DOI: 10.2507/32nd.daaam.proceedings.xxx

FORMAL SPECIFICATION OF A MOBILE ROBOT WITH MODULAR ARCHITECTURE FOR NAVIGATION IN DYNAMIC ENVIRONMENT

Victor Andreev & Valerii Kim

This Publication has to be referred as: Andreev, V[ictor] & Kim, V[alerii] (2021). Formal Specification of a Mobile Robot with Modular Architecture for Navigation in Dynamic Environment, Proceedings of the 32nd DAAAM International Symposium, pp.xxxx-xxxx, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-xx-x, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/32nd.daaam.proceedings.xxx

Abstract

This article discusses the organization of inter-module information interaction in a mobile robot with a hierarchical architecture of the control system. The use of the principle of the full functionality of the modules allows real-time reconfiguration of such mobile robots and the distribution of the computational load between the modules to provide a real-time mode.
The purpose of the work is to present a formal description, mathematical and computer models of the control system of a modular mobile robot, and inter-modular information interaction in the process of solving the general problem of a mobile robot. The two-level hierarchical architecture of the control system is considered, which consists of an intelligent control module, transport and sensor modules. The described robot control system provides navigation in an environment with dynamic obstacles. The functionality of the main modules of the robot and the requirements for their inter-module interfaces have been determined. An analysis of well-known methods for avoiding moving obstacles is given, which made it possible to establish the requirements for the functionality of the transport and sensor modules and their inter-module interfaces.
The article presents the results of a computer simulation of the movement of a modular mobile robot in a dynamic environment. The specificity of the work of a modular system with a hierarchical organization of the control system is shown. A modular mobile robot is proposed to be represented as a network of intelligent agents in a multi-agent environment, the mathematical model of which can be formally described using the theory of finite automata.

Keywords: modular robot; finite-state machine; multi-agent system; collision avoidance; computer networks.

1. Introduction

This article is a continuation of the works [1], [2], [3], aimed at the implementation of a hierarchical modular architecture of the control system (CS) of mobile robots (MR). A review of similar solutions and the rationale for the proposed approach are considered in [3]. The hierarchical modular architecture of CS is based on the principle of distributed computing by analogy with multi-agent systems, which allows distributing the general MR task between modules. This approach leads to a simplification of the algorithms for the operation of the control systems of the

modules, which, in turn, makes it possible to use relatively simple and cheap computing devices in the control systems of the modules - the so-called embedded systems (microprocessors and single-board computers). The use of embedded systems allows reducing the total cost of robotic systems (RTS), energy consumption, and their dimensions. When organizing distributed computing, the general computational control process is parallelized, which makes it possible to provide a real-time mode even with an increase in the RTS functionality.

According to our approach, the control system of a mobile robot consists of 6 main modules, while the functional solution of each module is determined by the principle of full functionality formulated in our work [1]: *each robot module must be capable of performing its target function in any way convenient for it, using only its means to execute commands from an external control system*. In other words, each module must be fully functional. **A fully functional module** is a module that is capable of independently performing the task assigned to it by the supervisor (external control module). To obtain a formal description of such a modular architecture, it is necessary to solve a number of the following problems. First of all, it is necessary to limit ourselves to a specific task that a mobile robot must perform. Then one should define what specific functions each module is responsible for. Based on this, the requirements for inter-modular interfaces and inter-modular interaction protocols are formed.

Thus, the **purpose of the work** is to provide a formal description of the information interaction of modules, a mathematical and computer model of a control system for a mobile robot, consisting of an intelligent control module, transport, and sensor modules. We restrict the general MR task to navigation in an environment with static and dynamic obstacles.

The article is organized as follows: Section 2 provides a formal description of the information interaction of the mobile robot modules in the process of solving the task assigned to the MR. Section 3 provides a brief overview of various algorithms for avoiding moving obstacles. This overview is provided to reasonably distribute the computational load between the transport and sensor modules when avoiding obstacles. Section 4 provides a mathematical model of the transport module and intelligent control module, section 5 - a computer model of the system and the results of the simulation. At the end of the article, conclusions are given, including the direction of the following research.

2. Formal description of information interaction between mobile robot modules

The key feature of the architecture under consideration is the use of the principle of full functionality of modules.

Let a modular robot move on a flat underlying surface in some partially known environment. The robot can be surrounded by static and dynamic (movable) obstacles. The robot receives a general task from an external supervisor (usually a human operator). The general task is to move the MR from the current position to the target position with coordinates X_G, Y_G relative to some world coordinate system. In the problem under consideration, the orientation angle at the target position is not taken into account.

The intelligent control module (ICM), the transport module (TM), and the short-range sensor module (SRSM) in the process of information interaction cooperatively solve the problem of moving the MR with the division of responsibility (Table 1).

Module	Main functions
ICM	Global planning and task formation for other modules
TM	Local planning and movement
SRSM	Collection and processing of information about the environment in the near zone

Table 1. Main functions of the modules

Basic assumptions when solving the problem:

- the TM path does not have to be optimal in terms of the distance traveled and the trajectory;
- the maximum permissible time of arrival (deadline) of the TM at a given position must be set;
- if possible, the TM should avoid all obstacles (including moving ones); in the event of a collision with an obstacle (or with a high probability of collision), the TM must report this to the ICM and wait for further instructions;
- all obstacles are solid bodies with dimensions that allow them to be detected by the sensor system;
- all obstacles fall into the scanning plane of the SRSM range finders;
- the number of objects in the field of view of the SRSM should not exceed some critical value (for example, no more than 10).

The general scheme of information interaction of fully functional modules is shown in fig. 1.

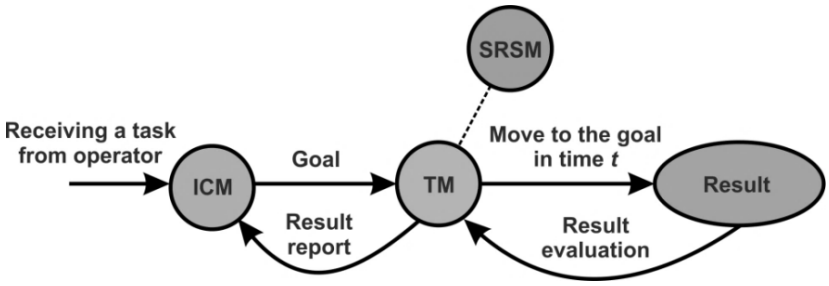


Fig. 1. The general scheme of information interaction of modules

The ICM receives a general task from an external supervisor, which he divides into tasks for each module. Such a task for the TM is to move from an initial position to a target position $[X_i, Y_i]^T$ with some acceptable error in coordinates. In general, this target position is one of the many points of the route planned by the ICM, the final position of which has coordinates X_G, Y_G.

According to the principle of full functionality, the TM must independently reach a given position using the control methods embedded in its control system and information received from its sensory system. It is assumed that the TM sensor system should have a minimum functionality, sufficient only for avoiding static obstacles. For avoiding dynamic and static obstacles with complex shape, the data received by the SRSM should be used.

Consider the information interaction between the ICM and the TM.

The ICM forms a path from the starting position to the target position using data about environment presented in the form of a global map. Usually the map contains information about static obstacles, the data about dynamic obstacles is directly received from the sensor module while moving.

There are various methods of global planning on the map to find the optimal path. These include well-known graph search algorithms: A* [4], D* [5], random sampling algorithms: RRT [6], BIT* [7], etc. The choice of a specific search algorithm depends on the formal description of the map.

As a result of a path planning, the ICM must form a set of sub-goal points of the desired path that will be reached by the TM. In this case, the trajectory of the TM to the next point does not have to be optimal. The interaction between the ICM and the TM can be schematically represented as follows (fig. 2).

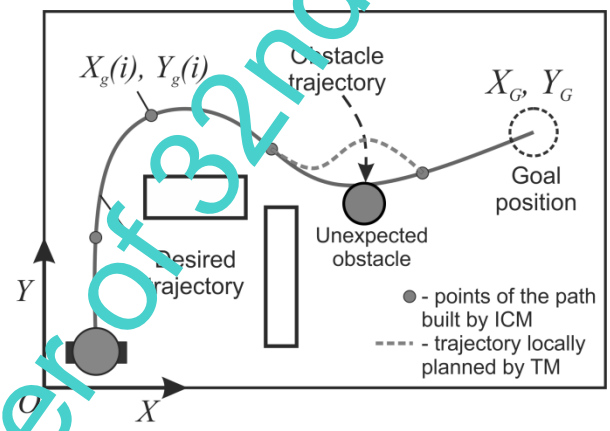


Fig. 2. Scheme of a modular robot movement in an environment with a moving obstacle known to the ICM

Let the intelligent control module construct a path from the initial position to the target one. The way of constructing the path can be any, while the path must be a finite set of points $\{(X_i, Y_i)\}$, $i = 1, 2, ..., n$ on the estimated trajectory. The distances between the points must be at least a certain value (for example, 3 m is a range of the SRSM reliable work), otherwise the interaction between the ICM and the TM will approach continuous, which negates the advantages of the principle of full functionality.

Figure 2 shows that in the event of a moving obstacle, the TM should try to avoid it and get to the next target point using the data of the SRSM and its local path planner. In the general case, the procedure for interaction between the TM and the ICM is described by the following algorithm (fig. 3).

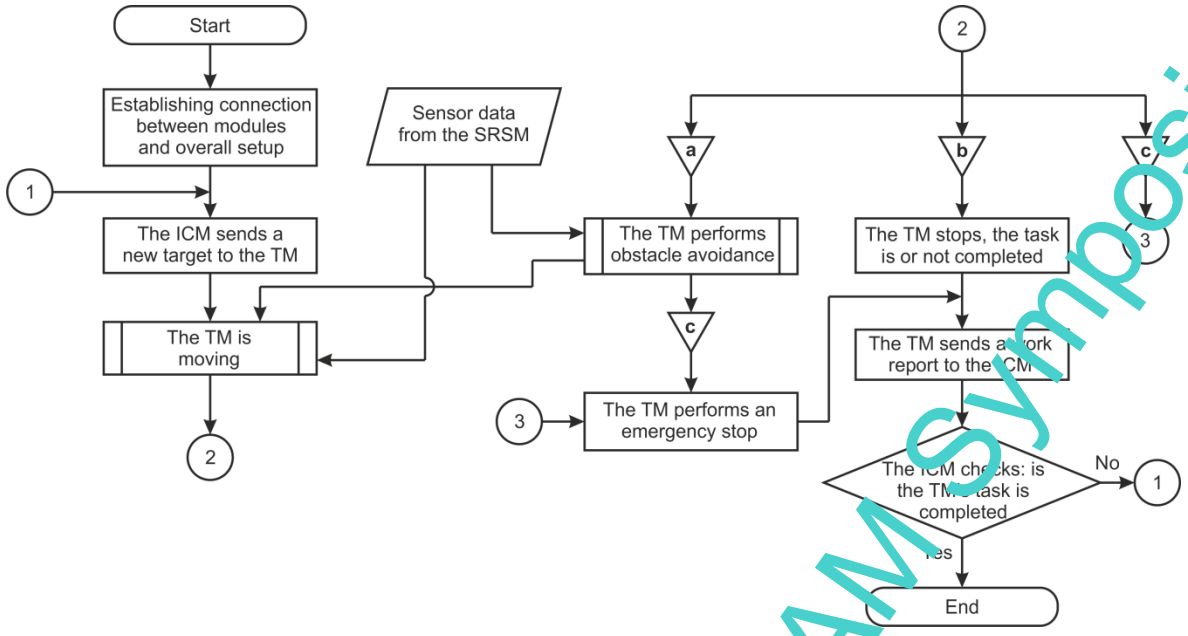


Fig. 3. General algorithm of interaction of ICM, TM and SRSM when the modular mobile robot moves

In this block diagram, the triangles with the symbols a, b, c represent the transfer of control from one process to another:

a – transition to the process of avoiding an obstacle by the transport module in the event of a static or dynamic obstacle on the way. If the obstacle is successfully passed, then there is a return to the process of normal movement to the next point.

b – transition to the stop process, when the TM considers that it has completed the task according to the data of its control system.

c – transition to the emergency stop process, when there was a collision with an obstacle (or a high possibility of a collision) or a failure of the TM internal systems. The block diagram (fig. 3) shows that this interruption can occur when the TM is moving towards the target and the path is clear, or the TM is avoiding an obstacle.

A movement task sent by the ICM to the TM is represented by a data structure with the following parameters:

1. target point coordinates: X_i and Y_i, m;
2. coordinates of the current initial point where the robot is located: X_0 and Y_0, m;
3. admissible error in a target position, m;
4. arrival time limit (deadline), s.

The TM sends a work report to the ICM for any outcome, as can be seen in the block diagram above. The report is presented as a data structure with parameters:

1. target points coordinates: X_i and Y_i, m;
2. elapsed time, s;
3. own assessment of the task (completed or not, emergency stop).

It should be noted that the target and current positions of the robot are set relative to the world coordinate system associated, for example, with the room. At the same time, the ICM knows the location of the fixed base coordinate system, and the coordinate system of the robot itself.

The block diagram (fig. 3) shows that the execution of the movement to the target and the obstacle avoidance are carried out while receiving information from the SRSM. This information should allow the TM to avoid both static and dynamic obstacles.

It should be noted that, according to the principle of full functionality, the division of the computational load between the SRSM and the TM when avoiding obstacles can be implemented in many ways. For example, the SRSM can scan the environment in the near zone, detect obstacles and issue a motion vector for the TM, which in turn must only move in this direction. On the other hand, the SRSM can only be responsible for scanning and periodically send an array of points to the TM for further processing. It is necessary to divide the task of avoiding obstacles so that the computational load between the SRSM and the TM is divided approximately equally. To solve this problem, it is first necessary to analyze various algorithms for avoiding moving obstacles.

3. Algorithms for avoiding moving obstacles and analysis of the interaction of the transport and sensor module

The review includes mainly algorithms designed for a flat environment: [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20] and [21]. In our case, the following characteristics of the algorithms are of interest:

- the need for a global map of the environment,
- the path planner used,
- representation of the local environment,
- the shape and size of the robot and obstacles,
- required information about moving/static obstacles,
- kinematic model of the robot,
- dynamic characteristics of the robot, taken into account in the work.

The analysis of considered algorithms showed that a global map of the environment is rarely used since in a changing environment with dynamic obstacles, local scanning is important for operational decision making..

The path planner is used in those algorithms where it is necessary to construct an optimal path on a global occupancy grid map or a state graph. The A* algorithm and various optimization methods are used as the path planner.

The local environment around the robot can be defined in various ways. For example, in the classical method of velocity obstacles [15], the local environment is represented as the region of attainable robot velocities, built on the basis of its accelerations and "velocity cones" of moving obstacles. In papers [8], [11], the local environment is presented as an occupancy grid map, which is convenient for localizing both static and dynamic obstacles. At the same time, grid maps work well together with rangefinder sensors [22].

In the overwhelming majority of the considered algorithms, the robot is represented as a moving circle, less often as a polygon [13]. Dynamic obstacles are also represented as circles. This is due to the fact that the circles allow the most simple calculation of the nearest distance from the robot to the obstacles, which reduces the requirements for the computing resources of the robot. If a grid map is used, then obstacles are displayed as a set of occupied map cells [8], [11].

The analysis of considered algorithms showed that to avoid moving obstacles, the local planner of the robot must know the coordinates of the obstacles, the vectors of their velocities (projections of the velocities on the axes of the coordinate system) and the radii of the overall circles. This is usually enough to avoid a collision and get to the target position. There are methods that require a set of points on the trajectory of the obstacle [18]. However, there are ways to predict the trajectory of an obstacle based on its current speed, position and kinematic model.

Modern algorithms for avoiding obstacles take into account the kinematic constraints of robots (nonholonomicity), since this can narrow the range of permissible movements of the robot. Also, most of the methods take into account the dynamic characteristics of robots, in particular, acceleration. Taking acceleration and deceleration into account limits the permissible range of motion, but reduces a collision probability.

So, to solve the problem of avoiding obstacles in a partially known environment, the following division of the computational load between TM and SRSM is proposed. SRSM should perform the following main functions:

1. Scanning the environment for obstacle detection and elimination of "false" scan points, i.e. points that are not obstacles (points filtering). The field of view (FOV) of the sensor system must be at least 180° and cover the front of the robot.
2. Solution of the SLAM problem.
3. Tracing moving obstacles. Obstacles should be presented as moving circles.

The SLAM problem should be solved by the SRSM in conjunction with the ICM, which stores a global map of the environment, as well as when using data received from a long-range sensor module (LRSM). The LRSM uses range sensors such as video cameras or lidars.

The TM cannot perform the localization task, since its sensors do not have the required accuracy. Moving obstacles can only be traced by the SRSM, since it is equipped with appropriate sensors and performs the localization of the robot, without which it is impossible to distinguish a moving obstacle from a stationary one, especially if the robot is moving.

TM, in turn, must perform the following functions:

1. Local planning within the near zone of the SRSM.
2. Obstacle avoidance.
3. Control of actuators.
4. Sampling of low-level sensors (limit switches, simple rangefinders).

The near zone of the SRSM can have different sizes and shapes depending on the assigned task, the size of the robot and the size of the terrain. In this work, it is assumed that the near zone is limited by a circle with a radius of 3 m centered at the characteristic point of the robot. The robot itself is circular.

The algorithm for avoiding obstacles can theoretically be any, if the information received from the SRSM is sufficient to execute the algorithm. It should be taken into account that the time to avoid an obstacle is limited from above by the maximum arrival time.

As part of inter-modular interaction, the SRSM sends a data structure with the following parameters to the TM:

1. coordinates of the robot current position with respect to world coordinate system: X_a and Y_a, m;
2. set of the static obstacles points: $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, where $\mathbf{x}_i = (x_i, y_i)^T$ – coordinates of the i-th point in the world coordinate system, n – the number of scanning points in the near field after filtering points in the SRSM and setting a limit on the maximum number of points (see below);
3. moving obstacles dataset: $\{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_m\}$, where $\mathbf{O}_i = (x_{O_i}, y_{O_i}, v_{x_i}, v_{y_i}, r_{O_i})$ – tuple of the i-th obstacle's parameters, m – number of obstacles within the near zone of the robot, x_{O_i}, y_{O_i} – coordinates of the center of the overall circle of the obstacle with a radius r_{O_i} in the world coordinate system, v_{x_i}, v_{y_i} – the projections of the obstacle velocity vector on the axes of the robot coordinate system.

The number of points obtained as a result of scanning obstacles can be large: for example, a laser scanner with an angular resolution of 0.1 ° and a full all-round view can produce up to 3600 points at a frequency of 10-20 Hz. In this case, some of these points will be discarded after filtering. Not all scan points should be sent to TM, but only those n points that are within the near zone (up to 3 m).

Since the requirements for the accuracy of scanning the surrounding space in mobile robotics strongly depend on the specific task, conditions and the robot itself (a mechanical design, software, electronic components), in this work it is proposed to set the limit for the number of scan points sent by the SRSM to the TM. Modern mobile robots use high-resolution laser rangefinders for navigation, which can detect both static and dynamic obstacles (table 2).

Paper	Laser scanner	FOV	Resolution	Maximum number of points
[14]	TIM551	270°	1°	270
[11]	SICK LD-MRS	110°	0.125°	880
[8]	Hokuyo URG-04LX	240°	0.36°	667
[10]	SICK 2D LRF	270°	0.33°	818

Table 2. The main characteristics of typical laser scanners in various mobile robots for the task of motion in a dynamic environment

In all the articles indicated in the table, mobile robots successfully reach the target position by avoiding both static and dynamic obstacles. The number of scan points in these studies rarely exceeds 900. Therefore, in this work, we accept with a small margin that the maximum number of points of static obstacles transmitted to the TM should not exceed 1000 pcs. If the SRSM sensor system is capable of receiving a scan with a large number of points, then the SRSM should select only 1000 of them, excluding the rest.

The frequency with which the SRSM sends data to the TM primarily depends on the frequency of the module's rangefinders and the processing time. For laser rangefinders, the typical scanning frequency is 10-30 Hz [23]. The SRSM should send data as often as its sensors allow.

TM sends information about its position and speed, obtained using odometry, to the SRSM. This is necessary to improve the accuracy of the SLAM algorithms. The sending frequency must be at least the odometer polling rate.

The structure of the sent data has the following parameters:

1. coordinates of the robot current position with respect to world coordinate system: X_a and Y_a, m;
2. the projections of the TM velocity vector on the axes of the robot coordinate system, m/s.

Sending information about the speed to the SRSM can be useful for adjusting the settings of the sensors of the sensor module. For example, if the speed of the TM is below a predetermined threshold, then the frequency of the range finders polling should be reduced to decrease the traffic of inter-module interaction and the power consumption of the SRSM.

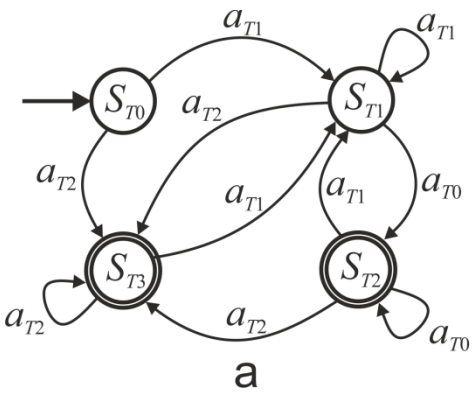
4. Mathematical model of the mobile robot with modular architecture

The mathematical model of a modular mobile robot should take into account the interaction of modules and their functioning as independent intelligent agents. In this paper, it is proposed to use the theory of finite-state automata to formally describe some of the main parts of the robot control system.

The transport module of the robot can be described by the finite state machine of the form [24]:

$A_T = (S_T, \Sigma_T, \delta_T, s_{T0}, F_T)$, where

the finite non-empty set of the TM states: $S_T = \{S_{T0}, S_{T1}, S_{T2}, S_{T3}\}$,
the input alphabet: $\Sigma_T = \{a_{T0}, a_{T1}, a_{T2}\}$,
$s_{T0} = S_{T0}$ - is an initial state,
δ_T - the state-transition function: $\delta_T : S_T \times \Sigma_T \to S_T$,
F_T - the set of final states, $F_T \subseteq S_T$, $F_T = \{S_{T2}, S_{T3}\}$.
The state diagram (a directed graph of the automata A_T) is shown in the fig. 4.



S \ a	a_{T0}	a_{T1}	a_{T2}
S_{T0}	-	S_{T1}	S_{T3}
S_{T1}	S_{T2}	S_{T1}	S_{T3}
S_{T2}	S_{T2}	S_{T1}	S_{T3}
S_{T3}	-	S_{T1}	S_{T3}

b

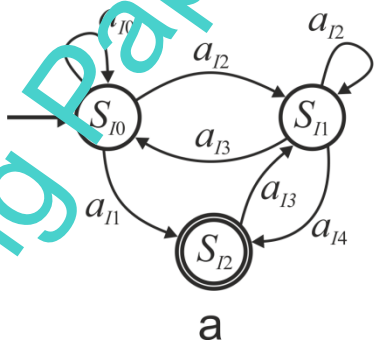
Fig. 4. The state diagram of the TM (a), state transition table (b)

The transport module can be in four main states: S_{T0} – the TM waits for a new target (position), S_{T1} – the TM moves to a new target, S_{T2} – the TM has reached the target, S_{T3} – an emergency stop caused by a collision, expiration of time to move to a target, etc. The initial state of the TM is the expectation of a new goal, therefore $s_{T0} = S_{T0}$. In this case, there are two final states: S_{T2} and S_{T3}, in which the TM is waiting for new instructions from the ICM. For the transport module the S_{T2} state is the same both in the case of arriving at an intermediate point and the final one.

The input alphabet describes the events (symbols) that relate to the work of the TM: a_{T0} – the next goal is achieved, a_{T1} – the new goal is received, a_{T2} – an emergency situation has arisen: a collision with an obstacle (or a high probability of a collision), the time of arrival at a new position has expired.

The state-transition function δ_T is most easily represented in the form of a table (fig. 4, b). Although the function δ_T is not defined for the combination (S_{T0}, a_{T0}) and (S_{T3}, a_{T0}), the presented state machine can be classified as deterministic (DFA), since for each combination of a state and input there is only one unique next state.

Consider the finite-state machine for ICM: $A_I = (S_I, \Sigma_I, \delta_I, s_{I0}, F_I)$. This module is a rather complex device, but for the task under consideration it is possible to reduce its operation to several states (fig. 5, a): $S_I = \{S_{I0}, S_{I1}, S_{I2}\}$, where S_{I0} – the robot moves along the planned path, S_{I1} – creating a new path, S_{I2} – task completed or not completed, stop; the task may not be completed if movement along the original path is impossible and a new one cannot be created. The initial state of the automata is: $s_{I0} = S_{I0}$, i.e. the machine starts working by moving along the planned path. The final state of the automata is: $F_I = S_{I2}$.



S \ a	a_{I0}	a_{I1}	a_{I2}	a_{I3}	a_{I4}
S_{I0}	S_{I0}	S_{I2}	S_{I1}	-	-
S_{I1}	-	-	S_{I1}	S_{I0}	S_{I2}
S_{I2}	-	-	-	S_{I1}	-

b

Fig. 5. The state diagram of the ICM (a), state transition table (b)

The input alphabet $\Sigma_I = \{a_{I0}, a_{I1}, a_{I2}, a_{I3}\}$ consists of the next symbols: a_{I0} – the current target is reached, a_{I1} – the last target (end position) is reached, a_{I2} – the transport module reports an emergency, a_{I3} – continue the task (movement), a_{I4} – impossible to create a new path.

The state-transition function δ_I is described by the table (fig. 5, b). The transition function is also not defined for all combinations of states and inputs, however the state machine A_I can be considered deterministic.

The implementation of the two presented automata in the computer model of the robot is described in the next section.

According to the scheme of information interaction between TM and ICM, the transport module must reach the next target point with a given accuracy in a certain time (state S_{T1} from finite automata). When an unexpected obstacle appears on the robot's path, TM must try to avoid it. In this work, we assume that the TM is a wheeled platform with an omnidirectional drive. Then it is possible to control the module by adjusting the speed vector of the wheeled platform – $\mathbf{v}_r = \left(v_x, v_y\right)^T$.

To solve the problem of moving to the target point, we use the simplest linear control law:

$$\begin{cases} v_x = KL\cos\left(\gamma - \varphi\right), \\ v_y = KL\sin(\gamma - \varphi), \end{cases} \tag{1}$$

where K – controller coefficient, L – distance from the current robot position to the target, γ – the angle between the OX axis of the world coordinate system and the line segment connecting the current position of the robot and the target point.

To eliminate the oscillatory movements of the robot near the target point, the TM should stop when the position error becomes less than a certain predetermined threshold.

In this work, the classic Velocity Obstacle (VO) method [15] was chosen to avoid moving obstacles. This is one of the most well-known algorithms that provides motion in a dynamic environment and has a large number of modifications [9], [13], [19], [21]. We will not give a description of the algorithm here, but note only that to avoid moving obstacles $B_1, B_2, ..., B_m$ the robot's new speed at the current time interval must be chosen outside the union of the VO sets:

$$VO_{A|B_1,B_2,...,B_m} = \bigcup_{i=1}^{m} VO_{A|B_i},$$

where $VO_{A|B_i}$ - velocity obstacle - area of forbidden robot speeds relative to a moving obstacle B_i.

The motion of the transport module is controlled at each time interval Δt, which is set based on the computational capabilities of the module and the frequency of obtaining new sensor data. The magnitude of the speed that the TM can reach through this time interval is determined by the maximum possible acceleration of the module [15], [16]. The robot's speed $\mathbf{v}_r(t + \Delta t)$ on the next interval Δt must satisfy the condition:

$$v_x(t + \Delta t) \in \left[v_x(t) - \dot{v}_x \Delta t, v_x(t) + \dot{v}_x \Delta t\right],$$
$$v_y(t + \Delta t) \in \left[v_y(t) - \dot{v}_y \Delta t, v_y(t) + \dot{v}_y \Delta t\right],$$

where $\dot{v}_x$ and $\dot{v}_y$ – accelerations along axes of the coordinate system fixed to the TM.

This condition can be graphically represented by a rectangular area of reachable velocities (RV). In general this area can intersect with cones $VO_{A|B_i}$. The portion of the reachable zone that does not belong to VO is the set of reachable speeds to avoid moving obstacles. In the article [15] this set is denoted as RAV:

$$RAV(t + \Delta t) = RV(t + \Delta t) \setminus VO_{A|B_1,B_2,...,B_n}(t)$$

Any RAV speed will allow the robot to avoid an obstacle, however the TM must select a speed that is close to the preferred speed $\mathbf{v}_r^{pref}$. This is the speed at which the TM should move towards the target point if there are no obstacles. For example, it can be estimated according to formula (1).

The choice of speed at the next time step is a typical optimization problem, which can be written as follows [25]:

$$\mathbf{v}_r^{new} = \underset{\mathbf{v}_r \in RAV(t+\Delta t)}{\arg\min} \left\| \mathbf{v}_r - \mathbf{v}_r^{pref} \right\| \quad (2)$$

This condition takes into account only dynamic obstacles, however, as noted above, static obstacles can be in the path of the robot. Usually, VO regions are also built for static obstacles, but it is difficult to do this for arbitrary-shaped obstacles. Therefore, in this work, all those speeds that will lead to a collision with a static obstacle at a time $t+\Delta t$ are excluded from the RAV set. Excluding all collision-related velocities from the RAV set, we get the RAV* set. Then finally condition (4) can be rewritten:

$$\mathbf{v}_r^{new} = \underset{\mathbf{v}_r \in RAV^*(t+\Delta t)}{\arg\min} \left\| \mathbf{v}_r - \mathbf{v}_r^{pref} \right\|$$

Thus, the algorithm of motion in a dynamic environment using the VO method is formulated as follows:

1. **Input parameters of the algorithm**: list of moving obstacles $\{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_m\}$, points of static obstacles $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.
2. Calculate the preferred velocity $\mathbf{v}_r^{pref}$.
3. Construct the set of reachable velocities *RV*.
4. For every $\mathbf{O}_i$ from $\{\mathbf{O}_1, \mathbf{O}_2, ..., \mathbf{O}_m\}$:
5. Construct $VO_{A|\mathbf{O}_i}$.
6. Construct *RAV*: exclude $VO_{A|\mathbf{O}_i}$ from *RV*.
7. Construct *RAV** (using known points of obstacles).
8. If set *RAV** is empty, then
9. Stop TM: $\mathbf{v}_r^{new} = (0,0)^T$.
10. The TM informs the ICM that the task has not been completed (failure).
11. else
12. Calculate new velocity $\mathbf{v}_r^{new} \in RAV^*$, which is close to $\mathbf{v}_r^{pref}$

Algorithm 1. Algorithm of TM motion in a dynamic environment

In this algorithm lines 9 and 10 indicate the transition to the S_{T3} state of the TM state machine. If there are no dynamic obstacles, then steps 4-6 are not performed, and the new speed is selected from the subset of RV*, which eliminates collision with static obstacles. In fact, the ICM route built for the robot must be safe (collisions with static obstacles are excluded), so this check is needed for additional safety.

5. Computer model of the mobile robot with modular architecture

To assess the performance of the proposed modular architecture, computer simulation was carried out in the MATLAB (Simulink). Simulation of the movement of a robot consisting of the TM, ICM and SRSM was carried out in a flat environment with moving and static obstacles. The block diagram of the model is shown in fig. 6.

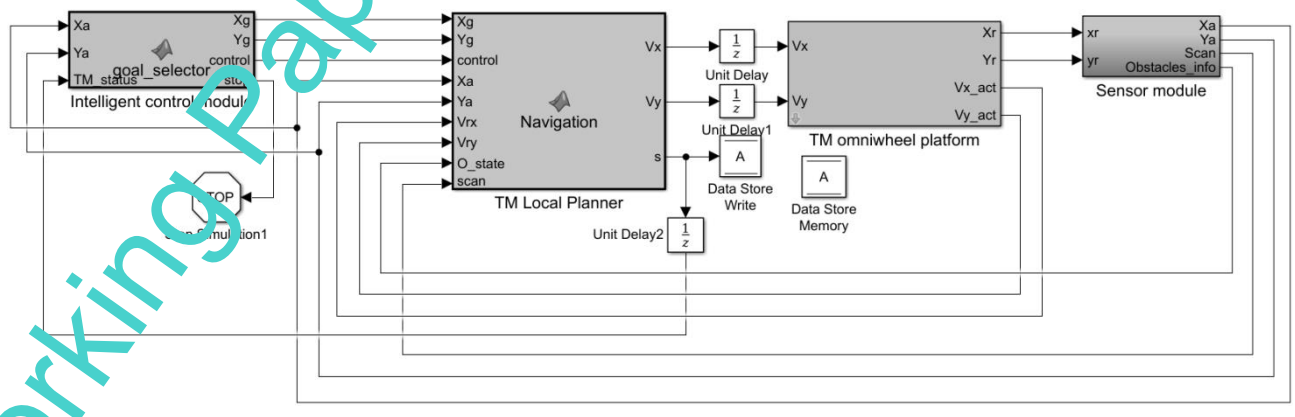


Fig. 6. Computer model of the mobile robot consisting of three main modules

The main blocks of the model:

1. **Intelligent control module** – a block that performs part of the ICM functions: stores data about the path of the robot, compares the position of the robot received from the TM with the given one. The model assumes that the TM receives information about its position using odometry. If the next target position is reached with an acceptable error (no more than 5 cm in this model), the ICM sends the coordinates of the next point to the TM.

The block partially implements the automaton shown in fig. 5, a. In the model, a new path is not built after a transition to a state S_{I1}; instead, an alternative path is used for a known map configuration (this is acceptable since the ICM must store a map of the environment).

2. **TM Local Planner** – block of TM local planning which implements the obstacle avoidance algorithm considered above (Algorithm №1) and the law of motion given by the formula (1). The block accepts the coordinates of the target position as input – Xg, Yg; coordinates of the current robot position – Xa, Ya; the TM speed projections – Vrx, Vrx; obstacles data presented in the form of a matrix with dimensions $m \times 5$, where m is the number of obstacles in a near zone of the TM and the points of the scan of the SRSM. The model assumes that the SRSM consists of a sensor system with 36 idealized rangefinders.

The TM Local Planner block outputs the projection of the linear speed, which ensures the avoidance of obstacles and is the closest to the reference linear speed, which is determined by the formula (1).

The block implements the state machine shown in fig. 4, a. The state of the TM is described by the "s" variable, which is issued to the block output for interaction with the ICM:

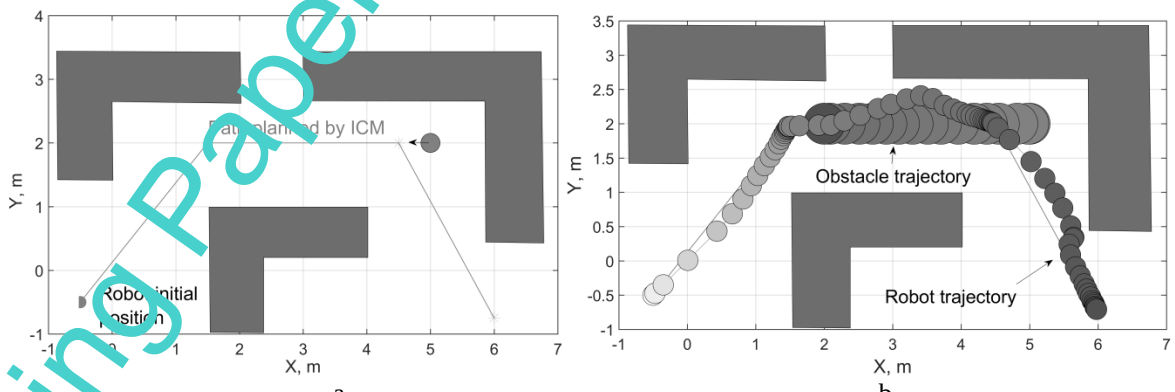
- s = 1 – the TM moves to a next target (the S_{T1} state),
- s = 2 – the TM has reached the target (the S_{T2} state),
- s = 3 – an emergency situation (the S_{T3} state): a collision has occurred, the RAV* set is empty or the time of arrival at the target position has expired.

Here, a collision is understood as a situation when the distance between the robot and the obstacle becomes less than a predetermined threshold. There is no S_{T0} state in the model since at the beginning of the simulation the robot immediately has a point to which it must arrive. When s = 3 the ICM starts issuing points of the new path.

3. **TM omniwheel platform** – a block containing a dynamic model of the TM with an omnidirectional drive. The block receives the projection of the given speed as input and outputs the actual speeds, as well as the coordinates of the platform position.

4. **Sensor module** – the block that scans an area around the robot using ideal rangefinders. The block returns the scan of obstacles obtained by determining the intersection points of virtual beams of rangefinders with polygons that are models of static obstacles. Inside the block there are models of moving obstacles, information about which is grouped into a matrix.

Consider the movement of the robot in the environment in the presence of both dynamic and static obstacles (Fig. 7). At the initial moment, the robot is in the position with coordinates (-0.5, -0.5). Let the ICM planned a path consisting of three points with the following coordinates: (1.5, 2), (4.5, 2) and (6, -0.75). TM must consistently pass these positions. The TM receives the next target point from the ICM after the distance between the actual position of the robot and the target becomes less than 0.05 m. There is one moving obstacle in the environment that moves to the left from the position with coordinates (5, 2), it starts moving 6 seconds after starting the robot and stops at point (2, 2).



a b

Fig. 7. The environment configuration with the planned path (a), trajectories of the robot and moving obstacle (b)

The trajectory of the modular robot and obstacle after the computer simulation is shown in Fig. 7, b. It can be seen from the graphs in the figure that the robot successfully avoided a moving obstacle and avoided collision with walls.

Consider the following situation. The robot is in a room with one rectangular obstacle in the center. In the environment map, this obstacle was taken into account and the ICM planned two identical optimal paths (fig. 8). Let the ICM choose path №1 with the coordinates of the target points (1, 3.5), (4, 3.5) and (5, 2). These points must be visited by the TM in sequence. At the same time, an obstacle moves towards it from the position (4, 3.5), which could not be taken into account at the stage of path planning.

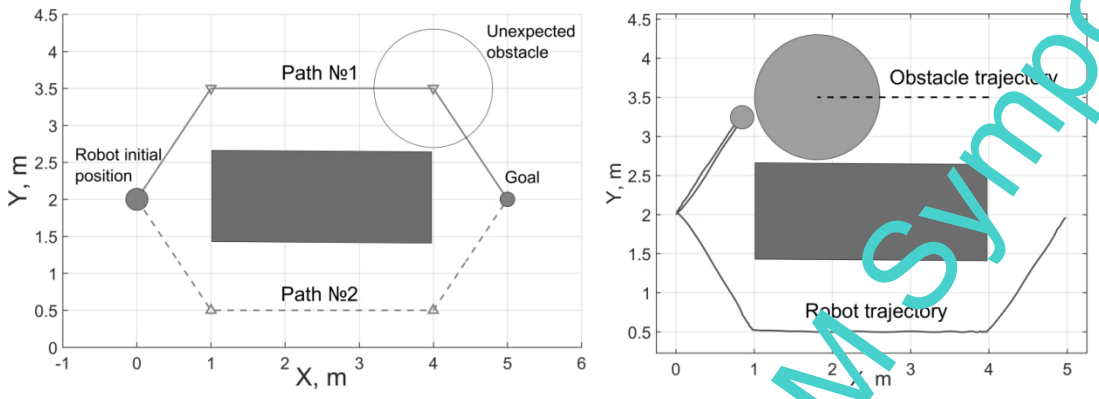


Fig. 8. The model of the environment with one static obstacle and one unknown moving obstacle (a), trajectories of the robot and moving obstacle after simulation (b)

At first the TM moved to the point (1, 3.5) in the normal mode (S_{T1} state), but a large obstacle moving towards it did not allow reaching the first point and a collision with the robot occurred. Due to the collision the TM changes its state to S_{T3} and reports this to the ICM. The ICM also changes its state and chooses the second path, which is similar to the first, and starts sending the coordinates of the target points of the second path to the TM. In this case, the first point to be visited by the TM is the initial position of the robot before starting a simulation. Figure 8 (b) shows the trajectory of the robot, which shows that the robot first tried to get to position (1, 3.5), but after colliding with an obstacle, it returned to its original position and began moving along path №2.

Consider a similar situation: there are two corridors on a map and the ICM builds two identical paths to the goal (Fig. 9, a). At the same time, one of the passages is blocked by obstacles that the ICM does not know about. This can happen if these obstacles appeared after the map was built. Let the ICM choose the path №1 and the TM begins to consistently visit the points of the path.

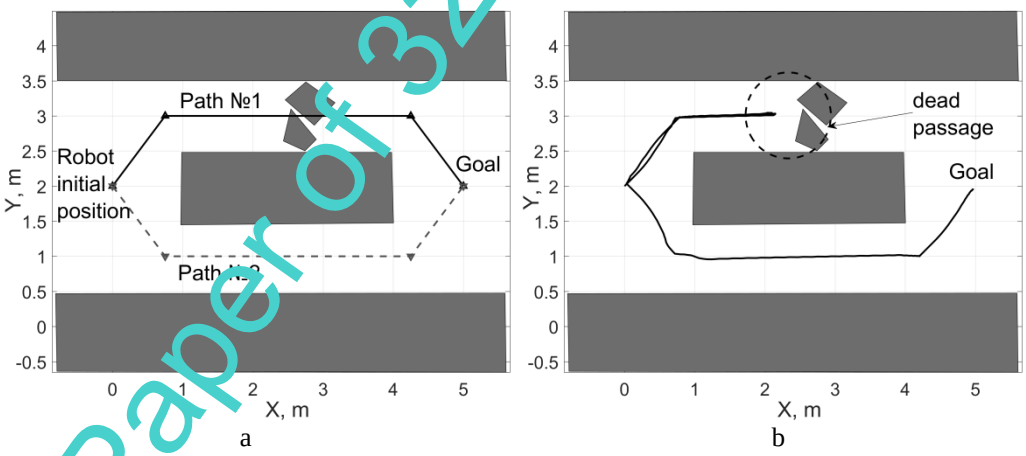


Fig. 9. The model of the environment with two corridors, one of which is blocked (a), the trajectory of the robot after simulation (b)

Figure 9 (b) shows the trajectory of the robot after the simulation. The TM visited only one point of the first path. Further in the corridor, obstacles did not allow TM to pass to the next target, and he began to make small movements in the dead passage and almost didn't move trying to go forward. Before the start of the simulation, a deadline was set to move to the next point (15 sec). When the time to reach the next point exceeded the deadline, the TM switched to the state S_{T3} (s. 3). This switch was captured by the ICM, the state of which changed from S_{I0} to S_{I1}. The ICM switched to performing a movement along path №2, and began sending points of this path to TM. After that, the TM has successfully reached the final position with coordinates (5, 2).

The results of the computer simulation showed the efficiency of the mathematical model of the modular robot. The use of the apparatus of the theory of finite automata makes it possible to describe the robot modules as independent agents. In this case, the state machine model can be easily implemented in the software of a particular module.

6. Conclusions

This article presents a developed formal description of the information interaction of the ICM, TM and SRSM modules, as well as a mathematical and computer model of a mobile robot consisting of these modules.

The analysis of information inter-modular interaction was carried out based on the principle of full functionality for the problem of navigation in a flat environment. The division of functions between the modules has been substantiated and a general algorithm for the interaction of the ICM, TM and SRSM has been developed. The data structures that modules must send to each other are defined, which is necessary for drafting a specification for inter-modular interfaces of modules. To formalize the information interaction between TM and SRSM, it was necessary to conduct an analytical review of the known algorithms for avoiding moving obstacles. The analysis of known works showed that many algorithms use the same information about moving obstacles, which made it possible to formulate requirements for the TM-SRSM interface.

It is proposed to consider the control systems of the TM and ICM as finite automata. This approach allows studying modules as hardware and software agents in a multi-agent environment, changing their state and behavior depending on external and internal conditions. To avoid moving obstacles, it was decided to use the VO method in the TM control system.

Analysis of the results of computer modeling allows us to conclude that the proposed mathematical model of the control system reflects the specifics of the modular hierarchical architecture. On the one hand, the modules work independently: the TM avoids obstacles; the ICM chooses the path on the map. On the other hand, the state of one module affects the operation of another, which was shown in the results of computer simulation.

7. Future work

In the course of further research, it is necessary to expand the mathematical model of a mobile robot with a modular architecture, namely: to investigate the behavior of the system under various internal and external conditions, to complicate the TM model, taking into account the sub-modules that are part of it, i.e. introduce the 3rd level of the hierarchy. This will increase the performance of the TM control system by further parallelizing the computational process. A big task is to add elements to the robot's computer model that are responsible for simulating the operation of communication channels to assess inter-module traffic in various situations.

8. Acknowledgments

The Russian Foundation for Basic Research supports this research: Grant 19-07-00892a.

9. References

[1] Andreev, V.; Kim, V. & Pletenev, P. (2017). The Principle of Full Functionality – the Basis for Rapid Reconfiguration in Heterogeneous Modular Mobile Robots, Proceedings of the 28th DAAAM International Symposium, 08-11th November 2017, Zadar, Croatia, ISSN 1726-9679, ISBN 978-3-902734-11-2, B. Katalinic (Ed.), pp.0023-0028, Published by DAAAM International, Vienna, Austria, DOI: 10.2507/28th.daaam.proceedings.003.

[2] Andreev V. (2019). Control System Mobile Robots with Modular Architecture as a Multi-Agent System with a Hierarchical Topology, Proceedings of the 30th DAAAM International Symposium, 23-26th October 2019, Zadar, Croatia, ISSN 1726-9679, ISBN 978-3-902734-22-8, B. Katalinic (Ed.), pp.0010-0019, Published by DAAAM International, Vienna, Austria, DOI: 10.2507/30th.daaam.proceedings.002.

[3] Andreev, V.; Kim, V. & Eprikov S. (2020). ModRob: the hardware-software framework for modular mobile robots prototyping, Proceedings of the 31st DAAAM International Symposium, 21-24th October 2020, Hosted in Mostar, BiH, ISSN 1726-9679, ISBN 978-3-902734-29-7, B. Katalinic (Ed.), pp. 0391–0402, Published by DAAAM International, Vienna, Austria, DOI: 10.2507/31st.daaam.proceedings.054.

[4] Hart, P. E.; Nilsson, N. J. & Raphael, B. (1968), "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics, Vol. 4, No. 2, 1968, pp. 100-107.

[5] Stentz, A. (1997). Optimal and efficient path planning for partially known environments, In: Intelligent Unmanned Ground Vehicles, Martial H., Charles E., Anthony Stentz, (Ed.), pp. 203–220, Springer US, ISBN 978-0-7923-9833-2, Germany.

[6] LaValle, S.M. (1998). Rapidly-Exploring Random Trees: A new tool for path planning, Available from: http://lavalle.pl/papers/Lav98c.pdf Accessed: 2020-10-09.

[7] Gammell, J. D.; Srinivasa S. S. & Barfoot, T. D. (2014). Batch informed trees (bit*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs, arXiv preprint arXiv, vol. 1405, pp. 5848.

[8] Gerasimov, V.N. (2015). Sistema navigatsii servisnogo robota v srede s dinamicheskimi prepyatstviyami [service robot navigation system in an environment with dynamic obstacles], Ph.D. Dissertation, Bauman Moscow State Technical University, Moscow, Russian Federation.

[9] Berg, J. van den; Lin M., & Manocha, D. (2008). Reciprocal Velocity Obstacles for real-time multi-agent navigation, in 2008 IEEE International Conference on Robotics and Automation, 19-23 May 2008, Pasadena, CA, USA, Print ISSN: 1070-9932, pp. 1928–1935, IEEE, DOI: 10.1109/ROBOT.2008.4543489.
[10] Chen, Y. & Liu M. (2018). RRT* Combined with GVO for Real-time Nonholonomic Robot Navigation in Dynamic Environment, Available from: http://arxiv.org/abs/1710.07102 Accessed: 2021-10-01.
[11] Cherubini, A.; Spindler, F. & Chaumette, F. (2014). Autonomous visual navigation and laser-based moving obstacle avoidance, IEEE Transactions on Intelligent Transportation Systems, Vol. 15, No. 5, 2014, pp. 2101–2110, Print ISSN: 1524-9050, DOI: 10.1109/TITS.2014.2308977.
[12] Guy S.J. et al. (2009). ClearPath: highly parallel collision avoidance for multi-agent simulation, in Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 1-2 August 2009, New Orleans, Louisiana, USA, ISBN 978-1-60558-610-6, pp. 177-187, Association for Computing Machinery, New York, USA, DOI: 10.1145/1599470.1599494.
[13] Claes, D.; Hennes, D.; Tuyls, K. & Meeussen, W. (2012). Collision avoidance under bounded localization uncertainty, in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 7-12 Oct. 2012, Vilamoura-Algarve, Portugal, Print ISBN:978-1-4673-1737-5, Print ISSN: 2153-0858, pp. 1192–1198, IEEE, DOI: 10.1109/IROS.2012.6386125.
[14] Chen, Y. F.; Liu, M.; Everett, M. & How, J. P. (2017). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning, IEEE International Conference on Robotics and Automation (ICRA), 29 May – 3 June 2017, Singapore, Electronic ISBN: 978-1-5090-4633-1, pp. 285–292, IEEE, DOI: 10.1109/ICRA.2017.7989037.
[15] Fiorini, P. & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles, The International Journal of Robotics Research, Vol. 17, No. 7, 1998, pp. 760–772, ISSN 0278-3649, DOI: 10.1177/027836499801700706.
[16] Fox, D.; Burgard, W. & Thrun, S. (1997). The dynamic window approach to collision avoidance, IEEE Robotics & Automation Magazine, Vol. 4, No. 1, 1997, pp. 23–33, ISSN 10709932, DOI: 10.1109/100.580977.
[17] Shahriari, M.; Svogor, I.; St-Onge, D. & Beltrame G. (2018). Lightweight collision avoidance for resource-constrained robots, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1-5 Oct. 2018, Madrid, Spain, Electronic ISBN: 978-1-5386-8094-0, Electronic ISSN: 2153-0866, pp. 1–9, IEEE, DOI: 10.1109/IROS.2018.8593841.
[18] Narayanan, V.; Phillips, M. and Likhachev, M. (2012). Anytime Safe Interval Path Planning for dynamic environments, in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2012, Vilamoura-Algarve, Portugal, Print ISBN: 978-1-4673-1737-5, Print ISSN: 2153-0858, pp. 4708–4715, IEEE, DOI: 10.1109/IROS.2012.6386191.
[19] Alonso-Mora, J.; Breitenmoser, A.; Rufli, M.; Beardsley, P. and Siegwart, R. (2013). Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots, Distributed Autonomous Robotic Systems, Vol. 83, 2013, pp. 203–216, 2013, Print ISBN 978-3-642-32722-3, DOI: 10.1007/978-3-642-32723-0_15.
[20] Savkin, A. & Wang, C. (2013). A simple biologically inspired algorithm for collision-free navigation of a unicycle-like robot in dynamic environments with moving obstacles, Robotica, Vol. 31, No. 6, 2013, pp. 993–1001, ISSN 0263-5747, DOI: 10.1017/S0263574713000313.
[21] Snape, J.; Berg, J. van den; Guy, S.J. & Manocha, D. (2011). The Hybrid Reciprocal Velocity Obstacle, IEEE Transactions on Robotics, Vol. 27, No. 4, 2011, pp. 696–706, ISSN 1552-3098, DOI: 10.1109/TRO.2011.2120810.
[22] Siegwart, R.; Nourbakhsh, I. and Scaramuzza, D. (2011). Introduction to Autonomous Mobile Robots, Second edition. The MIT Press, ISBN-13: 978-0262015356.
[23] Wang, D.; Watkins, C. and Xie, H. (2020). MEMS Mirrors for LiDAR: A Review, Micromachines, Vol. 11, No. 5, p. 456-480, ISSN 2072-666X, DOI: 10.3390/mi11050456.
[24] Balogh, R. & Obdržálek, D. (2019). Using Finite State Machines in Introductory Robotics, in Robotics in Education, vol. 829, W. Lepuschitz, M. Merdan, G. Koppensteiner, R. Balogh, and D. Obdržálek, Eds., pp. 85–91, Springer International Publishing, DOI: 10.1007/978-3-319-97085-1_9.
[25] Snape, J.; Berg, J van den; Guy, S. J. and Manocha, D. (2009). Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles, in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 10-15 Oct. 2009, St. Louis, MO, USA, pp. 5917–5922, DOI: 10.1109/IROS.2009.5354821.