FINITE-STATE MACHINE CONTROL AND POTENTIAL FIELD METHOD FOR MOBILE ROBOT NAVIGATION IN AN ENVIRONMENT WITH OBSTACLES

V. E. Karpov

National Research Center Kurchatov Institute, Moscow, Russia

✉ karpov-ve@yandex.ru

Abstract. This paper considers a local navigation method for a mobile robot operating in extreme conditions, i.e., the absence of global navigation and centralized control means, as well as limited-capability sensors. Local navigation is based on the potential field method, and the motion programs developed with this method are transferred to the level of elementary actions. Complex behavior is formed by meta finite-state machines (FSMs) implementing a control hierarchy. The robot's basic task is to move to a given area or region of space. Within this task, an elementary procedure—obstacle avoidance—is considered and implemented based on the potential field method as well. The results of experiments are provided for three different levels, namely, computations, simulations (the physical level), and real robots, with the same computational model and FSMs used in all experiments. As shown, the successful solution of the local navigation task is determined by the size of the robot's field of view, but the latter can be naturally expanded due to the universality of control mechanisms.

Keywords: potential field method, finite-state machine control, meta finite-state machine, movement models, mobile robot, local navigation, obstacle avoidance, control system hierarchy.

INTRODUCTION

In the development of group robotic systems, an applied field is the concept of deploying large groups of ground-based mobile robots in extreme conditions with losses. This concept involves the use of numerous relatively simple autonomous mobile platforms capable of operating under severe constraints. These constraints include:

– no communication with the operator,
– autonomy and decentralized control,
– no global navigation means,
– no map of the terrain,
– and limited sensor capabilities (no active detection means, such as powerful lidars).

Note that the goals of operation are determined by the group.

Among the tasks to be performed by a group of such robots, the basic one is to deliver a certain payload to a specified area or region of space. The robots are given a target bearing and an approximate distance from the starting point. The robots are equipped with cameras, rangefinders, or sensors to detect obstacles or impassable segments, as well as with local communication devices to determine the status of neighboring robots in the group. In the absence of global navigation and a map of the terrain, the robots can rely on an analysis of their local environment. By assumption, the robots move as a compact group, avoiding collisions and bypassing obstacles while maintaining the specified direction. The group-based nature of such systems is due to, first, the required low sensitivity to losses and, second, the expected synergistic effects arising from group movement (interaction).

In this paper, we consider the basic procedure for solving the delivery task, i.e., the robot's bearing-based motion with obstacle avoidance.

1. BEARING-BASED MOTION

Among the multitude of path planning methods for environments with obstacles (e.g., see the review [1]), local navigation models based on potential fields (PFs) hold a special place. Importantly, these methods not only construct individual paths but also implement rules for the coordinated motion of a group of agents. The general idea behind the PF method is to move along the lines of a vector field whose potential func-

tion reflects the configuration of observed objects, obstacles, and their shapes, as well as the goal of motion. Formally, the PF method belongs to a wider family (the so-called potential guidance methods) and, despite a history spanning for over 50 years, it is still actively investigated; for example, see the works of A.B. and N.B. Filimonovs [2, 3], or the review by H. Xing [4]. This is primarily due to the obvious and insurmountable limitations of the PF method: a tendency to get stuck in local minima, oscillations, and path planning challenges when avoiding complex-shaped obstacles [5]. Moreover, it is difficult to apply the PF method when the robot differs from a convenient system with a differential drive [6]. Overall, the general evolution trend of PF-based control methods consists in the development of hybrid control systems, where the PF mechanism is used to construct paths at the tactical level.

Motion in an environment with obstacles is typically categorized as a complex task requiring the preliminary analysis of the environment. To this end, special path planning algorithms with obstacle avoidance are being developed, utilizing both "conventional" production mechanisms (e.g., see the paper [7]) and rather computationally intensive graph-based search methods (see the publication [8] or O'Rourke's classical work [9]). The so-called Bug algorithms stand somewhat apart: a robot follows the contour of an obstacle, permanently estimating the deviation between its current orientation and the required bearing [1]. This approach has a significant limitation: as a matter of fact, the robot moves to the next path point located behind the obstacle, performing an excessive bypass. But the greatest difficulty is that the contour-following procedure requires advanced sensors and the capability to reconstruct this contour.

Nevertheless, in this paper, the PF method is used for local path planning, and obstacle avoidance issues are resolved by introducing an additional control level of finite-state machines (FSMs). The PF method is of interest due to its computational simplicity and generality, which are crucial for the specifics of the delivery task.

This paper does not address the application of the PF method to group tasks. The implementation of various swarm motion laws was described in [10]. Here, we will focus on solving a particular problem, i.e., obstacle avoidance.

First, let us describe a realization of the PF method for a mobile robot.

1.1. Potential Field Method

The agent's model. An agent is an object equipped with sensors and capable of moving, generally speaking, in three-dimensional space. Agent A is defined by a five-tuple of the form

$$Agent = \langle name,\ C,\ R,\ S,\ F \rangle,$$

where *name* specifies its name; C is the set of the agent's characteristics; R is its spatial position; S denotes the state of the sensory system; finally, F is a control function (law).

The control law F determines the agent's spatial position:

$$F(C, S, R) \to R.$$

For the problem under consideration, the basic characteristics of the agent are a triple of the form

$$C = \langle t,\ r,\ \omega \rangle,$$

where t denotes the agent's type, $t \in \mathbb{N}$; r is the agent's size (its geometric characteristics, $r \in \mathbb{R}$); and ω specifies the agent's weight $(\omega \in \mathbb{R})$. The types, weights, and sizes of agents are introduced for implementing group control functions, when an agent needs to orient itself relative to the other group members, based on the observed characteristics of its neighbors.

The spatial position R is specified by a triple of the form $R = \langle X,\ v,\ A \rangle$ with the following designations: X is the agent's coordinates, $X \in \mathbb{R}^3$; v is its velocity $(v \in \mathbb{R})$; and $A = \langle \alpha_{\text{yaw}},\ \alpha_{\text{pitch}},\ \alpha_{\text{roll}} \rangle$ is the triple of Euler angles, where α_{yaw}, α_{pitch}, and α_{roll} are the yaw, pitch, and roll angles, respectively.

The agent's sensory system provides information about the state, coordinates, velocities, and orientation angles of a set of objects Ω_s inside its field of view, i.e., a certain region of space $\mathbb{R}^3_s$ near the agent, e.g., its forward hemisphere. Each robot may be equipped with various sensors, and their readings complement the set Ω_s (Fig. 1).

This set is defined as $\Omega_s = \{o_i^k\}$, where o_i^k denotes the objects detected by the robot's sensors (neighbors, obstacles, target landmarks, etc.). More precisely,

$$o_i^k = \langle (x_1, x_2, x_3), k \rangle,$$

where $(x_1,\ x_2,\ x_3)$ are the object's coordinates, and k indicates its class (neighbors, obstacles, target objects, etc.). Each element of o_i^k determines its contribution (sign and magnitude) to the resulting motion vector. The most important characteristic of the region Ω_s is its radius r_Ω, i.e., the distance at which the agent can see the object. The radius r_Ω can be determined both

by the agent's sensory capabilities and by external-source information (the agent's prior knowledge of the environment). If the radius r_Ω is commensurate with the size of the agent's operation region, then we obtain a situation of complete knowledge of the environment, and local path planning will turn into global navigation.

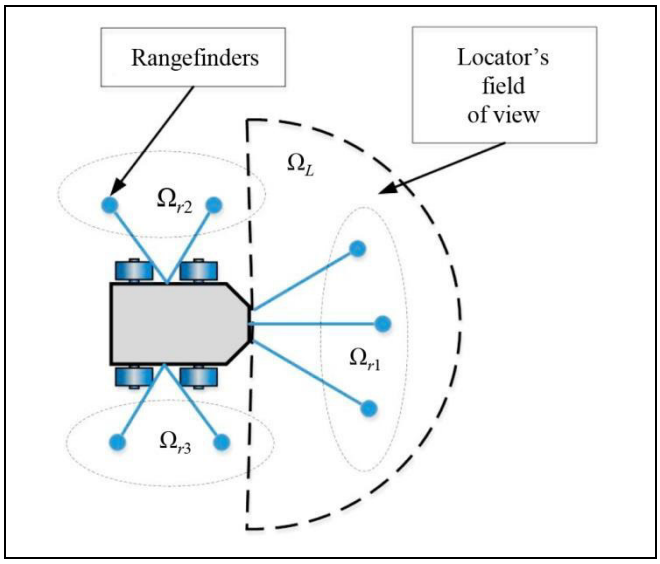


Fig. 1. The agent's field of view Ω_s, formed by the sets of objects detected by the locator Ω_L and the rangefinders Ω_{ri}.

The agent's motion. The velocity and direction of the agent's motion are determined by the sum of the field potentials at its position. In other words, each object in the field of view Ω_s is treated as a source of potential (positive or negative), and the sum of these potentials determines the control law as a pair

$$F_{\text{ctl}} = \langle v_{\text{ctl}}, A_{\text{ctl}} \rangle, \quad A_{\text{ctl}} = \langle \alpha_{\text{yaw}}, \alpha_{\text{pitch}}, \alpha_{\text{roll}} \rangle,$$

where v_{ctl} and A_{ctl} are the velocity and direction of motion, respectively.

Let $\langle X_s, v_s, A_s \rangle$ denote the average values of the coordinates, velocity, and orientations of the objects observed in the region Ω_s. For the observed objects, the components of the coordinates of their center of mass are calculated as

$$X_s^k = \frac{\sum\limits_{o_i \in \Omega_s} x_k^i}{\sum\limits_{o_i \in \Omega_s} \omega_i}, \; k = 1, 2, 3, \qquad (1)$$

where x_k^i mean the components of the coordinate vector of the observed object, and ω_i is its weight. The average velocities v_s and the orientation vector A_s are determined by analogy.

These characteristics of the region Ω_s can be used to define flexible motion laws: not only to move in the direction determined by the center of mass X_s, but also to coordinate the direction of motion and velocity with other agents, to pursue a target, to avoid obstacles, etc.

Thus, the input consists of the agent's state $\langle X, v, A \rangle$ and the observed variables $\langle X_s, v_s, A_s \rangle$; the interaction law is defined by the function $F_A(X, X_s, A_s)$; and the desired output is the control vector $\langle v_{\text{ctl}}, A_{\text{ctl}} \rangle$. The function F_A determines the components of the control vector $\langle v_{\text{ctl}}, A_{\text{ctl}} \rangle$ as follows:

$$v_{\text{ctl}} = |F_A(X, X_s, A_s)|, \; A_{\text{ctl}} = \arg\left(F_A(X, X_s, A_s)\right).$$

For example, consider the cohesion law F_{coh}, when the agent should stay close to its neighbors; then the resulting velocity can generally be determined by the agent's velocity and the distance r_s to the group's center of mass, determined by the coordinates X and X_s:

$$v_{\text{ctl}} = |F_{\text{coh}}| = \min\left(v_{\max}, \frac{k_v}{r_s}\right),$$

where $v_{\max}$ is the maximum velocity, and k_v is a certain coefficient. And the direction of motion can be determined from the position of the center of mass of the observed neighbors.

To avoid additional operators and transformations in the considerations below, we will assume that the vector F is naturally represented as an explicit <length, orientation> pair: $F = \langle |F|, \arg(F) \rangle = \langle v, a \rangle$.

In the case of the agent's simple bearing-based motion with the given angles α_{yaw} and α_{pitch}, we have

$$F_{\text{bear}} = \langle v_{\text{ctl}}, A_{\text{ctl}} \rangle = \langle v_{\text{ctl}} = \text{const}, \langle \alpha_{\text{yaw}}, \alpha_{\text{pitch}}, 0 \rangle \rangle. \qquad (2)$$

The general control law of the agent's motion. Let the region Ω_s allow separating the classes of objects Ω_s^i with different laws of motion F_i; some laws depend on the field of view (approach, repulsion), while the others are independent of it (free motion, bearing-based motion, etc.). The control law F_{ctl} will generally be defined by the total effect of all forces acting on the agent via the two components $\langle v_{\text{ctl}}, A_{\text{ctl}} \rangle$:

$$F_{\text{ctl}} = \langle v_{\text{ctl}}, A_{\text{ctl}} \rangle = \sum_{i=1}^{n} k_i F_i(\Omega_s^i), \qquad (3)$$

where $F_i\left(\Omega_s^i\right)$ is the sum of all force vectors determined by the position of the objects from the region Ω_s^i relative to the agent; k_i are coefficients whose values determine the agent's behavior (how it responds to objects of class *i*). Let *K* denote the set of these coefficients. By tuning the coefficients, one can implement, e.g., the classical Reynolds model of group motion [11]. According to the motion rules, an agent must keep a given distance from other agents and move in the same direction as they do. In addition, it is required to follow a specified direction of motion (a target bearing) and to avoid approaching obstacles and/or hazardous regions. Formally, this means that the agent's resulting direction of motion is defined by the sum of repulsive and attractive forces F_{res} of the form

$$\begin{aligned} F_{\text{res}} &= k_{\text{tar}}F_{\text{tar}} + k_{\text{sep}}F_{\text{sep}} \\ &+ k_{\text{align}}F_{\text{align}} + k_{\text{coh}}F_{\text{coh}} + k_{\text{obst}}F_{\text{obst}}, \end{aligned} \quad (4)$$

where F_{tar} is the vector defining bearing-based motion with a target bearing; F_{sep} is the repulsive force from neighbors (separation for collision avoidance); F_{align} is the course alignment force (maintaining the average course of the neighbors); F_{coh} is the attractive force toward the neighbors (orientation toward their average position for cohesion); finally, F_{obst} is the repulsive force from obstacles. The coefficients *k*, with appropriate subscripts, specify the significance of the contribution of the corresponding forces.

The law (4) can be naturally extended by adding new factors that determine the agent's movement trend. As a result, one arrives at the general control scheme (3).

1.2. Motion Control

1.2.1. Meta FSMs

The agent's motion procedure in a potential field can be treated as a certain action. More complex behavior (a sequence of actions) requires different implementation mechanisms. One of such mechanisms is finite-state machine control, e.g., the use of Mealy machines. A Mealy machine has quite limited capabilities, and describing complex behavior requires constructing complex FSMs, which will negate simplicity and clarity as important properties of machine-based control. At the same time, there exists a methodology of meta-machine control where elementary actions, their sequences, and complexes of sequences are represented as hierarchies of corresponding FSMs.

A meta-machine is a Mealy machine in which the arc labels determine the procedures executed by the agent. If a machine activates low-level motor functions, then such a machine has rank 0 (and is denoted by M^0). A meta-machine M^1 (of rank 1) is intended to control machines M^0. By analogy, a meta-machine M^n (of rank *n*) controls machines M^{n-1}. Thus, the control hierarchy is structured in the machine representation; for details, see the paper [10]. Note that, first, the class of algorithms realized by meta-machines is still limited to FSM representations; second, the hierarchy of meta-machines is merely a control decomposition method, in contrast to, e.g., the nested machine paradigm. (For instance, we mention A. Shalyto's machine-based programming school [12].)

Thus, the two-level control system is organized as follows:

1. The basic level. Reactive direct control models are used to determine the motion vector based on an analysis of the region Ω_s. At this level, motion programs such as approach, obstacle avoidance, pursuit, etc., are executed. These action-programs are realized using the potential field method. From the standpoint of the generalized control model (3), this means setting the coefficients from the set *K*.

2. The behavioral level. It is implemented based on discrete-event (FSM) models. At this level, meta-machines form sequences of motion programs, realizing complex behavioral procedures such as bearing-based motion with obstacle avoidance, motion along a given route (patrolling), and so on.

Two motion procedures are required to move to a desired region of space with a target bearing, namely, a free-motion program implementing the generalized control (3) and a program for moving around an obstacle.

1.2.2. Free-motion program

This program is executed when the robot sees no obstacles on its path. The robot's motion is defined by a set of the coefficients *K*, e.g.,

$$K = \left\{k_{\text{bear}},\ k_{\text{obj}},\ k_{\text{coh}}\right\},$$

where k_{bear}, k_{obj}, and k_{coh} are the parameter values determining the significance of the forces of bearing-based motion, motion toward an object of interest, and cohesion, respectively.

By assumption, in the absence of obstacles, the robot moves with a given bearing; if an object of interest (the finish point) is detected, the robot moves toward it while maintaining the direction of the group's motion (the cohesion rule: "I will go where everyone goes").

Motion is performed as follows:

• **Bearing-based motion.** It is defined by the given angles α_{yaw} and α_{pitch} according to formula (2):

$$F_{bear} = \langle v_{ctl}, A_{ctl} \rangle = \langle v_{ctl} = \text{const}, \langle \alpha_{yaw}, \alpha_{pitch}, 0 \rangle \rangle.$$

• **Motion toward the object of interest.** Let the robot detect a set of target objects Ω_s^{obj}. Based on their coordinates, the agent's direction is defined as motion toward their center of mass, see formula (1):

$$F_{obj} = \langle v_{ctl}, A_{ctl} \rangle = \langle v_{ctl} = \text{const}, A_{obj} \rangle,$$

where A_{obj} is the total direction vector toward the center of mass of the observed objects.

• **Cohesion (attraction).** Let the robot detect the set of its neighbors Ω_s^{coh}. If the coordinates of the group members—the elements of the set Ω_s^{coh}—are known, the agent's velocity and direction will be defined as motion toward their center of mass.

The resulting direction of motion is defined as

$$F_{ctl} = \langle v_{ctl}, A_{ctl} \rangle = k_{bear} F_{bear} + k_{obj} F_{obj} + k_{coh} F_{coh}.$$

The specifics of the current task are determined by assigning weights to the corresponding motion factors. For example, if $k_{coh} \ll k_{bear} \ll k_{obj}$, the main priority will be target search; in the case $k_{coh} \ll k_{obj} \ll k_{bear}$, bearing-based motion.

1.2.3. Motion around obstacles

The motion program allowing the robot to bypass an obstacle is more complex. In fact, this is a procedure for moving around observable landmark objects (obstacles or their fragments). Consider the basic procedure for the agent's motion around landmarks using the potential field method. This procedure has many parameters determining the motion nature and returns a control law in the form of a pair $\langle v_{res}, A_{res} \rangle$, i.e., the velocity and direction of motion. The procedure is reentrant and is called at every simulation step.

The input parameters are:

• the agent's coordinates $c_a = (x_a, y_a, z_a)$ and orientation angles $\alpha_a = \alpha_{yaw}^a, \alpha_{pitch}^a, \alpha_{roll}^a$;

• the set of observed objects $\Omega = \{s_i\}$; for each element, the coordinates (x_s^i, y_s^i, z_s^i) are known (all objects are supposed to be point objects);

• v_0, the agent's linear cruising speed;

• α_{add}, the angle of additive correction to the calculated course, which defines the angle of the agent's motion relative to the landmarks;

• $\dim_{attr}$, the distance degree between objects for attraction;

• $\dim_{sep}$, the distance degree between objects for repulsion (the larger it is, the more "precisely" the agent will move around spots (obstacles));

• d_f, the coefficient determining the threshold for repulsive (attractive) forces (the smaller it is, the closer the agent will move to obstacles).

Ultimately, the parameters $\dim_{attr}$ and $\dim_{sep}$ define the path accuracy, while d_f defines the nature of transients.

The core of this procedure is that every object within the field of view Ω possesses both attractive and repulsive potentials. The repulsive field near the object must be significantly stronger than the attractive one. This is achieved by introducing a certain threshold distance to the obstacle, as well as by the nature of the repulsive field. Of course, the repulsive potential must increase infinitely in the immediate vicinity of the object.

In addition, it is important to consider the influence of objects located along the robot's path.

Attractive force. Let d_i^a denote the misalignment between the agent's orientation vector and the direction vector toward the object: $d_i^a = A - A_i$. Then the tuple $\langle v_s, A_s \rangle$ is defined as the sum of the vectors $\langle v_{res}, A_{res} \rangle = \sum_{o_i \in \Omega_B} (F_i^{attr}, A_i)$, where F_i is the "attractive force" of object o_i from the region Ω:

$$F_i^{attr} = \begin{cases} 0 \text{ if } r_i < r_{min} \\ \dfrac{k_{cos}}{r_i^{\dim_{attr}}} \text{ otherwise,} \end{cases}$$

$$k_{cos} = \cos \max_j (d_i^a)_j,$$

and r_i is the distance to the object. The coefficient k_{cos} is introduced to assign higher priority to the landmarks located along the agent's path. To avoid the greater influence of closer landmarks, it is reasonable to raise the distance r_i in the denominator to a power

of $\dim_{attr} \geq 1$. In addition, very close landmarks should not participate in determining the direction of motion. To achieve this, a certain minimum distance r_{min} is used (e.g., a multiple of the agent's size).

The repulsive force F_i^{sep} is defined by analogy, provided that $\dim_{sep} > \dim_{attr}$.

After determining the direction of the resulting field $F_i = F_i^{attr} + F_i^{sep}$, an extra orientation angle α_{add} is added to its value. This angle determines the agent's side to navigate around the obstacle.

Based on these considerations, we present an algorithm for the agent's motion along a chain of objects.

1. function SpotMv($c_a, \alpha_a, \Omega, v_0, \alpha_{add}, dim_{attr}, dim_{sep}, d_f$)
2. %-- Stage 1. If landmarks are not visible, the agent will rotate in place
   %-- (searching for landmarks).
3. if $\Omega = \emptyset$ then
4. $v_{res} := {}^{v_0}/_2, A_{res} := \langle \alpha_{yaw}^a + d_{yaw}, 0, 0 \rangle$
5. return $\langle v_{res}, A_{res} \rangle$
6. Endif
7. %-- Stage 2. Initializing the value of the resulting control,
8. %-- and the attractive and repulsive forces.
9. $\langle v_{res}, A_{res} \rangle := \langle 0, \langle 0,0,0 \rangle \rangle$
10. $\langle v_{attr}, A_{attr} \rangle := \langle 0, \langle 0,0,0 \rangle \rangle$
11. $\langle v_{sep}, A_{sep} \rangle := \langle 0, \langle 0,0,0 \rangle \rangle$
12. %-- Stage 3. Organizing a loop through all objects s in the field of
    %-- view Ω.
13. for s inΩ do
14. %-- 3.1. Determining the normalized distance d to the object and
    %--the angle α_s.
15. $A := (x_s - x_a, y_s - y_a, z_s - z_a)$
16. $d := |A|/d_f$
17. if $(d_{min} > d)$ then $d_{min} := d$
18. $\alpha_s = \langle \alpha_{yaw}^s, \alpha_{pitch}^s, \alpha_{roll}^s \rangle := EulerAngles(A)$
19. %-- 3.2. Taking the direction into account. The law of cosines:
20. %-- the relevant landmarks are those ahead.
21. $k_{cos} := cos(\alpha_{yaw}^a - \alpha_{yaw}^s)$
22. %-- 3.3. Taking the closeness factor (visibility) into account
23. $k_{near} := \begin{cases} 1 \; if \; d > 0.5 \\ 0 \; otherwise \end{cases}$
24. %-- 3.4. Calculating the attractive/repulsive forces
25. if $(d > 1)$ then %-- Attraction
26. $force := k_{cos} k_{near} d^{dim_{attr}}$
27. $\langle v_{attr}, A_{attr} \rangle := \langle v_{attr}, A_{attr} \rangle + \langle force, \alpha_s \rangle$
28. else %-- Repulsion
29. $\alpha_{sep} := \langle \alpha_{yaw}^s + \frac{\pi}{2}, \alpha_{pitch}^s + \frac{\pi}{2}, \alpha_{roll}^s \rangle$
30. $force := k_{cos} k_{near} d^{dim_{sep}}$
31. $\langle v_{sep}, A_{sep} \rangle := \langle v_{sep}, A_{sep} \rangle + \langle force, \alpha_{sep} \rangle$
32. endif
33. endfor
34. %-- Stage 4. Calculating the total force
35. $\langle v_{res}, A_{res} \rangle := \langle v_{attr}, A_{attr} \rangle + \langle v_{sep}, A_{sep} \rangle$
36. %-- Adding the extra value α_{rot} to the yaw angle for strafe motion.
    %-- This value depends on the distance to the obstacle r.
37. $r := max(d_{force}(d_{min} - 1),\ 0)$
38. $\alpha_{rot} := \alpha_{add} e^{-r}$
39. $A_{res} := \langle \alpha_{res}^{yaw} + \alpha_{rot}, \alpha_{res}^{pitch}, \alpha_{res}^{roll} \rangle$
40. %-- Constant velocity
41. $v_{res} := v_0$
42. return $\langle v_{res}, A_{res} \rangle$

Thus, the procedure for moving to a given region of space requires implementing two motion programs (free motion and motion around landmarks). We proceed to the machine control level responsible for switching between these programs.

1.3. Control Machines

To unify control, we will assume that the motion procedures are launched by *dctl*, a machine with a single state. When the machine is launched, the argument *arg*—the name of the motion program—is specified. This machine outputs the control law $F_{\text{ctl}} = \langle v_{\text{res}}, A_{\text{res}} \rangle$. For the task under consideration, the arguments are the names of the free-motion and obstacle-avoidance motion procedures, *FreeMv* and *SpotMv*, respectively.

The meta-machine for a bearing-based motion with obstacle avoidance implements the following behavior algorithm: if the path is clear, the robot moves according to the free-motion program (the machine *dctl* with the argument *FreeMv* is active); otherwise, control is transferred to the machine that activates the procedure for moving along a chain of objects (*dctl* with the argument *FObstAlong*). The program *FObstAlong* is a realization of the function *SpotMv* with parameters defining the required specifics of the robot's motion (moving closer to or farther from obstacles, accuracy, etc.).

Fig. 2 shows the machine *dctl* and the meta-machine of a bearing-based motion. The edge labels contain the transition condition (in the numerator) and the procedure to be executed. For the meta-machine, this is the launch of the machine *dctl* with the corresponding argument.

More complex behavior may require creating corresponding meta-machines of the same rank M^1 and a second-rank machine to activate the machines of rank M^1. For example, the machine M^2 can be responsible for determining motion termination conditions, launching the pursuit program (of the meta-machine), and so on.

2. EXPERIMENTS

The experiments were conducted at three levels: computations in *Kvorum* (a discrete-event multi-agent simulation system), simulations in *Gazebo* (a physical simulation environment), and real robots (both in laboratory conditions and in open spaces).

In all experiments, the robot was a wheeled platform with a differential drive, equipped with a locator-rangefinder in the forward half-plane and seven rangefinders arranged as shown in Fig. 3.

In addition, the robots were equipped with a gyroscope-based compass. At a frequency of 10 Hz, the rangefinders output the spherical coordinates of the detected obstacles, which are converted into the coordinates of the obstacle objects in the set Ω. The locator, both the model and the real one, has a discrete step as well. (The length of the locator's reading vector is 180 due to the minimum step of 1°, although the real step is 10° due to using rangefinders with a working angle of about 15°.) Therefore, the resulting potentials are formed by a relatively small set of objects.

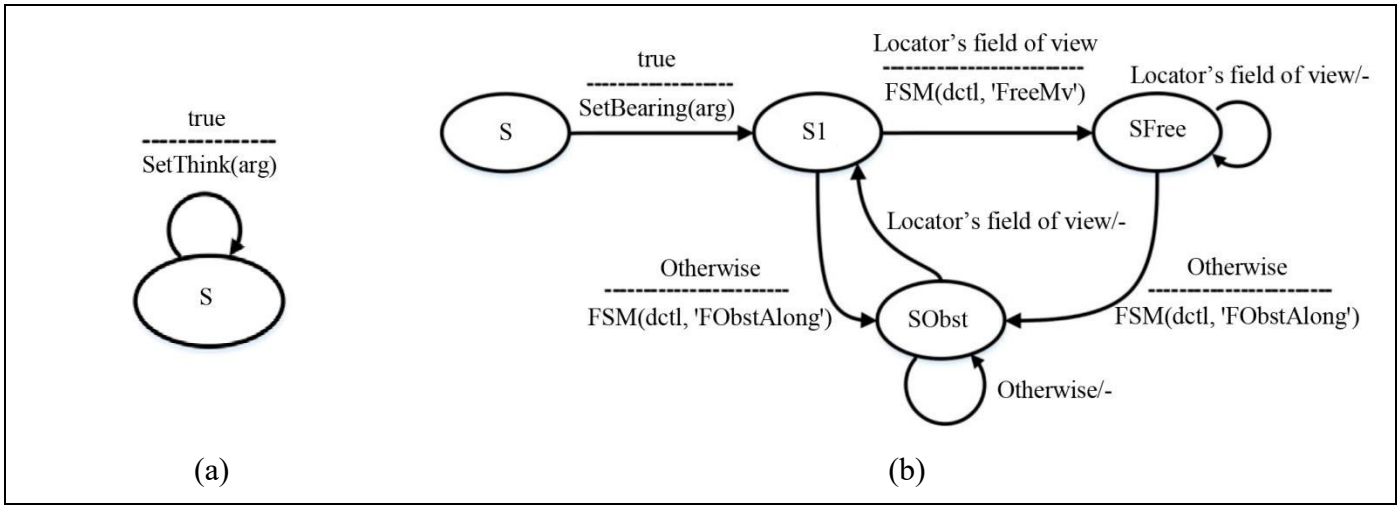


Fig. 2. FSMs: (a) the level 0 machine (*dctl*, launch of motion programs) and (b) the meta-machine for bearing-based motion with obstacle avoidance.

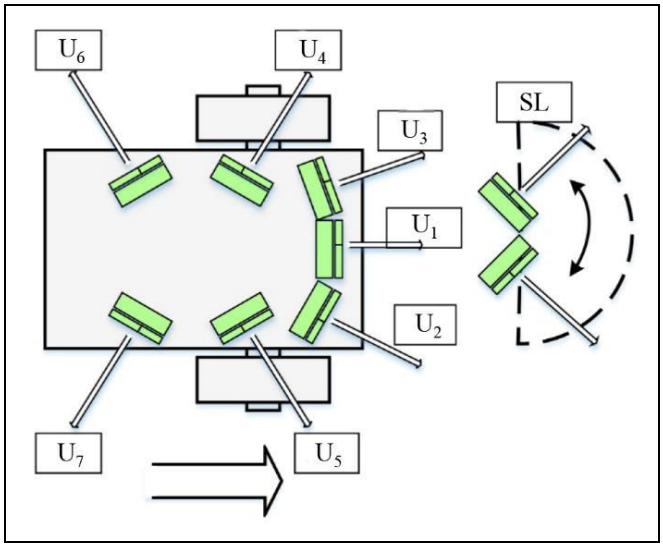


Fig. 3. The arrangement of sensors: rangefinders U_1–U_7 and 180°-view locator SL.

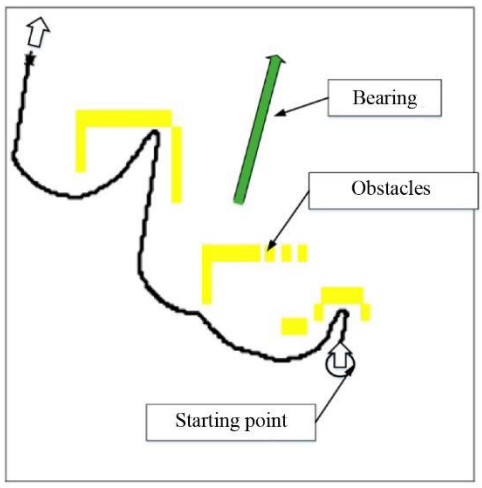
Note that in all experiments (computational, simulation, and real), the same computational core was used to calculate field potentials, as well as the same control machines.

2.1. Computational Experiments

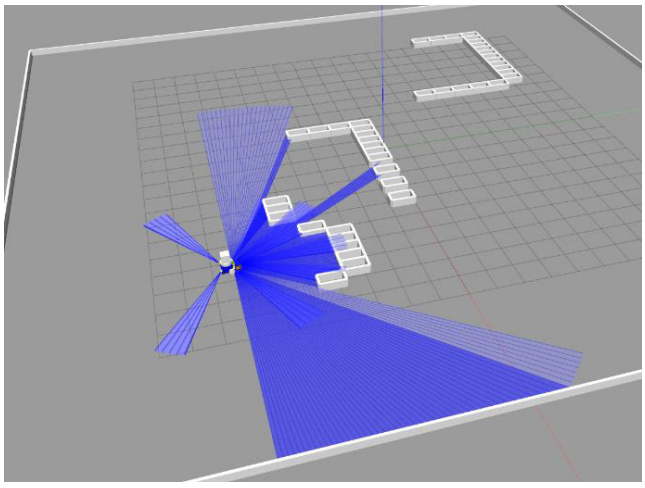
The main series of experiments was conducted in *Kvorum*. Recall that *Kvorum* is a discrete-event simulation system in which an agent environment is represented by a discrete (cellular) field [13]. Since the motion procedures based on the PF are continuous, the resulting control of the form $F_{\text{ctl}} = \langle v_{\text{res}}, A_{\text{res}} \rangle$ was translated into discrete commands for the model object: a step of constant-velocity motion and a discrete turn. Nevertheless, this coarsening of the control law had no effect on the final result. Figure 4a shows an example of a field with obstacles and the agent's path. The physical model of this field is presented in Fig. 4b.

The experiments were intended to determine the success rate (the share of successful missions to a given half-plane) E depending on the field's obstacle density ρ, i.e., $E(\rho)$, where $\rho = N_{\text{obst}}/N$, N_{obst} is the number of obstacle cells, and N is the total number of cells in the model field. The value of ρ was selected from the range [0.005, 0.06] with a step size of 0.005. For each value of ρ, ten experiments were conducted: fields were generated with a random uniform distribution of obstacles, and the agent was launched from the same starting point. The mission was considered successful if the agent reached the target region after at most $T_{\max} = N$ cycles (as a rough pessimistic estimate). For each series, the value of $E(\rho)$ was determined. The results of the experiments can be seen in Fig. 5.

The curves $E(\rho)$ (Fig. 5) were obtained by simulation modeling in *Kvorum* (the kvor plot) and the wave algorithm (the wave plot) under the same conditions. The wave algorithm was chosen as the reference. For both algorithms, the starting point (the robot's initial position) was set at the bottom edge of a rectangular field, and the target point was at the top. In addition, the wave algorithm was executed on a field with obstacle scaling: a safety zone is required when moving in an environment with obstacles (the robot must keep a certain distance from them).



(a)



(b)

Fig. 4. Examples: (a) an illustrative experiment in *Kvorum* and (b) the robot's model in *Gazebo*. The obstacle topology is the same.

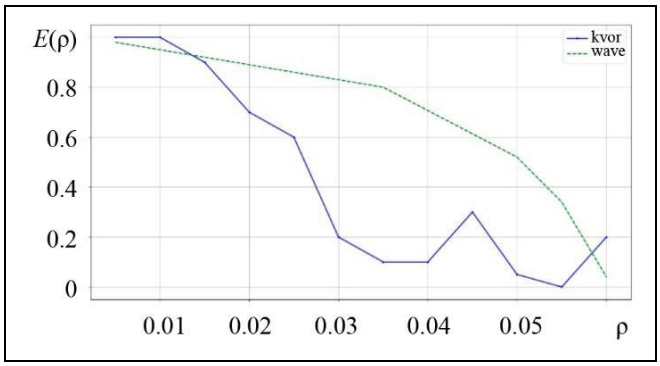


Fig. 5. The results of computational experiments.

Of course, the wave algorithm performs better due to the guaranteed solution calculation for environments in which a path exists. Nevertheless, for both sparse environments and environments with high obstacle density ρ, the success rate values are clearly similar. For the PF method, the earlier deterioration of the performance when increasing ρ is due to potentially getting stuck at a local extremum. For the wave algorithm, a decrease in $E(\rho)$ implies the emergence of theoretically impassable zones.

Obviously, the introduction of additional control levels still cannot ensure successful local navigation using the PF method. A guaranteed correct motion with obstacle avoidance is achieved only within a region determined by the radius r_Ω of the field of view.

In fact, the experiments can be interpreted as determining the dependence of navigation efficiency on the radius r_Ω and the average distance between obstacles (obstacle density).

Indeed, let S be the total area of the field. Then each obstacle is the center of a region with the area $S_0 = S / N_{\text{obst}}$ and radius $r_0 = \sqrt{S_0}$. Since $S = N$, we have $S_0 = \dfrac{N}{N_{\text{obst}}} = \dfrac{N}{\rho N} = \dfrac{1}{\rho}$, and the average distance r_0 between obstacles can be estimated as

$$r_0 = \sqrt{S_0} = \frac{1}{\sqrt{\rho}}.$$

In the experiments, the (constant) value of r_Ω was set equal to 10% of the linear size of the entire region, i.e., $r_\Omega = 0{,}1\sqrt{N}$.

The following result was established: if the radius r_Ω of the agent's field of view exceeds the minimum distance r_0 between obstacles, then the agent successfully completes the route.

Comparison with other path planning algorithms and methods. Formally, the experiments were conducted on maps of the *random*(*m*×*n*, *R*) class: a rectangular grid consisting of *m* rows and *n* columns, with *R*% of randomly selected cells being blocked (representing obstacles). Among the many metrics describing both the properties of the environment and the effectiveness of the solution, two were selected: the *obstacle density* ρ and the *success rate E*. In studies, the properties of the algorithms and methods under consideration are often assessed using examples of fixed obstacle maps. The set of metrics can be quite diverse, see [14]. For example, *traversability* is a widely used metric for assessing the complexity of static environments. It is defined as the average traversable distance for all uniformly selected positions and directions:

$$Tr = \frac{1}{N_s}\sum_{i=1}^{N_s} d_i,$$

where d_i is the traversable distance at the *i*th selected position and direction; N_s is the total number of selected positions and directions. However, due to computational costs, the use of this metric is impractical for experiments involving the generation of a set of random maps.

Perhaps, an estimate closest to $E(\rho)$ was given in [15]: the dependence of E on a parameter characterizing the obstacle density; however, the comparison of algorithms provided therein is only qualitative in nature, showing a monotonically decreasing function $E(\rho)$. Therefore, in this paper, we compare the results of the PF method with those of the wave algorithm as a reference.

2.2. Experiments in *Gazebo* and with Real Robots

These experiments were primarily intended to illustrate the effectiveness of the above models in real conditions. Furthermore, the control law $F_{\text{ctl}} = \langle v_{\text{res}}, A_{\text{res}} \rangle$ in the experiments was more naturally translated into motion commands specified by the pair $\langle v_{\text{lin}}, v_{\text{ang}} \rangle$, where v_{lin} and v_{ang} are the robot's linear and angular velocities, respectively.

Since A_{res} in the control law defines the resulting direction of motion, the value of v_{ang} was determined as the rotation eliminating the misalignment between the desired and current compass orientation of the robot.

2.2.1. Physical Simulation in *Gazebo*

Gazebo is a physical simulation tool widespread in robotics [16]. It allows adding realistic sensors and using 3D agent models, thereby facilitating the transition from control models to real robotic devices.

The images of the robot model and a test field are provided in Fig. 6.

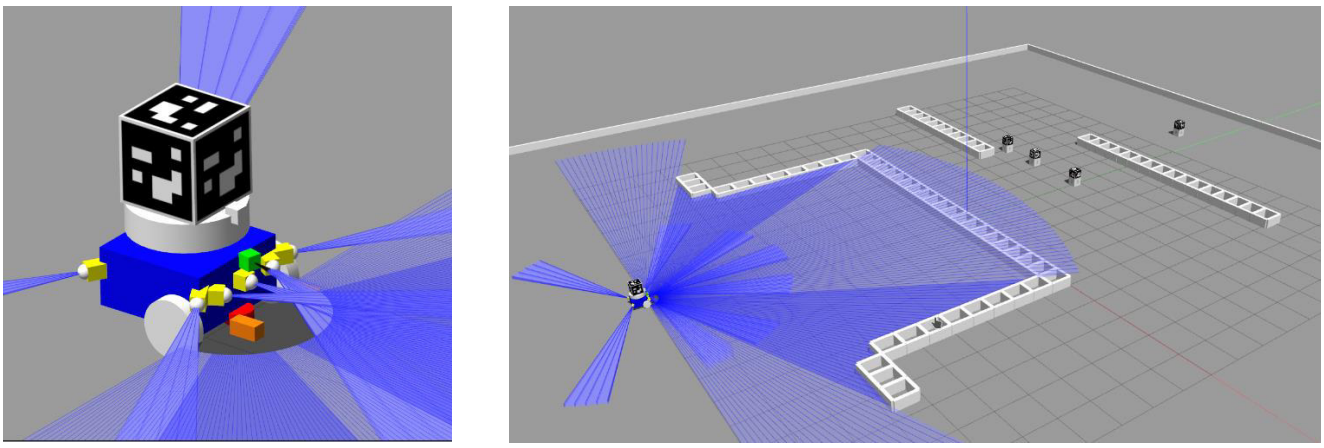


Fig. 6. Simulation modeling in Gazebo: (a) the robot's model and (b) the robot on the test field.

2.2.2. Experiments with real robots

The experiments were conducted both in laboratory conditions and in open space with the corresponding robots. Based on a common architecture, the robots differ in size and the types of rangefinders used on their locators: the laboratory robot has VL53L0X laser rangefinders (15° angle and range up to 2 m with 3% accuracy), while the "outdoor" robot has HC-SR04 ultrasonic rangefinders (15° angle and range up to 4 m).

The robot's locators are paired rangefinders mounted on a servo motor. This provides a 180° field of view when the servo motor rotates 90° (six steps at an effective angle of 15°). The compass is based on the MPU6050 IMU sensor (an accelerometer and a gyroscope with a standard deviation of 0.3°). Compass drift was ignored.

The outdoor (designed for open spaces) and laboratory robots are presented in Fig. 7.

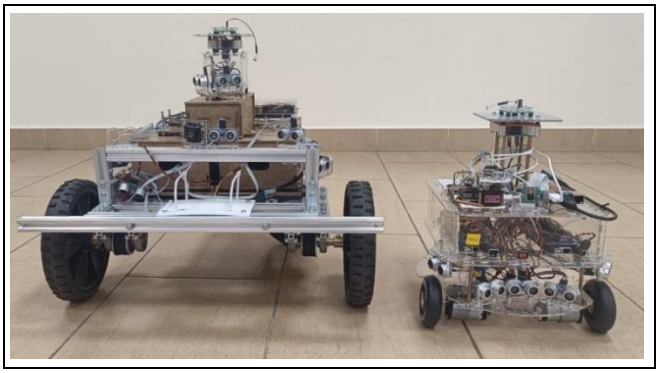


Fig. 7. The outdoor (left) and laboratory (right) robots.

For ease of debugging and control, computations were performed on a remote computer, while the Raspberry Pi 4 onboard computer served to transmit sensor data and process velocity control commands. The robots' task was to move in the direction determined by their initial orientation. The hardware reflexes––the response to obstacles in the immediate vicinity––were disabled. Overall, the obstacles comparable in size to the robot were satisfactorily avoided (no collisions). This is due to the wide coverage angle of the robot's rangefinders.

The errors of the sensory system. How do measurement errors affect the system's performance? This question arises both in computational experiments and when working with real robots. The errors are categorized into obstacle detection failure, accuracy, and inertia. Despite the discrete-time operation of the above locator, the use of rangefinders with a wide 15° angle of view allows eliminating obstacle detection failure: the locator operates with a significant overlap. A measurement accuracy of 3% is sufficient for the safe operation of the system: if the minimum safe distance to obstacles equals one linear dimension of the robot (0.5 m), the permissible error will not exceed 20% for a velocity of 1 m/s and a sensor updating time of even 0.5 s. The same applies to the inertia of the sensory system. (Since the robot is moving, the rangefinder provides a distorted picture of obstacles.) How far will the robot travel during the rangefinder updating time? This question is resolved by analogy with that of accuracy. Thus, under the above parameters, inertia is also negligible.

CONCLUSIONS

Navigation in extreme conditions, when a robot relies only on the data from relatively simple sensors and has no global navigation means or prior

knowledge of its environment, requires specific control methods. These specifics are determined, first, by limited computational resources; second, by the flexibility and universality of control models; and third, by weak requirements for the quality (accuracy) of solutions. In this regard, the potential field method stands out favorably among a wide range of navigation mechanisms.

Due to fundamental limitations, the continuing interest in the PF method is driven by the development of various modifications, the design of new hybrid architectures, the application of machine learning methods, and so on. In this paper, we have proposed an architectural approach to local navigation in which the PF-based control is transferred to the level of elementary motion, while situation analysis and switching between motion programs are transferred to the level of meta-machine control.

Of course, the success of local navigation depends on the size of the robot's field of view; however, we emphasize once again that this field can be naturally expanded owing to the uniformity of control mechanisms.

Acknowledgments. *This work was carried out within the state assignment of NRC Kurchatov Institute.*

REFERENCES

1. Katona, K., Neamah, H.A., and Korondi, P., Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robots, *Sensors*, 2024, vol. 24, no. 11, art. no. 3573.
2. Filimonov, A.B. and Filimonov, N.B., Issues of Motion Control of Mobile Robots Based on the Potential Guidance Method, *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2019, vol. 20, no. 11, pp. 677–685. (In Russian.)
3. Filimonov, A.B. and Filimonov, N.B., New Approach for Use of the Potential Fields Method in Mobile Robotics, *Proceedings of the 4th International Conference on Control in Technical Systems (CTS'2021)*, St. Petersburg: IEEE, 2021, pp. 314–317. (In Russian.)
4. Xing, H., Review of Unmanned Aerial Vehicle Obstacle Avoidance Planning Based on Artificial Potential Field, *Appl. Comput. Eng*., 2023, vol. 10, no. 1, pp. 55–63.
5. Koren, Y. and Borenstein, J., Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, USA, 1991, vol. 2, pp. 1398–1404.
6. Xu, Z., Hess, R., and Schilling, K., Constraints of Potential Field for Obstacle Avoidance on Car-Like Mobile Robots, *IFAC Proceedings Volumes*, 2012, vol. 45, no. 4, pp. 169–175.
7. Sidorenko, A.V. and Saladukha, N.A., Obstacle Avoidance Algorithm in Mobile Robot Motion, *System Analysis and Applied Information Science*, 2024, vol. 4, pp. 29–33. (In Russian.)
8. Aldoshkin, D.N. and Tsarev, R.Yu., Mobile Robot Path Planning in the Presence of Obstacles and Lack of Information about the Environment, *Mechatronics, Automation, and Control*, 2016, vol. 17, no. 7, pp. 465–470. (In Russian.)
9. O'Rourke, J., *Computational Geometry in C*, Cambridge: Cambridge University Press, 1998.
10. Karpov, V.E., Vorobiev, V.V., and Rovbo, M.A., On Some Aspects of Applying Finite State Machine Models to Group Control, *Mechatronics, Automation, and Control*, 2023, vol. 24, no. 4, pp. 171–180. (In Russian.)
11. Reynolds, C.W., Flocks, Herds, and Schools: A Distributed Behavioral Model, *Proc. 14th Annual Conf. Comput. Graph. Interact. Tech. (SIGGRAPH'1987)*, Anaheim, 1987, vol. 21, no. 4, pp. 25–34.
12. Polikarpova, N.I. and Shalyto, A.A., Programming of Finite-State Machines, St. Petersburg, 2008. (In Russian.)
13. Karpov, V.E., Rovbo, M.A., and Ovsyannikova, E.E., Kvorum – the System for Modeling the Behavior of Robotic Agent Groups with Elements of Social Organization, *Software & Systems*, 2018, vol. 31, no. 3, pp. 581–590. (In Russian.)
14. Shi, M., Chen, G., Serra Gómez, A., et al., "Evaluating Dynamic Environment Difficulty for Obstacle Avoidance Benchmarking," *arXiv:2404.14848*, 2024. DOI: https://doi.org/10.48550/arXiv.2404.14848
15. Diab, M., Mohammadkarimi, M., and Rajan, R.T., Artificial Potential Field-Based Path Planning for Cluttered Environments, *Proceedings of the 2023 IEEE Aerospace Conference*, Big Sky, MT, USA, 2023, pp. 1–8. DOI: 10.1109/AERO55745.2023.10115857
16. *Gazebo*. URL: https://gazebosim.org/ (Accessed September 3, 2025.)

This paper was recommended for publication by E.Ya. Rubinovich, a member of the Editorial Board.

Received September 4, 2025, and revised January 27, 2026. Accepted March 16, 2026

Author information

Karpov, Valery Eduardovich. Dr. Sci. (Eng.), National Research Center Kurchatov Institute, Moscow, Russia
✉ karpov-ve@yandex.ru
ORCID iD: https://orcid.org/0000-0002-9364-1223

Cite this paper

Karpov, V.E., Finite-State Machine Control and the Method of Potential Fields for Mobile Robot Navigation in an Environment with Obstacles. *Control Sciences* **3,** 86–99 (2026).

Original Russian Text © Karpov, V.E., 2026, published in *Problemy Upravleniya*, 2026, no. 3, pp. 72–82.

Translated into English by *Alexander Yu. Mazurov*,
Cand. Sci. (Phys.–Math.),
Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia
✉ alexander.mazurov08@gmail.com