DOI 10.15622/sp.60.1

M. NEGRETE , J. SAVAGE , L. CONTRERAS -TOLEDO

A MOTION-PLANNING SYSTEM FOR A DOMESTIC SERVICE ROBOT

Negrete M., Savage J., Contreras L. **A Motion-Planning System for a Domestic Service Robot.**

Abstract. Service robots are intended to help humans in non-industrial environments such as houses or offices. To accomplish their goal, service robots must have several skills such as object recognition and manipulation, face detection and recognition, speech recognition and synthesis, task planning and, one of the most important, navigation in dynamic environments. This paper describes a fully implemented motion-planning system which comprehends from motion and path planning algorithms to spatial representation and behavior-based active navigation. The proposed system is implemented in Justina, a domestic service robot whose design is based on the ViRBot, an architecture to operate virtual and real robots that encompasses several layers of abstraction, from low-level control to symbolic planning. We evaluated our proposal both in simulated and real environments and compared it to classical implementations. For the tests, we used maps obtained from real environments (the Biorobotics Laboratory and the Robocup@Home arena) and maps generated from obstacles with random positions and shapes. Several parameters were used for comparison: the total traveled distance, the number of collisions, the number of reached goal points and the average execution speed. Our proposal performed significantly better both in real and simulated tests. Finally, we show our results in the context of the RoboCup@Home competition, where the system was successfully tested.

Keywords: Autonomous navigation, behavior-based robotics, domestic service robots, path planning.

1. Introduction. According to the International Organization for Standardization a service robot is a robot that performs useful tasks for humans or equipment excluding industrial automation applications [13]. This norm also states that a robot, in general, requires a certain degree of autonomy which, in this context, is considered as the ability to perform intended tasks based on current state and sensing, without human intervention. The degree of autonomy in a service robot ranges from partial autonomy, which includes human-robot interaction, to full autonomy (without human intervention). Although by definition service robots do not need to be fully autonomous, the more degree of autonomy, the more useful the robot will be. For example, in space applications, due to the delay caused by the enormous distances, teleoperation is not a feasible solution, thus a fully autonomous robot is needed. In domestic applications, if a robot is going to clean a room, it should do it without human help. Thus, a high degree of autonomy is desired in a domestic service robot.

Domestic service robots need several skills: motion planning in dynamic environments, object recognition and manipulation, human-robot interaction, which can include human detection, gesture recognition and speech understanding and synthesis, task planning, real-time awareness, among others. The problem of motion planning refers to the robot's ability to localize itself, map an environment and plan secure paths from one point to another [6].

A motion planning system consists of many components that are interconnected and whose performance depend on other modules' performance. For example, following a path depends on the path planner and the low level control, and both depend on the localizer, but also the localizer depends on the control module because if robot moves abruptly or shows oscillations, localization will be more likely to fail. In principle, one can test each module in an isolated manner and the reliability of the whole system can be argued based on the reliability of each component. This is possible when only a feedforward interaction is present, nevertheless when feedback is involved, the whole system performance must be tested since it can differ from each module's performance.

As stated in [1], robotic competetions are useful to test and compare different algorithms and systems by providing an objective performance evaluation under controlled and replicable conditions. Robocup@Home [30] and RoCKIn@Home [1] are examples of competitions for domestic service robots to evaluate specific and general robot's performance. Contrary to experiments, where specific hypothesis are tested, a competition usually evaluates general abilities of robot systems. Testing a robot system in a competition also helps to find solutions to problems that only arise in fully integrated systems.

There is a large amount of work reported in the scientific literature about motion planning for autonomous robots, nevertheless, it is still an open and challenging problem when talking about robots navigating in real environments [5]. This can be seen, for example, in the Robocup@Home 2018 rulebook where the tasks of three tests are focused, among others, in safe navigation and obstacle avoidance. Autonomous navigation is developed to the point that there are open libraries to implement navigation systems, but these libraries commonly implement basic skills and a lot of work can be done, for example, for improving path planning in hard-to-navigate environments such as messy rooms. Robocup@Home rulebook also contemplates tests where the robot has to evade hard-to-see obstacles such as an apple (small object), glass or Lego brick. The fact that very recent competitions still pose challenges for testing autonomous navigation in domestic service robots shows that navigation is still an open problem.

This paper is organized as follows: In Section 2 we present the related work and highlight the novelties of our approach. Section 3 is dedicated to explain ViRBot, the architecture our domestic service robot is based on. In Sections 4 to 8 we explain in detail every module of the proposed motion planning. In Section 9 we describe the results of applying the proposed system in the context of the tasks required by the international competitions Robocup@Home and we compare the performance with one of the most used packages for navigation. We state our conclusions and sketch the future work in Section 10.

2. Related Work.

2.1. Complete motion planning systems. Most of the literature in this topic reports isolated problems and experiments, such as mapping algorithms, path planning algorithms, localization or obstacle avoidance but there is only a few works reporting fully implemented and tested motion planning systems for domestic service robots. Examples of these works are the robots Markovito [4], LISA [26] and Cosero [27].

The work of [4] propose a navigation system based on a three-level software architecture: functional, execution and decision. This motion planning system uses dynamic programming for path planning, particle filters and geometric landmarks (corners and lines) for localization, occupancy grids for environment representation and include semantic information for a better motion planning. This motion planning system is based on a well defined architecture both, contrary the work presented in this paper, they do not use biology models of behavior.

The work of [27] is focused on the design and building of a cognitive service robot. To building a map, their motion planning system uses simultaneous localization and mapping based on Rao-Blackwellized particle filters and the Adaptive Monte Carlo Localization for estimating the robot's position. Authors of this work use 3D surfel grids from which they extract 2D navigation maps by exploring the traversability of surfels. When using a 2D sensors, they extract line segments from 2D scans and align them with the surfels in the map. Their software is also designed based on an architecture with four layers: task, subtask, action and perception, and sensori-motor.

The work of [26] developed a ROS-based autonomous service robot called LISA. Its motion planning system also uses particle filters for map building and localization. A* algorithm on the occupancy grid is used for path planning and the calculated path is post processed to get smooth trajectories. The smooth path is followed using waypoints along the planned path. Authors mention the close integration with the graphical user interface as the main benefit of their motion planning system. Contrary to this work, [26] do not mention the use of reactive behaviors to deal with unexpected obstacles.

2.2. Localization and Mapping. In order to fulfill their tasks, service robots need a representation of the environment (a map either geometric, topological or semantic), and its current configuration within such environment. The problem of building a map while estimating robot's position at the same time is called SLAM (Simultaneos Localization and Mapping) and is commonly solved using Kalman [9] or particle filters [11]. More recent work has tackled this problem using cameras and data compression techniques such as [8] and [7]. To solve only the localization problem, one of the most used techniques in the development of service robots is the Adaptive Monte Carlo Localization [28] which is also used in this work.

Roadmaps are a class of topological maps that are useful for mobile robots' navigation in structured environments. There are several techniques for roadmap construction when a geometrical representation of the objects in the environment is available, for example, Voronoi diagrams [17], visibility maps [19] or probabilistic roadmap methods [15]. In this work we describe how we build roadmaps using data from RGB-D cameras and vector quantization techniques. A similar work is presented in [24]. Vector quantization has a very high computational cost and processing can be very slow. In this proposal computation time is significantly reduced by implementing such algorithms in parallel.

2.3. Obstacle Avoidance. Artificial Potential Fields [16] are a technique for path planning and obstacle avoidance. In this work, potential fields are used as a behavior and they are part of the behavior-based methods of the ViRBot architecture. Costmaps are also a technique commonly used for obstacle avoidance and path planning, for example, the work of [20] uses a scheme of layered costmaps to plan paths in dynamic environments. Nevertheless a costmap requires a map while artificial potential fields can be purely reactive. As it will be discussed later, costmaps may become uneffective when robot is navigating in narrow spaces.

2.4. The ROS package *nav2d*. Robot Operating System (ROS) [23] is an open source middleware which provides the functionality commonly needed in the development of software for autonomous mobile robots such as message passing and package management. ROS also has several packages with algorithms to implement the most common skills needed in an autonomous robot such as perception of objects and people, knowledge representation and navigation. *Nav2d* is an open source package for 2D navigation (http://wiki.ros.org/nav2d) whose main features, according to its web page, are a purely reactive obstacle avoidance, a simple path planner and a graph-based SLAM that allows multiple robots to cooperatively build a map. The last update was done in 2017 (according to the web page). This package was used to compare the performance of the proposed system.

3. ViRBot: an architecture for the operation of service robots. To develop domestic service robots a minimum hardware is required and also a design architecture must be followed in order to integrate the huge amount of software that is required to operate this kind of autonomous robots. To perform the intended tasks, domestic service robots must have from low-level controls, to make actuators move according to a set point, to high-level task planning, to understand, plan and execute commands given by humans. Designing a domestic service robot requires much more than merging and coordinating top trending technologies in machine learning, computer vision, control, navigation, real time mapping, artificial intelligence, and human-computer interaction. A service robot interacts with an ever-changing environment in which the "optimal" conditions of a research laboratory are almost never met. For these reasons, a design architecture is mandatory for succeeding in the development of autonomous robots.

ViRBot is an architecture to design, organize, integrate and test software for autonomous service robots [25]. In the ViRbot architecture the operation of a service robot is divided in four general layers: Inputs, planning, knowledge management and execution, having each of them several subsystems, see Figure 1. Each subsystem has a specific function that contributes to the final operation of the robot. This architecture has similar features to the ones presented in the INTERRAP agent architecture [21].

Input layer. This layer encompasses all propioceptive and exteroceptive robot's sensors. Every sensor has also a simulation mode so that, when a virtual robot is used, algorithms can be tested using simulated sensor data. This layer also includes data coming from the human-robot interaction subsystem, which can be recognized voice or gesture commands. Digital signal processing techniques are applied to data comming from all sensors to obtain a symbolic representation of the environment and, with this representation, a series of beliefs are generated. A set of programmed tasks are also included in this layer since they stablish a set of goals used to plan actions.

Planning layer. In cooperation with the Knowledge Management layer, beliefs are validated in the Planning layer and thus, a situation recognition is created. Given such recognized situation, a set of goals are activated and then, the action planner finds a sequence of physical operations to achieve the generated goals. The motion planner module is in charge of planning the movements of the actuators (mobile base and manipulators) to perform the sequence of physical operations generated by the action planner. If during the execution of a plan, something not considered in it happens, the exception recognizer module will try to solve the problem by setting new goals and re-planning the overall sequence of operations.

Knowledge Management layer. This layer builds and manages the robot's knowledge both declarative and procedural. The cartographer module is in charge of three tasks: to keep a representation of the environment using geometric maps and roadmaps, to build such representation using SLAM and

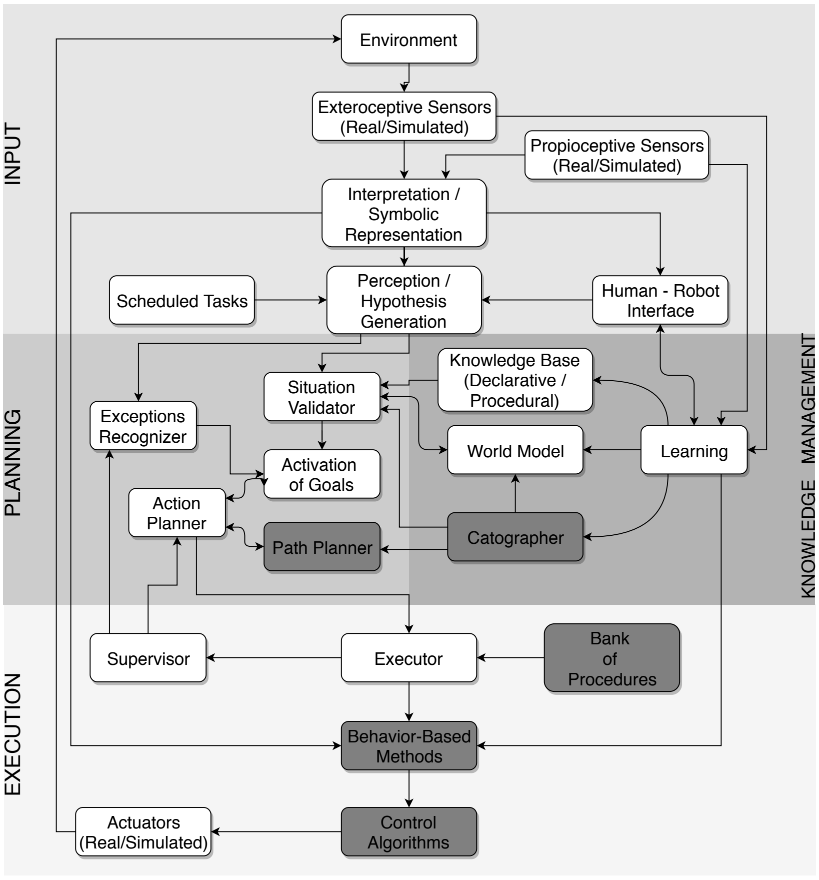


Fig. 1. The ViRBot architecture: several layers and subsystems are integrated to operate autonomous service robots. Highlighted modules are those involved in this paper

clustering algorithms, and to localize the robot using an Extended Kalman Filter and the Adaptive Monte-Carlo Localization. This layer also includes a knowledge base which keeps both procedural and declarative knowledge. The first one encodes the knowledge of an expert using a rule-based system written in CLIPS, a logic language developed by NASA. The declarative knowledge encompasses patterns of objects and faces, predefined locations and regions in the environment and positions of objects and persons. Learning algorithms are used to train patterns and to generate new behaviors.

Execution layer. The first module of this layer is the executor. It takes a set of hardwired procedures to transform plans generated by the action planner into simpler sequences of actions. The executor informs the result of the execution of each hardwired procedure to the supervisor, which checks that all actions and movements are performed according to the plans. The bank of hardwired procedures is a set of augmented finite state machines used to partially solve specific problems such as taking and object or ask and memorize a name. Once the executor generates simple sequence of actions, a set of behaviors are used to cope with problems not foreseen by the action planner, like the avoidance of unknown obstacles. Behaviors generate signals taken as desired values by the control algorithms and finally, the outputs of such low-level controls are used to move the actuators.

The ViRBot and the proposed system. The ViRBot architecture is an abstraction of the tasks and interactions between of the software used to operate a service robot. A program running on the robot can perform more than one task of the ViRBot and also a subsystem of the ViRBot can be implemented in more than one programs. The proposed system is part of the path planner, cartographer, human-robot interface, bank of procedures, behaviors and control algorithms. Figure 1 shows, highlighted in green, those subsystems involved in this proposal. In the following sections we describe each module of the proposed system and their corresponding parts in the ViRBot architecture.

4. Control Algorithms. In Sections 5 to 8 it is explained how the motion planning system plans paths, represents the environment and uses behaviors to avoid obstacles, nevertheless, all those algorithms will require control laws to guaranty that all plánned motions will be performed correctly. For that purpose, we designed a non-linear control law.

Consider an omnidirectional mobile base like the one shown in Figure 2, where configuration is determined by three variables $[x_r, y_r, \theta_r]$. If only the kinematic part is considered and assuming there is no slip, the model of the

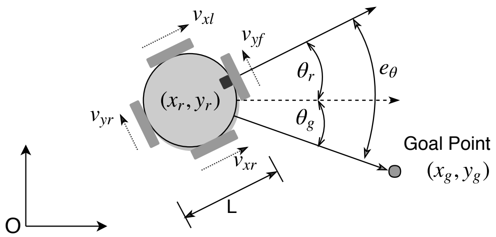


Fig. 2. Omnidirectional mobile base, the current and the goal position

robot is given by

$$\dot{x_r} = v_x \cos\theta_r - v_y \sin\theta_r, \tag{1}$$

$$\dot{y_r} = v_x \sin\theta_r + v_y \cos\theta_r, \tag{2}$$

$$\dot{\theta_r} = \omega, \tag{3}$$

where v_x, v_y and ω are the frontal, lateral and angular speeds, respectively, taken as input signals.

Note that the dynamic part of the robot is not modeled, i.e., the configuration $[x_r, y_r, \theta_r]$ is also considered as the state vector. Also, it is assumed that v_x, v_y and ω can be set arbitrarily. In the real system this is not the case, the real input signal is the voltage set in the motor terminals, nevertheless, we can assume that electrical and mechanical responses of the four motors are fast enough and thus, we can set any desired motor speed and it will be reached in a very small time.

The objective of the control laws is to determine signals v_x, v_y and ω such that it can be ensured that the robot will reach the goal position (x_g, y_g) even in the presence of uncertainties (such as non modeled dynamics) and perturbations. Although the mobile base is omnidirectional, in this work all movements are done taking it as a differential mobile base. Omnidirectionality is used in our service robot in other tasks where fine lateral movements are needed (e.g., when the robot needs to move laterally for grasping and object) but that is out of the scope of this work.

Consider the scheme shown in Figure 2. Let $theta_g$ be the desired angle, which corresponds to the angle of the position error vector $[x_g - x_r, y_g - y_r]^T$, calculated as

$$\theta_g = \text{atan2}\left(y_g - y_r, x_g - x_r\right),$$

and then, angle error can be calculated as

$$e_\theta = \theta_g - \theta_r = \operatorname{atan2}(y_g - y_r, x_g - x_r) - \theta_r. \tag{4}$$

It is important to note that e_θ, same as any other angular measure, must be always in the interval $(-\pi, \pi]$. If the difference given by (4) results in an angle greater than π or less than $-\pi$, it must be corrected so that $e_\theta \in (-\pi, \pi]$ and the correct performance of the control laws can be guaranteed.

To model an omnidirectional base as a differential one, it is enough to set $v_y = 0$. Assuming we have a mobile robot whose model is given by (1)-(3), then the control law

$$v_x = v_{max} e^{-\frac{e_\theta^2}{\alpha}}, \tag{5}$$

$$\omega = \omega_{max}\left(\frac{2}{1+e^{-\frac{e_\theta}{\beta}}} - 1\right), \tag{6}$$

ensures the robot will reach the goal position $(x_g, y_g)^T$. Given v_x, v_y and ω, the four motor speeds (left v_{xl}, right v_{xr}, front v_{yf} and rear v_{yr}) can be calculated as

$$\begin{aligned} v_{xl} &= v_x - \frac{L}{2}\omega, \qquad v_{yf} = v_y + \frac{L}{2}\omega, \\ v_{xr} &= v_x + \frac{L}{2}\omega, \qquad v_{yr} = v_y - \frac{L}{2}\omega, \end{aligned}$$

where L is the robot diameter (see Figure 2).

The control law (5)-(6) has four parameters: v_{max} and ω_{max} are the maximum linear and angular speeds, respectively, that the robot can reach during its movement. These parameters, in principle, can be set arbitrarily, nevertheless, in a real implementation, they are bounded by the actuators capabilities.

For a better understanding of constants α and β, consider the Figure 3. It can be seen that constant α determines how fast linear speed v decreases when error angle increases. A small α will make linear speed to decrease too fast, that is, robot will not move forward until it is pointing directly to the goal point. In other words, it will first turn until angle error is very small and then it will move towards the goal point.

Constant β determines how fast increases angular speed ω when error angle increases. In general, a small β will make the robot to keep always "point-

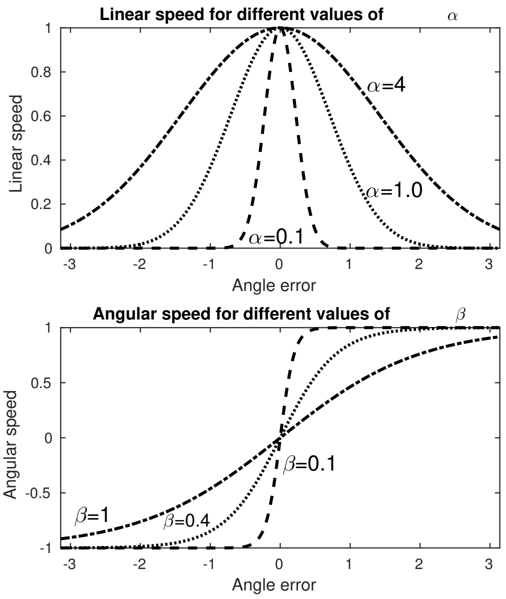


Fig. 3. Linear and angular speeds for different values of α y β

ing" to the goal point, i.e., it will help to a better path tracking, nevertheless, if β is too small, it can generate undesired oscillations.

5. Cartographer. Cartographer, in the ViRBot architecture, is part of the Knowledge Management layer. It does not make any movement plan but builds and keeps a representation of the environment. Using this representation and the information coming from sensors, it also estimates the robot configuration.

5.1. Environment Representation. Occupancy grids are a type of geometric maps in which the space is dicretized with a given resolution. Each resulting cell is assigned to a number $p \in [0,1]$ which indicates the occupation level. In the simplest interpretation, $p = 1$ means the cell is occupied and $p = 0$, the cell is free space. Occupancy grids can also be used with a probabilistic approach where p indicates the certainty that a given cell is occupied. In this work we use occupancy grids for representing the environment due to the ease to build them and the availability of standard formats to represent them such as the *OccupancyGrid* message in the platform ROS.

5.2. Localization. Adaptive Monte Carlo Localization is a probabilistic method used to determine robot's position based on particle filters to track the 2D position against a known map. It is implemented in the open source ROS package *amcl*. The theorical base is explained in [28] and the implementation details are documented in the package web page. In this work, we used the *amcl* package to localize our domestic service robot.

5.3. Roadmap Construction using Clustering Algorithms. Roadmaps are useful for mobile robots' navigation in structured environments. If such roadmaps are constructed with a fast enough sampling time and based on information extracted from robot's sensors, they can be used for obstacle avoidance. If a geometric representation is not available, it can be obtained by vector quantization (VQ) methods and, based on this representation, it is possible to build a roadmap. Roughly explained, the process for building such roadmaps is as follows: we process the point cloud acquired from a Kinect device to separate free and occupied space. Then, both the free and occupied space are clustered. Centroids and sizes of the occupied space clusters are used as a geometrical representation of the objects in the environment. Centroids of the free space are used as nodes of the roadmap.

Separation of Free and Occupied Space. RGB-D cameras provide information through an RGB image and a point cloud that represents the spatial position of each pixel of the captured image. This research uses only the spatial information which comes as a set R of triplets of the form $r_j = \left(x_{screen_j}, y_{screen_j}, d_j\right)$, where $\left(x_{screen_j}, y_{screen_j}\right)$ is the pixel location in the image and d_j, the distance to the object on the line of sight. The point cloud S expressed in cartesian coordinates $s_j = (x_j, y_j, z_j)$ of the objects w.r.t. the RGB-D camera plane, can be obtained with the transformation

$$s_j = Mr_j, \tag{7}$$

with M, the matrix of intrinsic parameters of the RGB-D camera, which can be obtained by several methods (e.g., the work of [14]).

Since the camera is mounted on a mechatronic head with pan and tilt movements, it is convenient to apply a homogeneous transformation to express point with respect to the mobile base. After transformation, a point p_j is classified as free space if its z component is less than a constant K_h and, as occupied space, otherwise.

Vector Quantization. Vector Quantization [18] is commonly used for data compression in telecommunications and digital signal processing. In the field of robotics, it is also used to compress data and get a smaller but significant set of data. In this work, we use VQ to cluster the free and occupied space and, based on this clusters, to construct a roadmap.

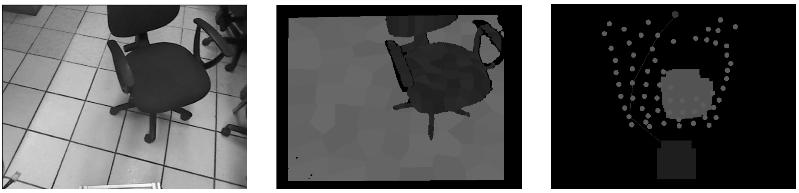
Given a point cloud P, i.e., a set of n_v vectors $p_j = (x_j, y_j, z_j); j = 1, ..., n_v$ that represent the position in space, a set of centroids that represent these vectors is found. A collection of centroids is called a codebook and it is designed from a long training sequence that is representative of all vectors p_j to be encoded. The codebook is created with the Linde-Buzo-Gray (LGB) algorithm [18] as explained in Algorithm 1.

Algorithm 1. Vector Quantization by the Linde-Buzo-Gray algorithm

```
Require:
  Point cloud $P = \{p_j | p_j = (x_j, y_j, z_j), j \in [1, n_v]\}$
  $L_d$ the desired codebook size
Ensure: Codebook $D = \{C_{L_m}\}$

  $m \leftarrow 1$ //Current iteration
  $L_m \leftarrow 1$ //Current number of centroids
  //First centroid is the average of all points
  $C_1 \leftarrow \frac{1}{n_v} \sum_{j=1}^{n_v} p_j$
  $D_m \leftarrow \{C_1\}$ //Initial codebook
  while $L_m < L_d$ do
    for $\forall C_i \in D_m$ do
      $\psi \leftarrow$ disturbance of small magnitude
      Obtain two new centroids by adding $\pm\psi$
    end for
  //$D_{m+1}$ will now contain $L_{m+1}$ new centroids
    $L_{m+1} \leftarrow 2L_m$
  //Set of clusters for each centroid
    $R \leftarrow \{R_k\}, k \in [1, L_{m+1}]$
    $\overline{d_t} \leftarrow \infty$ //Average change in centroid values
    while $\overline{d_t} > \varepsilon$ do
      for $\forall p_j \in P$ do
        Assign $p_j$ to the nearest cluster $R_k$ whose corresponding centroid is
  $C_k \in D_{m+1}$.
      end for
      for $\forall R_k \in R$ do
        Recompute centroid $C_k$ by averaging all vectors $p_j \in R_k$
      end for
      Calculate $\overline{d_t}$ as the average distance between the current centroid $C_k$
  and its value in the previous iteration
    end while
  end while
```

The desired codebook size L_d (number of regions in the environment) is chosen with a tradeoff between computation time limitations for real time operation and the desired precision. In this work, we used $|\psi| = 0.01$, $\varepsilon = 0.03$ and $L_d = 64$. The LGB algorithm is applied to cluster both the free and occupied space and a total of 128 centroids are calculated, 64 for the free space and 64 for the occupied space. Figure 4, left and center, shows the resulting clusters.



a) b) c)

Fig. 4. a) original image captured by the Kinect; b) free space clusters are colored in light gray and occupied space clusters, in dark gray. The black regions are those points with no depth information; c) resulting environment representation. Dots represent the nodes (free space centroids) used to build the roadmap and calculate a path. Light gray rectangles represent obstacles from which paths are calculated by A* algorithm to reach a goal point

Roadmap Construction After the free and occupied spaces are separated and clustered, the roadmap is built following the Algorithm 2. In this algorithm $e(v, v\prime)$ represents the edge between nodes v and $v\prime$ and $Vis(v, v\prime, p)$ is a function that determines if it is possible to reach the node v from node $v\prime$ without crashing with the obstacle whose centroid is p. NV means Not Visible and function Vis return this value when node v is not visible from $v\prime$.

Algorithm 2. Roadmap construction from clusterized space

```
Require:
  $P = \{p_i\}$ //Centroids of the quantized occupied space
  $C = \{c_j\}$ //Centroids of the quantized free space
Ensure:
  Roadmap $G(V, E)$ with
  $E$: The edges of the roadmap
  $V$: The nodes of the roadmap

  $E \leftarrow \emptyset$
  $V \leftarrow C$
  $i \leftarrow 0$
```

```
for $\forall v \in V$ do
   for $\forall v\prime \in V$ do
      if $e(v,v\prime) \notin V \wedge \mathrm{Vis}(v,v\prime,p) \neq NV\ \forall p \in P$ then
         $E \leftarrow E \cup \{e(v,v\prime)\}$
      end if
   end for
end for
```

Figure 4 shows the original image, the clusterized free and occupied space and the resulting roadmap. In general, clustering algorithms have a very high computational cost, which results in latencies that could make not possible to use them online as a method for obstacle avoidance. Nevertheless, in this work vector quantization was implemented in parallel allowing to get much shorter latencies. Details of this implementation are given in [22].

6. Path Planner. In the ViRBot architecture, the Path Planner is part of the planning layer. It takes information coming from the Action Planner and the environment representation kept by the Cartographer to calculate safe paths.

If environment is represented with an occupancy grid, the problem of planning a path can be solved by applying a search algorithm in graphs. In this case, each cell represents a node in the graph and it is considered to be connected only with the neighbor cells belonging to the free space. To determine the neighbor nodes, either four or eight-connectivity can be used.

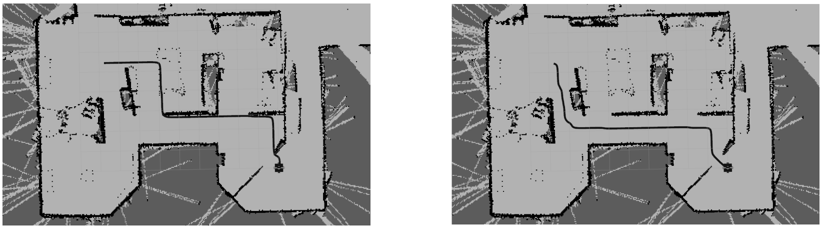
A* is a search algorithm which explores the path with the lowest expected cost. For a given node n, the expected cost $f(n)$ is calculated as

$$f(n) = g(n) + h(n),$$

where $g(n)$ is the cost of the path from the start node until node n and $h(n)$ is an heuristic function which determines *a cost* that would be expected from node n to the goal node. The expected cost $h(n)$ must actually subestimate the real value, i.e., it must be hold that $h(n) \leqslant g(n) \quad \forall\, n \in\ Graph.$

The A* algorithm finds a path that is optimal with respect to the cost function $g(n)$. This function should be designed according to what is needed in the robot navigation: traveling the shortest distance, the fastest path or the most energy-efficient trajectory, for example. In this proposal, the risk of collision is taken as part of the cost function such that the robot will always navigate through the safest paths. To achieve this, we define the cost function as

$$g(n) = d(n) + r(n), \tag{8}$$



a) b)

Fig. 5. Comparison of paths using: a) only distance as cost function; b) distance plus nearness to obstacles

where $d(n)$ is accumulated distance from the starting node to the current node n and $r(n)$ is a value inversely proportional to the distance from node n to the nearest obstacle. Figure 5 shows a comparison with and without the component $r(n)$ in the cost function. As can be seen in the upper Figure, if only distance is taken as a cost function, the path is calculated with positions next to the walls, since this is the shortest distance. Lower Figure shows the result considering the nearness to obstacles as a cost function and, as it can be seen, it is much safer for navigation.

A similar path can be obtained by augmenting the size of all obstacles in the map, nevertheless, with this approach the planner may be unable to calculate paths through small spaces. Costmaps are also a similar approach but, as it will be shown in the results section, they also have the problem of being unable to calculate paths through narrow spaces.

The heuristic function we use is the Manhattan distance from the current node to the goal node. After the path is calculated, it is post-processed to obtain a smooth path such that the control signals can also be smooth and avoid damages to the motors.

7. Behavior-Based Methods. Deliberative and reactive are two paradigms in robotics each one with advantages and disadvantages. Deliberative paradigm assumes an environment representation and this gives robots high prediction capabilities and thus, more complex tasks can be solved. Since the robot actuates based on a knowledge representation, the response is slow and in general it is difficult to handle fast changes in the environment. In the opposite side, the reactive paradigm does not depend on a knowledge representation and its response is faster. Reactive paradigm is better to handle high dynamic environments but their prediction capabilities are poor.

Behavior-based robotics states that intelligence in robots can be achieved as an emergent property of a large enough set of behaviors interacting with each other [3]. A behavior is a pair of stimulus-response and, since there is no symbolic planning, response is fast and computational cost is low. Since each behavior represents a "direct connection" of sensors with actuators, it is possible to have several behaviors trying to control the robot's actuator. A solution for this problem is the use of an arbiter which can make a selection of only one behavior, based on a hierarchy or the amplitude of the response, or it can produce a response by weighting all behaviors responses.

Hybridizing paradigms allows to take the advantages of each one and that is the case of the ViRBot architecture. Environment representation and planning algorithms allows the robot to perform more complex movements while behavior-based methods allows it to deal with changes in the environment such as obstacles not considered in the map. In our work, once a path is planned, it is executed with a behavior-based approach. There are three behaviors running concurrently: *go-to-goal-point* behavior, *avoid-obstacle* behavior and *collision-risk* behavior. To assemble all behaviors, we use a priority-based arbiter where the first behavior has the lowest priority and the last one, the highest priority.

7.1. The Go-To-Goal-Point Behavior. This behavior takes as input signals the desired positions calculated by the path planner (see Section 6) and the robot's position estimated by the localization algorithms (see Section 5). The output of this behavior are the tire speeds calculated according to the control laws explained in Section 4. This behavior is considered to be activated any time a new path is calculated.

7.2. The Avoid-Obstacles Behavior. We use the potential fields method as a behavior to avoid unexpected obstacles. The potential fields method proposes to design a potential function $U(q) : R^n \to R$, where q is the robot position, such that it has a minimum in the goal point and local maxima in every obstacle's position. If U is taken as a potential energy function, then its gradient $\nabla U(q)$ is a force vector whose direction points to the direction of maximum change. If the robot moves always in the opposite direction to the gradient (a method called gradient descent), then it will stop its movement in a local minimum. Thus, if the potential function is correctly designed (with a minimum in the goal point and local maxima in every obstacle) and the robot moves following the gradient descent, then it will reach the goal point evading obstacles at the same time [16].

There are several methods to design the potential function U and one of the most used is the attractive and repulsive field method. Since the movement will be guided by the gradient ∇U and not the function U, it is easier to design directly ∇U, i.e., to design the attraction and rejection forces. In this work we use the forces proposed by [2] given by

$$F_a = \zeta \frac{(q-q_g)}{\|q-q_g\|}, \qquad \zeta > 0, \tag{9}$$

$$F_r = \begin{cases} \eta \left(\sqrt{\frac{1}{d} - \frac{1}{d_0}}\right) \frac{q_{oi}-q}{d} & \text{, if} \quad d < d_0, \\ 0 & \text{, otherwise,} \end{cases} \tag{10}$$

where q is the current robot position, q_g is the goal position, q_{oi} is the position of obstacle i and $d = \|q_{oi} - q\|$. These forces have three design parameters $\zeta > 0$, $\eta > 0$ and $d_0 > 0$. Constant ζ defines how big is the attraction force and constant η, how big is the rejection force. d_0 is called the influence distance. If the distance between an object an the robot is greater than d_0, then that object will no affect the robot's movement.

The resulting force is calculated as the sum of the attractive force and the mean of all repulsive forces:

$$F = F_a + \frac{1}{n}\sum_{i=1}^{n} F_r, \tag{11}$$

Once calculated the gradient, we move the robot using the Algorithm 3 (gradient descent). Constant $\alpha > 0$ must be small enough to avoid oscillations but also considering that a too small α will imply a high computational cost.

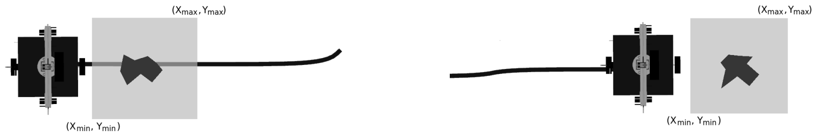
Algorithm 3. Gradient descend to move robot through a potential field

```
Require: Initial position $q_s$, goal position $q_g$, obstacle positions $q_{oi}$
Ensure: A sequence of points $\{q_0, q_1, q_2, \dots\}$
  $q_0 \leftarrow q_s$
  $i \leftarrow 0$
  while $\|\nabla U(q_i)\| > 0$ do
    $q_{i+1} \leftarrow q_i - \alpha \nabla U(q_i)$
    $i \leftarrow i + 1$
  end while
```

Finally, to move the robot towards each point q_i we use the control laws given by (5)-(6). Designing the potential function U using attractive and repulsive fields has the problem of local minima, i.e., the robot can stop in a point (local minimum) because gradient ∇U at that point is zero, but it is not the goal point. Nevertheless this problem can be overwhelmed if behaviors are assembled correctly, as it will be explained in the following subsections.

It is worth to note that, although Algorithm 3 assumes all obstacle positions are known, it is not necessary to have an environment representation with all obstacle positions. Since the robot has a laser sensor, every reading can be regarded as an obstacle and thus potential fields can be calculated directly from robot's sensors.

7.3. The Collision-Risk Behavior. The goal of this behavior is to stop the robot if a risk of collision is detected. Since the laser sensor makes only 2D scans, it is not possible to detect obstacles that do not cross the plane of readings, that's why we use the RGB-D camera mounted on the robot's head. The criterium for determining if there is a risk of collision is the detection of an obstacle in front of the robot but only if the robot is going to move forward. To determine the presence of an obstacle, we count the number of pixels in the RGB-D image that have a Z-value greater than zero and are within a rectangular area in front of the robot; if the counting exceeds a threshold K_obs, then there is an obstacle in front of the robot. Consider the Figure 6. The yellow square in front of the robot is the detection region and is defined by the bounding box $[(x_{min}, y_{min}), (x_{max}, y_{max})]$. Collision risk detection is performed according to Algorithm 4.



a) b)

Fig. 6. a) risk of collision is detected since the obstacle is in the robot's path; b) obstacle is not in the path of the robot, thus, no risk of collision is detected

Algorithm 4. Detection of collision risk

```
Require:
  Point cloud $P = \{p_j | p_j = (x_j, y_j, z_j), j \in [1, n]\}$
  Control signals $v$ and $\omega$
  Minimum number of points $K_{obstacle}$
  Minimum linear speed $v_{fwd}$ to consider the robot is moving forward
Ensure: Collision risk flag
  $i \leftarrow 0$
  Collision risk $\leftarrow$ False
  for $\forall p_j \in P$ do
    if $x_j > x_{min} \wedge x_j < x_{max} \wedge y_j > y_{min} \wedge y_j < y_{max}$ then
      $i \leftarrow i + 1$
    end if
  end for
```

```
**if** $v > v_{fwd} \wedge i > K_{obstacle}$ **then**
//The number of points is greater than a threshold and the robot is moving forward
    Collision risk $\leftarrow$ True
**end if**
```

The output of this behavior is always zero since it is intended to stop the robot in case of a risk of collision, thus, this behavior is considered activated any time the conditions described in Algorithm 4 are hold.

7.4. The Arbiter. Figure 7 shows a block diagram of the behavior based movement. Arbiter takes the activated behavior with the highest priority. The *go-to-goal-point* behavior activates any time a new path is planned. *Avoid-obstacles* behavior activates when the calculated repulsive force is greater than zero and the *collision-risk* behavior, when a collision risk is detected. With this scheme, robot is able to avoid unexpected obstacles and also the problem of local minima in potential fields is overwhelmed since, every time an obstacle is in front of the robot (possible causing local minima), the robot is stopped and a new path is calculated using a map built as described in Section 5.3.

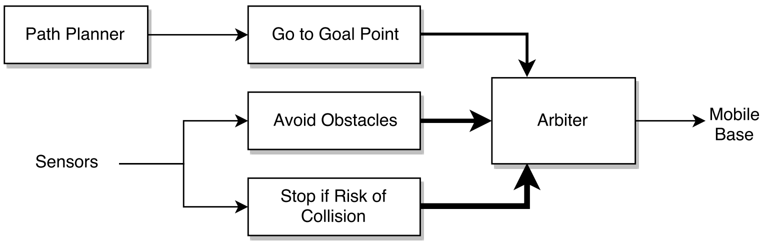


Fig. 7. The priority-based arbiter. Arrow thickness indicate the behavior priority

8. Bank of Procedures. In the ViRBot architecture, the bank of procedures is a set of state machines used to perform simple tasks such as taking an object (once position has been determined using computer vision algorithms), asking and memorizing a name and following a path, among others.

The control law (5)-(6) has the disadvantage of depending only on the angle error, i.e., robot does not decelerates as it approaches to the goal point. This will cause the robot to have strong oscillations when it is in a small region around the goal point. There are two ways to address this problem. The easiest one is to execute the control law only if the distance to the goal point $d = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2}$ is greater than a tolerance ε:

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{cases} \begin{bmatrix} v + \frac{L}{2}\omega \\ v - \frac{D}{2}\omega \end{bmatrix}, & \text{if} \quad d > \varepsilon, \\ \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{otherwise.} \end{cases}$$

In this case the robot will stop abruptly when the error distance is less than ε, which is not desirable since the robot should decelerate slowly as it approaches to the goal point. It is also desirable that robot increases it linear speed slowly at the beginning of the movement, which is not yet considered, because according to (5)-(6), the robot can have a linear speed of v_{max} since the begining of the movement.

In the other hand, control law (5)-(6) is designed to reach a goal point, but what we have to follow is not a point but a path composed by a sequence of points. Acceleration and deceleration can be achieved by parameterizing with respect to time the full path. This is commonly done by adjusting a fifth degree polynomial, or a higher degree if more variations of speed and acceleration are required. Nevertheless, expressing positions x and y as a function of time is easy when only straight-line movements are required, but in the case of a service robot moving in a domestic environment, paths can be too complex and parameterization with respect to time can become unfeasible.

Augmented Finite State Machine. An easy way to make the robot to accelerate and decelerate, that is, to get a speed profile, is through an Augmented Finite State Machine (AFSM) that establishes the value of v_{max} in control law (5)-(6). Let v_{sm} be the new maximum speed, such that

$$v = v_{sm} e^{-\frac{e_\theta^2}{\alpha}}, \tag{12}$$

$$\omega = \omega_{max}\left(\frac{2}{1+e^{-\frac{e_\theta}{\beta}}} - 1\right). \tag{13}$$

Control given by (12)-(13) is designed to reach a goal point (x_g, y_g), nevertheless, what we want is to track a path given by N points $(x_i^p, y_i^p) \quad i \in [0, N-1]$. To follow this path is is enough to set as a goal point, the point (x_i^p, y_i^p) which is "in front of" the robot, that is, a point in the path whose position is a small distance away from the current position. In this work, we consider "small distance" a distance of 0.3 [m].

Figure 8 shows the AFSM used to determine v_{sm} and the goal point. In general, v_{sm} is increased slowly until it reaches the final value of v_{max} and it starts to decrease when robot is “near” the last point of the path, that is, when distance r is less than a constant r_d. When distance to current goal point is less than 0.3 [m], then it is changed by the next point of the path. Values of r_d and Δv are determined depending on how fast the robot should accelerate and decelerate.

9. Results.

9.1. The Service Robot Justina. Justina is a domestic service robot built at the Biorobotics Laboratory of the National Autonomous University of Mexico and developed under the ViRBot architecture (see Section 3). This robot and its predecessors have been participating in the Robocup@Home league [30] since 2006 performing several tasks such as cleaning a table, serving drinks and several other tasks that humans ask for. To sense the environment, Justina has several sensors: two laser range finders, an RGB-D camera, a directional microphone, an array of omnidirectional microphones and encoders in each motor. The actuators of Justina consist of an omnidirectional mobile base, a 2-DOF head (pan and tilt movements) where the RGB-D camera and the directional microphone are mounted, two 7-DOF anthropomorphic manipulators and a 1-DOF torso to modify Justina’s height. Figure 9 shows robot Justina and the location of its sensors and actuators.

As mentioned in Section 2, the *nav2d* package was used as a baseline to compare the performance of our proposal. Experiments were conducted in real and simulated environments as follows.

9.2. Simulation Results. To test the performance of our proposal and compare it with the *nav2d* package, we generated 10 artificial maps consisting, each one, of polygons with random shapes and positions. Figure 10 shows examples of the artificial maps and the random points. We also used 2 real maps: The Biorobotics Laboratory and the @Home arena of the Robocup 2018 (shown in Figure 13). The *nav2d* pacakge was used with all constants and parameters in their default values, except for the maximum linear speed, which was set to 0.7 to be the same than the value used with the control laws (5)-(6), and the *map inflation radius*, whose value of 0.2 was selected as the greatest value that allowed *nav2d* to calculate a path through the doors of the environment.

The low-level control was implemented using as constants $\alpha = 0.6$, $\beta = 0.09$, $\omega_{max} = 1.0$ and $V_{max} = 0.7$. Path planning was implemented according to the algorithms described in Section 6. For RoC detection we used $x_{min} = 0.3$, $y_{min} = -0.25$, $x_{max} = 0.9$, $y_{max} = 0.25$, $K_{obstacle} = 30$, $K_\Pi = 0.05$ and $v_{fwd} = 0.1$. Potential fields used the constants $\zeta = 1.0$, $\eta = 5.0$ and $d_0 = 0.8$.

The baseline (*nav2d* package) and the proposal were tested with 100 random goal points for each map. Comparison was made base on three param-

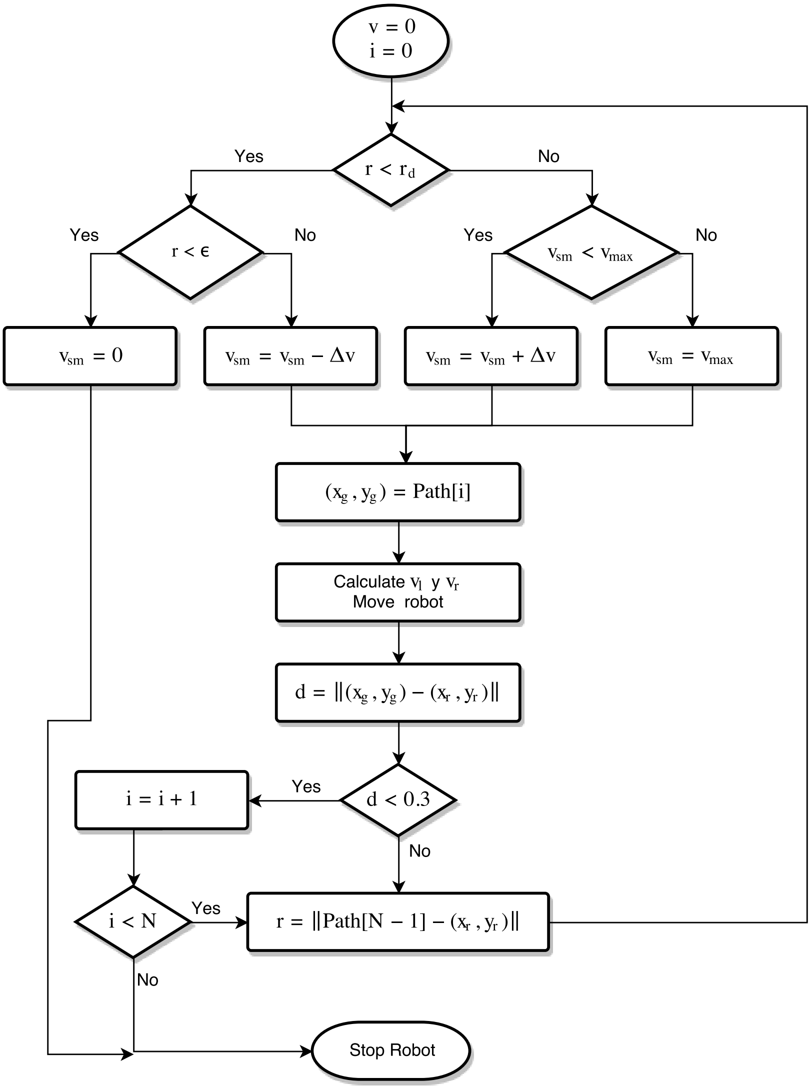


Fig. 8. Augmented Finite State Machine used to generate a speed profile

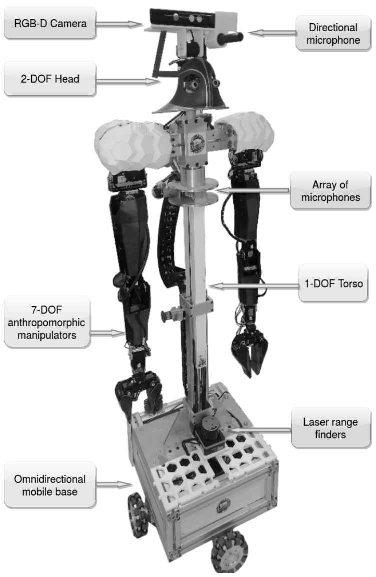


Fig. 9. The domestic service robot Justina

eters: **TDEDR**. Traveled Distance to Euclidean Distance Ratio. Calculated as the total traveled distance divided by the euclidean distance from the start to the goal point. We consider that the longer the path traveled, the less efficient the moving planning system. **AMpS**. Average Meters per Second. Calculated as the total traveled distance divided by the total time spent to reach the goal. This measure can give an insight of how fast the robot moves towards the goal. **NoC**. The total number of collisions (NoC) with any part of the map. Since these tests were simulated, we counted the times the robot touched any part of the map.

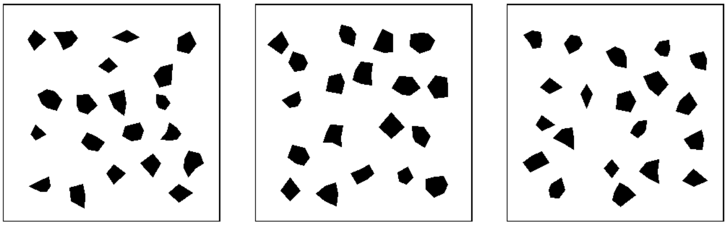


Fig. 10. Examples of the worlds used to test path planning and the random goal points

Table 1 shows the mean and standard deviation of the TDEDR for the 100 random goal points for each world. In order to determine if there is a significant difference between both navigation systems, a two-tailed t-Student test was performed. Last columns show the statistic and the p-value. In a similar way, tables 2 and 3 show the means, standard deviations and t-Student tests for AMpS and NoC, respectively.

Table 1. Ratio of the Traveled Distance to the Euclidean Distance (TDEDR) for our proposal and the baseline (the *Nav2d* package) for simulated scenes. One-hundred random paths were evaluated per map

	Proposal		Nav2d		Significance	
World	Mean	SD	Mean	SD	t	p-value
Biorobotics L.	1.29	0.39	1.18	0.20	2.45	0.01534
@Home arena	1.56	0.98	1.19	0.18	3.73	0.00030
Random 1	1.31	0.35	1.12	0.20	4.41	1.88E-5
Random 2	1.26	0.27	1.13	0.22	3.76	0.00022
Random 3	1.30	0.24	1.10	0.13	7.55	3.4E-12
Random 4	1.31	0.26	1.11	0.15	6.62	5.5E-10
Random 5	1.27	0.22	1.13	0.21	4.43	1.54E-5
Random 6	1.24	0.21	1.11	0.17	4.73	4.22E-6
Random 7	1.24	0.18	1.10	0.13	6.19	3.83E-9
Random 8	1.22	0.27	1.15	0.22	1.96	0.05032
Random 9	1.22	0.27	1.13	0.26	2.44	0.01540
Random 10	1.26	0.22	1.08	0.14	7.01	5.6E-11

Table 1 shows that TDEDR was greater for the proposal, i.e., distance traveled using the *nav2d* package was significantly less than the distance using the proposed navigation system in 11 out of 12 tested worlds (using 95% of confidence). This is due to the cost function described in Section 6. Since the nearness to obstacles is taken as part of the cost function, paths are in general longer because robot will tend to be away from all obstacles. Figure 11 shows a comparison of the paths calculated by Justina and by the *nav2d* package. As mentioned in Section 6, safer paths can be obtained by the *nav2d* package by incrementing the obstacle inflation radius, but doing so, disables the package to calculate paths through small doors. Despite the longer paths, Justina reached the goal points in shorter times, as it can be seen in table 2, which shows that the Average Meters per Second (AMpS) was significantly greater for the proposed system than the *Nav2d* package.

For both navigation systems, we counted the NoC (any touch to some part of the map was considered a collision). Table 3 shows the statistics of NoC for the 100 goals for every map. As it can be seen, NoC with Justina's system was significantly lower for all maps, as a result of the safer paths calculated by the path planner. Figure 11 shows a comparison of the paths calculated by the

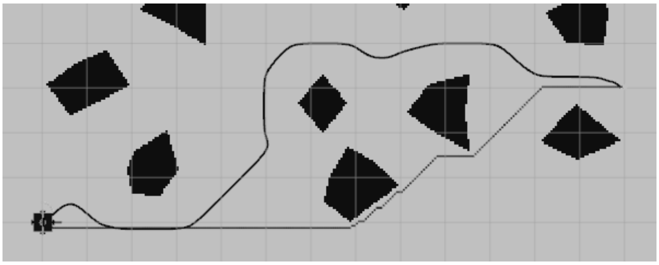


Fig. 11. Lower path: Calculated by *nav2d*. Upper path: Calculated by Justina. Justina's path is safer

Table 2. Average Meters per Second (AMpS) for our proposal and the baseline (the *Nav2d* package) for simulated scenes. One-hundred random paths were evaluated per map

	Proposal		Nav2d		Significance	
World	Mean	SD	Mean	SD	t	p-value
Biorobotics L.	0.43	0.107	0.20	0.042	20.08	2.2E-16
@Home arena	0.43	0.109	0.19	0.048	19.62	2.2E-16
Random 1	0.41	0.110	0.20	0.046	17.98	2.2E-16
Random 2	0.43	0.099	0.20	0.040	21.61	2.2E-16
Random 3	0.46	0.091	0.20	0.033	22.81	2.2E-16
Random 4	0.43	0.092	0.21	0.047	21.73	2.2E-16
Random 5	0.42	0.086	0.20	0.037	23.42	2.2E-16
Random 6	0.43	0.090	0.21	0.036	23.14	2.2E-16
Random 7	0.44	0.091	0.20	0.034	24.64	2.2E-16
Random 8	0.42	0.113	0.20	0.035	18.37	2.2E-16
Random 9	0.42	0.111	0.20	0.046	18.32	2.2E-16
Random 10	0.43	0.089	0.21	0.036	23.09	2.2E-16

baseline and the proposal. Justina's system gets safer paths which results in less number of collisions.

9.3. Results with the real robot. Similar to the simulated results, we tested our proposal and the *nav2d* package with 20 points but only in the Biorobotics Laboratory. For security reasons, once we detected the robot is going to crash or even touch some obstacle, we stopped the robot, thus, contrary to the simulated results, we don't have a measure of the number of collisions, instead, we have a Number of Reached Goal Points (**NoRGP**). For all goal points, we put some obstacles in front of the robot to check the performance for obstacle avoidance. All parameters and constants were the same than those used in the simulated results.

Of the 20 goal points, Justina reached 18 goals without crashing or touching the unexpected obstacles using the proposed system. When using the

Table 3. Number of Collisions (NoC) for our proposal and the baseline (the *Nav2d* package) for simulated scenes. One-hundred random paths were evaluated per map

	Proposal		Nav2d		Significance	
World	Mean	SD	Mean	SD	t	p-value
Biorobotics L.	0.22	0.48	1.96	1.76	-9.51	3.7E-16
@Home arena	0.43	0.97	2.70	1.82	-11.00	<2.2E-16
Random 1	0.10	0.30	1.88	1.66	-10.55	<2.2E-16
Random 2	0.08	0.31	2.19	1.61	-12.90	<2.2E-16
Random 3	0.13	0.34	1.91	1.62	-10.74	<2.2E-16
Random 4	0.08	0.27	2.27	1.68	-12.86	<2.2E-16
Random 5	0.05	0.22	1.85	1.59	-11.20	<2.2E-16
Random 6	0.07	0.25	2.08	1.64	-12.12	<2.2E-16
Random 7	0.15	0.48	2.40	1.73	-12.50	<2.2E-16
Random 8	0.14	0.38	2.43	2.10	-10.71	<2.2E-16
Random 9	0.15	0.39	1.99	1.50	-11.82	<2.2E-16
Random 10	0.07	0.26	1.93	1.68	-10.92	<2.2E-16

Table 4. Statistics of the Traveled Distance to Euclidean Distance Ratio (TDEDR) and the Average Meters per Second (AMpS) for the proposal and the baseline in the real environment. Last row shows the Number of Reached Goal Points (NoRGP)

	Proposal		Nav2d		Significance	
Param	Mean	SD	Mean	SD	t	p-value
TDEDR	1.06	0.337	1.36	0.365	-2.6376	0.01253
AETPM	0.19	0.052	0.17	0.054	1.4153	0.1661
NoRGP	18		14			

nav2d package, it only reached 14 goal points. Figure 12 shows the performance of the behavior-based approach for obstacle avoidance. The upper Figure shows the robot at the beginning of the movement. At this point, *go-to-goal-point* and *avoid-obstacles* behaviors are activated. The center Figure shows the moment when a collision risk is detected and the robot is stopped. Although a person was in front of the robot, what could cause a local minimum, robot was able to reach the goal due to the activation of the *collision-risk* behavior and the calculation of a new path, as shown in the lower Figure.

Table 4 shows the statistic data for the TDEDR and AMpS for both navigation systems. As it can be seen, in the real world with unexpected obstacles the mean distance traveled using the *nav2d* package was significantly longer than the distance with the proposed system. This means that the proposed system avoids objects more efficiently since it travels a shorter distance. The AMpS had no significant difference for both systems, which is actually a good indicator: the proposed system traveled shorter paths which, at the same average speed, means shorter travel times.

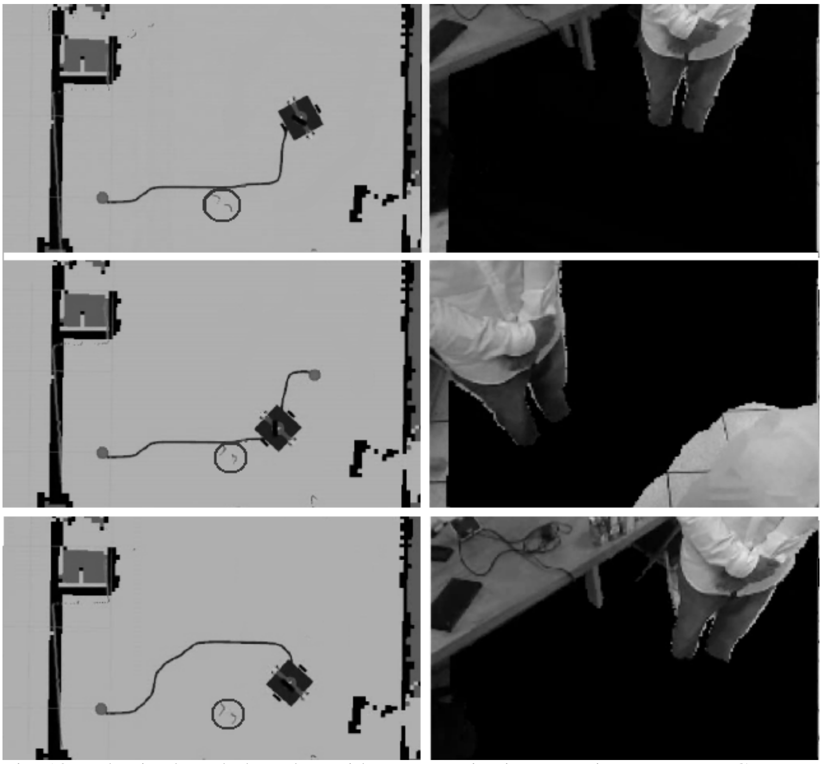


Fig. 12. Behavior-based obstacle avoidance. Top: justina starts its movement. Center: a risk of collision is detected. Bottom: a new path is calculated

9.4. Service Robots Competitions. To promote and foster development of autonomous robots, there are international competitions, such as Robocup [10] and RoCKIn [1]. Both competitions have the advantage of providing a standard problem where a wide range of technologies and approaches can be integrated, examined and evaluated. As stated in [12], competitions are useful when it is necessary to evaluate the general performance of a whole system and not only isolated hypothesis. For these reasons, robot Justina and its predecessors have been participating in the Robocup@Home since 2006. The motion-planning system described in this paper was tested in the last Robocup@Home edition held in Montreal, Canada, where Justina achieved the second place. In the @Home league, navigation was necessary in 6 of 7 tests [29]. In all of them, Justina successfully performed all the required motions. Figure 13 shows the map of the @Home arena where the proposal was tested.

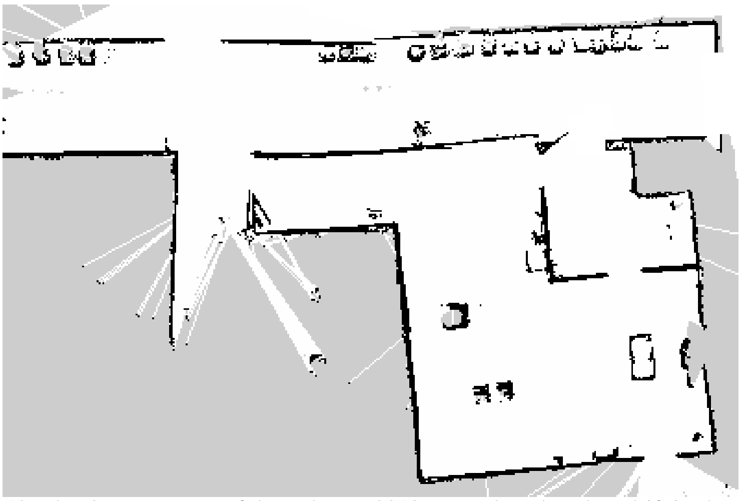


Fig. 13. The @Home arena of the Robocup 2018. Even though a clear drift is observed in the map during the slam process, our system performs well due to the active navigation module

10. Conclusions. We built a motion-planning system following the guidelines proposed by the ViRbot architecture. The motion planning was implemented in the service robot Justina and was tested both in real and simulated worlds. To evaluate the performance, we compared it with the nav2d package taking the Traveled Distance to Euclidean Distance Ratio (TDEDR), Average Meters per Second (AMpS), Number of Collisions (NoC) and Number of Reached Goal Points (NoRGP) as evaluation parameters. Distances traveled using the proposed system were significantly longer, in the simulated test, than the distances traveled using the *Nav2d* package, according to the TDEDR. This is due to the cost function used in the A* algorithm which results in longer but safer paths. Safety of paths was verified by the significantly lower Number of Collisions (NoC) when using our motion-planning system, in the simulation tests, and the greater Number of Reached Goal Points (NoRGP), in the real experiments. Contrary to the simulation results, in the experiments with the real robot, distances using our proposal were significantly shorter. This is due to the presence of unexpected obstacles, i.e., the proposed system avoided obstacles more efficiently.

As stated in the introduction of this work, testing whole systems is important when there is a feedback between the involved subsystems and thus, the reliability of each module does not guarantee the reliability of the whole system. In this work, we implemented several algorithms already described in the literature but we integrated them following the ViRBot architecture and we tested the performance of the whole system in the context of the Robocup@Home competition, where navigation was required in 6 of the 7 tests. Justina was able to successfully perform all the required motions.

As future work, we plan to make a more extensive comparison with other service robot architectures in similar conditions by promoting open source access and research collaboration.

References

1. Amigoni F. et al. Competitions for benchmarking: task and functionality scoring complete performance assessment. *IEEE Robotics&Automation Magazine*. 2015. vol. 22(3). pp. 53–61.
2. Arambula F., Padilla M.A. Autonomus robot navigation using adaptive potential fields. *Mathematical and Computer Modelling*. 2011. Vol. 40. pp. 1141–1156.
3. Arkin R.C. Behavior-based robotics. MIT Press. 1998. 491 p.
4. Avilés-Arriaga H.H. et al. Markovito: A Flexible and General Service Robot. Springer. 2009. pp. 401–423.
5. Banino A. et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*. 2018. vol. 557. no. 7705. pp. 429.
6. Choset H.M. et al. Principles of Robot Motion. MIT Press. 2005. 626 p.
7. Contreras L., Mayol-Cuevas W. Trajectory-driven point cloud compression techniques for visual slam. IEEE/RSJ International Conference on Robots and Systems (IROS). 2015. pp. 133–140.
8. Contreras L., Mayol-Cuevas W. O-poco: Online point cloud compression mapping for visual odometry and slam. IEEE/RSJ International Conference on Robots and Systems (IROS). 2017. pp. 4509–4514.
9. Dissanayake M.W.M.G. et al. A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on robotics and automation. 2001. vol. 17(3). pp. 229–241.

10. Ferrein A., Steinbauer G. 20 years of robocup. *KI-Künstliche Intelligenz*. 2016. vol. 30(3-4). pp. 225–232.
11. Grisetti G., Stachniss C., Burgard W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*. 2007. vol. 23(1). pp. 34–46.
12. Iocchi L. et al. RoboCup@ Home: Analysis and results of evolving competitions for domestic and service robots. *Artificial Intelligence*. 2015. vol. 229. pp. 258–281.
13. Robots and robotic devices - vocabulary. ISO 8373-201. International Organization for Standardization. Geneva. Switzerland. 2012.
14. Jin B., Lei H., Geng W. Accurate intrinsic calibration of depth camera with cuboids. European Conference on Computer Vision. 2014. pp. 788–803.
15. Kavraki L., Svestka P., Latombe J-C., Overmars M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*. 1996. vol. 12(4). pp. 566–580.
16. Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*. 1986. vol. 5(1). pp. 90–98.
17. Latombe J.C. Robot Motion Plannin. Kluwer Academic. 1991.
18. Linde Y., Buzo A., Gray R.M. An algorithm for vector quantizer design. *IEEE Transactions on Communications*. 1980. vol. 28(1). pp. 84–95.
19. Lozano-Pérez T., Wesley M.A. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*. 1979. vol. 22(10). pp. 560–570.
20. Lu D.V., Hershberger D., Smart, W.D. Layered costmaps for context-sensitive navigation. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2014 . 2014. pp. 709–715.
21. Müller J.P. The design of intelligent agents: a layered approach. Springer Science & Business Media. 1996. vol. 1177. 233 p.
22. Negrete M., Savage J., Cruz J., Marquez J. Parallel Implementation of Roadmap Construction for Mobile Robots using RGB-D Cameras. Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW). 2014. 184 p.
23. Quigley M., Gerkey B., Smart W.D. Programming Robots with ROS: a practical introduction to the Robot Operating System. "O'Reilly Media, Inc.". 2015. 448 p.
24. Savage J. et al. Construction of roadmaps for mobile robots' navigation using rgb-d cameras. *Intelligent Autonomous Systems*. Springer. 2016. vol. 13. pp. 217–229.
25. Savage J. et al. VIRbot: a system for the operation of mobile robots. RoboCup 2007: Robot Soccer World Cup XI. Springer. 2008. pp. 512–519.
26. Seib V., Memmesheimer R., Paulus D. Aros-based system for an autonomous service robot. Robot Operating System (ROS). Springer. 2016. pp. 215–252.
27. Stückler J., Schwarz M., Behnke S. Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero. *Frontiers in Robotics and AI*. 2016. vol. 3. 58 p.
28. Thrun S., Burgard W., Fox D. Probabilistic robotics. MIT Press. 2005. 672 p.
29. Van Beek L. et al. RoboCup@Home 2018: Rules and Regulations. 2018. 120 p. Available at: http://www.robocupathome.org/rules/2018_rulebook.pdf.2018 (accessed: 29.08.2018)
30. Wachsmuth S., Holz D., Rudinac M., Ruiz-del-Solar J. RoboCup@Home-benchmarking domestic service robots. Association for the Advancement of Artificial Intelligence (AAAI). 2015. pp. 4328–4329.

Negrete Marco — Master in Engineering, Ph.D. student of Laboratory of Biorobotics of Faculty of Engineering, National Autonomous University of Mexico (UNAM). Research interests: domestic service robots, control and automation, autonomous navigation, computer vision, cognitive and behavioral sciences. The number of publications — 12. mnegretev@gmail.com; Circuito Exterior S/N, Ciudad Universitaria, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Savage Jesús — Ph.D., professor, full professor of the Electrical Engineering Department, National Autonomous University of Mexico (UNAM). Research interests: autonomous mobile robots, digital signal processing, computer architectures. The number of publications —100. robotssavage@gmail.com; Circuito Exterior S/N, Ciudad Universitaria, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Contreras-Toledo Luis Angel — Ph.D., postdoctoral researcher of Advance Intelligence and Robotics Research Center, Tamagawa University. Research interests: computer vision, visual SLAM, robotics. The number of publications — 20. tenshi.unam@gmail.com; 6-1-1, Tamagawa-gakuen, Machida, Tokyo, 194-8610, Japan; office phone: +5215510323399.

Acknowledgements. This research is supported by UNAM-DGAPA-PAPIIT (grant IG110818).

УДК 006.72 **DOI** 10.15622/sp.60.1

М. Негрете , Х. Саваж , Л.Э. Контрерас -Толедо

СИСТЕМА ПЛАНИРОВАНИЯ ДВИЖЕНИЯ БЫТОВОГО СЕРВИСНОГО РОБОТА НА ОСНОВЕ АЛГОРИТМОВ ПРОСТРАНСТВЕННОГО ПРЕДСТАВЛЕНИЯ И АКТИВНОЙ НАВИГАЦИИ

Негрете М., Саваж Х., Контрерас-Толедо Л.Э. **Система планирования движения бытового сервисного робота на основе алгоритмов пространственного представления и активной навигации.**

Аннотация. Главным предназначением сервисных роботов является помощь людям в непромышленных средах, таких как дома или офисы. Для достижения своей цели сервисные роботы должны обладать несколькими навыками, например распознавание и манипулирование объектом, обнаружение и распознавание лиц, распознавание и синтез речи, планирование задач и одним из самых важных навыков — навигация в динамических средах. В статье описывается полностью внедренная система планирования движения, которая учитывает все: начиная от алгоритмов движения и планирования пути до пространственного представления и активной навигации на основе поведения. Предлагаемая система реализована в бытовом сервисном роботе под названием «Юстина», конструкция которого основана на робототехнической архитектуре под названием «ViRBot», использующейся для контроля действий виртуальных и реальных роботов, которая охватывает несколько уровней абстракции от низкоуровневого управления до символьного планирования. Мы оценили наш проект как в симулированной, так и в реальной среде и сравнили его с классическими реализациями. Для тестов мы использовали карты, полученные из реальных сред (Лаборатория биороботов и Robocup@Home arena), и карты, созданные из препятствий со случайными положениями и формами. Для сравнения использовалось несколько параметров: общее пройденное расстояние, количество столкновений, количество достигнутых целей и средняя исполнительная скорость. Наш проект значительно улучшился как в реальных, так и в симуляционных тестах. Представлены экспериментальные результаты успешно протестированной системы в контексте конкурса RoboCup@Home.

Ключевые слова: Автономная навигация, поведенческая робототехника, бытовые сервисные роботы, планирование маршрута.

Негрете Марко — аспирант лаборатории биоробототехники, инженерного факультета, Национальный автономный университет Мексики. Область научных интересов: сервисный робот, управление и автоматизация, автономная навигации, компьютерное зрение, когнитивистика, бихевиоризм. Число научных публикаций — 12. mnegretev@gmail.com; Сиркуито Экстериор, Сьдад Университария, Мехико, 04510, Мексика; р.т.: +52(55)56223041.

Саваж Хесус — Ph.D., профессор, профессор электротехнического отдела, Национальный автономный университет Мексики. Область научных интересов: автономные мобильные роботы, обработка цифрового сигнала, компьютерная архитектура. Число научных публикаций — 100. robotssavage@gmail.com; Сиркуито Экстериор, Сьдад Университария, Мехико, 04510, Мексика; р.т.: +52(55)56223041.

Контрерас-Толедо Луис Энджел — Ph.D., научный сотрудник передового исследовательского центра искусственного интеллекта и робототехники, Университет Тамагава. Область научных интересов: компьютерное зрение, визуальный метод одновременной локализации и построения карты, робототехника. Число научных публикаций

— 20. tenshi.unam@gmail.com; Тамагава Гакуэн, 6-1-1, Матида, Токио, 194-8610, Япония; р.т.: +5215510323399.

Поддержка исследований. Работа выполнена при финансовой поддержке UNAM-DGAPA-PAPIIT (грант IG110818).

Литература

1. *Amigoni F. et al.* Competitions for benchmarking: task and functionality scoring complete performance assessment // IEEE Robotics&Automation Magazine. 2015. vol. 22(3). pp. 53–61.
2. *Arambula F., Padilla M.A.* Autonomus robot navigation using adaptive potential fields // Mathematical and Computer Modelling. 2011. Vol. 40. pp. 1141–1156.
3. *Arkin R.C.* Behavior-based robotics // MIT Press. 1998. 491 p.
4. *Avilés-Arriaga H.H. et al.* Markovito: A Flexible and General Service Robot // Springer. 2009. pp. 401–423.
5. *Banino A. et al.* Vector-based navigation using grid-like representations in artificial agents // Nature. 2018. vol. 557. no. 7705. pp. 429.
6. *Choset H.M. et al.* Principles of Robot Motion // MIT Press. 2005. 626 p.
7. *Contreras L., Mayol-Cuevas W.* Trajectory-driven point cloud compression techniques for visual slam // IEEE/RSJ International Conference on Robots and Systems (IROS). 2015. pp. 133–140.
8. *Contreras L., Mayol-Cuevas W.* O-poco: Online point cloud compression mapping for visual odometry and slam // IEEE/RSJ International Conference on Robots and Systems (IROS). 2017. pp. 4509–4514.
9. *Dissanayake M.W.M.G. et al.* A solution to the simultaneous localization and map building (SLAM) problem // IEEE Transactions on robotics and automation. 2001. vol. 17(3). pp. 229–241.
10. *Ferrein A., Steinbauer G.* 20 years of robocup // KI-Künstliche Intelligenz. 2016. vol. 30(3-4). pp. 225–232.
11. *Grisetti G., Stachniss C., Burgard W.* Improved techniques for grid mapping with raoblackwellized particle filters // IEEE transactions on Robotics. 2007. vol. 23(1). pp. 34–46.
12. *Iocchi L. et al.* RoboCup@ Home: Analysis and results of evolving competitions for domestic and service robots // Artificial Intelligence. 2015. vol. 229. pp. 258–281.
13. Robots and robotic devices - vocabulary. ISO 8373-201 // International Organization for Standardization. Geneva. Switzerland. 2012.
14. *Jin B., Lei H., Geng W.* Accurate intrinsic calibration of depth camera with cuboids // European Conference on Computer Vision. 2014. pp. 788–803.
15. *Kavraki L., Svestka P., Latombe J-C., Overmars M.H.* Probabilistic roadmaps for path planning in high-dimensional configuration spaces // IEEE Transactions on Robotics and Automation. 1996. vol. 12(4). pp. 566–580.
16. *Khatib O.* Real-time obstacle avoidance for manipulators and mobile robots // The international journal of robotics researc. 1986. vol. 5(1). pp. 90–98.
17. *Latombe J.C.* Robot Motion Plannin // Kluwer Academic. 1991.
18. *Linde Y., Buzo A., Gray R.M.* An algorithm for vector quantizer design // IEEE Transactions on Communications. 1980. vol. 28(1). pp. 84–95.
19. *Lozano-Pérez T., Wesley M.A.* An algorithm for planning collision-free paths among polyhedral obstacles // Communications of the ACM. 1979. vol. 22(10). pp. 560–570.

20. *Lu D.V., Hershberger D., Smart, W.D.* Layered costmaps for context-sensitive navigation // IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2014 . 2014. pp. 709–715.
21. *Müller J.P.* The design of intelligent agents: a layered approach // Springer Science & Business Media. 1996. vol. 1177. 233 p.
22. *Negrete M., Savage J., Cruz J., Marquez J.* Parallel Implementation of Roadmap Construction for Mobile Robots using RGB-D Cameras // Open German-Russian Workshop on Pattern Recognition and Image Understanding (OGRW). 2014. 184 p.
23. *Quigley M., Gerkey B., Smart W.D.* Programming Robots with ROS: a practical introduction to the Robot Operating System // "O'Reilly Media, Inc.". 2015. 448 p.
24. *Savage J. et al.* Construction of roadmaps for mobile robots' navigation using rgb-d cameras // Intelligent Autonomous Systems. Springer. 2016. vol. 13. pp. 217–229.
25. *Savage J. et al.* VIRbot: a system for the operation of mobile robots // RoboCup 2007: Robot Soccer World Cup XI. Springer. 2008. pp. 512–519.
26. *Seib V., Memmesheimer R., Paulus D.* Aros-based system for an autonomous service robot // Robot Operating System (ROS). Springer. 2016. pp. 215–252.
27. *Stückler J., Schwarz M., Behnke S.* Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero // Frontiers in Robotics and AI. 2016. vol. 3. 58 p.
28. *Thrun S., Burgard W., Fox D.* Probabilistic robotics // MIT Press. 2005. 672 p.
29. *Van Beek L. et al.* RoboCup@Home 2018: Rules and Regulations. 2018. 120 p. Available at: http://www.robocupathome.org/rules/2018_rulebook.pdf.2018 (дата обращения: 29.08.2018)
30. *Wachsmuth S., Holz D., Rudinac M., Ruiz-del-Solar J.* RoboCup@Home-benchmarking domestic service robots // Association for the Advancement of Artificial Intelligence (AAAI). 2015. pp. 4328–4329.