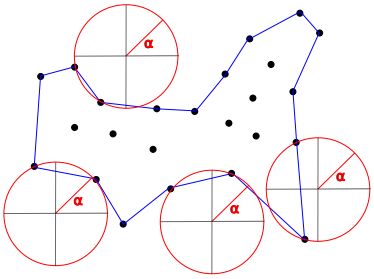

Figure 3. Alpha shape example in 2D. Alpha shapes can be described as the shape which is created if a disk (or a sphere in 3D) with the radius α is rolled along the point cloud, starting with a border point. Each time a point is hit by the disk, it becomes the new pivoting point and the previous and the new pivoting point are connected.

and are mathematically well defined. For each point in a segmented region R, a vertex is created. Two vertices p_i and p_j are connected if there exists a 3D sphere with radius α which has p_i and p_j on the boundary and does not contain any other point in R. In our implementation, we use the alpha shape reconstruction provided by [4]. Every alpha shape is then projected to a 2D plane for later shape analysis, e.g. eccentricity, size, shape, center of gravity. Note the dependency of the recovered shape on the radius α. If $\alpha \rightarrow 0$, each point in R forms a separate shape, if $\alpha \rightarrow \infty$ the convex hull is calculated. Figure 4 shows our lab scenery and the segmented planes and structures. For our approach we exploit the hypothesis that most of man-made indoor structure is of a rectangular shape. Examples for such structures are, for instance, tables, shelves, doors, walls, but also usable common objects of every-day use, such as monitors, TV-sets, microwave ovens, refrigerators, etc. In order to detect such objects, it is important to detect rectangular shapes from the set of alpha shapes calculated in the prior steps.

The detection of the rectangular shapes is done using Hough transformation. The shapes are therefore projected in a discrete 2D bitmap with fixed cell size. On this 2D image plane, the shapes can be analyzed straight forward. Based on the line segmentation results, the detected polygons are checked whether they intersect in four points and their dihedral angle is around 90°.

The detection of rectangular shapes is one of the important features needed to form the spatial feature vector described in the next section. In addition to the "rectangularness" of the shape, important features like the area covered and the expansion can be extracted. Before we introduce the spatial feature descriptors in the next chapter, we summarize the processing steps for unorganized point clouds.

- KNN search and storage in a priority queue for each point.
- Application of region-growing algorithm using the KNN-priority queues as growing frontier.
- Extraction of concave polygons using alpha shape recovery
- Projection of alpha shapes into discrete 2D bitmap.
- Detection of lines using Hough transformation.
- Detection of rectangular shapes using line analysis.
- Extraction of form features from the 2D bitmap.

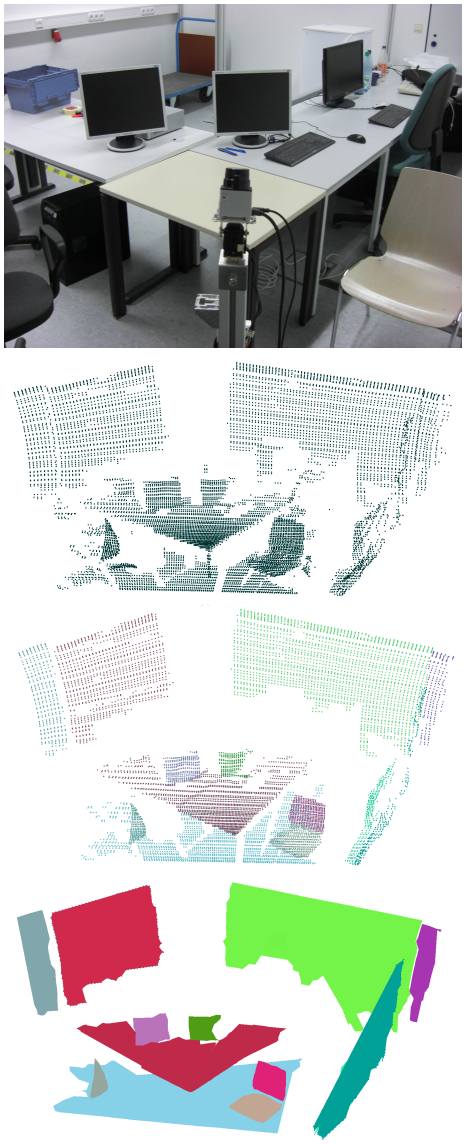

Figure 4. A lab scene and the corresponding segmentation steps. After the point cloud has been segmented and after the generation of the alpha shapes, some 2D planes are left for further feature analysis. As a human it is easy to detect the floor, the table, and the chair based only on the knowledge of spatial relationships between the objects.

3 SPATIAL FEATURE DESCRIPTION

Once the shapes have been recovered from the unorganized point cloud, the goal is to classify the structure the robot perceives and to label the structure with semantics. To make semantic labeling possible in indoor environments, we make use of some basic assumptions. If we look around in a typical indoor environment like a household environment or an office environment, it is clear that most structures are of rectangular shape and mostly parallel or orthogonal to each other.

We will explain our semantic labeling approach using a simple example: Think of two persons who are able to communicate with each other. One person is able to perceive his environment, the other is not but has a perfect knowledge about the environment. One can think of the first person being in the spatial domain (without any understanding) and the other person being in the semantic domain.

Now think of how the person in the spatial domain would describe a *table* without knowing what a table is. A few spatial features would be sufficient until the person in the semantic domain can guess what is meant by the description (e.g. rectangular shape, parallel to the floor (or ceiling), height less than one meter from the floor, etc.).What happens is that the person in the semantic domain matches the available information to its internal model. Similar to the processing in a decision tree, every additional information given will increase the likelihood for a certain entity in the semantic model space.

Similar to the example above, the robot has to extract a vector of feature descriptors of the spatial entities in order to compare them with the semantic knowledge database. In a first approach, we define a set of vectors which are able to describe spatial entities of an environment. The feature vector Φ is defined as

$$\Phi \quad = \quad (A, E, R, \Theta)^T,$$

where $A \in \mathbb{R}^2$ is the area covered by the shape, $E \in \mathbb{R}$ describes the maximum extension of the detected entity, and $R \in [0, 1]$ describes the "rectangularness" of the entity, with $R = 1$ describing a perfect rectangle. In our current approach, we only consider perfect rectangles as a binary decision. In later implementations, we also want to consider similarities to rectangles in order to increase the robustness of our approach. The reason to look for a rectangular structure is given by the observation that most artificial objects have a rectangular plane in their structure (e.g. doors, shelves, closets, walls, monitors, fridges).

$\Theta \in [0, 1]$ describes the relationship to other detected entities and is given by

$$\Theta \quad = \quad P(\omega, \sigma)$$

where $\omega \in \Omega$ is the target of the relationship and $\sigma \in \Sigma$ is the definition of spatial relationship. Ω is a set of targets, i.e. labels in the semantic domain; Σ holds the attribute space which maps semantic attributes to spatial attributes. The labels in the semantic domain are defined in a model data base and include entities like *desk, table, door*, etc.

The attributes describe the spatial relationship between the detected entities (i.e. the parent of the relationship) and the target entity. An example for an attribute is

$$\begin{array}{lcl} \sigma & \rightarrow & [0, 1] : \\ above & \rightarrow & (Pos\_Z_{parent} - Pos\_Z_{target}) < 0, \end{array}$$

which means that the target is below the parent entity. In our current implementation, we again consider a likelihood function in order to deal with uncertainties. For instance, two shapes can be parallel with the certainty of 0.9 due to noise and rounding differences in the extraction process.

$P \in [0, 1]$ maps the relationship between the parent entities and the target entity, where 1 is the maximum likelihood. Another advantage of our approach is that it can be chained forward. For instance, an object on the table can be identified as a monitor with likelihood 0.8. Because the likelihood of the table is 0.5, the likelihood of being a monitor for the object is reduced to 0.4

Mapping semantic attributes to spatial relationships is the main contribution in our approach. A mapping between attribute *A above B* and a geometric expression would include that the z-value of the spatial entity A is higher than the z-value of B.

Simply speaking, we solve the semantic classification by recursively calling the relationship function Θ until a **spatial axiom** is reached. The spatial axioms are defined by entities which do not depend on a relationship with other entities. They are defined as shapes having the spatial feature vector

$$\Phi \quad = \quad (A, E, R, 1)^T,$$

implying that there is no relationship needed in order to put semantic labels on spatial axioms, so the likelihood is set to 1.

An example of a spatial axiom is, for instance, a floor which is the lowest horizontal entity in reference to the robot coordinate system. So the floor is not identified by a relationship to other entities.

From the spatial axioms, the relationships Θ are resolved until the root of the classification tree is reached. The function Θ is a recursive call to the semantic domain space. In order to define a match between a model entity and an extracted spatial entity, we define the following similarity equation. The spatial feature descriptor of a model Φ_{model} and an extracted spatial entity Φ_{entity} are similar if

$$\Phi_{model} \quad \odot \quad \Phi_{entity} < \delta,$$

where δ is the similarity threshold. The feature disparity function $\odot$ between Φ_{model} and Φ_{entity} is defined as:

$$\begin{array}{rcl} \Phi_{model} \odot \Phi_{entity} & := & \|A_{model} - A_{entity}\| \\ & & +\|E_{model} - E_{entity}\| \\ & & +\|R_{model} - R_{entity}\| \\ & & +\|\Theta_{model} - \Theta_{entity}\| \end{array}$$

When classifying structures, we are not dealing with full 3D perception but with a projection of 2D shapes in 3D space which is typical for a LIDAR recovered structure. Considering shape analysis, all detected shapes are projected onto a 2D plane. In order to analyze the recovered shapes (cf. Section 2), the planes are quantized by a projection in a 2D occupancy grid with a fixed grid of 1cm per pixel.

In our first approach, we focus on rectangular shapes, which will cover most of the objects found in indoor environment. In order to detect rectangular shapes in 2D, a Hough transformation is used [5]. To achieve this, all detected alpha shapes are projected into a discrete 2D bitmap. In order to detect a rectangular shape, all lines are analyzed with regard to their intersection in four points and whether the angles between the lines are $\sim 90^\circ$.

We now summarize our approach for the semantic labeling of spatial entities.

- A spatial database with labels and spatial description is set up. Each entity is represented by a spatial feature descriptor SFD_{model}. Each element is described by a spatial feature descriptor $\Phi_{model} = (A, E, R, \Theta)^T$. Some entities must be spatial axioms and not depending on the relationship to any other entity (i.e. $\Theta = 1$).

- The parameters A,E,R of Φ are extracted from the detected shapes in the spatial domain using the rectangle detection. Θ is evaluated until a spatial axiom is found.
- The disparity function $\Phi_{model} \odot \Phi_{entity}$ is evaluated. If the spatial feature descriptors are similar, the detected entity is matched with the model.

4 PRELIMINARY RESULTS

We tested our algorithm in a typical indoor environment using the robot setup pictured in Figure 5. For the scan, we chose 100.000 points at an opening angle of 180° horizontally and 90° vertically. The segmentation results are shown in Figure 6.

It is obvious that important spatial entities are segmented correctly

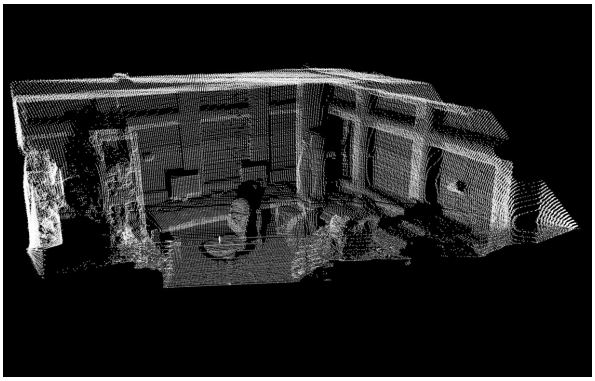

Figure 5. A point cloud of an office environment including 100.000 points

and the shape is recovered correctly. Besides the correct outer shape, important spatial information can be derived from the shapes, e.g. their appearance, the plane normal, and the metric extension. By this means, the 3D scene can be interpreted using constraints based on pre-knowledge about the environment. In a typical office environment, a table can be described as a rectangular shape parallel to the floor. The floor itself may be identified as being the largest horizontal plane with the lowest z-value, while the perpendicular, rectangular shape above the desk may be classified as flat screen monitors. Figure 7 shows an example in order to extract the features A,E,R from the shape polygons. In order to classify the shapes recursively, we define three semantic entities, i.e. *desk, screen, floor* with their spatial relationships.

$$\begin{array}{lcl} \Phi_{desk} & = & (1.28, 1.60, 1, \Theta(floor, parallel))^T \\ \Phi_{screen} & = & (0.24, 0.57, 1, \Theta(desk, orthogonal) \\ & & \wedge \Theta(desk, above))^T \\ \Phi_{floor} & = & (2.0, 2, 0, 1)^T \end{array} \quad (1)$$

The relationships *parallel*, *orthogonal*, and *above* map the corresponding entities to the spatial relationships

$$\begin{array}{lcl} \sigma & \rightarrow & [0, 1] : \\ parallel & \rightarrow & (N_{parent} \cdot N_{target}) \\ & & -(|N_{parent}| \cdot |N_{target}|) < \epsilon \\ orthogonal & \rightarrow & N_{parent} \cdot N_{target} < \epsilon \\ above & \rightarrow & (Pos\_Z_{parent} - Pos\_Z_{target}) > 0 \end{array} \quad (2)$$

In the relations above, N denotes the normal vector of the extracted regions belonging to the regarded shape (cf. Section 2). Pos_Z denotes the position of the shape (represented by the center of mass of the corresponding shape) in the vertical direction. Note that alpha

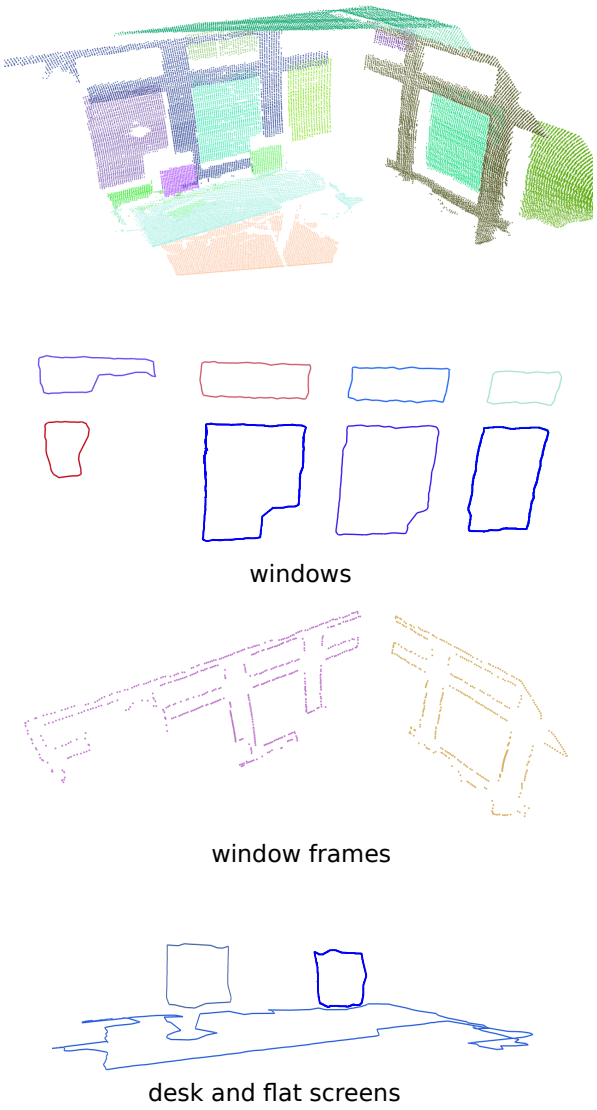

Figure 6. The results of our segmentation algorithm. The planes in the point cloud are segmented correctly and assigned to different colors. Some selected alpha shapes are presented. The window shapes, the shape of the window frame, and the shape of the flat screens on the desk are clearly recognizable after the polygonization.

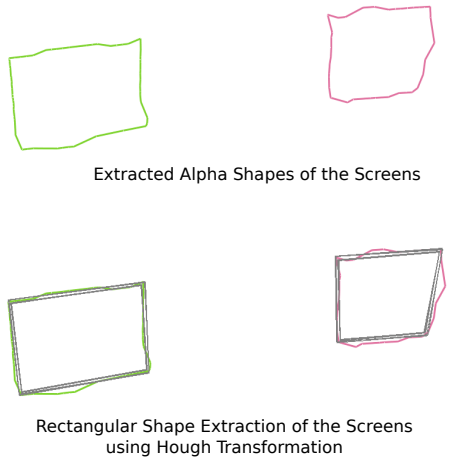

Figure 7. The two flat screen monitors from our scenery in Figure 6. First, the alpha shapes are extracted. Using our rectangle detection algorithm, the shape is clearly defined as rectangular. From the shape, the parameters A (area) and E (extension) can be easily extracted. The rectangularness (R) is set to 1 for each of the shapes. The relation function Θ is omitted in this example because no other entities are related to the shapes.

shape or Hough transformed structures do not contain any relationship to each other. The extracted relationship features are processed during the region-growing process, e.g. center of gravity, or normal vector.

Currently, we are able to extract all the spatial features mentioned in this paper and chain the extraction of the relationship function Θ. What is still missing to finally prove our concept is the implementation of the search algorithm that is able to match the spatial feature descriptors of detected entities to existing model feature descriptors. First experiments are promising considering only three simple entities, such as floor, table and monitor standing on the table. The next step is to match the features described in this paper to a semantic database of objects.

5 CONCLUSION

In this paper, we combined a method for recovering structure from unorganized 3D point clouds in the robotics domain. We presented two algorithms: The first transforms the point cloud into independent plane segments. The planes are then processed by computing the concave hull using an alpha-shape algorithm. By this means, the shapes of objects can be recovered efficiently. We showed how rectangular structures can be detected from the extracted shapes after the polygonization and which features can be extracted in order to apply semantic labeling to spatial entities. We introduced a spatial feature description together with a spatial relationship mapping, allowing to find labels for detected entities. We finally provided first results of the geometrical extraction process.

Future work focuses on the implicit mapping between semantic and spatial entities. For the time being, we are able to fill the feature vectors for known entities and define a similarity function. The last step, i.e., the automatic classification of detected objects is still to be realized. Another research problem is to build a descriptive ontology of the semantic space, allowing spatial reasoning in the semantic space and using our approach for bridging the gap between the semantic and the spatial domain. Our main research goal is to have a descriptive language for spatial entities which can be searched for object classification based on similarities in the feature space.

ACKNOWLEDGEMENTS

The work is sponsored by the German Federal Ministry of Education and Research (BMBF) within the “SEMPROM” project under contract number 01IA08002.

REFERENCES

[1] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, ‘The Ball-Pivoting Algorithm for Surface Reconstruction’, in *IEEE Transactions on Visualization and Computer Graphics*, volume 5, pp. 349–359, Los Alamitos, CA, USA, (1999). IEEE Computer Society.

[2] Fausto Bernardini and Chandrajit L. Bajaj, ‘Sampling and reconstructing manifolds using alpha-shapes’, in *CCCG*. IEEE Computer Society, (1997).

[3] P. J. Besl and H. D. Mckay, ‘A method for registration of 3-d shapes’, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **14**(2), 239–256, (1992).

[4] CGAL Editorial Board, *CGAL User and Reference Manual*, 3.5 edn., 2009.

[5] R.O. Duda and P.E. Hart, ‘Use of the Hough transformation to detect lines and curves in pictures’, *Communication of the ACM*, **15**, (January 1972).

[6] C. Galindo, J.A. Fernández-Madrigal, J. González, and A. Saffiotti, ‘Robot task planning using semantic maps’, in *Robotics and Autonomous Systems*, volume 56, p. 955966. Elsevier, (2008).

[7] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.A. Fernandez-Madrigal, and J. Gonzalez, ‘Multi-hierarchical semantic maps for mobile robotics’, in *Proc. IROS*, p. 34923497. Citeseer, (2005).

[8] C Kuo and H Yau, ‘A Delaunay-based region-growing approach to surface reconstruction from unorganized points’, *Computer-Aided Design*, **37**(8), 825–835, (2005).

[9] N. Lomenie, ‘A generic methodology for partitioning unorganised 3D point clouds for robotic vision’, in *First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.*, pp. 64–71. Ieee, (2004).

[10] Z.C. Marton, R.B. Rusu, and M. Beetz, ‘On Fast Surface Reconstruction Methods for Large and Noisy Datasets’, in *Proceedings of the IEEE International Conference on Robotics*, (2009).

[11] A. Nüchter and J. Hertzberg, ‘Towards semantic maps for mobile robots’, *Robotics and Autonomous Systems*, **56**(11), 915926, (2008).

[12] T. Rabbani, F. van Den Heuvel, and G. Vosselmann, ‘Segmentation of point clouds using smoothness constraint’, in *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume 36, p. 248253. Citeseer, (2006).

[13] R. Schnabel, R. Wahl, and R. Klein, ‘Efficient RANSAC for Point-Cloud Shape Detection’, *Computer Graphics Forum*, **26**(2), 214–226, (Juni 2007).

[14] Wei Shen, ‘Building boundary extraction based on lidar point clouds data’, in *ISPRS08*, p. 157. ISPRS, (2008).

[15] Pravin M. Vaidya, ‘An optimal algorithm for the all-nearest-neighbors problem’, in *SFCS ’86: Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pp. 117–122, Washington, DC, USA, (1986). IEEE Computer Society.

[16] N. Vaskevicius, A. Birk, K. Pathak, and J. Poppinga, ‘Fast Detection of Polygons in 3D Point Clouds from Noise-Prone Range Sensors’, in *IEEE International Workshop on Safety, Security and Rescue Robotics, 2007. SSRR 2007*, p. 16. IEEE Press, (2007).