Towards Semantic Navigation in Mobile Robotics

Adam Borkowski, Barbara Siemiatkowska, and Jacek Szklarski

Institute of Fundamental Technological Research, Polish Academy of Sciences,
Pawinskiego 5b, PL-02-106 Warsaw, Poland
abork@ippt.gov.pl
http://www.ippt.gov.pl/~ztiwww/en/DIShome.html

Abstract. Nowadays mobile robots find application in many areas of production, public transport, security and defense, exploration of space, etc. In order to make further progress in this domain of engineering, a significant barrier has to be broken: robots must be able to understand the meaning of surrounding world. Until now, mobile robots have only perceived geometrical features of the environment. Rapid progress in sensory devices (video cameras, laser range finders, microwave radars) and sufficient computational power available on-board makes it possible to develop robot controllers that possess certain knowledge about the area of application and which are able to reason at a semantic level.

The first part of the paper deals with mobile robots dedicated to operate inside buildings. A concept of the semantic navigation based upon hypergraphs is introduced. Then it is shown how semantic information, useful for mobile robots, can be extracted from the digital documentation of a building.

In the second part of the paper we report the latest results on extracting semantic features from the raw data supplied by laser scanners. The aim of this research is to develop a system that will enable a mobile robot to operate in a building with ability to recognise and identify objects of certain classes. Data processing techniques involved in this system include a 3D-model of the environment updated on-line, rule-based and feature-based classifiers of objects, a path planner utilizing cellular networks and other advanced tools. Experiments carried out under real-life conditions validate the proposed solutions.

1 Introduction

At the onset of Mobile Robotics (MR) most researchers believed that the ultimate goal is to develop robots able to act without human guidance. Most early papers and conferences on MR quote Autonomous Mobile Robots as their subject. Despite the cognitive challenge, this attitude turned out to be unjustified by needs of practice. The most spectacular application of mobile robots — the exploration of Mars — was carried out by means of remotely operated rovers [1]. Apart from units that perform very simple tasks, like cleaning a floor [2] or mowing a lawn [3], mobile robots used to-day in practice depend more or less on human assistance. Moreover, a capability to interact with people and an ability to understand their intentions seems to be a prerequisite for further progress in MR [4].

In order to perform any task, a mobile robot must be able to recognise its environment and to move safely inside it. Such an ability is secured by a *navigation system* of the robot. This system solves the following tasks:

1. *Mapping* means building and updating a *map*, i.e. an internal representation of the surrounding world.
2. *Self-localisation* amounts to determining a current *position* of the robot with respect to a certain reference frame.
3. *Path-planning* means generating a collision-free *trajectory* that leads from the current position to the goal position.

Taking into account many spectacular solutions, shown in the literature (compare, e.g., the overview in [5]), it may be considered that the mapping problem has been satisfactorily solved. Contemporary mobile robots equipped with cameras and laser range finders are able to build 2D-maps of quite cluttered and complex in-door environments. Under such circumstances it seems to be possible to add further dimensions to the mapping problem (2.5D, 3D, 4D maps) without changing the following basic premises:

1. The map is built from scratch, without any initial knowledge about the explored environment.
2. The map describes only geometry of the surrounding world.

The first assumption makes the problem cognitively challenging, though less practically oriented. Service robots are supposed to work in particular buildings, or in particular types of buildings. Digital documentation of any building erected contemporary is available in one of the CAD-formats. A scan of an older building can be easily generated by a laser-based geodetic tool, like Riegl System [6].

A rescue robot might need to build a map of a demolished building, but even then the knowledge of the state, preceding the disaster, is likely to be available. Hence, it seems to be reasonable to relax the first premise by assuming that the layout, taken from the documentation of the building, can serve as the initial assumption about the environment. Moving around the building, the robot is able to detect objects that were not shown on the initial map (e.g. pieces of furniture). It might happen sometimes that certain features of the real layout differ from those described in the documentation of the building (e.g. a door may become blocked or even removed). As an outcome of the process of exploration, the robot obtains a current map of its working environment. Taking floor plans provided by the designers of the building as the initial step of the mapping procedure considerably reduces the computational burden. Instead of generating the map from scratch, the robot registers only missing elements and changes in the environment. Such procedure is illustrated by Fig. 1, where the robot detects an object in the room and recognizes it as a wardrobe.

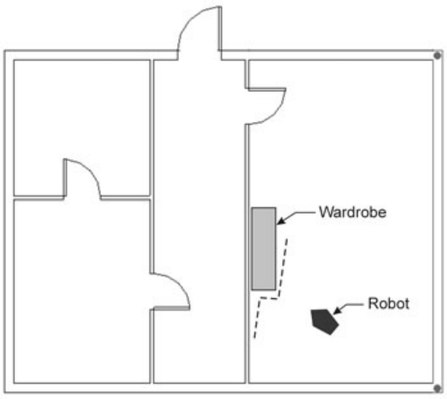

Fig. 1. Exploring a floor of a building

In order to act purposefully in the building, the mobile robot must be aware of the meaning of a surrounding world. A capability of breaking the barrier that separates the acquisition of geometrical features from the understanding of semantics will determine the success of robots in the future. Most researchers recognize this challenge and the first attempts to add meaning to environment maps are reported in the literature [7,8]. These works follow the pattern recognition scheme. It is assumed that the robot is given a certain knowledge about the building (ontology). Such knowledge allows the robot to recognize the components of the layout (windows, doors, etc.) on the current map, or even to detect the functional role of a particular subspace of the building (a hall, a staircase, etc.). Learning semantics autonomously is a difficult task. An obvious solution is to do it interactively with the help of a human operator.

The layout of our paper is as follows. In Section 2 we recall primaries of the geometry-oriented navigation known in the literature, we discuss limitations of this approach and we introduce our proposal of the navigation based upon semantics of the surrounding world. Section 3 is devoted to the new standard of the Architecture-Engineering- Construction (AEC) industry called Building Information Model (BIM). We show that this knowledge representation scheme contains a lot of information useful for in-door class mobile robots. By extracting this information from BIM-files, a preliminary mapping of the building is obtained. This mapping can be validated and enriched by the robot exploring its working area.

In Section 4 we demonstrate how a mobile robot equipped with a laser range scanner can recognise characteristic objects in the environment. Such objects, like a door or a piece of furniture, are stored in the lowest level of a sematic map proposed in Section 2. The path planning by means of Cellular Neural Networks (CNN's) is the subject of Section 5. Here the novelty lies in the mixed metric-symbolic format of the map that is used. A brief summary of results and a list of referencies conclude the paper.

2 Semantic Navigation

Until now most effort in mobile robotics has been spent on *geometric navigation*. Given a start position of the robot S and a goal position G, the navigation system has to find a trajectory T that brings the robot from S to G avoiding obstacles on the way. It is assumed that the positions S and G, as well as the location and shape of obstacles, are given with respect to a certain global reference frame. Under such premises, finding T would be a purely geometrical problem leading to an infinite number of solutions. Therefore, usually additional constraints are imposed on the trajectory, like the minimum length or the minimum energy consumption. This leads to alternative solutions shown in Fig. 2 by a solid line and a dashed line.

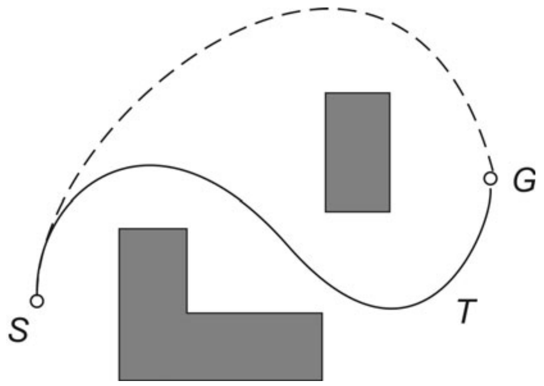

Fig. 2. Main components of geometrical navigation: start position A, goal position B, trajectory T, free space and obstacles

The advantage of the geometric navigation is that it allows the control system to move the robot more or less safely around its working area. The drawback of this procedure lies in the lack of links to the task planning performed at a higher level of abstraction than pure geometry. Let us consider a robot that should perform transportation tasks in an office building. Its current task could be to bring a parcel of books to the library. Assume that this parcel was placed on the robot while it was standing in the hall of the building. Thus, *Hall* is the starting place and *Library* is the goal place. Note that there is no need of geometrical data about start and goal, provided the layout (topology) of the building is given. Let it be a graph depicted in Fig. 3. Nodes of this graph correspond to particular elements of the building. They bear names reflecting functionality of these elements, which simplifies the human-machine interface. The order *Go to the library* can be given via a touch screen or via a natural language recognition system.

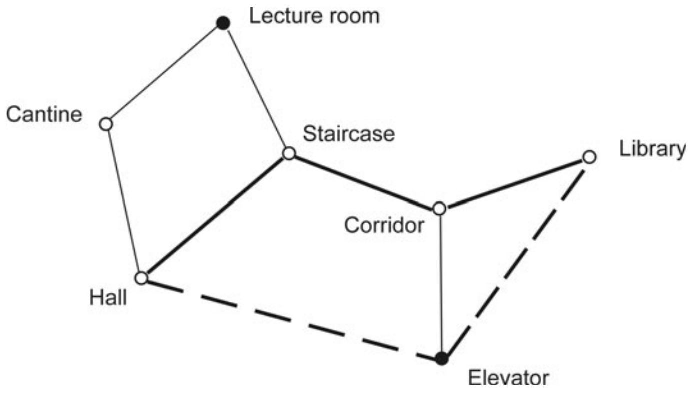

Fig. 3. Main components of semantic navigation: start place *Hall*, goal place *Library*, passable place *Staircase*, impassable place *Elevator*, trajectory {*Hall, Staircase, Corridor, Library*}, alternative trajectory {*Hall, Elevator, Library*}

Edges of the layout graph (usually called topological map) reflect accessibility relations between places. Knowing them, it is relatively easy to generate paths that lead from the starting place to the goal place. Many efficient graph search algorithms can be used for this purpose. Similarly as in the case of geometrical navigation, additional requirements should be posed in order to obtain an unique trajectory. A cost of traversing an edge could reflect either a distance between places or any other parameter. It is also possible to mark nodes of the graph as temporarily passable or impassable. In Fig. 3 the latter are drawn as circles filled with black color. So, despite the circumstance that the trajectory {*Hall, Elevator, Library*} is less "expensive", the robot will go through the staircase and corridor, because the elevator is blocked for the time being.

Representing the layout of the building by a graph seems to be natural but flat graphs are insufficiently expressive. Therefore, we prefer to use multilayered or hierarchical graphs. The usage of such graphs for representing knowledge about layout and functionality of buildings was subjected to our joint study with the group led by Prof. Manfred Nagl [17,18,19,20].

Let us take an exemplary office building as a case study (Fig. 4). Omitting details, we can present its layout in a three-level graph, as depicted in Fig. 5. The top level describes blocks of the building. It is seen that long distance paths require transversing the *Central unit*, where elevators are located. Each floor has similar layout with rooms accessible directly from a corridor. As it will be shown in the following Section, the functionality of a particular place (room) can be stored in an attribute of the object. Such an attribute is rather static in its nature, whereas other attributes (e.g. accessibility or passability) can be time dependent.

The lowest level encompasses a single place. Here nodes of the graph correspond to characteristic objects (landmarks) that can be detected by the means of the sensoric devices mounted on the robot. At the room level we change the

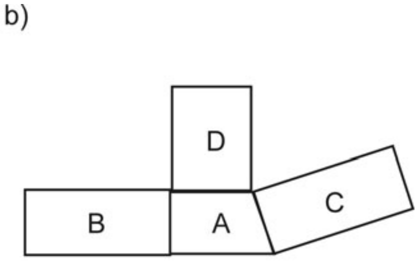

Fig. 4. Exemplary office building: a) overall view; b) footprint

meaning of edges. Instead of accessibility relations attached to edges at higher levels of the graph, we now introduce the global frame of reference *Nord-South, East-West* and store in the edges relative positions of landmarks with respect to this frame. Thus, a door may be situated to the north from the center of the room, the desk to the east, etc. This requires, of course, the robot to be equipped with a digital compass, which is a low cost sensor nowadays.

Within the frame of semantic navigation positioning means knowing the name of a place in which the robot currently is. This can be achieved either by reading a properly placed label (barcode, RFID, etc.) or by recognising the place on the basis of its characteristic features. In the following Sections we show some ways how such recognition can be accomplished.

3 Extracting Semantic Features from Documentation of Buildings

3.1 Data Models in AEC-Industry

The support of 3D modeling of buildings can be traced back to the early 1970s. At first, modeling was based on geometrical primitives, like octahedron, cylinder or pyramid, that were combined by means of Boolean operations (union, inter-section, difference, etc.) to represent more complex shapes (Fig. 6a). Later, Constructive Solid Geometry (CSG) was introduced, defining a final shape via a sequence of Boolean operations ordered in a tree. An alternative way was the so-called Boundary Representation (B-rep). In this model the shape is described by vertices and edges, as shown in Fig. 6b. Both representations have played a useful role in the development of CAD-systems – they have simplified the integration of efforts in different domains of engineering. However, this methodology was not able to represent more than pure geometry of the considered object.

At present AEC industry tries to catch up to aerospace and automotive industries that extensively use digitized models in design, manufacturing and maintenance of their products. It seems that Building Information Modeling (BIM)

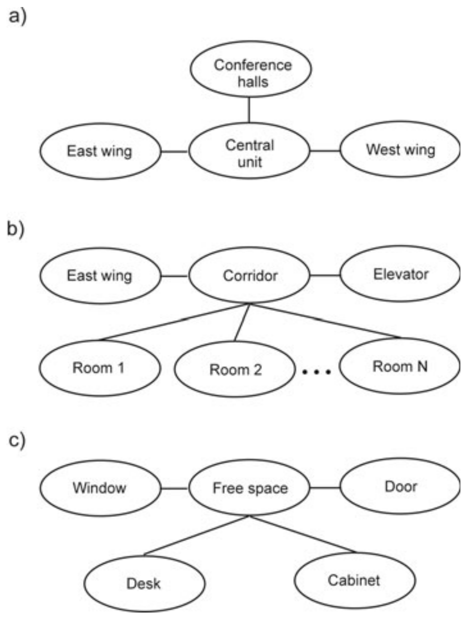

Fig. 5. Topological map of building: a) building level; b) floor level; c) room level

gives a chance of success in this endeavor. The object-based parametric modeling that is the core of BIM-methodology takes the following attitude towards representation of knowledge [9]:

1. Building components are described as hierarchically nested objects that have attributes and application rules.
2. The data structure is consistent, non-redundant, and allows the extraction of multiple views of the object.

A building within the BIM-framework is an assembly of instances of object classes, like walls, floors, ceilings, etc. There is information about how these components have been assembled together and about the constraints that make the design feasible. It is obvious that intelligent components may ,,be told" to take into account the needs of mobile robots as well. For example, a floor could refuse to accept steps and a corridor could check whether it is wide enough for two robots to pass each other.

One of the main advantages of BIM-compliant systems is their ability to interchange data with applications coming from different disciplines. Such interoperability can be achieved using various data formats, but the format Industry Foundation Classes (IFC) [10] seems to be most suitable with respect to the linkage between AEC and MR.

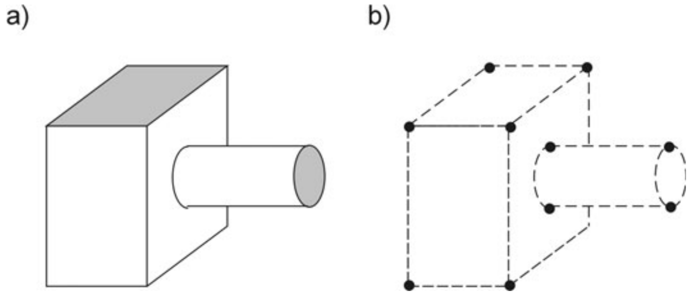

Fig. 6. Defining geometry: a) in CSG style, b) in B-rep style

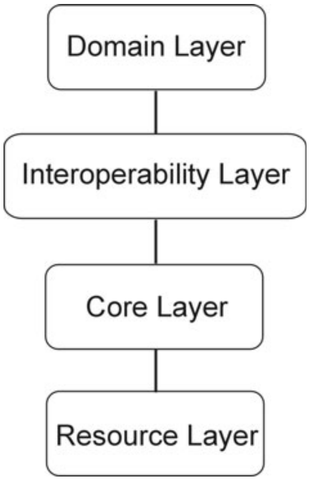

Fig. 7. Main layers of the IFC model

As shown in Fig. 7, the IFC model consists of four hierarchically nested layers. The bottom layer contains 26 reusable constructs like *Geometry, Topology, Materials,* etc. At the level of *Interoperability Layer* these constructs are combined into *Shared Objects.* Such objects include not only generic elements of the building itself, like walls, floors, columns, etc., but also generic elements of the service, management and facility domains. This allows the IFC model to cover the entire life-cycle of the building from its design through construction and usage up to final dismantling.

The top layer refers to specific subdomains of knowledge related to AEC. The most frequently used are the domains *Architecture, Structural Analysis, Heating and Ventilation.* The domains *Building Control* and *Telecommunication* become increasingly important in newly erected buildings.

The IFC model follows the known rules of the object-oriented data representation. It uses the EXPRESS data modeling language [12], developed within the frame of the ISO-STEP (Standard for the Exchange of Product Model Data) initiative. All objects in EXPRESS are treated as entities subject to enumerations and types. Objects stay in hierarchical parent-child trees with child nodes inheriting properties from their parents. At each level of such a tree user-defined attributes and relations can be introduced, which makes the IFC model extensible and adjustable to various needs.

Let us take the EXPRESS definition of *Space* as an example:

```
ENTITY IfcSpace

SUBTYPE OF (IfcSpatialStructureElement);

InteriorOrExteriorSpace:    IfcInternalOrExternalEnum;
ElevationWithFlooring:      OPTIONAL IfcLengthMeasure;
INVERSE
HasCoverings:   SET OF IfcRelCoversSpaces FOR RelatedSpace;
BoundedBy:      SET OF IfcRelSpaceBoundary FOR RelatingSpace;
END_ENTITY;
```

A space represents an area or volume bounded physically or virtually. Spaces are areas or volumes that provide certain functions within a building. Hence, the description of space entity bears important information for a mobile robot that is supposed to act inside this building.

A space can be either interior or exterior. An interior space is associated with a building storey, whereas an exterior space is associated with a building site. A space may span over several connected subspaces. Therefore, a space group provides for a collection of spaces included in a storey. A space can also be decomposed in parts, where each part defines a partial space. This is defined by the *CompositionType* attribute of the supertype *IfcSpatialStructureElement* which is interpreted as follows:

```
COMPLEX = space group
ELEMENT = space
PARTIAL = partial space
```

The inheritance chain of *IfcSpace* is rather long:

```
IfcRoot - IfcObjectDefinition - IfcObject - IfcProduct -
IfcSpatial-Element - IfcSpatialStructureElement - IfcSpace.
```

Due to this chain IfcSpace obtains several attributes that carry semantic information (*Name, Description, LongName, ObjectType*). The functional category of space is usually stored in the last attribute. Unfortunately, BIM-capable systems available at present are not very good at representing spaces and their assemblies. ArchiCAD [13] allows the user to define *Zones* and Revit Architecture [14] is able to extract automatically *Rooms* as spaces bounded by *Walls, Floors* and *Ceilings.*

An important ingredient of the IFC model is a *Property Set* (P-set). Property sets are defined by the following entity:

```
ENTITY IfcPropertySet

SUBTYPE OF (IfcPropertySetDefinition);

HasProperties:      SET [1:?] OF IfcProperty;
WHERE
WR31: EXISTS(SELF\IfcRoot.Name);
WR32:  IfcUniquePropertyName(HasProperties);
END_ENTITY;
```

The P-set *SpaceCommon* is a property set common for all types of spaces. The exemplary set *SpaceFireSafetyRequirements* contains the fire safety requirements for all types of spaces. When needed, new P-sets can be defined by the user.

A property of an object is defined as:

```
ENTITY IfcProperty

ABSTRACT SUPERTYPE OF   (ONEOF(IfcComplexProperty, IfcSimpleProperty));

Name     :      IfcIdentifier;
Description: OPTIONAL IfcText;
INVERSE
PartOfPset: SET OF IfcPropertySet FOR HasProperties;
PropertyForDependance: SET OF IfcPropertyDependencyRelationship
            FOR DependingProperty;
PropertyDependsOn: SET OF IfcPropertyDependencyRelationship
            FOR DependantProperty;
PartOfComplex: SET OF IfcComplexProperty FOR HasProperties;
END_ENTITY;
```

It is seen from this definition that a property can be either complex or simple. The definition of the latter reads:

```
ENTITY IfcSimpleProperty

ABSTRACT SUPERTYPE OF   (ONEOF(IfcPropertySingleValue,
         IfcPropertyEnumeratedValue, IfcPropertyBoundedValue,
         IfcPropertyTableValue, IfcPropertyReferenceValue,
         IfcPropertyListValue))
SUBTYPE OF (IfcProperty);
END_ENTITY;
```

The above definitions of properties fulfill most needs of AEC industry. A serious drawback with respect to mobile robotic applications is the deterministic nature of properties in the IFC model. Moreover, the present release of IFC contains a very limited number of properties describing the assumed function of a space, a required security level in a specific zone of the building or other functionality-oriented features. Some proposals how to overcome this deficiency will be given in Section 3.3.

IFC also provides means of expressing relations between objects. These relations are grouped into abstract classes as follows:

1. *Assigns* defines relations between instances and parent entity.
2. *Decomposes* describes assemblies and their parts.
3. *Associates* relates information shared in different parts of the model.
4. *Connects* defines topological relationships between adjacent entities.

It is clear that the present set of relations must be enriched in order to make BIM robot-friendly. This issue is also discussed in Section 3.3.

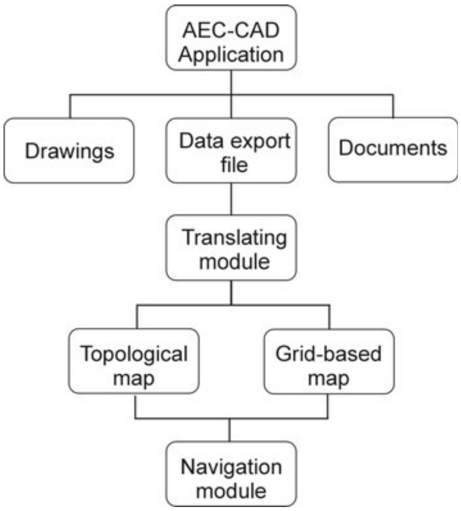

Fig. 8. Data transfer from AEC-format to MR-format

3.2 Taking Advantage of Existing BIM

The attempts to use documentation created when the building was designed for building metric maps applicable in mobile robotics were made several years ago [15,16]. However, at that time BIM was not available and extracting anything more than geometry from CAD drawings was almost impossible. In principle, one could think about representing maps of an in-door environment in one of the data formats popular in the AEC-industry. However, there are too many of them and they are less suited for sensor-based mapping and navigation than formats developed by the MR-community. Therefore, the scheme depicted in Fig. 8 seems to be preferable.

The simplest way of transferring data from one application to another is to use a proprietary file exchange format supplied by a CAD-vendor. We use this procedure in the currently running project, worked up in cooperation with the research group from the Jagiellonian University in Krakow [17].

The aim of this project is to develop a prototype software that translates floor layouts produced by Revit Architecture into three types of maps suitable for mobile robots:

1. Occupancy grids.
2. Accessibility graphs.
3. Feature maps.

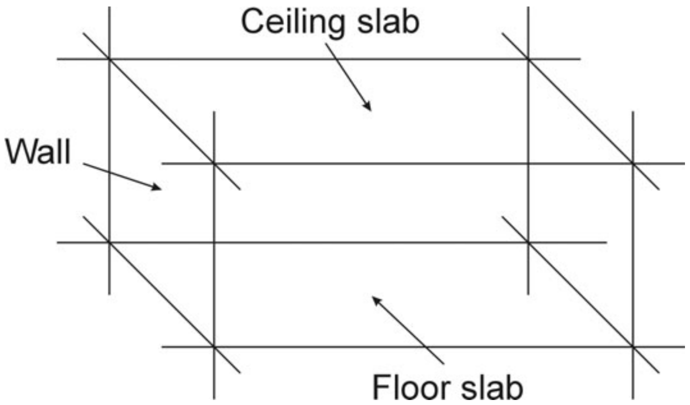

Fig. 9. Room as a subspace of building

Occupancy grids serve as ground knowledge for the mapping module, according to the scheme described in Section 4. Accessibility graphs and feature maps help the navigation module in planning the routes that bring the robot into desired places in the building. At present our translator takes into account only the information present in the standard AEC-oriented BIM model of the building. In such a model a building is composed of *IfcWalls*, and *IfcSlabs* – the entities defined in *IfcShared-BldgElements*. Knowing these entities and relations *Connects* between them, Revit Architecture automatically generates rooms as such spaces bounded by walls, floors and ceilings (Fig. 9).

Rooms are treated as *IfcSpace* entities possibly related by *Connects* relation (adjacency). There is no relation *Accesses* in the presently available BIM model. Fortunately, it is easy to check which pairs of rooms are mutually accessible: they must be adjacent and have a door in a common wall. IFC data sets uniquely define locations of doors and their belonging to particular walls. Hence, a topological map containing adjacency and accessibility relations can be generated relatively easily. Figure 10 shows a simple example of such a map.

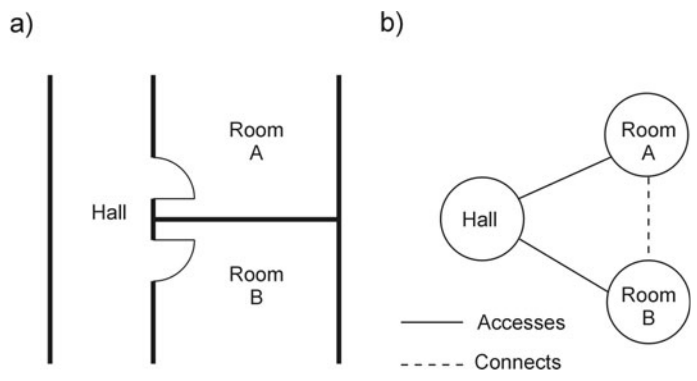

Fig. 10. Adjacency and accessibility relations: a) part of floor layout; b) topological map

The nodes of this graph can be attributed to values taken from P-sets of rooms. A label indicating the functional role of the room is probably the most valuable information for the mobile robot. Such a label could be stored by the designer either in *LongName* or in *ObjectType*. Other attributes, like an area of the room, could also be of interest for a service robot.

Most commercially available AEC-CAD tools support architects, structural engineers and other specialists in the phase of detailed design. Prior to entering this phase, designers must take many conceptual decisions that are crucial for the quality of the building. In our earlier papers [18,19] we investigated how an architect can be supported by a computer in the phase of preliminary design.

It turned out that, at least for a functionality-driven design, a certain formalization of reasoning about the goals, that are to be fulfilled by the designed building, helps the designer to find proper conceptual solutions. Such a formalization can be based upon the theory of graph transformations [20]. It allows the designer to convert informal wishes expressed by the investor into a formal specification of the building. This specification is stored in a form of the functionality graph serving as a framework for various special layouts that are considered in the conceptual phase of design. Functionality graphs bear close resemblance to topological maps of the building. Therefore, we intend to incorporate their usage in the future release of the tools transferring knowledge about the building from the AEC-domain into the MR-domain.

3.3 Adapting BIM for Mobile Robots

Mobile robots can be seen as facilities inside intelligent buildings. The *Domain Layer* of IFC includes *Facilities Management*. Base entities of this domain give the user a possibility to include information relevant for mobile robots into BIM-based description of the building.

Let us consider a typical layout of a control system for an intelligent building (Fig. 11). Such a building has multiple stationary sensors that measure

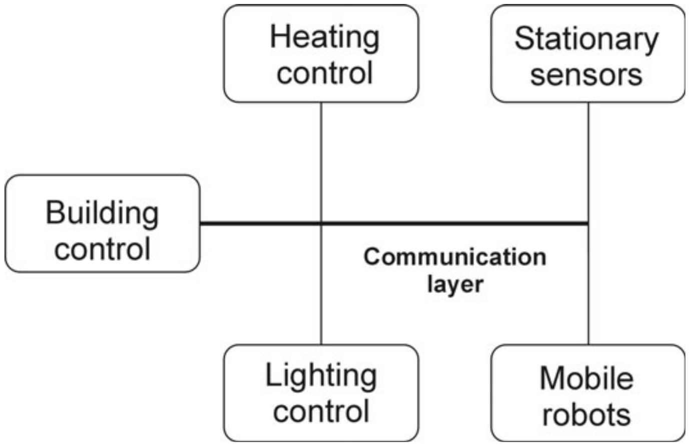

Fig. 11. Mobile robots embedded in an intelligent building

temperature, lighting level and other parameters determining living comfort in particular rooms. The data from the sensors are transmitted via an internal communication network to controllers responsible for heating, ventilation, lighting, etc. The overall state of the building is monitored by the main controller that could take under its supervision mobile robots as well. These robots could perform various tasks, like cleaning floors of the building, transporting goods inside it, etc. In order to be more precise, let us fix our attention on one particular, though quite important task: protecting the building against intruders. The security system could be based upon a combination of stationary sensors and mobile sentry robots. After receiving a signal that a certain suspicious motion has been detected by a sensor in a particular room, the building controller could send a sentry robot to check this room.

Can such a scenario be described by means of BIM? The answer to this question is to a large extent positive. Stationary sensors are quoted in *IfcBuildingControlsDomain*:

```
TYPE IfcSensorTypeEnum = ENUMERATION OF

(   FLOWSENSOR, GASSENSOR, HEATSENSOR, HUMIDITYSENSOR, LIGHTSENSOR,
MOVEMENTSENSOR, PRESSURESENSOR, SMOKESENSOR, SOUNDSENSOR,
TEMPERATURESENSOR, USERDEFINED, NOTDEFINED);
END_TYPE;
```

A possibility to control various facilities of the building is granted by the following entity:

```
ENTITY IfcControl;
. . .
Controls: SET OF IfcRelAssignsToControl FOR RelatingControl;
END_ENTITY;
```

included in the *Facility Management Domain*. Examples of possible actions encompass moving an object inside the building:

```
ENTITY IfcMove

SUBTYPE OF (IfcTask);

MoveFrom:   IfcSpatialStructureElement;
MoveTo:     IfcSpatialStructureElement;
. . .
END_ENTITY;
```

or triggering an action:

```
ENTITY IfcOrderAction

SUBTYPE OF (IfcTask);
. . .
END_ENTITY;
```

when certain conditions hold. Thus, the controller of an intelligent building can be implemented as a rule-based inference engine.

Moreover, BIM introduces *Actors* that play certain *Roles*:

```
ENTITY IfcActorRole;

Role: IfcRoleEnum;
UserDefinedRole: OPTIONAL IfcLabel;
Description: OPTIONAL IfcText;
WHERE

WR1:    (Role <$>$ IfcRoleEnum.USERDEFINED) OR
((Role = IfcRoleEnum.USERDEFINED) AND
EXISTS(SELF.UserDefinedRole));
END_ENTITY;
```

Until now, this part of BIM has served for describing human actors that take part in designing, constructing and exploiting buildings:

```
TYPE IfcRoleEnum = ENUMERATION OF

(   SUPPLIER, MANUFACTURER, CONTRACTOR, SUBCONTRACTOR,
ARCHITECT, STRUCTURALENGINEER, . . .,
USERDEFINED);
END_TYPE;
```

The open character of BIM allows the user to define his or her own actors and their roles. Thus, mobile robots as active elements of intelligent buildings can be easily modeled in the future. Moreover, BIM provides the possibility to define *Views*. These are submodels tailored for the purpose of specific applications. At present only the AEC-oriented view is available. It takes into account needs of the major players in designing conventional buildings: an architect and a structural engineer. A shift towards intelligent buildings will justify the effort to develop a control-oriented view. Such a view should enable architects, developers of mobile robots, suppliers of sensors and software engineers to cooperate efficiently on elaborating complex solutions for comfortable housing.

4 Extracting Semantic Features from Laser Scans

Robotics was defined as *the intelligent connection of perception with action*. A variety of sensing techniques is available to provide the perception. In mobile robotics usually laser range finders are used in order to measure the distance to objects in the operation area of the robot and the odometry to measure internal parameters of the vehicle. The advantage of 2D laser range finders is high accuracy and reliability of the measurements, however they scan the environment in a single plane. Obstacles placed below or above that plane cannot be detected. Today a lot of methods for 3D sensing are based on CMOS/CCD techniques. Typical CMOS/CCD 3D systems are based on the stereo vision and like all passive sensors they have difficulties providing reliable data in an environment with changing illumination. Since 2006 the 3D range cameras have been available. Active methods like 3D laser scanners give a better robustness.

To create a global model of an environment the scans have to be represented in the same coordinate system. This process is called registration. Many representations have been proposed, one of the most popular is 2D representation [21,22,5]. However, when the service robot acts in a domestic environment the 2D-world model assumption is not fulfilled and a 3D method is preferred. Methods of 3D map building can be roughly divided into following groups:

– Full 3D representation of the environment: meshes [23,24], point clouds [25], 3D evidence grids. Methods of this kind allow to represent an unstructured environment very precisely but are computationally expensive and consume a large amount of memory.
– In the second group the environment is represented as a set of features for instance walls [26,27,28]. The advantage of this approach is the compact data representation but it cannot be used directly for a collision-free path planning.
– The third approach combines 3D map and 2D grid-based representation. 2.5D elevation maps, extended elevation maps and multilayer maps are examples of this method [27,29].

The algorithm described in Section 2 belongs to the third group but the map represents not only metric information but also nonmetric properties of the environment. We call this kind of representation semantic map. It means assigning meaning to data. This kind of representation has simplified human-robot communication. For instance the goal for the robot can be given using semantic labels.

The path planning problem is typically formulated as follows: Knowing the goal position and the robot position, find a collision free path leading from the robot to the goal. The path should satisfy certain optimization criteria (for instance it has to be the shortest one). Many path-planning algorithms have been proposed [32], they can be classified according to two factors: the type of the environment (static or dynamic) and the knowledge about the environment (global or local). The global path planning algorithms requires a complete knowledge about the entire environment which is computationally expensive, and a large amount of memory is required. The local path planning algorithms make use of local knowledge only, which is faster, a small amount of memory is needed, it is easier to respond to any local environmental change. However, the solution, e.g., the shortest path, is not necessarily optimal.

Our approach combines the advantages of local and global methods: the planned path is optimal, the path is re-planned in response to changes in the environment, the problem of local minima does not exist. Moreover, it is easy to implement CNN on a parallel architecture in order to increase efficiency.

The scanning system used in our experiments is built on the basis of a SICK LMS 200 laser range finder which is mounted on a rotating support. Sick LMS 200 measures distances to the obstacles in 2D plane with resolution 0.1°, 0.5° or 1.0° and swiping space from −90° to +90°. A plane with 181 (or 361) data points is scanned in 13ms (or 26ms). The laser scanner is mounted on a special support which rotates vertically from −15° to +90°. Depending on the scanning resolution it takes from 100ms to 3s. Figure 12 depicts the scanning system.

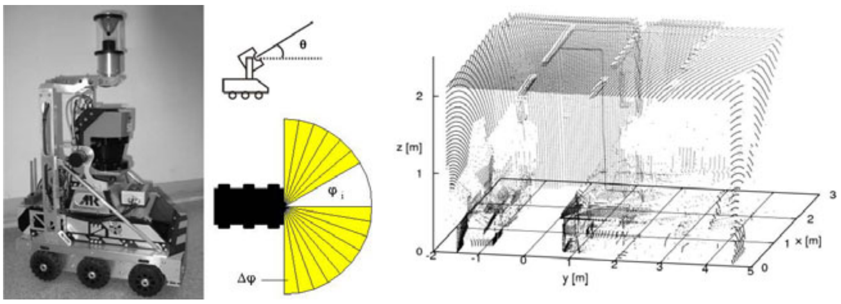

Fig. 12. 3D scanning system: a) Robot Elektron1, b) rotating support and LMS 200 scanning system, c) a sample cloud of points

Data from the laser are obtained in the local polar coordinate system $(r_{i,j}, \phi_i, \theta_j)$, where: $r_{i,j}$ - the distance to an obstacle [meters], θ_i - the vertical angle, ϕ_j - the horizontal angle. The local Cartesian coordinates are computed as follows:

$$
\begin{aligned}
x_{i,j} &= r_{i,j} \cdot \cos\theta_i \cdot \cos\phi_j ,\\
y_{i,j} &= r_{i,j} \cdot \sin\phi_j ,\\
z_{i,j} &= r_{i,j} \cdot \sin\theta_i \cdot \cos\phi_j ,
\end{aligned}
\tag{1}
$$

where $(x_{i,j}, y_{i,j}, z_{i,j})$ are Cartesian coordinates of the point (i, j).

Usually the next step is to transform these values into a point cloud which is a set of 3D points in the Cartesian coordinate system with the robot in its center (Eq. 1). Such a point cloud can be analyzed for a single 3D scan, or all the points can be combined from different scans to form a global representation of the environment in which the mobile robot is embedded.

For most applications, direct data from point clouds are not sufficient and for some, usually quite sophisticated ones, additional processing is required. Generally, the aim of such processing is to detect and then gather information about specific objects present in the environment. In the field of modern robotics it is usually expected not only to trace the objects of interest, but also to assign some semantic meaning to them.

Often the first step is to find some regular structures in the point cloud: flat or curved smooth surfaces like spheres or cylinders, line segments joining surfaces, etc. Therefore, each point from the cloud can be assigned to its specific element. Algorithms used in this process often have their counterparts in the image analysis field, like the split-and-merge method, surface growing methods, or the generalization of Hough transform [34]. Afterwards a single detected element or a group of connected elements can be recognized as an object. Of course applications of the discussed process are not only limited to robotics. Due to the popularity of high-resolution 3D scanners they are used in architectural modeling [35], industrial applications transforming real scenes into virtual reality [36] and many more. However, it should be noted that it is of vital importance that all the algorithms applied for real-time semantic object detection in mobile robotics are fast.

Below we shortly discuss our simple approach to a very fast scene segmentation with use of standard methods for image analysis. Essentially, the method is based on the direct transformation of a point cloud from a single scene into the RGB image. Afterwards we outline our knowledge-based expert system for semantic labeling detected objects.

4.1 Converting Acquired Data into a 2D Image and Image Segmentation

The most straightforward way to transform data from the laser scanner into an image is to use (ϕ, θ) as the pixel coordinates and assign its color according to the measured distance. A sample scene is shown in Fig. 13.

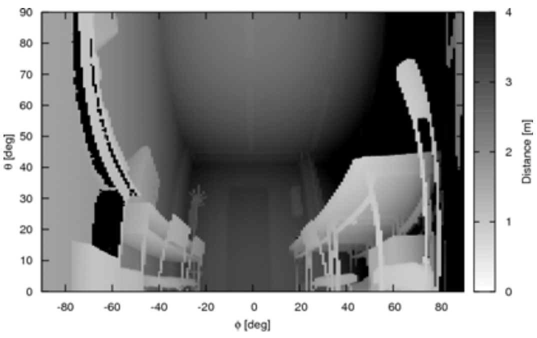

Fig. 13. A sample gray-scale image representing distance from the robot in (ϕ, θ) space. Maximum distance measured by the scanner is 8 meters. (Here, for clarity, we show all distances greater than 4 meters as black.)

As first step of our process a flood fill algorithm is run on the image representing distance. The threshold for the algorithm is constant and it corresponds to about 5 cm (the difference between neighbouring pixels is considered when flooding). All areas which are too small to be classified are marked with number 0 and are not considered in any later stage of the process.

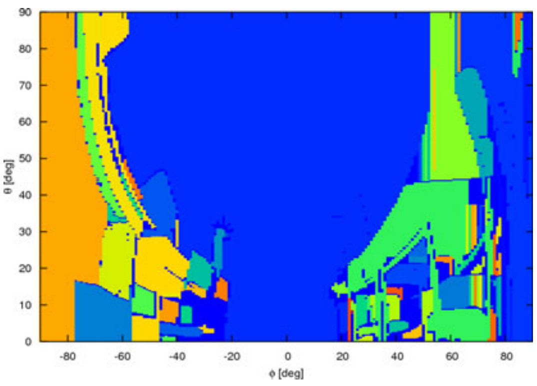

Fig. 14. Areas detected after the first stage of the segmentation process applied to data presented in Fig. 13. Each area is marked with a different, arbitrary chosen, color.

After the first step is finished, one has a list of areas which represent ,,continuous structures" of the environment. For example, a chair standing in front of a wall will be assigned to a different area than a wall since there is a large change of distance between the chair's edge and the wall. On the other hand, walls, ceiling, and floor will be classified as a single area since the change of the distance in all corners is assumed to be small.

In order to obtain images more suitable for the information extraction we map three coordinates associated with a normal vector for each pixel to RGB values of an ordinary color image. For each pixel (i, j) we obtain its position $\mathbf{p}$ in 3D Cartesian coordinates with the robot in its center. Then a normal vector at (i, j) is calculated as a sum of cross products of $\mathbf{p}$ and vectors associated with the four closest neighbours of (i, j).

A color RGB image is constructed by assigning values of the coordinates $\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z$ as colors red, green and blue, accordingly. Then $\mathbf{p}$ is normalized and each coordinate is mapped to an 8-bit color component as $(-1, 1) \rightarrow (0, 255)$. Note that, for example, a ceiling or a floor will have red and green components equal to 128, while the blue one will be larger than 128 for the floor (making it blueish) and smaller than 128 for the ceiling (making it look more yellowish). On the other hand, all of the planes perpendicular to the laser scanner will have the blue component equal to 128. Moreover, walls which are placed along the line of sight of the robot will be pink on their left-hand side and cyan on the other side. An image generated in this way is depicted in Fig. 15.

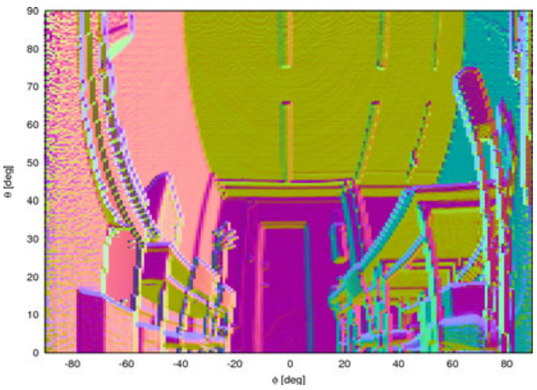

Fig. 15. Data for the scene presented in Fig. 13 were transformed in order to obtain normalized normal vectors $\mathbf{n}$ for each pixel. Absolute values of (x, y, z) coordinates of $\mathbf{n}$ are presented as red, green and blue components of this 8-bit RGB image (e.g., $\mathbf{n}_x = 0$ gives red=0, $\mathbf{n}_z = \pm 1$ gives blue=255). Note the increasing noise for ϕ near $\phi = \pm 90°$.

Naturally, the straightforward mapping of angles ϕ, θ as pixel position does not have to lead to the construction of images optimal for subsequent processing. For example, it is evident that all measurements with $\phi = \pi/2$ (for $\phi = -\pi/2$ as well) represent in fact the same point regardless of the value of θ. Consequently, the resulting image in the region where $\phi \approx \pm\pi/2$ is strongly distorted, also a significant noise is observed. The noise arises due to the method for calculation of the normal vectors – the entire region represents a small area of the real environment and statistical errors from the laser are exaggerated accordingly. Therefore, in our research we consider other mappings as well.

For example it is convenient to convert data in (ϕ, θ) coordinates to the usual spherical coordinate system $(\hat{\phi}, \hat{\theta})$ in which the robot is in the center, $\hat{\phi}$ is the longitude and $\hat{\theta}$ is the latitude. Having done this, it is possible to use, e.g., the Albers equal-area conic projection [37] which produces images representing the environment in a somehow more natural way. Obviously, one has only a limited number of measured points and therefore it is necessary to apply the appropriate interpolation when projecting from (ϕ, θ) coordinates.

The purpose of the discussed segmentation is to perform a *fast* decomposition of the gathered data into areas, each one representing a flat polygon in the real scene. Along the most important areas are, of course, a ceiling, a floor, walls, doors, etc. Besides the list of polygons, some numbers characterizing physical properties of a polygon can also be extracted. These can be used later for better object classification.

Having obtained a list of n areas by running the flood-fill algorithm on an image representing distance, Figs. 13, 14, the next step is performed. It consists of dividing the n areas into smaller ones by running the flood-fill algorithm on the RGB image constructed from normals. For each area i, $1 \leq i \leq n$, it is used as a mask, so only pixels belonging to the area i are used as seed points and all pixels from area outside i act as borders for the filling process. Eventually, the second step gives a list of $m \geq n$ areas each representing a more or less flat surface of the real scene.

In the final phase of the segmentation process the areas from step two are converted into polygons in the full three-dimensional space. For each area i, $1 \leq i \leq n$, we consider its pixels belonging to the *inner* edge of the area on the border between i and the rest of the image. Of course every pixel directly corresponds to a point in 3D space from the point cloud. Taking every 3D point from all pixels forming the edge we obtain a list of vertices defining a connected component – polygon – related to the area i. These polygons are used in a classification procedure.

After the segmentation process is finished, some geometric features of the scene are extracted and a list of flat surfaces is obtained. In order to assign a semantic feature to such a surface we use the following characteristics (see also, e.g., [34]).

- Size – usually an object which is supposed to be detected is characterized by some reasonable geometrical size.
- Orientation – for example: walls or doors are always vertical, the ceiling is always horizontal.
- Topology – relations between surfaces is important.

Based on the above characteristics a simple rule-based classifier is applied. In such a system an object is labeled as *door* if it is a single, vertical surface with the width within the range 1-2m, the height 1.5-2.5m, it connects directly to a *floor* at its bottom and to a *wall* from all other sides.

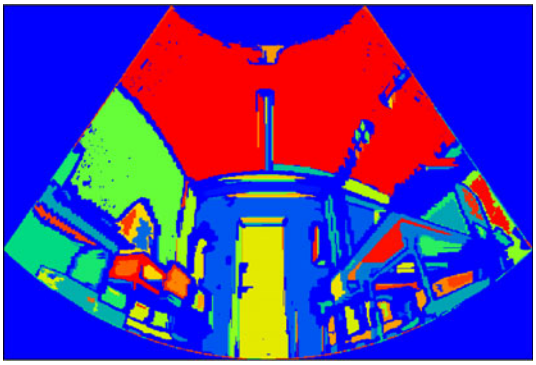

Fig. 16. Areas detected after the segmentation process. However, here processing is done on an image which has been firstly projected using Albers conic projection.

4.2 Rapid Object Detection Using Haar-Like Features

Treating laser scanner data as an image makes it also possible to directly apply well known methods for object detection and pattern recognition from an image analysis field. Here we show how to enrich our classification system by using a scheme based on a boosted cascade of simple features to detect objects. Algorithms which are applied are available in the OpenCV library and they implement methods proposed by [38] for basic set of Haar-like features and by [39] for rotated features which enhance the basic set. After the system is trained for recognition of certain objects, new images can be analysed very fast while maintaining a good hit rate and a reasonably low false alarm rate. This makes the method interesting and practical for our purposes. Moreover, having direct geometrical information about a detected object we can often reject false classifications just by analysing its real size.

Images generated from laser data in the way proposed above have, of course, different properties than usual visual images gathered by cameras. For example, illumination and any lighting conditions are not of our interest here. Either in bright light with many shadows or in a completely dark room one gets the same image. On the other hand, when the robot equipped with the scanner moves on a floor, red and green components of the image change so there is some dependency upon its position. This, however, does not have to lead to problems with the object detection with use of the image analysis, since many of the methods used for that purpose operate on gray-scale images.

In the first stage of our classifier it is necessary to train it for objects of interest. Here we show examples how to detect a sink visible from the perspective of the robot. In order to perform training a Haar-like classifier one needs a large set of positive and negative example images. All of the images should be scaled to the same size, we use 20x20. As the set of negative examples we use a large number of arbitrary images representing a scene without any object of our interest, i.e., without any image of the sink in this case.

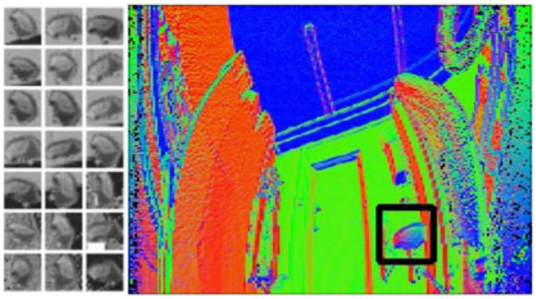

Fig. 17. The left panel shows 21 images – a small subset of set of positive samples which are used to train classifier for a sink-like object. On the right the result of classification for test image is presented (the black rectangle). The test image was not used in the training process. Here when converting the normals to RGB absolute values $|\mathbf{n}_x|, |\mathbf{n}_y|, |\mathbf{n}_z|$ instead of $\mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z$ were used.

In order to get a set of positive examples we take several snapshots with the laser scanner of a scene with the sink. Then, after constructing images representing the scene, we crop a sub-image with a sink only, making the background translucent. In the next step all the sub-images are rotated about x, y, z axis by random angles, intensity is randomly modified and sinks are placed onto a random background image. Finally, we have 1000 different images of a sink with random transformations placed onto different backgrounds. Large images containing a sink serve as testing images after training.

After the training process is completed, the classifier can be applied to any region of an image giving *true* if the region is likely to contain a pattern similar to one of those from the positive samples set, *false* otherwise. The analysis is very fast so one can try many different regions with varied sizes from all parts of the image. By doing this in a loop one can search the entire image for an object of interest. Figure 17 shows the result of such an analysis when searching with a classifier trained for the detection of objects similar to a sink and stairs respectively.

Areas corresponding to objects detected with the discussed method can be processed later in our classification scheme when building a semantic map. Naturally, each object which is going to be recognized has to have its own specifically trained Haar-features classifier. In a more sophisticated approach it is possible to combine object recognition from both: visual images and images constructed from laser scans.

Figure 18 depicts a top-view of a sample global 3D map of the environment. In such a map each cell corresponds to a cube of size 10 cm x 10 cm x 10 cm in reality. If a certain object has been detected, appropriate cells in the map are marked with specific semantic labels. On this sample map the following objects have been detected: floor, ceiling (both omitted in the figure), wastebasket, door, stair and washbasins.

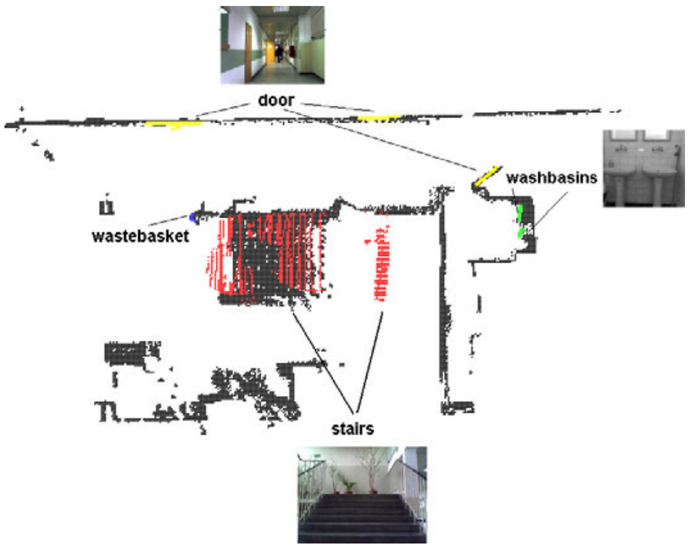

Fig. 18. A top-view of a sample global map of the environment. Cells represent unidentified and identified obstacles, floor and ceiling on the image are omitted.

5 Path Planning

A task planning for mobile robots usually relies on the spatial information. Although this kind of information is necessary for performing basic operations, the use of semantic knowledge is useful at a higher degree of autonomy and intelligence. In this section we show how this type of knowledge can be profitably used for the robot path planning. We start by defining a specific type of semantic maps, which integrate the metric information and the semantic knowledge.

The map of the environment is represented as a grid of cells, a list of semantic labels is attached to each cell. This kind of representation allows us to find easily the position of a specified object in the environment, and the relationship between objects. We can also ask the robot to move to a door or along a corridor. The path-planning algorithm which is proposed in this paper is implemented using the Cellular Neural Network.

5.1 Cellular Neural Network

The Cellular Neural Network (CNN, also known as systolic arrays) was proposed by L. O. Chua in 1988, as a very efficient tool for image analysis [30,31]. CNN consists of neurons (cells) which interact locally. Usually cells are arranged in a form of an $N \times M$ array. The cell in strip i and column j is denoted by the symbol c_{ij}, its state is described by symbol x_{ij} .

The cell c_{kl} belongs to the neighbourhood of c_{ij}, if for a parameter r (radius of neighbourhood) the following condition is fulfilled:

$$\max(|i-k|,|j-l|) \leq r. \tag{2}$$

The neighbourhood of the cell c_{ij} is denoted by the symbol N_r^{ij}. The CNN is defined by the following parameters: input signals $u_{ij}^{kl} \in R$, output signals $y_{ij}^{kl} \in R$, a bias $I \in R$, a_{ij}^{kl} - interconnection weight between the cells c_{kl} and c_{ij}, b_{ij}^{kl} - the feed forward template parameter. The dynamics of the CNN is described as follows:

$$x_{ij}(t+1) = \sum_{c_{kl} \in N_r^{ij}} a_{ij}^{kl} y_{kl}(t) + \sum_{c_{kl} \in N_r^{ij}} b_{ij}^{kl} u_{kl}(t) + I, \tag{3}$$

$$y_{ij}(t+1) = f(x_{ij}(t+1)), \tag{4}$$

were: f is an output function. Chua extended the definition of CNN. It is assumed that CNN consists of cells that interact locally and can be modelled as locally connected finite state machines. The new definition allows to widen the area of possible applications.

5.2 Path Planning

The algorithm for path planning consists of the following parts:

- The $N \times M$ grid-based traversability map is created based on the dual map of the environment, each cell of the map represents the traversability level (u_{ij}) of a corresponding area.
- The $N \times M$ *CNN* is built, each cell of the map corresponds to the cell of *CNN*.
- The weights a_{ij}^{kl} of the interconnection between c_{ij} and c_{kl} are computed. a_{ij}^{kl} is proportional to the distance between centres of gravity of areas which are represented by cells c_{ij} and c_{kl}.
- u_{ij} is the input signal to the cell c_{ij}, the value of u_{ij} is computed based on semantic knowledge.
- A set c_G which represents the position of the goal is distinguished. Usually this set consists of more than one neuron, the neuron c_R - represents the position of the robot.
- the value f_{ij} is computed for each cell of CNN:

  $$f_{ij}^r = f_o(u_{i-r,j-r}, ..., u_{ij}, u_{i+r,j+r}), \tag{5}$$

  where r is the radius of the neighbourhood, f_o is the function of input signals.
- The state of the cell c_{ij} is described as follows:

$$x_{ij}(t) = \begin{cases} K - f_{ij}^r & if\ c_{ij} = c_G, \\ \max(\max_{kl \in N_2^{ij}}(y_{kl}(t) - a_{ij}^{kl}) - f_{ij}^r, 0) & if\ c_{ij} \neq c_G. \end{cases} \tag{6}$$

If a cell represents the goal position and is free of the obstacles, then its state is static and equals K (very large number). If it is occupied then its state equals to 0 . If a cell is not a goal and is free from the obstacles then its state equals to the maximum value of the neighbouring cells minus the traversing cost in other cases the state equals to 0.

– The output signal y_{kl} of the cell c_{kl} is described as follows:

$$y_{kl}(t) = f(x_{kl}(t)). \tag{7}$$

In our approach $f(x) = x$.

The process of diffusion is continued until stability

$$\forall_{i=1,..,N,j=1,..,M} \;\; x_{ij}(t) = x_{ij}(t+1). \tag{8}$$

If the cell c_{ij} represents the position of the robot then the next position of the vehicle is indicated by the neuron $c_{kl} \in N_2^{ij}$ for which the following formula is fulfilled:

$$x_{kl}(t) = \max(\{x_{nm}(t)\}), \text{ where } c_{nm} \in N_2^{kl} \tag{9}$$

The planned path (an ordered list of cells) depends on the values of input signals u_{ij}, function f_o and the radius of the neighbourhood.

In most of the algorithms of path-planning, the robot is represented as a point and the obstacles are extended by some radius in order to take into account the dimensions of the robot. In many practical situations it is difficult to indicate the best value of the radius and the method fails when a group of robots with different sizes plan their paths based on the same map. In our approach, the dimension of the robot is taken into account during the path planning - values of parameters b_{ij}^{kl} are computed. b_{ij}^{kl} indicates the influence of an obstacle in the area represented by the cell c_{ij} to the cell c_{kl}.

If, for instance, the task for the robot is to move to the nearest table, then all cells which correspond to the position of any table are activated and the path is planned automatically. If the task for the robot is to move to the specified table, then the cells which correspond to this table are activated.

Figure 19 presents the result of the path plannig to a washbasin. The dotted line indicates the shortest path and the continuous line referes to the safest one. R - represents the initial position of the robot.

In order to determine low-level primitives, a set of pairs (v_t, ω_t), where v_t is the linear velocity and ω_t the rotational velocity of the robot at the moment t, a modified dynamic window approach is used [33]. In this method the search space consists of velocities that can be achieved by the robot, given its current linear and angular velocities and its acceleration capabilities, within a given time interval. This time interval corresponds to a servo tick of the control loop. In the case of semantic navigation the set of admissible velocities depends also on the environment. The maximal velocity of the robot is large in a big, empty environment but the robot has to slow down near a door.

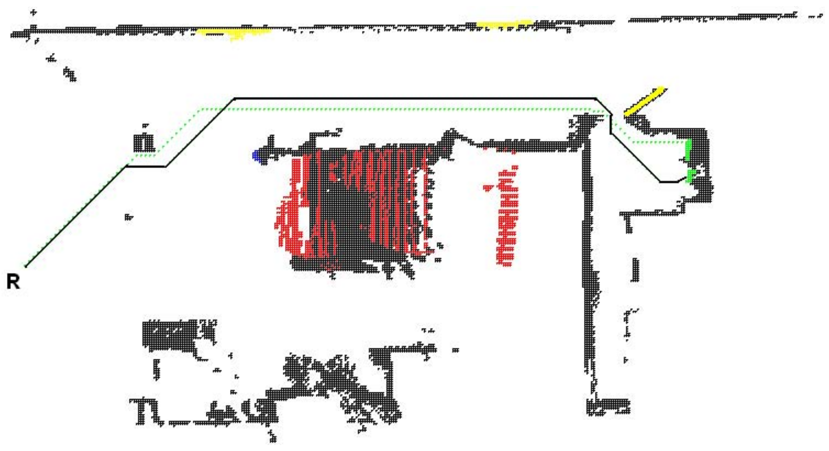

Fig. 19. The path of the robot: the shortest path - dotted line, the safest path - continuous line

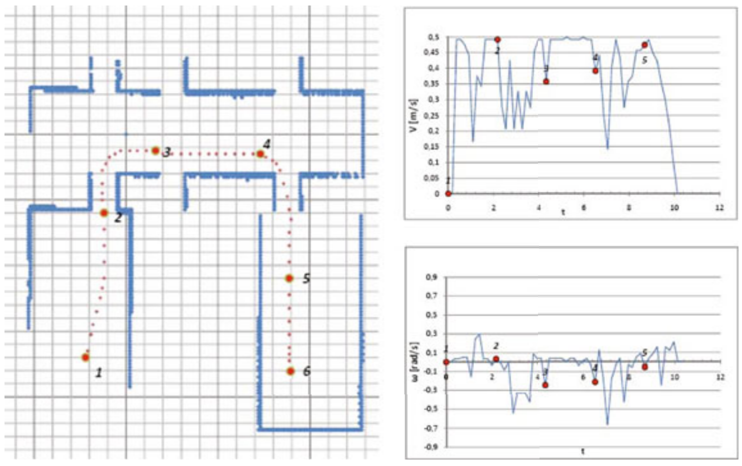

Fig. 20. The executed path of the robot: *Left:* the path of the robot, *Right, top:* linear velocities, *Right, bottom:* angular velocities

Velocities which maximize the function $G(v, \varphi)$ are taken as optimal values.

$$G(v, \omega) = a \cdot \mathrm{Dyf}(v, \omega) + b \cdot \mathrm{Dyf}_{\alpha}(v, \omega), \tag{10}$$

where $\mathrm{Dyf}(v, \omega)$ is the difference between the state value of a cell which represents the current position of the robot and the value of the state after time Δt if controls (v, ω) are applied, $\mathrm{Dyf}_{\alpha}(v, \omega)$ is a function which favours trajectories that lead straight towards the cell (c_{max}). c_{max} has the maximum state value in the neighbourhood of the cell which corresponds to the current position of the robot, parameters a and b are scaling.

6 Conclusions

The mobile robotics is on the verge of transition from cognitive-driven research to full scale applications. In the foreseeable future in-door class robots will become important ingredients of facilities provided by public domain infrastructure and by newly erected living habitats. In order to achieve the best performance, mobile robots must be fully integrated into the functional concept of the building and the building must be designed taking into account the presence of robots. Such synergy requires the opening of a new interdisciplinary domain of research, where people interested in designing buildings and mobile robots could meet and exchange their needs and ideas. The present paper can be seen as a modest step toward this goal.

Acknowledgement. This work has been partially supported by the Polish Ministry of Science and Higher Education (grant: 4311/B/T02/2007/33).

References

1. Mars Exploration Rover Mission. Jet Propulsion Laboratory, California Institute of Technology (October 3, 2009), `http://marsrovers.nasa.gov/home`
2. Household Robots. iRobots (October 4, 2009), `http://www.irobot.com/uk/home_robots.cfm`
3. Robomow by Friendly Robotics (July 22, 2008), `http://www.friendlyrobotics.com`
4. Fox, D.: Toward High-level Reasoning for Autonomous Systems. In: Lilienthal, A., Petrovic, I. (eds.) Proc. of the 4th European Conference on Mobile Robots (ECMR 2009), Mlini-Dubrovnik, Croatia, pp. 1–6 (2009)
5. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT-Press, Cambridge (2005)
6. RIEGL Laser Measurements Systems (2008), `http://www.riegl.co.at`
7. Martinez Mozos, O., Triebel, R., Jensfelt, P., Rottmann, A., Burgard, W.: Supervised semantic labeling of places using information extracted from sensor data. Robotics and Autonomous Systems 55(5), 391–402 (2007)
8. Vasudevan, S., Harati, A., Siegwart, R.: A Bayesian Approach to Conceptualization and Place Classification: Using the Number of Occurrences of Objects to Infer Concepts. In: Burgard, W., Gross, H.-M. (eds.) Proc. of the 3rd European Conference on Mobile Robots (ECMR 2007), Freiburg, Germany, pp. 1–6 (2007)
9. Eastman, C., et al.: BIM Handbook. A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors. Wiley, Hoboken (2008)
10. BuildingSMART (former International Alliance for Interoperability IAI), Industry Foundation Classes, Version 2X4 (December 14, 2008), `http://www.iai-international.org/index.html`
11. Industry Foundation Classes, Release 2x, Platform Specification (IFC2x Platform) (2005), `http://www.iso.org/iso/catalogue_detail.htm?csnumber=38056`

12. Schenck, D.A., Wilson, P.R.: Information Modeling the EXPRESS Way. Oxford University Press, New York (1994)
13. ArchiCAD, ArchiCAD 12 - Accelerating the Design Experience (December 14, 2008), `http://www.graphisoft.com/products/archicad/ac12`
14. Autodesk (2008). Revit Architecture (December 14, 2008), `http://usa.autodesk.com/adsk/servlet/index?id=3781831&siteID=123112`
15. Fennema, C., et al.: Model-Directed Mobile Robot Navigation. IEEE Transactions on Systems, Man and Cybernetics 20, 1352–1369 (1990)
16. Murarka, A., Kuipers, B.: Using CAD Drawings for Robot Navigation. IEEE Transactions on Systems, Man and Cybernetics 2, 678–683 (2001)
17. Borkowski, A., Grabska, E., Palacz, W.: Modeling Buildings for Mobile Robots. In: Proc. ASCE International Workshop on Computing in Civil Engineering, Austin, Texas (2009)
18. Borkowski, A., Grabska, E.: Converting function into object. In: Smith, I.F.C. (ed.) EG-SEA-AI 1996. LNCS, vol. 1454, pp. 434–440. Springer, Heidelberg (1998)
19. Borkowski, A., Szuba, J.: Graph transformations in architectural design. Computer Assisted Mechanics and Engineering Sciences (CAMES) 10, 93–109 (2003)
20. Kraft, B., Nagl, M.: Visual Knowledge Specification for Conceptual Design: Definition and Tool Support. Advanced Engineering Informatics 21, 67–83 (2007)
21. Leonard, J.J., Durrant-Whyte, H.F.: Direct Sonar Sensing for Mobile Robot Navigation. Kluwer Academic Publishers, Boston (1992)
22. Elfes, A.: Sonar-based real-world mapping and navigation. IEEE Transactions on Robotics and Automation, 249–265 (1987)
23. Sakas, G., Hartig, J.: Interactive visualization of large scalar voxel fields. In: Vizualization, pp. 29–36 (1992)
24. Schroeder, W., Zarge, J., Lorensen, W.: Decimation of triangle meshes. Computer Graphics, 65–70 (1992)
25. Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M.: Towards 3d point cloud based object maps for household environment. Journal of Robotics and Autonomous Systems 56, 927–941 (2008)
26. Weingarten, J., Siegwart, R.: EKF-based 3D SLAM for structured environment reconstruction. In: Proc. of IROS 2005 (2005)
27. Triebel, R., Pfaff, P., Burgard, W.: Multi-level surface maps for outdoor terrain mapping and loop closing. In: Proc. of IROS, pp. 1–2 (2006)
28. Siemiatkowska, B., Gnatowski, M., Chojecki, R.: Cellular neural networks in 3d laser data segmentation. In: 9th WSEAS International Conference on NEURAL NETWORKS, pp. 84–88 (2008)
29. Gu, J., Cao, Q., Huang, Y.: Rapid traversability assesment in 2.5d grid based map on rough terrain. Internationl Journal of Advanced Robotic Systems 5(4), 389–394 (2008)
30. Chua, L., Young, L.: Cellular Neural Networks. IEEE Transaction on Circuit System 2, 985–988 (1988)
31. Chua, L., Wu, C.W.: On the Universe of Stable Cellular Neural Networks. IEEE Transaction on Circuit System 42, 559–577 (1995)
32. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion - Theory, Algorithms, and Implementations. MIT-Press, Cambridge (2005)
33. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. 4, 23–33 (1997)

34. Gorte, B., Vosselman, G., Sithole, G., Rabbani, T.: Recognizing structure in laser scanner point clouds. In: Proceedings of Conference on Laser Scanners for Forest and Landscape Assessment and Instruments (2004)
35. Pu, S.: Vosselman, G.: Knowledge based reconstruction of building models from terrestrial laser scanning data. ISPRS Journal of Photogrammetry and Remote Sensing, 0924-2716 (2009) (in Press) (Corrected Proof)
36. Yu, Y., Ferencz, A., Malik, J.: Extracting objects from range and radiance images. IEEE Transactions on Visualization and Computer Graphics 7, 351–364 (2001)
37. Snyder, J.P.: Map Projections – A Working Manual, United States Government Printing, Washington, DC, pp. 98–103 (1987)
38. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE CVPR (2001)
39. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: IEEE ICIP 2002, vol. 1, pp. 900–903 (2002)