MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL AVIATION UNIVERSITY

FACULTY OF AVIATION, ELECTRONICS AND TELECOMMUNICATIONS

DEPARTMENT OF AEROSPACE CONTROL SYSTEMS

TO THE PROTECTION

Head of Department

_____

“_____”_____ 20___y.

QUALIFICATION WORK

(EXPLANATORY NOTE)

GRADUATE DEGREE OF EDUCATION

"BACHELOR"

Topic: Automatic ground vehicle control system in obstacle avoidance mode

Performer: Moskalenko Nikita Oleksandrovich _____

Head: Melnyk Yury Vitaliyovych _____

Normocontroller: Divnych Mykola Polikarpovych _____

KYIV 2024

NATIONAL AVIATION UNIVERSITY

Faculty of Aeronautical Electronics and Telecommunications

Department of aerospace control systems

Specialty 151 "Automation and computer-integrated technologies"

APPROVED

Head of Department

_____

«_____»_____20__ y.

TASK

for the performance of qualification work

Nikita Oleksandrovych Moskalenko's

(last name, first name, patronymic of the graduate in the genitive case)

1. Topic of qualification work Automatic ground vehicle control system in obstacle avoidance mode_____

approved by the rector's order from «01» 04 2024y. No 511/art.

2. The term of the work: from 13.04.2024 to 16.05.2024

3. Initial data for work:Devise an obstacle avoidance algorithm for ground mobile robots.

4. Contents of the explanatory note: The problem of creating systems of automatic control of ground moving objects in the mode of bypassing obstacles

5. List of illustrative presentation material:

Graphics of robot follow point for given data

6. Calendar plan-schedule

№ order	Task	Term implementation	A note about implementation
1.	Find literature for this problem	25.04.2024-29.04.2024	completed
2.	Create content of work	30.04.2024	completed
3.	Introduction to the thesis	02.05.2024-03.05.2024	completed
4.	Theoretical analysis and modules of ground moving objects	04.05.2024-06.05.2024	completed
5.	Creation of evolutionary algorithms	07.05.2024-10.05.2024	completed
6.	Creating a bypassing obstacle program	08.05.2024-10.05.2024	completed
7.	Designing the project	13.05.2024-16.05.2024	completed

7. Issue date of the task:“13" 04 2024 y.

Head of qualification work _____Melnyk Y.V.

(signature of the manager) (P.I.B.)

Accepted the task _____ Moskalenko N.O,

(signature of the graduate) (P.I.B.)

ABSTRACT

The text part of the work: 69 pages, 25 figures, 3 formulas

Research object: Application of genetic algorithms for solving the problem of path planning for autonomous robots.

Research subject: Navigation algorithms for mobile robotic systems.

The aim of the study is to devise an obstacle avoidance algorithm for ground mobile robots in MATLAB.

Research methods: Mathematical modeling and optimization techniques

An algorithm for controlling a mobile robot and simulation have been developed, allowing for efficient obstacle avoidance along the robot's path.

The materials from the qualification work can be utilized for conducting simulation modeling in the design of control systems for the motion of mobile ground robots.

AUTONOMOUS ROBOTS, AUTOMATIC CONTROL SYSTEM, MACHINE LEARNING, ARTIFICIAL INTELLIGENCE,GPS MODULES.

CONTENT

INTRODUCTION

In the modern world, the automation of ground moving objects, such as robots, cars, and drones, is becoming increasingly important. However, one of the main problems with automated movement is the need for effective obstacle avoidance, such as other objects, people, vehicles, or even natural barriers.

In some situations, obstacles may appear unexpectedly or change their position, complicating the task of trajectory planning. For example, in urban environments, cars must react quickly to other vehicles, pedestrians, and obstacles on the road to avoid accidents.

It is also important to consider that control systems must be able to operate in real-time and reliably respond to changes in the surrounding environment. This creates a need for the development of efficient and reliable automatic control systems that ensure the safe movement of the object around obstacles.

The research goal is to develop and test an automatic control system for ground moving objects in obstacle avoidance mode. Specific objectives aimed at achieving this goal may include:

1. Development of obstacle avoidance algorithms: Creating effective algorithms that allow the object to automatically determine the optimal path for avoiding obstacles in various conditions.
2. Sensor integration: Developing obstacle detection systems for the object's path using various sensors such as radars, LiDARs, cameras, etc.
3. Implementation of control system: Creating software and hardware for effective real-time control of the ground object's movement.
4. Testing and validation: Conducting system tests on specially created test sites or in simulators to verify its functionality and reliability.
5. Performance evaluation: Analyzing test results to evaluate the speed, accuracy, and safety of the system under different conditions and scenarios.
6. Capability demonstration: Demonstrating the operation of the developed system in real conditions or on virtual platforms.

Relevance of the automatic control system for ground moving objects in obstacle avoidance mode:

1. Safety and accident avoidance: In a world where automotive vehicles, robots, and other moving objects coexist closely with humans and other obstacles, effective accident avoidance and safety become critically important tasks.

2. Increased productivity: Automatic control allows moving objects to work more efficiently and effectively in conditions of limited space and time.
3. Technological development and artificial intelligence: The development of automatic control systems stimulates the advancement of advanced technologies and methods of artificial intelligence, which can have broad applications in other fields.
4. Modernization of the transportation system: In the modern world, the need for modernization of transportation infrastructure and management systems to ensure an efficient and safe transportation system becomes increasingly evident.
5. Economic benefits: Reducing the number of accidents and improving productivity can lead to economic benefits for companies and society as a whole.

CHAPTER 1. DESCRIPTION OF THE PROBLEM OF CREATING SYSTEMS OF AUTOMATIC CONTROL OF GROUND MOVING OBJECTS IN THE MODE OF BYPASSING OBSTACLES

1.1 Basic terms and definitions

An automatic control system consists of a controlled object and automatic measuring and controlling devices. Unlike automated systems, this operates independently, executing predefined process functions automatically, without human intervention, apart from initial setup and adjustments. Such systems are commonly known as Automatic Control Systems (ACS).Automatic regulation involves maintaining or adjusting specific levels of physical or chemical quantities characterizing the process according to predetermined laws. It encompasses various influences on the process selected from a predefined set. A control device generates a control effect on the controlled object based on its operational algorithm. When operating independently, it's referred to as an automatic control device.

An Automatic Control System (ACS) comprises a controlled object and a control device, interacting to achieve predefined objectives.

Additionally, it includes a reference device that establishes the necessary laws to alter the initial state of the controlled object. The concept of "control" encompasses "regulation," ensuring system parameters align with specified values. An automatic regulation system maintains parameters necessary for the desired technological process course, without human intervention.

The progression of a technological process or operation in a machine is viewed as a series of states, changing sequentially based on predefined criteria. Controlling a technological machine involves two tasks: ensuring the required sequence of transitions between states and maintaining the machine within specific states. Program control systems ensure the former task, orchestrating purposeful changes in the controlled object's state, enabling predefined sequences of operations.

Automatic regulation systems detect deviations in controlled parameters, influencing the object or process to rectify these deviations.

Elements of automatic control systems

Any control system consists of three main components: control program, control device, and controlled object.(Fig.1)

Fig. 1 Elements of control systems

The control program encompasses a series of instructions, providing a detailed step-by-step guide for a technological machine or a system of machines to execute specific technological operations or processes.

The control device interprets the control information from the program into a format understandable to the technological machine. It translates this data into commands for the various mechanisms, units, and components of the machine. The control unit's output signals are linked to the executive elements of the technological machine. To ensure the accurate execution of commands, feedback signals are sent back to the control device through dedicated channels. Consequently, the control device interacts with the controlled object by exchanging signals, including:

"u": commands from the control system to the executive mechanisms of the machine.

"u'": information about the execution of commands from the executive system to the control system.

This exchange of signals facilitates effective communication and coordination between the control device and the controlled object, ensuring the precise execution of commands within the technological process(Fig. 1.1).

Fig. 1.1 Controlled technological machine

Automatic control system (Fig. 1.2):

Fig. 1.2 Functional scheme of ACS

The automatic control system (ACS) provides for the automatic collection and processing of information, as well as the generation and implementation of control influences on the controlled object according to the control quality criterion. Therefore, it includes a measuring device, control, and executive elements (Fig. 1.3).

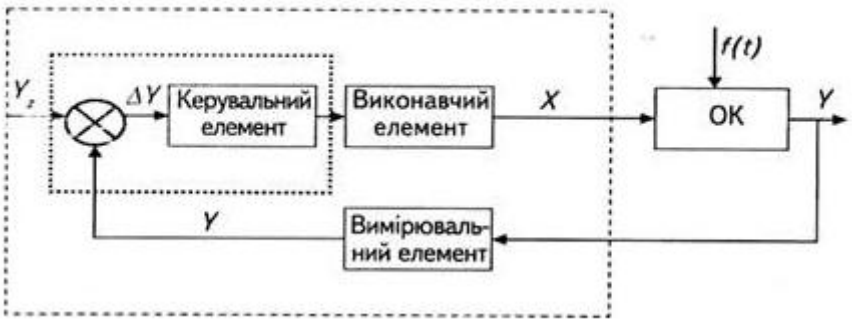

Fig. 1.3 The structure of the automatic control system

The measuring component serves the purpose of gauging technological parameters (such as velocity, temperature, etc.) and output variables of the controlled entity, which depict its condition. In practical automatic control setups, a sensor typically fulfills this role by transforming non-electric quantities (like displacement, speed, force, pressure, temperature, etc.) into electrical signals.

Control elements produce governing inputs for the executive component. Typical control components comprise conversion and summation segments. A shared characteristic among technical or technological systems engaged in control processes is the transmission of data regarding the ongoing procedures within distinct sections of the system via signals. In technical systems, tangible conveyors of information are denoted as signal carriers, capable of being modified based on the conveyed data. These carriers may encompass electrical voltage and current, pressure, mechanical shift, among others. The structural constituents of the system aim to translate one physical quantity and its corresponding signals into others. This process finds reflection in the cybernetic concept of system links.

A link represents a control part integrated into the automatic control system, where the input parameter undergoes conversion into the output parameter in a specified manner. While the graphic depiction of a link as a block does not capture the intricacies of its construction, the crucial aspect lies in the association between the input effect on the link and its resulting reaction at the output. This methodology facilitates the development of models for elements across diverse technical systems, independent of their particular technical realization.

Executive components directly execute the control effects of the automatic control system on the controlled entity. The prevalent devices utilized in technical systems encompass:

- DC and AC motors,
- Hydraulic drives and mechanisms,
- Pneumatic drives,
- Heating elements.

1.2 Types of automatic control

In control theory, automatic control types are categorized based on both the intended goal of control and the method of generating control inputs for the

controlled object. Concerning the control objective, automatic control is segmented into stabilizing, tracking, programmatic, coordinating, optimal, extreme, terminal, emergency, and restorative types. Regarding the generation of control inputs, all varieties of automatic control rely on just two fundamental principles:

- Disturbance regulation;
- Error detection

Disturbance regulation

Involves controlling the system while considering the disturbance magnitude. The disturbance level is gauged, and its value is relayed to the control device. This device assesses the disturbance signal and initiates a control action on the controlled object. This approach is often referred to as the Poncelet principle or compensation control, as it involves compensating for the disturbance's effect through control.

The main advantage of this method is its rapid response. Any change in the disturbance magnitude triggers an immediate reaction from the control device. However, this form of control doesn't permit alterations to the object's operating mode.

Despite its advantages, disturbance management has its drawbacks. Firstly, numerous factors can disturb the object, such as load variations, environmental temperature changes, or fluctuations in tank levels. To ensure effective control, each disturbance must be accounted for, necessitating a separate control loop for each one. This task is practically infeasible due to the multitude of potential influencing factors on any given object. Secondly, implementing disturbance control requires a thorough understanding of how the system responds to disturbances of varying magnitudes, which often demands meticulous investigation. Developing a regulator entails studying the system's behavior under

diverse disturbance conditions, a task that may not always be achievable with the required precision.(Fig. 1.4)

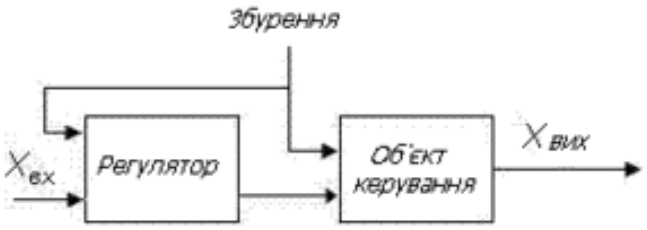

Fig. 1.4 Functional scheme of SAC with disturbance control

Error detection

In accordance with this principle, the output quantity's value is assessed at the system's output. Information regarding this value is transmitted to the control device. The control device generates a control signal based on the disparity between the setpoint signal and the output signal (feedback signal). This control principle is known as the Pozhunov-Watt principle.

Under this principle, control is executed based on the divergence between the output and setpoint values, regardless of the underlying cause of this deviation. It operates irrespective of the number or nature of disturbances influencing the deviation in the operational mode. Control is solely contingent upon the difference in values and can accommodate any influences on the system, constituting the advantage of error control.

However, error control suffers from system inertia, resulting in a time delay between disturbance changes and control actions, leading to oscillations and instability in system behavior. This is due to the fact that, according to this principle, the control device only intervenes with the object once its operational mode shifts and a difference between the output and setpoint quantities emerges.

Systems employing the error control principle are commonly known as closed-loop systems, featuring feedback between the system output and the control device(Fig. 1.5).

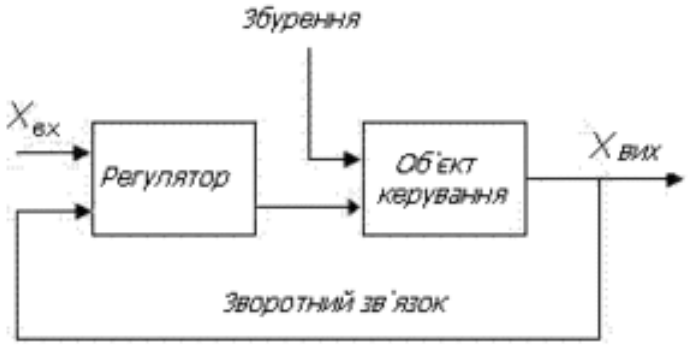

Fig. 1.5 Functional diagram of SAC with deviation control

Combined control principle

A combined control system(Fig. 1.6), operating on a combined principle, integrates two distinct control systems. Information regarding both the disturbance value and the output quantity value is transmitted to the control device. Each signal operates within its own control loop.

Fig. 1.6 Functional scheme of the SAC with a combined control principle

1.3 Process machine control program and classification of systems

Programmatic control involves orchestrating deliberate state changes within a technological system, resulting in the execution of a predetermined sequence of tasks. In this approach, the control program dictates the sequence of state transitions of the technological system, essentially outlining the sequence of

technological actions constituting the technological cycle. Systems that oversee the step-by-step progression of successive states in a technological machine are known as program control systems, commonly found in automatic machines producing discrete products.

Control schemes can be open-loop, where the previous state isn't monitored, and the transition to the next state occurs incrementally. More commonly, control schemes employ monitoring of each state of the technological machine, proceeding to the next state only after verification. Each state corresponds to the execution of a specific segment of the working cycle. Notably, modern packaging machines feature a distributed work cycle segmented into controlled stages, sometimes employing as many as 120-150 sensors.

Programmatic control ensures that the technological machine adheres to the planned sequence of technological actions. The control program must anticipate potential production scenarios; hence, most control systems incorporate feedback loops to provide information about completing each program step before proceeding to the next.

The control program encompasses a set of rules for issuing control commands to the machine's executive mechanisms, facilitating its operation in achieving the specified technological objective. Consequently, the movements of executive mechanisms in machines are governed by the control program, necessitating it to contain all essential information for coordinating these movements.

Automatic control systems can be classified based on various criteria, including the principle of informativeness, the number of controlled parameters and loops, the nature of static and dynamic characteristics, and structural features. Given the rapid advancements in information technologies, one prevalent classification principle is the informativeness principle, distinguishing systems with complete and incomplete initial information. Ordinary systems possess

sufficient initial information to solve tasks for the system's entire operation, while cybernetic systems require additional information acquisition during operation to formulate necessary control commands.

Water supply and drainage systems, as objects of automation, fall into the category of ordinary systems with complete initial information. They can be further divided into closed-loop automatic control systems, operating on the feedback principle by controlling deviations from the output quantity, and open-loop automatic control systems, which control disturbances regarding the same quantity.

Closed systems

Closed-loop systems, in turn, can be of three types: stabilization, programmed, and tracking.

Stabilization systems are intended to provide a constant value of the controlled variable (Y) of the control object: Y = const. Examples of such systems can be automatic control systems for air temperature in residential premises, automatic pressure control systems in the suction pipeline of pumps, and so on.

Programmed automatic control systems are designed to provide a change in the controlled variable through a pre-known program: Y = var.

Tracking automatic control systems also provide Y = var, but their main difference from programmed systems is that the law of change of the controlled variable required for operation is not known in advance and is formed during the system operation.

Open-loop systems

Open-loop systems can be of two types: compensatory and programmed. Compensatory systems provide the formation of control signals at the input of the object, which compensate for the effect of the corresponding disturbance on it.

Programmed control systems, unlike programmed automatic control systems, besides having an open-loop scheme, must also provide a change in the operating mode of the object according to a predetermined program. Examples of such systems can be elevator lifting installations, where the final switches provide the necessary changes in the operating mode of the electric drive depending on the position of the elevator cabin.

Static and astatic automatic control systems

The primary feature of these systems is the control characteristic, illustrating the relationship between the controlled variable (Y) at a static position and the flow rates of the working medium.

A system is deemed static if, when external disturbances alter the object, the controlled variable (Y) fluctuates within specific permissible limits after the transient process concludes, contingent upon the external disturbance. Typically, the control characteristic takes the form:

$$Y = \bar{Y} + \Delta(Y)$$

1.1 Perturbation function

Here, $\bar{Y}$ represents the average value of the controlled variable, while Y denotes the disturbance function, delineating the deviation of the controlled variable from its average value based on the disturbance within the control zone. The level of static stability is gauged by the ratio of the deviation of the controlled variable within the control zone to its average value, expressed as a percentage:

$$\delta = \frac{Ymax - Ymin}{\overline{Y}} * 100\%$$

1.2 The amount of statism

A system is considered static if the controlled variable Y settles on a strictly constant value after completing the transient process, regardless of different external disturbances. In such static systems, the control characteristic manifests as a horizontal straight line.

Continuous and discontinuous action

Regarding continuous and discontinuous action, a continuous system maintains a consistent structure of connections during operation, with each element's output being a continuous function of the disturbance and time. In contrast, discontinuous systems allow changes in connection structures during operation, leading to discontinuous signals at the outputs of elements or the system itself. Discontinuous systems can be further classified into relay and pulse systems, which incorporate relay or pulse elements respectively, impacting their characteristics.

Multi-dimensional systems

In terms of dimensionality, systems can be single or multi-dimensional. Multi-dimensional systems can be categorized as unlinked or linked regulation systems. Unlinked regulation systems feature several controlled coordinates and autonomous controllers that operate separately, though the controlled coordinates may be interconnected through the object. Linked regulation systems involve automatic controllers for different coordinates linked by additional connections, enabling autonomous regulation of individual variables.

Linear and nonlinear systems

Linear and nonlinear systems represent another classification criterion. While real systems often contain nonlinear elements, linear systems are those described by linear dependencies. These systems adhere to the principle of superposition, where the system's reaction to a combination of external actions equals the sum of reactions to each applied separately, akin to an additive function.

• Linear systems are described by linear dependencies, adhering to the principle of superposition. This principle states that the system's response to any combination of external actions equals the sum of responses to each action applied separately. This is equivalent to an additive function, expressed as:

$$x(U, Z) = x(U) + x(Z), \quad (1\text{-}2)$$

• Nonlinear systems contain at least one element with nonlinear characteristics. To facilitate analysis and synthesis tasks, nonlinear characteristics are often linearized. This process involves replacing the real nonlinear system with an equivalent linear (or linearized) system, simplifying the analysis.

1.4 Modules for autonomous navigation

To prevent collisions with obstacles, it's essential to outfit the operation with specialized modules designed to scan the surrounding environment and identify obstacles. Various sensors can be employed for obstacle detection, including an image sensor (camera), ultrasonic sensor, lidar, and Time-of-Flight (ToF).

Time-of-Flight (ToF) is a technique for measuring the distance between the sensor and an object. It relies on the time difference between emitting a signal and receiving its reflection from the object. ToF sensors utilize different types of signals, commonly light or sound. They measure distances by calculating the time it takes for photons to travel from the sensor emitter to the target and back to the sensor receiver. 3D ToF cameras can provide depth data in three dimensions. When using light, ToF is highly effective for distance and range determination. Compared to ultrasound, it offers a greater range, faster readings, and higher accuracy, all while maintaining a compact size, low weight, and minimal power consumption. ToF sensors find applications in robot navigation, vehicle monitoring, people counting, and object detection.

For distance measurement in robotics, ToF sensors such as the Teraranger Evo and IND-TOF-1 from Terabee company are suitable choices. The Teraranger Evo offers precise, high-speed distance measurement and object detection at close range, making it ideal for mobile robotics applications. Terabee offers several models of this sensor, varying in the distance range for object detection, with a maximum detection distance of up to 60 meters.

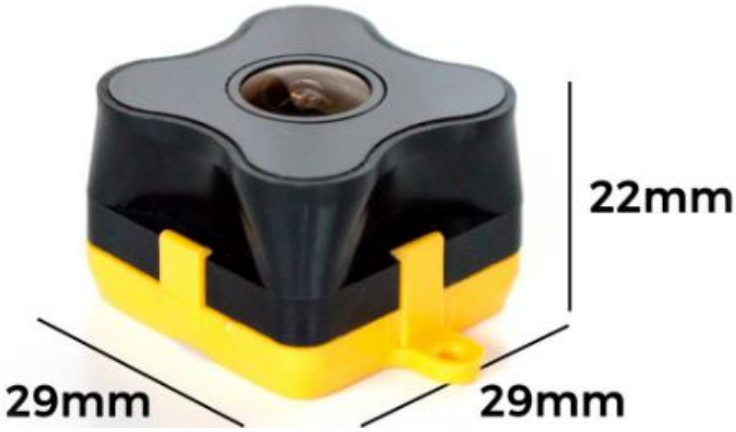

Fig. 1.7 TeraRanger Evo

The TeraRanger Evo sensors comprise an optoelectronic sensor module and a rear panel that seamlessly connects to provide communication channels and power management capabilities to the sensor, eliminating the need for adapters or intricate wiring. The choice of rear panel can be tailored to individual requirements, with options including USB, I2C, and UART interfaces. Moreover, the sensor is compatible with popular platforms such as Arduino, Raspberry Pi, Pixhawk, and ROS (Robot Operating System). Notably, one of its advantages is its ability to operate effectively in low light conditions and even in complete darkness.

LiDAR

Light Detection and Ranging, is a remote sensing technology that employs laser pulses to collect measurements, used for creating 3D models, object maps, and environmental maps. Similar to radar and sonar, LiDAR utilizes laser light waves instead of radio or sound waves. The system calculates the time taken for light to hit an object and return to the scanner, utilizing the speed of light to determine distance. LiDAR systems can generate approximately 1,000,000 pulses per second, with each measurement forming a point in a three-dimensional point cloud, representing shapes or objects. These point clouds consist of points with their own X, Y, and Z coordinates, and sometimes additional attributes. LiDAR finds applications in various tasks, including automotive radar.

In contrast to GPS, which provides location accuracy within a circle diameter of about 5 meters, LiDAR achieves accuracy up to 10 centimeters. Its advantages include high speed and accuracy of data collection, high penetration capability, independence from ambient light intensity, absence of geometric distortions, and seamless integration with other data collection methods. Additionally, LiDAR minimizes human involvement, which proves beneficial in certain fields.

One notable example of LiDAR technology is the LeddarTech Vu8, a compact semiconductor LiDAR capable of detecting multiple targets in eight independent segments. With a detection range of up to 215 meters and weighing only 75 grams, the Vu8 offers nearly double the range in half the volume compared to its predecessor, the Leddar M16.

Fig. 1.8 LeddarTech Vu8

The Vu8 utilizes a stationary laser light source, significantly enhancing the sensor's reliability and cost-effectiveness. Its design offers high resistance to noise and obstacles, rendering it impervious to interference from other sensors, varying lighting conditions (including direct sunlight), and ensuring consistent detection accuracy in diverse weather conditions, including rain and snow.

On the other hand, the LeddarOne is a full-beam sensor module tailored specifically for point measurement tasks, making it well-suited for applications such as level determination, security and surveillance, and presence detection.

The LeddarOne module boasts seamless integration into nearly any system due to its compact size, low power consumption, and high accuracy. These features provide developers and integrators with ample opportunities to enhance their products' performance and functionality.

Ultrasonic Sensors

Ultrasonic sensors are widely recognized for their remarkable reliability and versatility across various industrial sectors. Their capabilities extend to complex tasks such as object recognition or level measurement with millimeter precision, owing to the robustness of their measurement method, which operates effectively under diverse conditions.

Functioning on the basis of emitting high-frequency sound pulses, ultrasonic sensors excel at distance measurement. These pulses, emitted in a conical beam, reflect off surfaces they encounter. The sensor's operation relies on measuring the time taken for the signal to travel, enabling both object detection and distance measurement from the sensor to the objects.

Among the most prominent ultrasonic sensors for Arduino is the HC-SR04, prized for its widespread availability and affordability. With a measurement range spanning from 2 to 400 cm, this sensor remains operational even in the presence of electromagnetic radiation or solar energy, underscoring its

GPS Modules

GPS modules are employed for location determination by establishing connections with satellites and receiving location coordinates. Among the most prevalent GPS modules is the u-blox NEO-6m-001. It requires a duration to establish a connection with satellites. A cold start of this module typically takes

from 5 to 20 minutes, assuming it's under open sky or close to a window. Conversely, a hot start takes approximately 1 second. While this module aids in determining location, its accuracy may not be exceptionally high.

Fig. 1.9 NEO-6m-001

For enhanced accuracy in measurements, higher-priced models are necessary. Take, for instance, the NEO-8m module, boasting a data update frequency of up to 18 Hz (Single satellite) and 10 Hz (Multiple satellites). It has the capacity to connect to up to 72 satellites, a significant leap from the NEO-6m, which can only connect to 12. This increased satellite connectivity results in more precise location data. Moreover, the cold start time for the NEO-8m module is a mere 26 seconds, remarkably swift compared to other models.

Camera

Additionally, for analyzing surrounding objects, a conventional camera can be utilized. Through this camera, a computer equipped with a neural network can recognize various objects. Utilizing information about these objects and their positions within the frame, it can determine the appropriate direction to move or whether to remain stationary. Specifically designed for the Raspberry Pi microcomputer, the Pi High-Quality Camera (Figure 1.12) is available. This camera seamlessly connects to the Raspberry Pi and is easily configurable. It features a Sony IMX477R sensor matrix and requires lenses for operation with C

or CS type adapters. The module boasts a 12.3-megapixel sensor, with the Raspberry Pi High-Quality Camera offering a maximum resolution of 4056 x 3040 pixels (5K).

Fig. 1.10 Pi High Quality Camera without a lens

Accelerometer, magnetometer

Furthermore, to ascertain the device's orientation, both an accelerometer and a magnetometer are essential components.

An accelerometer gauges acceleration, which is the rate of change of velocity. By measuring dynamic acceleration's magnitude, it enables the determination of how rapidly and in which direction the device equipped with the accelerometer is moving.

On the other hand, a magnetometer measures the strength of the magnetic field. Modern electronic magnetometers, constructed using MEMS (Microelectromechanical Systems) technology, facilitate measurements along three perpendicular axes. They output projections of the magnetic field along these three axes within the magnetometer's coordinate system.

Through the accelerometer, the device's spatial position can be determined, while the magnetometer enables determination of the device's orientation.

Together, these sensors provide comprehensive information about the device's movement and direction.

Architecture of Intelligent Robots

Today, it is assumed that intelligent robots should consist of the following systems:

Actuators - these are manipulators, locomotion systems, and other devices through which a robot can interact with objects in its environment. Structurally, these are complex technical devices that include servo drives, mechatronic components, sensors, and control systems. Analogous to living organisms, these are the arms and legs of the robot.

1. Sensors - these are systems of technical vision, hearing, touch, distance sensors, locators, and other devices that allow gathering information from the surrounding world.
2. Control system - this is the brain of the robot, which should receive data from sensors and control actuators (effectors). This part of the robot is usually implemented through software. Components of the control system of an intelligent robot should include:
3. World model - reflects the state of the surrounding world for the robot in terms convenient for storage and processing. The world model serves the function of memorizing the state of objects in the world and their properties.
4. Recognition system - includes image recognition systems, speech recognition, etc. The task of the recognition system is identification, i.e., "recognizing" objects surrounding the robot and their positions in space. The operation of the recognition system components builds the world model.
5. Action planning system - performs a "virtual" transformation of the world model to obtain some action. At the same time, the feasibility of the set goal is usually checked. The result of action planning is the construction of plans, i.e., sequences of simple actions.

6. Action execution system - attempts to carry out planned actions by issuing commands to actuators and controlling the execution process. If the execution of a basic action turns out to be impossible, the entire process is interrupted, and new (or partially new) planning should be performed.
7. Goal management system - determines the sequence, i.e., the significance and order of achieving the set goals. Important properties of the goal management system include the ability to learn and adapt, i.e., the ability to generate action sequences for the set goal and adjust its behavior to changing environmental conditions to achieve the set goals.
8. Navigation system - designed to orient the robot in a three-dimensional world and to plan rational routes for the robot's movement.

1.5 Statement of problems of autonomous robots in their field

The problem of creating artificial intelligence

Creating a program for the autonomous operation of a robot requires writing new algorithms each time: the machine has no freedom of will even within the confines of executing assigned tasks; it merely follows the program code and explores options for the most rational execution of the given task. Once the task is completed, the robot loses its motive for functioning. It's impossible to give abstract instructions to a robot because there's no interpretation of the concept of "abstract instruction" as a "specific instruction." The solution is to create a system that will generate action (behavior or judgment) algorithms for itself for each specific case; this system should also accumulate information (experience) and analyze conclusions and generalizations from the accumulated information. Such a system, which generalizes, performs abstract constructions, not arithmetic calculations; it should be regarded as intelligence. Its artificial nature determines its characterization as "artificial intelligence." The autonomy of robots largely depends on the creation of artificial intelligence (AI).

Scientific developments in the field of artificial intelligence have reached an impasse. There is no universally accepted definition of what AI is. Instead, each new conference or symposium spawns new "concepts" with a common drawback. Very rarely do new ideas emerge, such as the concept of neural networks. Computers are getting better and better at playing chess. However, no computer learns or invents new chess algorithms on its own; they use schemes and methods provided by talented chess players or programmer-mathematicians, who are carriers of real intelligence. Computers simply explore options. Yet in the world, often what's needed is not just exploring options, or there are too many options, and the input data are unknown. Improvisation is needed, abstract thinking, which machines are not yet capable of and a way out of this situation has not yet been found either philosophically or mathematically.

The problem of power supply

One of the main problems in creating fully autonomous robots is providing them with power. One possible solution is to supply the robot with solar panels, but unfortunately, this method of obtaining energy may not always be available in the absence of a sufficiently bright light source. In countries close to the equator, this may not be a problem, but most developed countries are located in temperate zones, so the power supply problem for autonomous robots is urgent due to the current low efficiency of solar panels. A promising approach is to teach robots to generate energy independently, as animals do. However, in this case, there arises the problem of obtaining quality biological material for nutrition. There is an idea to teach the robot to feed on what can be easily found almost everywhere: for example, burnt leaves, dead insects, or human waste.

In 2004, Professor Chris Melhuish from the University of the West of England and his team developed the "EcoBot II" robot, which generated energy for its "life" from flies or pieces of rotten apples. A more advanced version was the "EcoBot III." In 2010, this robot was taught to dispose of the waste products of bacteria so that the bacteria supplying the operation of microbial fuel cells (MFC)

did not die from their own "impurities." Today, a new model of the robot "EcoBot-IV" is already being assembled, the fuel cells of which (as the developers assume) will be able to work for 20-30 years because there are no moving parts in MFC. Only bacteria function—meaning practically nothing can break[31].

Navigation Problem

The primary problem with all currently available mobile devices that move autonomously is navigation. Attempts to create autonomous means of transportation give rise to a series of problems collectively termed "navigation tasks." Navigation is the science of controlling the movement of a mobile robot (or, in other words, an autonomous object) in space. For successful navigation in space, the onboard system of the robot must be able to construct a route, control motion parameters (set the angle of wheel/rudder rotation and the speed of their rotation), correctly determine information about the surrounding world obtained from sensors, and constantly track its own coordinates. Typically, navigation tasks involve two subtasks that can be temporally divided: localization in space and path planning. Localization involves assessing the current position of the robot relative to certain known reference points in the environment, given in absolute coordinates. Planning involves searching for the shortest route possible and advancing to the destination point.

In goal-directed navigation, a minimum of three hierarchical levels of problem representation is distinguished: obstacle traversal, local navigation, and global route planning. Global planning algorithms involve information about the entire space to determine areas where movement is possible and then select the optimal path. Exact algorithmic solutions have been found for the planning task. However, precise algorithms have high computational complexity and also require accurate algebraic models of obstacles. Heuristic methods do not guarantee completeness of search and optimality even in global planning when all environmental data are available. However, heuristic global planning methods reduce the complexity of the task and sensitivity to errors in data in various ways.

By using genetic algorithms, an optimal route can be found, considering minimal travel time with various scenarios of real traffic conditions and different vehicle speeds.

Issues of Direct Danger from Machines

With the relentless development of robotics, robots and other automated systems are becoming increasingly intelligent and advanced. At the same time, they are being entrusted with more and more duties: driving cars, assisting with childcare, home security, and possibly even participating in military operations. The problem of complete trust in robots arises: there is no certainty that robots will never make decisions that harm humans[33].

First and foremost, the problem concerns combat robots. In modern armies, robots are primarily used for mine and bomb disposal, as well as for reconnaissance; however, they are increasingly being used as fully-fledged combat machines equipped with modern weapons. Currently, a living operator usually controls the combat robot, who is responsible for all actions of the entrusted device. However, if the machine is given the ability to autonomously select a target, the situation changes completely. Modern warfare must be conducted in such a way that those responsible for the deaths of civilians killed in the conflict can be identified—and the degree of their guilt determined. Since killings committed by autonomous robots cannot be assessed from this perspective, the concept of "responsibility" is fundamentally inapplicable to them. Therefore, the development of such machines should be banned on ethical grounds. Meanwhile, autonomous machines capable of killing already exist. For example, unmanned reconnaissance aircraft armed with missile weapons programmed to destroy targets that meet certain criteria. Such devices have been widely used by US troops during conflicts in the Middle East.

A direct consequence of the absence of human traits, and a potentially dangerous one, is the possibility of using robots in operations to suppress human

freedoms and rights. If the opportunity arises, robots will undoubtedly be used for illegal seizure and retention of power. International law does not guarantee protection from aggression by morally bankrupt individuals endowed with authority. Human rights activists consider "soulless machines" to be the ideal tool for suppressing rebellions, repression, etc., because (unlike most humans) a robot will not question orders and will execute everything it is instructed to do. The robot itself is not a sentient being capable of understanding the essence of punishment and reforming itself; punishing the military personnel who sent it on a mission is as futile as punishing the developers of its hardware and software.

Wendell Walla, an ethics expert from Yale University, and Colin Allen, a historian and philosopher of cognitive science at Indiana State University, speak of the inevitability of the widespread introduction of autonomous robots into our lives. As a partial solution to the danger posed by autonomous robotic systems to humans, they propose new laws of robotics, adopting which could reduce the danger from our high-tech creations[30]:

Locating robots in places where there is initially a low risk of developing hazardous situations: Before assigning tasks to robots, it is necessary to ensure that all computers and robots will never have to make decisions whose consequences cannot be predicted in advance. The workplace and tools used by robots should prevent even accidental harm to bystanders.

Do not give robots weapons: Although it is already too late to stop the construction of robots as weapons, it is not too late to limit their use to only certain types of weapons or restrict situations in which robots' weapons can be used.

Give robots the laws of robotics as in Asimov's works: Although Asimov's rules are poorly applicable due to the complexity of defining morality—good, evil, values, priorities, etc.—nevertheless, the rules can successfully restrict the behavior of robots and put them in very limited conditions.

Robotics laws should incorporate certain principles (rather than simple instructions): Adding motivation to robots, such as "doing the greatest good for the greatest number of people," is likely to be safer than setting simplified rules.

Teach robots like children (instead of loading them with a ready-made basic package of algorithms): Machines that learn and gradually "grow up" can develop an understanding of actions that humans consider right and wrong. Programming neuromorphic processors, promising bases for creating state-of-the-art autonomous robots, only allows for this approach (as opposed to algorithmic programming of instruction sets). The likelihood of success of this provision is quite promising, although this strategy requires several technological breakthroughs. Currently, there are almost no tools capable of teaching robots similar to humans.

Endow machines with emotions (artificial psyche):

Human abilities (such as empathy, emotionality, and the ability to read non-verbal signals of social communication) should give robots much greater abilities to interact with humans. Work in this direction has already begun, and it is planned that household robots in the future will possess such "emotional" properties. The likelihood of success of this approach is quite high. The development of emotionally sensitive robots will undoubtedly help implement the previous three laws of robotics. We use a lot of information to make choices and cooperate with other humans. Choice stems from our emotions as well as our ability to read gestures and intentions, representing events from another person's perspective.

1.6 Progress in commercial autonomous robots can be observed in various areas, including self-maintenance, environmental sensing, and task completion.

Self-maintenance is a critical aspect of robot autonomy, involving the ability to address their own needs independently. Today, many robots equipped with batteries can autonomously locate and connect to power sources, such as Sony's "Aibo" toy, which docks with charging stations. This capability relies on

proprioception, allowing robots to assess their internal status and take appropriate actions. For instance, a robot can recognize low battery levels and initiate charging by finding its docking station. Effective temperature control is also essential for managing heat exchange with the environment, particularly in challenging conditions or near humans.

Proprioceptive sensors play a crucial role in self-maintenance, including temperature sensors, Hall effect sensors, optical sensors, and touch sensors for object interaction. Additionally, environmental scanning, or exteroception, is vital for assessing surroundings and avoiding obstacles. Robots use a variety of sensors, such as those for the electromagnetic spectrum, sound, touch, chemicals, temperature, distance, and position estimation, to navigate and perform tasks effectively. For example, robotic lawnmowers adjust their operations based on grass growth rates, while cleaning robots analyze dirt levels to optimize cleaning durations.

Advancements in task completion involve mastering specific tasks autonomously. Compact vacuum robots like "iRobot" and "Electrolux" introduced in 2002 demonstrate progress in navigating homes using various sensors despite facing intelligence challenges. These robots efficiently cover large areas and tight spaces by devising work algorithms tailored to specific situations. Further progress involves robots handling complex conditional tasks, such as security robots identifying intrusions and responding based on intruder location and actions.

CONCLUSION

This chapter of my thesis focuses on the analysis of ground mobile objects. It covers general information about autonomous robots for ground movement, highlights automatic control, and thoroughly examines the navigation modules, their characteristics, and applications.

From this analysis, it is evident that ground mobile objects and robotic platforms are a significant area in robotics with substantial application potential. They are essential in industry, logistics, research, and the military sector, enhancing automation, improving efficiency, and ensuring safety across various fields.

Based on the current state and future development prospects of ground mobile objects, it is clear that they will continue to advance and improve. The integration of cutting-edge technologies such as artificial intelligence, machine learning, and sensor technology will result in more efficient, maneuverable, and autonomous robots. These advancements will expand their applications in different industries and play a crucial role in advancing industry, logistics, research, and other sectors of human activity.

CHAPTER 2: GENERAL STATEMENT OF THE PROBLEM OF PATH PLANNING BY AN AUTONOMOUS MOBILE ROBOT IN THE ENVIRONMENT

A crucial component of any navigation system is ensuring the destination is reached without getting lost or crashing into objects. Additionally, there may be other restrictions on a given route, such as speed limits or uncertain areas where routing is theoretically possible but not recommended.

Typically, the robot's path is autonomously planned using pre-existing data, without accounting for real-time obstacles. This approach can be effective, but only if the environment is perfectly known, unchanging, and the robot can follow the route precisely. In reality, however, conditions are far more complex.

Due to the limitations of autonomous planning, researchers have explored real-time planning (or online planning). This technique uses real-time data from the robot's sensors to navigate around unexpected obstacles as it moves through its environment.

2.1 Adaptive autonomous mobile control system

The autonomous mobile robot (AMR) system is designed to navigate along a predefined path in an unknown environment. Developing models that mimic human thinking is a significant scientific goal[34]. A critical component of modern robots is their sensor system, which gathers and processes data about the surrounding environment. This enables the robot to identify objects that may pose obstacles. The primary function of this system is to guide the AMR along the predetermined path. Human intelligence naturally excels at making decisions with incomplete and unclear information. A key requirement for AMRs is their ability to make decisions based on formalized data. The challenge is addressed using fuzzy logic and fuzzy knowledge bases. The fuzzy model is built upon expert knowledge of the process (system). This approach allows for the mathematical formalization and modeling of fuzzy information[35]. In this project, we will utilize a neuro-fuzzy system, which integrates fuzzy logic with a neural network[36]. The structure of this hybrid neuro-fuzzy network is illustrated in Figure 2.1.

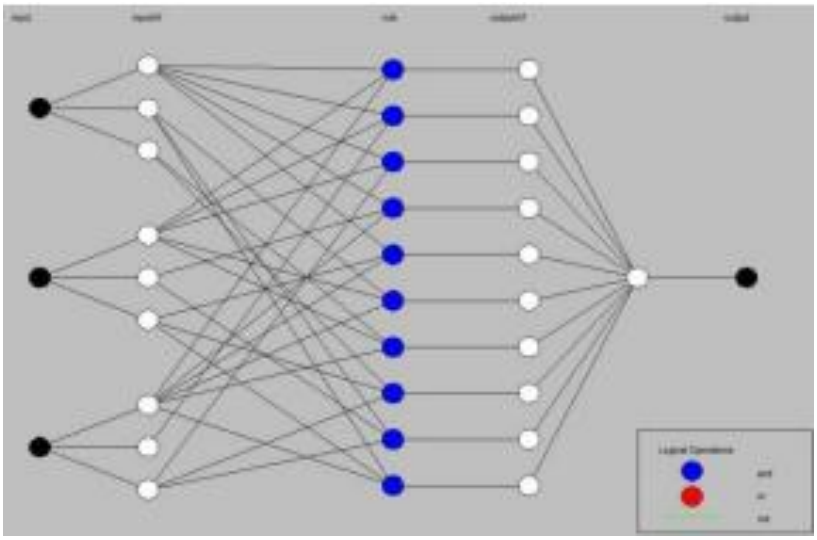

Fig. 2.1 The structure of a hybrid neuro-fuzzy network

Fuzzy hybrid neural networks are transparent in their logic, enabling them to absorb new knowledge effectively. They readily adjust to changes in the environment.

The system is structured in a neuro-like format, comprising 5 layers. These layers create a functional framework responsible for tasks such as:

- Assessing the degree of input signal membership in fuzzy sets,

- Determining rule validity,

- Normalizing rules,

- Generating the output fuzzy set and forming control decisions. The research was conducted using ANFIS models implementing Takagi-Sugeno rules.

The training process was conducted in two stages. Initially, the input signal values from the training dataset were fed into the control module. Based on this input, an output (control) action was generated. This signal propagated through the network in a forward direction, sequentially calculating the output values of the intermediate layers and the final output signal. The second stage involved backpropagation of the error. During this stage, the output response was compared with the reference value, and the comparison results were used to modify the weights. Subsequently, the weights of the connections and parameters of the network elements were adjusted, and the process moved to the next training sample. These iterations repeated until the artificial neuro-fuzzy network was properly trained. The training objective was to minimize the neural network error, which was determined using the least squares method. For a network with a single output (as in our case), the error is defined by the following ratio:

$$\Sigma = \frac{1}{2}\Sigma_{j=1}^{p}(y_i - b_i)^2$$

2.1 The Error

Where y_i denotes the output value of the neural network for each element of the output vector, while b_i stands for the desired output value of the neural network. Upon finishing the training process, the neural network generated a set of additional rules and established corresponding extra connections. The

configuration of the hybrid neuro-fuzzy network post-training. The training of the neuro-fuzzy network was carried out over 2500 cycles. The training error after this number of cycles for the specified sequence amounted to 4.3% (see Figure 2.2).

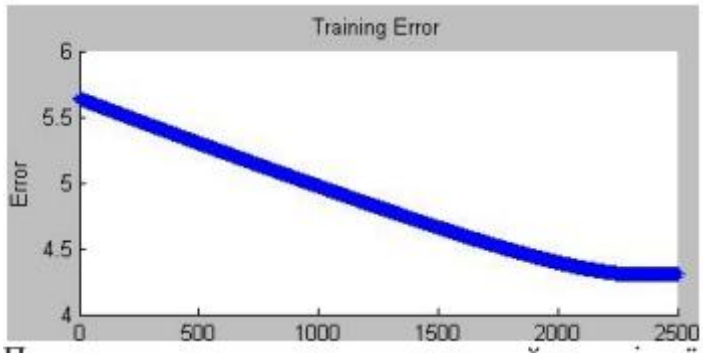

Fig. 2.2 An error based on the results of learning a neuro-fuzzy network

The research involved comparing two control methods for a robotic technical system: one employing fuzzy logic with the Mandmani algorithm, and the other utilizing the ANFIS model incorporating Takagi-Sugeno rules.

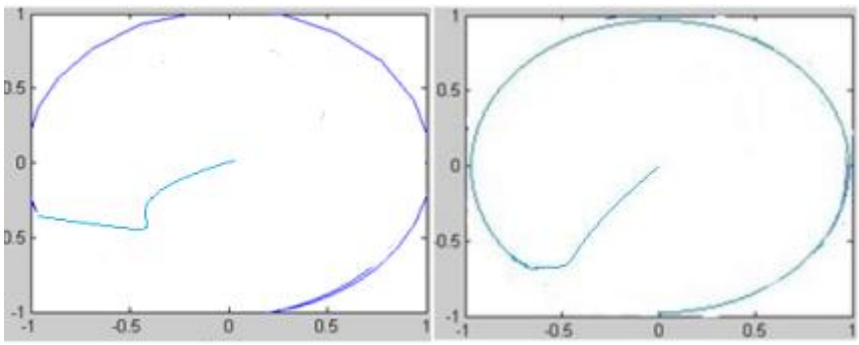

Fig. 2.3 The simulated motion trajectory of the autonomous model using fuzzy logic based on the Mandmani algorithm (first graph), the motion trajectory of the autonomous model using the ANFIS model where Takagi-Sugeno rules are implemented (second graph).

The application of the ANFIS model in constructing the control system displays superior qualitative features, evident in its minimal deviation from the desired movement trajectory. Compared to the control system implemented using fuzzy logic based on the Mandmani algorithm, there's a notable 40% reduction in the time taken for the model to reach the specified trajectory. The average deviation of the model's movement from the reference trajectory stands at 5%, which is half that of the classical algorithm using finite automata to compute

control actions. The outcome of this study is an enhanced solution for obstacle avoidance tasks (AMR). Further advancement involves integrating additional input values and rules to improve control precision. Consequently, this approach facilitates the development of a robot control system founded on a neuro-fuzzy (hybrid) network with adaptable properties.

2.2 Research and development of trajectory planning methods for autonomous mobile robots

The advancement of automation and research in autonomous navigation technology has significantly increased the use of mobile robots in various industrial applications. A crucial aspect of autonomous navigation is trajectory planning, which involves finding a feasible path for a mobile robot from a starting point to a destination within a given environment. This process considers optimization parameters such as path length, time, and trajectory smoothness. Trajectory planning is generally divided into two categories: global and local planning. Global planning requires complete information about the environment, including the positions of obstacles, while local planning deals with environments where such information is only partially known or unknown.

Finding paths that are both computationally feasible and optimal is a complex problem. A straightforward approach involves creating a connectivity graph where each node represents a working position and each link represents a pre-determined path of a specific length. Pathfinding can then be achieved by searching for the shortest path or using a table of precomputed optimal paths between all possible points. However, this method has limitations: it cannot handle arbitrary starting and ending points, and changing the list of working positions is time-consuming. This approach is effective in environments where robots follow repetitive routes with minimal changes.

Another method involves constructing a graph where nodes represent positions that can be connected by straight lines. This method, while simpler,

requires the graph to be updated with new nodes and links whenever new start and end points are specified, leading to a large and complex graph.A more recent approach divides the free space into convex polygons. In a convex shape, any two points can be connected by a straight line without leaving the shape, allowing the robot to navigate freely within these polygons. This method involves creating a connectivity graph where nodes represent convex polygons and arcs connect polygons with shared edges. However, this approach fails to fully utilize convexity and cannot dynamically reconfigure paths for optimality, particularly in large free areas.

R. Brooks proposed a method that combines the benefits of previous techniques by using generalized cones to represent free areas and considering the robot as a rectangular area rather than a point. This method avoids obstacles by moving the robot along the axis of free cones but sacrifices some optimality as it doesn't fully utilize convexity[52].

D. Kuan and colleagues improved Brooks's method by using cones for narrow spaces and non-overlapping convex polygons for larger areas. While effective in complex environments, this method still suffers from the drawbacks of non-overlapping areas and high graph complexity[53].

Building on these findings, a new path planning methodology is proposed that leverages convexity by identifying the largest rectangular free areas. A graph is created with nodes representing these areas, and intersecting shapes are connected as adjacent nodes. Path planning involves finding the best route from the source to the destination node through this graph. This method simplifies collision-free path planning by treating obstacles as enlarged and the robot as a point. Dynamic cost allocation ensures the optimal path is quickly determined. The proposed method effectively identifies near-straight paths when they exist, balancing optimality and computational efficiency.

2.3 Application of evolutionary algorithms for navigation problems

An evolutionary navigator is designed as a genetic algorithm system that integrates both autonomous and online planning modes, utilizing a simple, high-precision map and an efficient planning algorithm. The first component (autonomous planner) identifies the optimal global path from the start to the destination, while the second component (online planner) manages potential collisions or the avoidance of previously unknown objects by adjusting the global path with an auxiliary route. Notably, both parts of the evolutionary navigator use the same evolutionary algorithm but with different parameter settings.

Recently, other researchers have explored evolutionary computation methods for path planning. For instance, Davidor employs dynamic chromosome structures and variations in the crossover operator to optimize real-world processes, including path planning[13]. Other studies have introduced genetic algorithms for path planning and real-time multi-heuristic search strategies. These methods typically use maps made up of nodal points. Some researchers have applied classifier systems or genetic programming paradigms to tackle the path planning issue. The selected approach is unique because the evolutionary navigator operates across the entire free space without prior assumptions about permissible nodal points, combining both autonomous and online planning modes.

2.4 Selection of the genetic algorithm for the implementation of the evolutionary navigator

I'm choosing Zbigniew Michałewicz's evolutionary algorithm as the foundation for creating the evolutionary navigator (EN), detailed in the article "Path planning in a mobile robot environment"[17]. This approach stands out for its ability to operate across the entire free space without any predefined assumptions about permissible path nodes, seamlessly integrating both autonomous and online planning modes. Before delving into the algorithm's

intricacies, let's examine the map structure. To enable pathfinding in the continuous free space, we utilize vertex graphs to represent environmental objects. Currently, we simplify the environment to create a two-dimensional depiction of object areas. Consequently, while the robot is represented as a point, environmental objects can dynamically scale. We assume the mobile robot is equipped with sensors for environmental scanning. Known objects are described by an ordered list of their vertices, while online encounters with unknown obstacles are modeled as "wall" segments. Each segment, straight in nature, is delineated by its two endpoint vertices. This representation aligns with known objects, and in real-time, the robot updates the map with environmental data, though only partial information regarding unknown obstacles can be obtained through localized sensing. Ultimately, the environment is conceptualized as a rectangular area (the map).

Now, it's pivotal to define the paths that the EN should generate. A path comprises one or more straight-line segments, originating from the starting position, reaching the target, and potentially intersecting neighboring segments – referred to as nodes. A feasible path encompasses possible nodes, while an infeasible path contains at least one unattainable node.

Consider a path $p = (m1, m2, ..., mn)$ $(n \geq 2)$, where m1 and mn denote the initial and final nodes, respectively. A node mi $(i = 1, ..., n-1)$ is deemed infeasible if it's unattainable, unable to connect to the subsequent node mi+1 due to obstacles, or located within (or dangerously close to) an obstacle. We presume the initial and final nodes lie beyond obstacles and maintain a safe distance from them. However, it's noteworthy that the initial node isn't required to be feasible (if transitioning to the next node isn't possible), whereas the target node must invariably be feasible. Moreover, distinct paths may exhibit varying node counts.

Description of the evolutionary navigator algorithm

The evolutionary algorithm discussed here functions as an evolutionary navigator, integrating both autonomous and online planning modes with a straightforward high-precision map and an effective planning algorithm.

In the first segment of the algorithm (autonomous planner), it seeks optimal paths globally from the outset to the destination, while the second segment (online planner) manages potential collisions or previously unknown objects by substituting part of the original global path with an optimal subpath. It's worth noting that both components of the EN utilize the same evolutionary algorithm but with varying parameter values.

Initially, the EN reads the map and obtains the initial and target locations. Subsequently, the autonomous evolutionary algorithm (AEA) generates a nearly optimal global path, comprising partially straight paths consisting of permissible nodal points or nodes. Figure 1 demonstrates the operation of genetic algorithms in the evolutionary navigator.

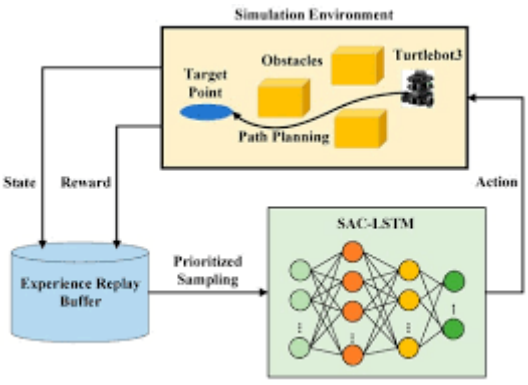

Fig. 2.4 Visualization of the evolutionary navigator

When the robot approaches an object in its vicinity, the AEA algorithm effectively expands the representation of this object on the map, shifting the local path away from it. This adjustment ensures that the path remains aligned with the original intention of the AEA algorithm, maintaining a straight trajectory.

Consequently, the robot can proceed along the current path to reach the next planned node successfully.

2.5 The use of machine learning tools to simulate the adversarial environment between robotic means and monitoring objects

Machine Learning (ML), a subset of artificial intelligence (AI) research, focuses on developing and studying statistical algorithms capable of learning from data and extrapolating to unseen data, thereby performing tasks without explicit instructions. Recently, generative artificial neural networks have surpassed many previous approaches in terms of productivity.

Machine learning methods have been applied across various domains, including large-scale language models, computer vision, speech recognition, email filtering, agriculture, and medicine, where developing algorithms for essential tasks would be prohibitively expensive. ML is widely known for its commercial applications, often referred to as "predictive analytics." Although not all machine learning relies on statistics, computational statistics serves as a crucial source of methods in this field.

The mathematical principles of ML provide techniques for mathematical optimization (mathematical programming). Data mining, a related field, focuses on exploratory data analysis through unsupervised learning. From a theoretical standpoint, a framework for describing machine learning offers probabilistically approximate correct learning.

The term "machine learning" was coined in 1959 by Arthur Samuel, an IBM employee and pioneer in computer gaming and artificial intelligence. During this era, the term "self-teaching computers" was also used interchangeably. While the earliest machine learning model was introduced in the 1950s by Arthur Samuel, who developed a program to calculate the odds of winning in checkers for each player, the history of machine learning spans decades of human curiosity and endeavor to understand human cognitive processes[54]. In 1949, Canadian

psychologist Donald Hebb published "The Organization of Behavior," proposing a theoretical neural structure formed by specific neuron interactions. Hebb's model laid the groundwork for how AI and machine learning algorithms function at the node level, or artificial neurons, which computers utilize to transmit data[55]. Other researchers exploring human cognitive systems also contributed to modern machine learning technologies, including logicians Walter Pitts and Warren McCulloch, who proposed early mathematical models of neural networks to develop algorithms simulating human thought processes[55].

Artificial Intelligence

As a scientific domain, machine learning has emerged from the quest for artificial intelligence (AI). During AI's early days as an academic field, certain researchers were intrigued by the idea of machines learning from data. They experimented with various symbolic techniques and what later became known as "neural networks," primarily perceptrons and similar models, which were essentially variations of generalized linear models from statistics. Additionally, probability-based reasoning was explored, particularly in the realm of automated medical diagnosis.

However, a shift towards a knowledge-based logical approach created a divide between AI and machine learning. Probabilistic systems encountered both theoretical and practical challenges in data gathering and presentation. By around 1980, expert systems took precedence in the AI domain, while statistics fell out of favor. Although work on symbolic or knowledge-based learning persisted within AI, leading to the development of inductive logic programming, the statistical orientation of research now lay beyond the traditional AI realm, focusing instead on tasks like pattern recognition and information retrieval. Concurrently, research on neural networks was sidelined by both AI and computer science. This line of inquiry, known as "connectionism," continued outside the AI and computer science realms, spearheaded by researchers such as Hopfield, Rumelhart, and Hinton.

Their significant breakthrough came in the mid-1980s with the reemergence of backpropagation[56].

Machine learning (ML Fig. 2.5), restructured and acknowledged as an independent field, began to thrive in the 1990s. Its focus shifted from achieving AI to tackling practical problems. Departing from symbolic AI approaches, ML embraced methods and models borrowed from statistics, fuzzy logic, and probability theory.

Fig. 2.5 Machine learning as a sub-branch of AI

Data extraction

Machine learning and data mining often utilize similar methodologies and significantly overlap, yet they have distinct focuses. While machine learning emphasizes making predictions based on known features learned from training data, data mining is centered around uncovering previously unidentified characteristics within data, constituting a step in knowledge discovery within database analysis. Data mining employs numerous machine learning techniques, albeit with different objectives. Conversely, machine learning also incorporates data mining techniques, either as "unsupervised learning" or as a preprocessing phase to enhance learning accuracy. Much of the confusion between these two research communities arises from their underlying assumptions: in machine learning, performance is typically assessed based on the ability to replicate

established knowledge, whereas in knowledge discovery and data mining, the primary goal is to unveil previously undisclosed insights. While unsupervised methods may struggle against supervised methods when evaluated against established knowledge, in typical knowledge discovery tasks, the use of supervised methods is impractical due to the absence of training data.

Additionally, machine learning is closely linked to optimization, as many learning objectives involve minimizing a loss function over a training dataset. These loss functions quantify the disparity between the model's predictions and the actual instances of the task. For example, in classification tasks, models are trained to accurately predict the pre-assigned labels of example instances.

The approaches in machine learning are traditionally divided into three broad categories, which correspond to learning paradigms and depend on the type of "signal" or "feedback" available to the learning system:

- Supervised Learning: In this approach, the computer learns a general rule mapping inputs to outputs by being provided with samples of inputs and their corresponding desired outputs, given by a "teacher."

- Unsupervised Learning: Algorithms in this category autonomously discover patterns or structure within the input data without being provided with explicit labels. Unsupervised learning can either be the main objective itself, such as identifying hidden patterns, or it can serve as a means to an end, like learning features.

- Reinforcement Learning: In this paradigm, a computer program interacts with a dynamic environment, aiming to achieve a specific goal, such as controlling a vehicle or playing a game. As the program navigates through tasks, it receives feedback, similar to rewards, which it seeks to maximize.

While each algorithm has its own set of advantages and limitations, no single algorithm is universally effective for all tasks.

Supervised Learning

Supervised learning algorithms develop a mathematical model based on a dataset containing both inputs and desired outputs. These datasets, termed training data, comprise a series of training examples. Each example includes one or more inputs along with an expected output, also referred to as a supervisory signal. In the mathematical representation, each training example is depicted by an array or vector, sometimes denoted as a feature vector, while the training data is represented by a matrix. Through iterative optimization of the objective function, supervised learning algorithms acquire knowledge of a function capable of predicting outputs for new inputs. This optimal function enables the algorithm to accurately infer outputs for inputs not included in the training data. An algorithm that progressively enhances the precision of its outputs or predictions is considered to have mastered this task shown in Fig. 2.6.

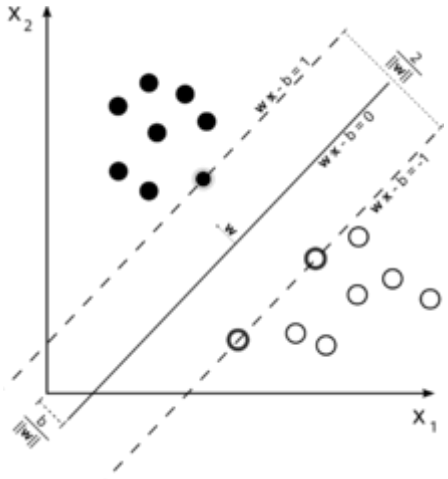

Fig. 2.6 A supervised learning model that divides the data into regions separated by a linear boundary.

Supervised learning encompasses active learning, classification, and regression algorithms. Classification methods are utilized when outputs are confined to a specific range of values, whereas regression approaches are

employed when outputs can span a continuum of numerical values within a defined range. For example, in the context of an email filtering system, a classification algorithm would analyze incoming emails, assigning them to specific folders based on their content.

Similarity learning, a subset of supervised machine learning, is closely linked to regression and classification but focuses on deriving insights from examples using a similarity function that measures the likeness or correlation between two objects. It finds practical application in various domains, including ranking systems, recommendation engines, visual identity tracking, facial recognition, and speaker authentication.

Unsupervised learning

Unsupervised learning algorithms identify patterns in data that haven't been tagged, classified, or categorized. Instead of reacting to feedback, they identify commonalities in the data and respond to the presence or absence of such commonalities in each new data point. Primary applications of unsupervised machine learning include clustering, reducing dimensionality, and estimating density. Unsupervised learning algorithms have also improved the process of identifying large haplotypes of a specific gene from the pangenome based on indels.

Cluster analysis involves dividing a set of observations into subsets, called clusters, so that observations within one cluster are similar based on one or more predetermined criteria, while observations from different clusters are dissimilar. Different clustering methodologies make various assumptions about the data structure, often defined by a measure of similarity and evaluated based on factors such as internal compactness or the similarity of cluster members, and separation, which denotes the difference between clusters. Other methods rely on estimated density and graph connectivity.

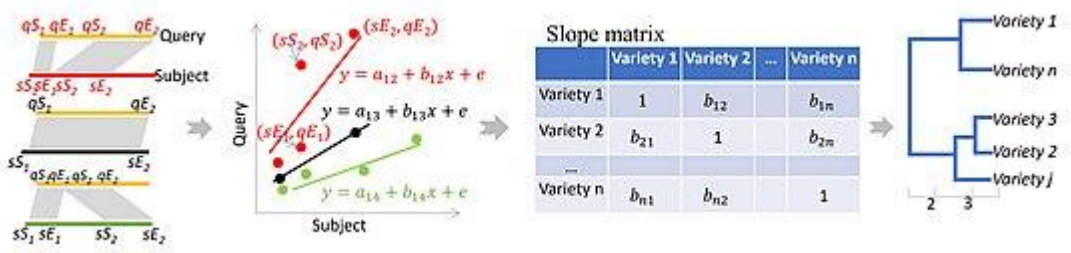

Fig. 2.7. Clustering with large indel permutation slopes turns the alignment image into a regression learning problem.

Semi-supervised learning

Semi-supervised learning falls in the middle ground between unsupervised learning, which lacks labeled training data entirely, and supervised learning, which relies on fully labeled training data. In this approach, some training examples are without labels, yet many machine learning experts have discovered that incorporating unlabeled data alongside a small portion of labeled data can notably enhance the accuracy of the learning process.

In weakly supervised learning, training labels are either noisy, scarce, or imprecise. Nonetheless, acquiring these labels is often more economical, leading to more cost-effective training datasets.

Reinforcement Learning

Reinforcement learning involves how software agents should make decisions in an environment to maximize cumulative rewards. This field is studied across various disciplines like game theory, control theory, operations research, information theory, model-based optimization, multi-agent systems, swarm intelligence, statistics, and genetic algorithms due to its broad applicability. In reinforcement learning, the environment is typically represented as a Markov decision process (MDP), and many algorithms in this field use dynamic programming techniques. These algorithms are utilized when precise models are not available. Reinforcement learning methods find application in autonomous vehicles and teaching agents to compete in games against human opponents.

2.6 Models

Engaging in machine learning often entails constructing a model trained on a set of training data, which can then analyze further data to make predictions. Various models have been explored and utilized in machine learning systems through research and application.

Artificial neural networks

Artificial neural networks (ANNs), also known as connectionist systems, are computational systems that draw inspiration from biological neural networks found in animal brains. These systems "learn" to perform tasks by observing examples, typically without being explicitly programmed with task-specific rules.

ANNs rely on interconnected nodes called "artificial neurons," which mimic neurons in the biological brain. Each connection, similar to synapses in biological neurons, can transmit information or "signals" from one artificial neuron to another. Upon receiving a signal, an artificial neuron can process it and then relay it to other connected artificial neurons. In ANNs, the signal passing through connections is usually represented by real numbers, and the output of each artificial neuron is determined by a nonlinear function applied to the sum of its inputs. Connections between artificial neurons are referred to as "edges," and both neurons and edges typically have weights that adjust during training. These weights modulate the strength of signals transmitted through connections. Additionally, artificial neurons may have thresholds, ensuring that signals are sent only when the cumulative signal surpasses a certain threshold value. Typically, artificial neurons are organized into layers, with each layer performing specific transformations on its inputs. Signals propagate from the input layer through hidden layers to the output layer, often undergoing multiple transformations along the way.

Initially, the primary objective of ANNs was to solve problems in a manner similar to the human brain. However, the focus has shifted towards achieving

specific task goals, leading to deviations from biological processes. ANNs have been applied to various tasks, including computer vision, speech recognition, machine translation, social network analysis, gaming, and medical diagnostics.

Deep learning, a subset of ANNs, involves the incorporation of multiple hidden layers into the network architecture. This approach aims to emulate the process by which the human brain processes sensory inputs such as light and sound into visual and auditory perception. Successful applications of deep learning include tasks such as computer vision and speech recognition.(Fig. 2.8.)

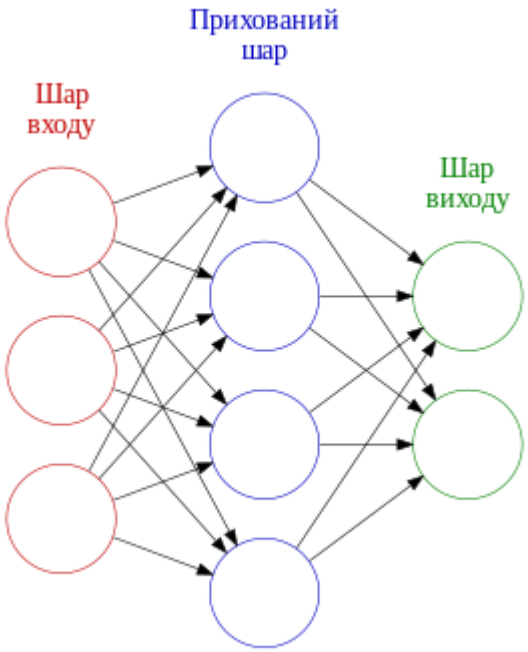

Fig. 2.8. Artificial neural network

Resistance vector machines

Support Vector Machines (SVM), also referred to as support-vector networks and the support-vector method, constitute a collection of related supervised learning techniques employed for classification and regression tasks. With a given set of training instances, each categorized into one of two groups, the SVM training process constructs a model to predict whether a new instance belongs to either category. The SVM training algorithm serves as an exceptional binary linear classifier, although approaches like Platt scaling are available for

utilizing SVMs in probabilistic classification scenarios. Furthermore, apart from conducting linear classification, SVMs can proficiently handle nonlinear classification through a method known as the kernel trick, which indirectly maps their inputs to high-dimensional feature spaces.

Regression analysis

Regression analysis encompasses a wide array of statistical techniques used to assess the relationship between input variables and associated features shown in Fig. 2.9 . Its most prevalent form is linear regression, which involves fitting a single line to the data based on a mathematical criterion like ordinary least squares. This method is often extended through regularization techniques to mitigate issues like overfitting, as seen in ridge regression. When dealing with nonlinear problems, common models include polynomial regression (e.g., for trend line fitting in Microsoft Excel), logistic regression (frequently used in statistical classification), or kernel regression, which introduces nonlinearity through a kernel trick to map input variables into a higher-dimensional space implicitly.

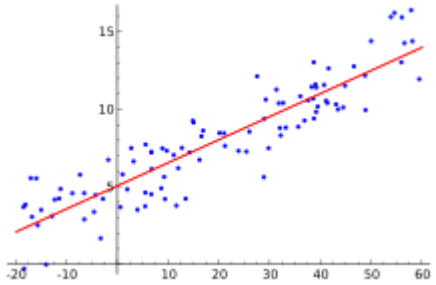

Fig. 2.9. Illustration of linear regression on a data set

Gaussian processes

A Gaussian process functions as a stochastic process where each finite collection of random variables within it adheres to a multivariate normal distribution. It relies on a predetermined covariance function, also known as a kernel, which dictates the relationship between pairs of points based on their positions.

Given a specified set of observed points or input-output pairs, the distribution of the output for a new, unseen point based on its input data can be directly calculated by considering the observed points and the covariance between these points and the new one.

Gaussian processes serve as popular surrogate models in Bayesian optimization, utilized for optimizing hyperparameters(Fig. 2.10.).

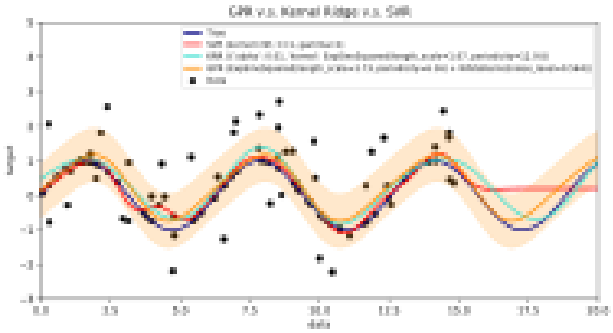

Fig. 2.10. An example of Gaussian regression (prediction) compared to other regression models

2.7 Creation of a mathematical model

Developing a mathematical model of the environment for utilization in reinforcement learning encompasses several stages.

1. Objective and Task Specification: This phase entails defining the objectives and tasks, such as tracking mobile entities, navigating towards objects, avoiding obstacles, and identifying objects.

2. Environment Modeling:

- Spatial Representation: This involves constructing a map of the area that may include obstacles, visibility zones, and various surface types.

- Dynamic Components:Identifying the parameters of moving entities, such as velocity, direction, and behavior.

- Sensor Models: Simulating the robots' sensors used for object detection and navigation, such as lidars, cameras, and ultrasonic sensors.

3. Definition of States, Actions, and Rewards:

- States: Establishing the possible states observable by the robots within the environment.

- Actions: Determining a set of actions that the robots can execute, such as forward movement, turning, or object grasping.

- Rewards and Penalties: Establishing a reward system to incentivize desired robot actions and penalize errors or ineffective actions.

Additionally, we will provide a verbal articulation of the system's goals and tasks:

Goals of Robotic Systems: The primary aim of robotic systems is to efficiently detect and track moving objects. This encompasses minimizing the time and distance required to reach a sufficient proximity for object identification. This includes the capability of robotic systems to traverse terrain affecting speed and maneuverability, and employing sensors for precise object localization.

Tasks of Robotic Systems:

1. Navigating towards objects while considering terrain and obstacles.

2. Avoiding obstacles and other robotic systems during movement.

3. Identifying objects as robotic systems approach a sufficient distance.

Goals of Moving Objects: Moving objects seek to evade close proximity to robotic systems by adjusting their speed, direction, and utilizing terrain for concealment.

From a mathematical formalization perspective, we will introduce the following notations and functions:

Spatial Modeling involves constructing a virtual map that accurately represents real-world conditions. This may involve:

- Data Collection: Utilizing available mapping services (e.g., Google Maps, OpenStreetMap) to acquire a foundational map of the area.

- Data Augmentation: Integrating data from stationary sensors, including cameras and motion detectors, to incorporate information regarding obstacles, visibility zones, and other pertinent features.

- Map Annotation: Manually or automatically appending metadata to the map, such as surface types, object heights, and additional physical attributes.

Dynamic Elements: The dynamic components of the environment encompass moving entities, which can be characterized by parameters such as:

- Velocity (v): A scalar or vector quantity denoting the rate at which the entity is moving through space.

- Direction (θ): A vector indicating the entity's direction of movement.

- Behavior: A set of rules or algorithms dictating the entity's response to environmental changes.

Following these steps, we will undertake a mathematical formalization of the environment for robotic systems and surveillance entities.

The environment states S comprise all potential observations accessible to the robot. These encompass:

- the robot's location p_r and the positions of mobile objects p_o ;

- the velocity of the robot v_r and that of mobile objects v_o ;

- the status of obstacles M and other environmental components;

- data regarding the recent positions of mobile objects.

Actions denote the array of feasible maneuvers the robot can execute, like advancing, turning, or seizing an object.

The reward function is devised based on diminishing the gap between the robot and mobile objects, relative to the "optimal" distance that would hypothetically exist if both entities were converging at maximum velocity unimpeded.

$$R_t^{(r)} = k(d_{t-1}^* - d_t);$$
$$R_t^{(o)} = -R_t^{(r)},$$

Where d_t is the actual distance between P3 and PO; d_{t-1}-"ideal" distance k-positive scaling factor. The simulation ends then

1) P3 reaches RO ($d \leq \in$)

2) PO goes beyond the defined environment ($p_o \notin$ Environment Bounds)

3) P3 falls into an area where it cannot function ($M(p_r)$= «untraversab»)

As a result, this model provides a clear framework for establishing a precise reward system, motivating the robot to efficiently pursue mobile objects (MO) while encouraging these objects to evade pursuit. It also outlines specific conditions for terminating the simulation. Now, by utilizing CNN, we will determine the coefficients k that denote how the environment influences the parameters of both the robot (RZ) and the mobile objects (MO).

1. Defining objectives and tasks: This involves setting clear goals for both RZ and MO and defining the tasks necessary for RZ to achieve these objectives.

2. Environment modeling: Creating a visual representation of the environment, including obstacles, visibility zones, and various surface types. It also entails determining dynamic elements such as the motion parameters of MO.

3. Defining states, actions, and rewards: Describing the possible states of the environment S, which include the positions and velocities of RZ and MO, as well as the status of obstacles M . This step also involves establishing the actions A that

RZ can undertake and designing a reward system that motivates desired actions while penalizing errors.

4. Formulating state transition functions: Developing equations that depict how the environment transitions between different states based on the actions of RZ.

5. Defining the reward function: Establishing a function that rewards behaviors leading to the reduction of distance between RZ and MO, among other criteria.

6. Setting termination criteria: Identifying conditions under which the simulation concludes, such as MO reaching a specific goal, MO leaving the environment boundaries, or RZ experiencing malfunctions.

7. Integrating the model into RL: Implementing the environment model into reinforcement learning (RL) algorithms and leveraging acquired data to optimize the behavior strategies of both RZ and MO. This method enables the creation of a detailed and adaptable environment model suitable for training RZ under diverse conditions and enhancing their real-world performance.

CONCLUSION:

Attempts to develop autonomous moving vehicles or mobile agents present a range of challenges known collectively as "navigation tasks." These primarily pertain to movement on terrestrial surfaces, though successes in this domain have found applications in underwater vehicle navigation and missile guidance, as well as in the realm of computer game development.

Global planning algorithms utilize comprehensive spatial information to identify viable movement paths and subsequently determine the most optimal route. In contrast, heuristic planning methods address task complexity and data error sensitivity using various approaches. Therefore, an evolutionary navigator algorithm was chosen to build a versatile navigation system for autonomous robots.

The adoption of this algorithm enables the consideration of numerous behavioral variations of robots and environmental factors during path planning. However, a primary concern with this algorithm selection remains the identification of crucial points. Resolving this issue serves as the cornerstone for further research.

In our study, we devised a comprehensive methodology for environment modeling tailored to the task of locating and tracking moving objects using robotic means within the framework of reinforcement learning. This methodology encompasses goal and task definition, environment modeling, state and action delineation, reward formulation, state transition function definition, reward function establishment, modeling termination criteria, convolutional neural network utilization for environment analysis, and model integration into reinforcement learning algorithms. The distinctive feature of our developed methodology lies in its adaptability to diverse environmental conditions and dynamic changes occurring within it.

CHAPTER 3: DEVELOPMENT OF PROGRAM

One of the primary challenges for these systems is their ability to effectively navigate unknown environments and avoid obstacles. Managing ground mobile objects automatically in obstacle avoidance mode is a crucial task in this context.Planning optimal trajectories for moving objects is complicated by the potential obstacles along the route. This requires the development of algorithms and systems that enable efficient obstacle avoidance and safe route completion.

In this regard, a robotic platform serves as a research and experimental subject. Leveraging modern sensors, powerful computational resources, and advanced control algorithms, it provides opportunities to explore and implement obstacle avoidance methods in various conditions. In this study, we will thoroughly explore the concept of such a robotic platform, discuss the critical aspects of its operation, including sensor equipment and control algorithms, and analyze the outcomes of our experiments. This approach will help us better understand the challenges of automatic control of moving objects in obstacle avoidance mode and identify potential directions for further research and development in this field.

A robotic platform can feature the following characteristics:

- Mobility: The platform may be designed for walking or wheeled movement, depending on specific needs and the intended environment.

- Sensors: It is equipped with a variety of sensors, such as laser rangefinders (LiDAR), cameras, ultrasonic sensors, and distance sensors, to collect data about the surrounding environment.

- Data Processing: The platform includes a powerful computational module to process sensor data and make decisions regarding obstacle avoidance routes.

- Control Mechanisms: It may utilize electronic servos or hydraulic systems to enable movement and obstacle avoidance.

- Control: The platform can be controlled by software that uses artificial intelligence and machine learning algorithms for autonomous route planning and obstacle avoidance.

This robotic platform can serve as a foundational tool for research and development in automatic control systems capable of avoiding obstacles.

The subsequent sections of this work will explore the theoretical foundations essential for developing control systems within the MATLAB environment. We will delve into the use of MATLAB for mathematical modeling and the application of Simulink for visual modeling and simulation of control systems.

Furthermore, we will analyze control methods and algorithms that can be implemented in MATLAB and Simulink to achieve efficient obstacle avoidance.

This theoretical exploration will provide insights into harnessing the power of MATLAB and Simulink for the development and testing of automatic control systems, including process modeling, result analysis, and system optimization.

3.1 Program for Robot - SMR (Small Mobile Robot)

The program for the robot, known as SMR (Small Mobile Robot), was developed and is available at DTU Electrical Engineering to test the solutions proposed in this project. This program is already utilized in numerous research projects and robotics courses. SMR is a general-purpose differential robot operating on a Linux platform, capable of being equipped with various sensors such as a line sensor, IR proximity sensors, wheel encoders, cameras, and a laser scanner.

SMR is fitted with a HOKUYO URG-04LX laser scanner, which can measure distances up to 4 meters and has a 240° coverage with a resolution of approximately 0.36º.

Two software interfaces are used for testing: smrdemo and the laser server. The smrdemo program implements the SMR-Control Language, providing a set of high-level commands for robot movement and commands for reading odometer data. The laser server offers commands for obtaining information from the laser scanner and setting scanning parameters.

Figure 3.1 illustrates the architecture used for the interaction between the SMR robot and MATLAB. Two connections are established between MATLAB and the respective servers. The smrdemo program receives commands necessary for controlling the robot and returns current odometer data when needed. The laser server provides new readings from the scanner as required and can adjust scanning parameters, including scan width.

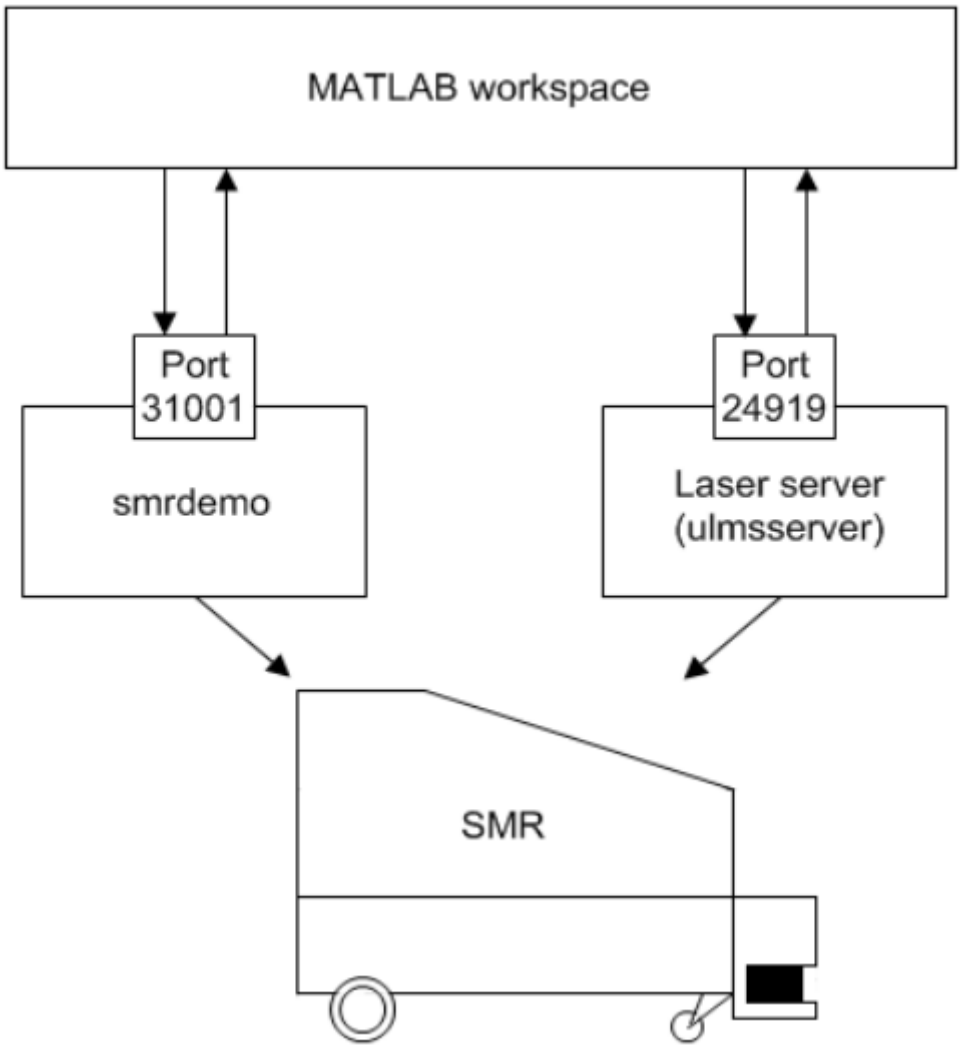

Fig. 3.1. The architecture used for the interaction between the SMR robot and MATLAB

3.2 Matlab

MATLAB is a software package for numerical analysis and also a programming language used within the package. Developed by The MathWorks, it is an effective tool for working with mathematical matrices, plotting functions, handling algorithms, and creating user interfaces with programs in other languages. While MATLAB specializes in numerical computation, it can work with Maple software through special toolboxes, making it a complete system for algebraic tasks. With over a million users in industry and academia, MATLAB is widely adopted. The basic commercial version without toolboxes costs around $2000, while the educational version with minimal toolboxes is about $100.

MATLAB offers a wide range of functions for data analysis that cover nearly all areas of mathematics, including:

- Matrices and Linear Algebra: Matrix algebra, linear equations, eigenvalues and eigenvectors, singularities, matrix factorization, and more.

- Polynomials and Interpolation: Polynomial roots, operations and differentiation of polynomials, curve interpolation, and extrapolation.

- Mathematical Statistics and Data Analysis: Statistical functions, statistical regression, digital filtering, fast Fourier transform, and others.

- Data Processing: Special functions including graph plotting, optimization, root finding, numerical integration, and more.

- Differential Equations: Solving differential and differential-algebraic equations, delay differential equations, constrained equations, partial differential equations, and more.

- Sparse Matrices: A special data class in MATLAB used for specialized applications.

- Integer Arithmetic: Performing integer arithmetic operations in MATLAB.

MATLAB is one of the oldest, most thoroughly developed, and time-tested systems for automating mathematical calculations, heavily utilizing matrix operations. This is reflected in its name, MATrix LABoratory. However, the programming language's syntax is so well-designed that users not focused on matrix calculations hardly notice this specialization. Matrices are fundamental in complex mathematical calculations, such as solving linear algebra problems and modeling static and dynamic systems. They form the basis for automatically constructing and solving state equations of dynamic systems. An example of this is MATLAB's extension, Simulink. This significantly enhances interest in MATLAB, which has incorporated the best advancements in fast matrix problem solving. Today, MATLAB has evolved far beyond a specialized matrix system to

become one of the most powerful universal integrated computing systems. The term "integrated" indicates that MATLAB combines a user-friendly interface, an expression and text comment editor, a calculator, and a graphical software processor.

The latest version includes advanced data types such as multidimensional arrays, cell arrays, structure arrays, Java arrays, and sparse matrices, opening up new possibilities for creating and debugging algorithms for matrix and parallel computations and large databases. Overall, MATLAB is a unique collection of implementations of modern numerical methods in computer mathematics, developed over the past thirty years. It incorporates the experience, rules, and methods of mathematical computation accumulated over millennia. This is combined with powerful tools for graphical visualization and even animated graphics. With extensive documentation, MATLAB can be considered a comprehensive multi-volume electronic reference for mathematical software, from personal computers to supercomputers. Unfortunately, the complete documentation is currently available only in English and partially in Japanese.

One of the challenges in modern science is developing and implementing research methods for the functioning of complex systems. Complex systems include technological, production, and energy complexes, automation control systems, and other entities. Modeling is one of the most powerful tools for studying such systems today. Modeling is a widely used method for studying various processes and phenomena. A model of an original object represents the object in a form different from its real existence. In engineering practice, a model is usually created to:

1. conducting experiments on the model that are impossible or challenging to perform on the actual object (providing the opportunity to acquire new insights about the object);

2. speeding up, reducing costs, simplifying, and otherwise improving the design process by working with a model, which is a simpler representation than the original object.

Today, various types of models and numerous modeling methods are well-known and widely applied in scientific research and engineering practice. Based on the degree of abstraction (the extent of difference from the real object), the following types of models can be identified:

1. physical (prototype) models (reproduce the studied process while maintaining its physical nature and serve as tools for physical modeling);

2. analog models (substitute one object with another that has similar properties);

3. mathematical models (abstract models that exist in the form of specialized mathematical constructs). Mathematical modeling involves studying various processes by examining phenomena with different physical content but described by the same mathematical relationships.

Among the numerous visual modeling packages, Matlab holds a unique position. Initially aimed at research projects, in recent years it has become a practical tool for engineers, students, managers, physicists, and communication specialists.

One of the primary reasons for Matlab's widespread use is the extensive range of tools it offers for solving diverse tasks across various fields of human activity. Within these tools, the Simulink subsystem stands out as particularly significant.

3.3 Simulink

Simulink is an interactive platform used to model and analyze diverse dynamic systems through block diagrams. It offers an interactive modeling environment where model behavior and results are displayed in real-time, allowing

for parameter adjustments even during execution. Simulink enables the creation of customized blocks and block libraries accessible from Matlab, Fortran, or C programs, facilitating integration with existing validated models. From version 3.0 onwards, specialized tools have been integrated into Simulink, enhancing its capabilities significantly:

- Stateflow: A graphical tool for designing complex control systems, allowing users to model event-driven behavior based on finite state machine theory.

- Stateflow Coder: Generates C code specifically for Stateflow diagrams, enabling seamless integration with Simulink models.

- Real-Time Workshop: Automatically generates C code from Simulink models, facilitating the development of code for discrete, continuous, and hybrid systems.

- DSP Blockset: Provides Simulink block libraries for creating and simulating digital signal processing systems.

- Nonlinear Control Design Blockset: Offers an interactive approach to automated control system design.

- Fixed-Point Blockset: Includes Simulink block libraries for modeling control systems and dynamic filters using fixed-point arithmetic.

- Simulink Report Generator: Allows for the creation of reports from Simulink and Stateflow models in various formats such as HTML, RTF, XML, and SGML.

3.4 Testing programs

This section focuses on testing the simulated solutions and strategies proposed earlier, alongside real-world testing conditions. Each scenario undergoes testing against various obstacle forms, with the most notable changes highlighted

in the results. The simulation of robot operations is conducted within the MATLAB workspace, with subsequent visualization of the outcomes. Conversely, real-world testing employs the SMR robot platform, as outlined in Appendix E. The subsection is organized into three primary sections, detailing the evaluation of each scenario under both simulated and real conditions, followed by discussions on the respective results. Finally, conclusions are drawn based on the findings.

To start, we'll write code within the programming environment to plot a designated route. We'll input the data defining our robot's path from start to end. Utilizing the specified points, we can generate the resulting visualization(Fig. 3.2.).

Program code for rabotic follow waypoints:

```
path = [8.00 6.25;
1.25 1.75;
5.25 8.25;
7.25 2.75;
12.75 4.75;
15.00 18.00];
robotCurrentLocation1 = path(1,:);
robotGoal1 = path(end,:);
initialOrientation1 = 0;
figure(1)
plot(path(:,1), path(:,2),'m-d','LineWidth',2)
title('Robot follow waypoints')
xlabel('X,m')
ylabel('Y,m')
xlim([0 20]); ylim([0 20]);grid
%Trajectory length computation
m=length(path(:,1));
s=0;
for i=1:m-1
s=s+sqrt((path(i,1)-path(i+1,1))^2+(path(i,2)-path(i+1,2))^2);
end
```

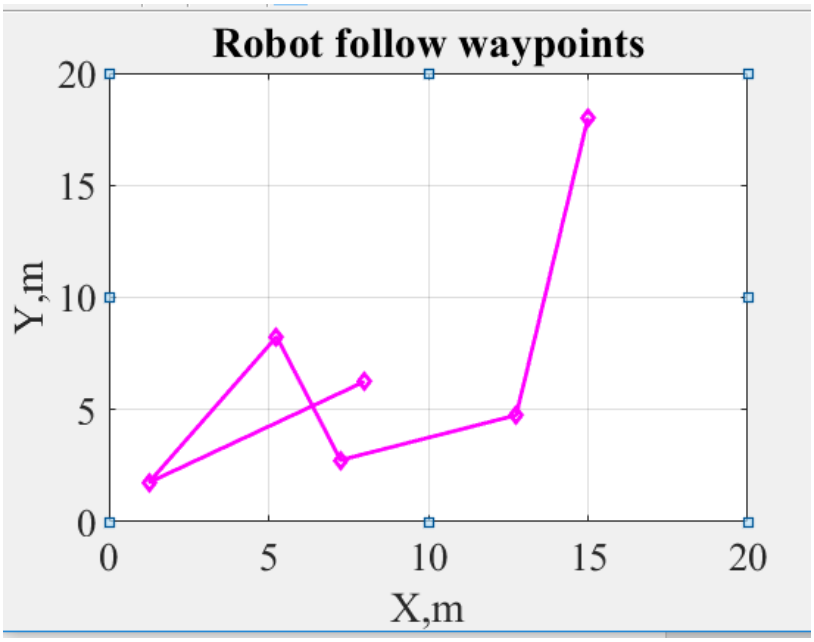

Fig 3.2 Robot follow waypoints, S=13.44 meters

3.5 Corridor with obstacles

The corridor has been filled with multiple obstacles, loaded through a specific map code. The modeling environment is configured appropriately. We are simulating conditions for the robot, initially opting for automatic movement, covering a distance of 16 meters from the starting point to the endpoint.

In the illustration(Fig. 3.3.), we notice that the robot has covered the distance but has come dangerously close to the obstacles, which could potentially lead to it becoming stuck. Hence, we will define the points in a way that allows it to travel a greater distance of 26 meters, ensuring a more precise arrival at the final destination. Automatic movement has been set to observe the robot's motion, enabling us to adjust its course as needed.

Program code:

```
filePath
=fullfile(fileparts(which('PathPlanningExample')),'data','exampleMaps.mat');
```

```
load(filePath);
   map = robotics.BinaryOccupancyGrid(simpleMap, 2);
   show(map);
   robotRadius = 0.2;

   path=[2.00 2.00;
   2.00 4.00;
   4.00 4.00;
   4.00 9.00;
   8.00 9.00;
   8.00 2.00;
   12.00 2.00]; %(path developed in 2.3)
   startLocation = [2 2];
   endLocation = [12 2];
   prm = robotics.PRM;
   prm.Map = map;
   prm.NumNodes = 100;
   path1 = findpath(prm, startLocation, endLocation);
   show(prm, 'Map', 'on', 'Roadmap', 'off');hold on
   robotCurrentLocation = path(1,:);
   robotGoal = path(end,:);
   initialOrientation = 0;
   m1=length(path1(:,1));
   plot(path(:,1), path(:,2),'m-d','LineWidth',2)
   xlim([0 12])
   ylim([0 12])
   m=length(path(:,1));
   s1=0;
   s2=0;
   % %Trajectory length computation
   for i=1:m1-1
   s1=s1+sqrt((path1(i,1)-path1(i+1,1))^2+(path1(i,2)-path1(i+1,2))^2);
   end
   for i=1:m-1
   s2=s2+sqrt((path(i,1)-path(i+1,1))^2+(path(i,2)-path(i+1,2))^2);
   end
```

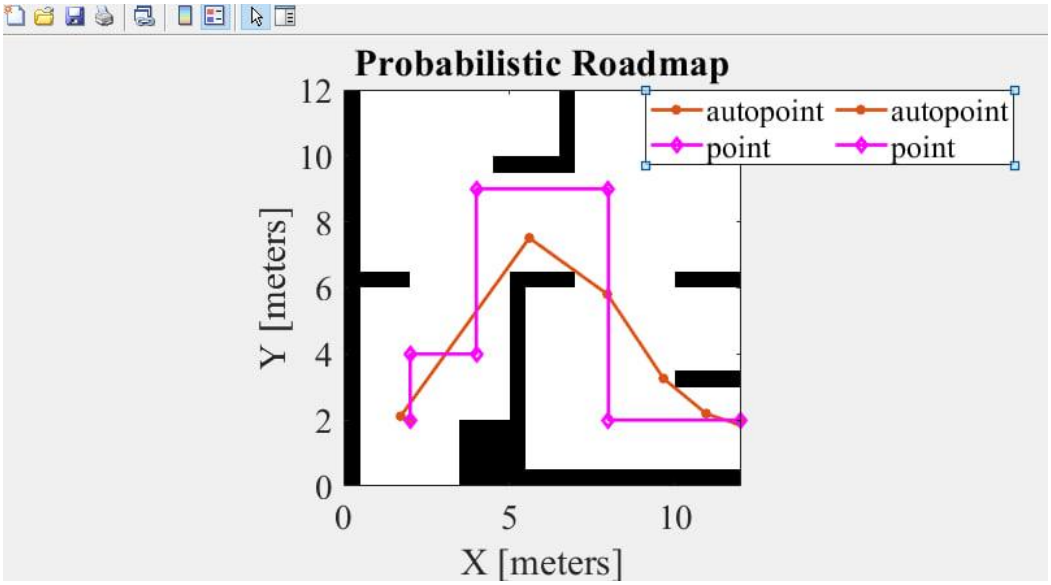

Fig. 3.3. Corridor with obstacles, s1=16.1251 meters (autopoint), s2=24 meters(point not auto).

3.6 Labyrinth

We repeat the same experiment as with the corridor. The result is shown in the picture Figure 3.4.

Matlab code:

```
      filePath
=fullfile(fileparts(which('PathPlanningExample')),'data','exampleMaps.mat');
      load(filePath);
      map = robotics.BinaryOccupancyGrid(complexMap, 1);
      show(map);
      robotRadius = 0.2;

      path=[7.00 5.00;
        7.00 9.00;
        31.00 9.00;
        31.00 18.00;
        45.00 18.00;
        45.00 28.00;
        31.00 28.00;
        31.00 39.00]; %(path developed in 2.3)
      startLocation = [2 5];
      endLocation = [31 39];
      prm = robotics.PRM;
      prm.Map = map;
      prm.NumNodes = 100;
      path1 = findpath(prm, startLocation, endLocation);
      show(prm, 'Map', 'on', 'Roadmap', 'off');hold on
      robotCurrentLocation = path(1,:);
      robotGoal = path(end,:);
      initialOrientation = 0;
      m1=length(path1(:,1));
      plot(path(:,1), path(:,2),'m-d','LineWidth',2)
      xlim([0 50])
      ylim([0 40])
      m=length(path(:,1));
      s1=0;
      s2=0;
      % %Trajectory length computation
      for i=1:m1-1
      s1=s1+sqrt((path1(i,1)-path1(i+1,1))^2+(path1(i,2)-path1(i+1,2))^2);
      end
      for i=1:m-1
      s2=s2+sqrt((path(i,1)-path(i+1,1))^2+(path(i,2)-path(i+1,2))^2);
      end
```

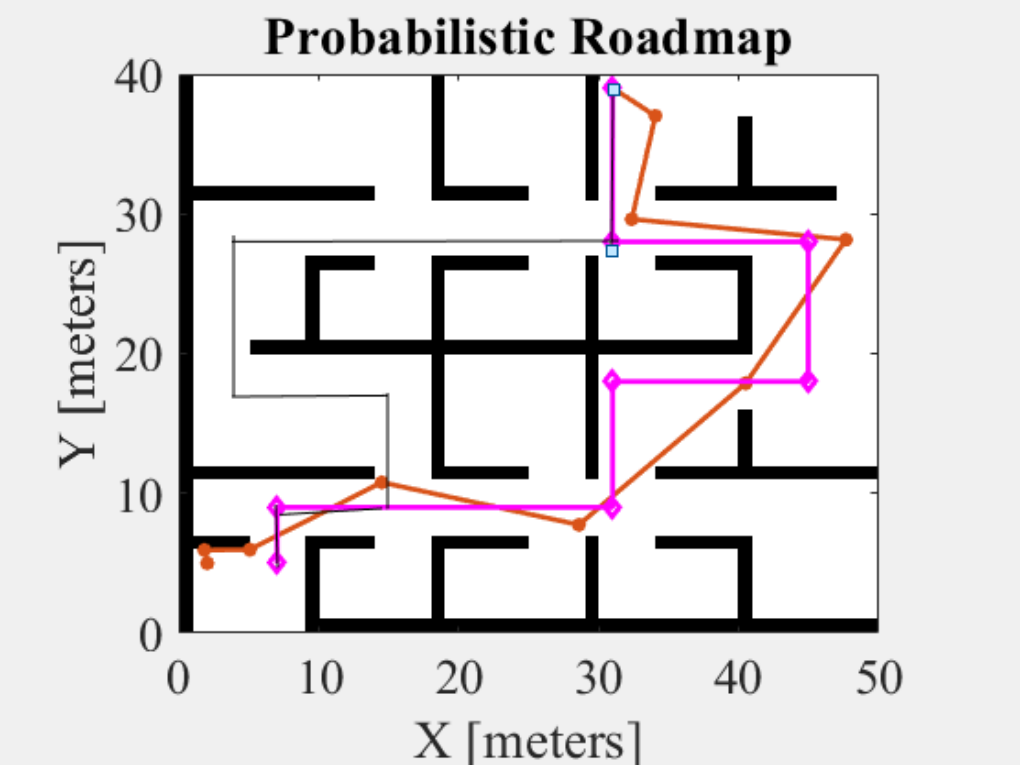

Fig. 3.4. Robot move in labyrinth, S1=84.1 meters, S2=86 meters;

3.7 Improvement of the passage of the path with the help of the controller

We'll build upon the previous content and improve the program. Establishing a path for the robot, it will traverse the distance independently using the controller.

We'll set the desired speed, angular velocity, and scanning distance. The outcome is illustrated on the diagram: the route is represented by the blue line, with our object depicted by an arrow.

Program code:

```
path = [2.0 1.0;
3.2 5.7;
6.3 6.7;
8.2 7.7;
12.7 14.7;
18.5 16.3];
robotRadius = 0.4;
robot = ExampleHelperRobotSimulator('emptyMap',2);
robot.enableLaser(false);
robot.setRobotSize(robotRadius);
robot.showTrajectory(true);
robotCurrentLocation = path(1,:);
robotGoal = path(end,:);
initialOrientation=0;
robotCurrentPose = [robotCurrentLocation initialOrientation];
robot.setRobotPose(robotCurrentPose);
```

```
plot(path(:,1), path(:,2),'k--d')
xlim([0 20])
ylim([0 20])
grid on
controller = robotics.PurePursuit
controller.Waypoints = path;
controller.DesiredLinearVelocity = 0.6;
controller.MaxAngularVelocity = 5;
controller.LookaheadDistance = 0.4; %;
goalRadius = 0.1;
distanceToGoal = norm(robotCurrentLocation - robotGoal);
controlRate = robotics.Rate(10);
while( distanceToGoal > goalRadius )
% Compute the controller outputs, i.e., the inputs to the robot
[v, omega] = step(controller, robot.getRobotPose());
% Simulate the robot using the controller outputs.
drive(robot, v, omega);
% Extract current location information ([X,Y])
% from the current pose of therobot
robotCurrentPose = robot.getRobotPose();
% Recompute the distance to the goal
distanceToGoal = norm(robotCurrentPose(1:2) - robotGoal);
waitfor(controlRate);
end

%Trajectory length computation
%delete(robot);
m=length(path(:,1));
s=0;
for i=1:m-1
s=s+sqrt((path(i,1)-path(i+1,1))^2+(path(i,2)-path(i+1,2))^2);
```

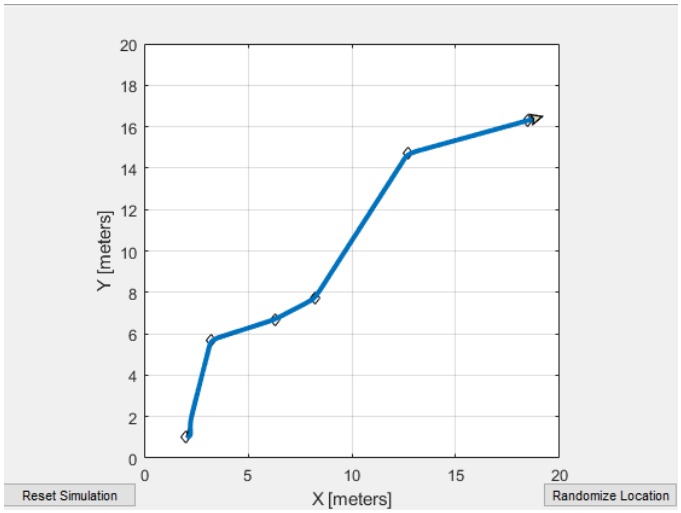

Fig. 3.5. Robot follow a way with help of controller

The effectiveness of a straightforward method for placing temporary points in a one-way maze to reach an exit is assessed in both simulated and real-world environments. In the practical scenario of "escaping a room," if there is a lack of positional reference relative to the robot's actual position, the map may become misaligned.

The diverse range of configurations that have successfully passed testing for this scenario demonstrates that solutions devised for task completion perceive the surrounding environment differently. Significant efforts have been made to improve the performance of the planned route, particularly when dealing with a real robot, to minimize task completion time while maintaining the robot's ability to navigate dynamic environments. Testing results from SMR suggest that while we can implement automatic robot movement, in certain situations, we opt for safer movements and guide the robot accordingly to prevent collisions with walls.

SUMMARY

An autonomous mobile robot operates in partially unknown environments, employing diverse strategies to tackle simple tasks.

Consequently, considerable effort has been devoted to optimizing the planned route, ensuring the robot completes tasks swiftly while retaining its adaptability to dynamic surroundings. A profound comprehension of the prevalent control scheme in mobile robots and the array of functions simulating robot behavior has been attained, laying the groundwork for testing new scenarios prior to real-world deployment. Furthermore, accurate methods were employed to map points across various coordinate systems within the environment. Substantial endeavors were also directed towards code optimization for addressing real-world challenges, drawing from extensive technical resources on linear indexing and vectorization in the MATLAB environment.

The primary aim of modeling is to showcase the functionality of the selected strategy for specific tasks. However, adapting these strategies to suit all possible configurations necessitates a significant time investment.

Lastly, it's worth mentioning that the SMR robot platform serves as a comprehensive tool for understanding autonomous mobile robots, even for users with limited knowledge of such equipment.

REFERENCES

1. Автоматичний рух керування [Електроний ресурс] Режим доступу: https://encyclopedia2.thefreedictionary.com/Automated+Control+Systems

2. Папушин Ю. Л., Білецький В. С. Основи автоматизації гірничого виробництва. — Донецьк : Східний видавничий дім, 2007. — 168 с. — ISBN 978-966-317-004-6.

3. Іванов А. О. Теорія автоматичного керування: Підручник. — Дніпропетровськ: Національний гірничий університет. — 2003. — 250 с.

4. Енциклопедія кібернетики. тт. 1, 2. — К.: Головна редакція УРЕ, 1973. — 584 с.

5. Поняття системи керування [Електроний ресурс] Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/ipv/page7.html

6. Курейчик В.М. Генетичні алгоритми та їх застосування / В.М.Курейчик. - Таганрог: ТРТУ, 2002

7. «Компоненти та технології» № 1'2008 [Електронний ресурс]. - Режим доступу:http://www.kit-e.ru/articles/sensor/2008_01_56.php

8. Чорноножкін В.А. Система локальної навігації для наземних мобільних роботів/ Чорноножкін В.А., Половко С.А./ Центральний науково-дослідний та дослідно-конструкторський інститут робототехніки та технічної кібернетики – Режим доступу до матеріалів: http://144.206.159.178/ft/9344/538978/11790042.pdf

9. Стоут Б.. Алгоритми пошуку шляху/Браян Стоут[Електронний ресурс]. – Режим доступу: http://pmg.org.ru/ai/index.html

10. Ємельянов В.В. Теорія та практика еволюційного моделювання / В. В. Ємельянов, В. В. Курейчик В. В., В. М. Курейчик. - М: Фізматліт,

11. Jean-Yves Bouguet. Visual Navigation using a single camera. – [Електронний ресурс]/ Jean-Yves Bouguet, Pietro Perona. – Режим доступа:http://www.vision.caltech.edu/bouguetj/

12. Lozano-Perz T. An Algorithm of Planning Colisions Free Paths among Polyhedral Abstracts/ T. Lozano-Perz. – ACM,1984. p 560-570.

13. Davidor Y. Genetic Algoritms and Robotics./ Y. Davidor. – World Scientific, 1991

14. Sibata T. Robot Motion Planning by Genetic Algoritms with Fuzzy Logics/ T. Sibata, T. Fukuda. – ISIC, 1993.

15. Shing M.T. Genetic Algoritms for the Development of Real-Time Multi-Historic Search Strategies/ M.T Shing, G.B. Parker. – 1993. 570 p.

16. Zhou H.H. Learning by Analogy in Genetic Classifier Systems/Zhou H.H. – CA, 1994. 550 p.

17. Zbigniev Michailevich. Genetic Algoritms plus Data Structures equal Evolution

18. Коваль В. Вдосконалений метод побудови локальної карти середовища мобільного робота // Вісник Тернопільського державного технічного університету ім. І.Пулюя. – 2003. – Т.8. – №2. – С. 80-88.

19. Коваль В. Спосіб виявлення перешкод мобільного робота з використанням технології злиття показів різнорідних сенсорів // АСУ и приборы автоматики. – Харьков. – 2004. – Вип. 126. – С. 128-135

20. Інтелектуальні роботиb [Електроний ресурс] Режим доступу: http://www.raai.org/about/persons/dobrynin/pages/kii2006-pln.html [Архівовано 27 січня 2012 у Wayback Machine.]

21. Мобільні роботи [Електроний ресурс] Режим доступу: http://www.mobilerobots.com/Libraries/Downloads/PeopleBot-PPLB-RevA.sflb.ashx [Архівовано 8 лютого 2013 у Wayback Machine.] Сайт mobilerobots.com

22. Principal Investigator: W. Kennedy, National Institutes of Health, NIH SBIR 2 R44 HD041781-02

23. Speci-Minder; see elevator and door access. *Архів* оригіналу *за 2 січня 2008*. Процитовано 7 лютого 2013.

24. Сайт go.amazone.de. *Архів* оригіналу *за 4 березня 2016*. Процитовано 10 лютого 2013.

25. FOXNews.com — Weapons Makers Unveil New Era of Counter-Terror Equipment — Local News | News Articles | National News | US News. *Архів* оригіналу *за 18 лютого 2013*. Процитовано 7 лютого 2013.

26. Штучний інтелект. Проблеми створення. *Архів* оригіналу *за 7 березня 2016*. Процитовано 10 лютого 2013.

27. Алгоритми пошуку шляху. *Архів* оригіналу *за 29 квітня 2012*. Процитовано 10 лютого 2013.

28. Sensor-based Autonomous Navigation for Mars Rover. *Архів* оригіналу *за 13 травня 2008*. Процитовано 10 лютого 2013.

29. Планування шляху автономного робота на основі еволюційних алгоритмів. *Архів* оригіналу *за 8 жовтня 2014*. Процитовано 10 лютого 2013.

30. Новые законы робототехники защитят людей от роботов. Архів оригіналу за 15 грудня 2012. Процитовано 10 лютого 2013.

31. Автономные роботы на топливных элементах нового типа. Архів оригіналу за 25 квітня 2013. Процитовано 10 лютого 2013.

32. Almeida P., Carvalho V., Simões A. Reinforcement Learning Applied to AI Bots in First-Person Shooters: A Systematic Review // Algorithms. 2023. Vol. 16, no. 7. P. 323. URL: https://doi.org/10.3390/a1607032 .

33. Военные роботы и нравственные проблемы. Архів оригіналу за 10 квітня 2013. Процитовано 10 лютого 2013.

34. Клебанова Т.С., Чаговець Л.О., Панасенко О.В., Нечітка логіка та нейронні мережі в управлінні підприємством: Монографія.-Х.: ВД «ІНЖЕК», 2011. – 240 с.

35. Кофман А. Введення в теорію нечітких множин / Кофман А. – М.: Радіо та зв'язок, 1982. – 432с.

36. Поспєлов Д. А. Логіко-лінгвістичні моделі в системах управління / Поспелов Д. А. - М.: Витрата енергії, 1981.-232 с

37. "Raspberry Pi 3B+ 32 bit and 64 bit Benchmarks and Stress Tests," [Online]. Available: https://www.researchgate.net/publication/327467963_Raspberry_Pi_3B_32_bit_an. [Accessed 6 червень 2022].

38. "Raspberry Pi Camera Board v1.3 Datasheet," [Електроний ресурс] Режим доступу: https://uk.pi-supply.com/products/raspberry-pi-camera-board-v1-3-5mp1080p. [Accessed 6 червень 2022].

39. "Raspberry Pi 3 Model B+ Datasheet," [Електроний ресурс] Режим доступу: https://raspberrypi.in.ua/wp-content/uploads/2018/03/datasheet_raspberry_pi_3_model_b.pdf. [Accessed 6 червень 2022].

40. "Camera Serial Interface," [Електроний ресурс] Режим доступу: https://www.mipi.org/sites/default/files/Camera%20Serial%20Interface_CSI 2_CSI. [Accessed 6 червень 2022].

41. "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms," [Електроний ресурс] Режим доступу: https://towardsdatascience.com/r-cnn-fastr-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed 6 червень 2022].

42. "Raspbian OS," [Електроний ресурс] Режим доступу: http://www.raspbian.org/. [Accessed 6 червень 2022]. 24. "imageZMQ: Transporting OpenCV images," [Online]. Available: https://pythonlang.dev/repo/jeffbass-imagezmq/. [Accessed 6 червень 2022].

43. "socket — Low-level networking interface," [Електроний ресурс] Режим доступу: https://docs.python.org/3/library/socket.html. [Accessed 6 червень 2022].

44. "LSM303 Accelerometer + Compass Breakout," [Електроний ресурс] Режим доступу: https://learn.adafruit.com/lsm303-accelerometer-slash-compassbreakout/coding. [Accessed 6 червень 2022].

45. "Interface NEO-6M GPS Module with Arduino," [Електроний ресурс] Режим доступу: https://microcontrollerslab.com/neo-6m-gps-module-arduino-tutorial/. [Accessed 6 червень 2022]. [28. "Introduction to L298N Motor Driver," [Online]. Available: https://www.electroduino.com/introduction-to-l298n-motor-driver-how-itswork/#:~:text=L298n%20motor%20driver%20module%20uses,polarity%20of%20its%20input%20voltage.. [Accessed 6 червень 2022].

46. Автономні роботи на паливних елементах нового типу. Архів оригіналу за 25 квітня 2013 року. Процитовано 10 лютого 2013 року.

47. Військові роботи та моральні проблеми. Архів оригіналу за 10 квітня 2013 року. Процитовано 10 лютого 2013 року.

48. http://www.popmech.ru/article/592-robotyi-ubiytsyi/ [Архівовано 28 жовтня 2012 у Wayback Machine.] Роботи-вбивці: війна машин як філософська проблема

49. Завтра та післязавтра бойових роботів. Архів оригіналу за 20 січня 2013. Процитовано 10 лютого 2013 року.

50. Нові закони робототехніки захистять людей від роботів. Архів оригіналу за 15 грудня 2012. Процитовано 10 лютого 2013 року.

51. М.Б. Ігнатьєв, Ф.М. Кулаков, А.М. Покровський. Алгоритми управління роботами-маніпуляторами. Л.: Машинобудування. 1977. 248 с.

52. R. A. Brooks, «Solving the Find-Path Problem by Good Representation of Free Space». IEEE Transactions on Systems, Man, Cybernetics, Vol. SMC-13, pp. 190-197.March. 1983.

53. D. T. Kuan, J. C. Zamiska and R. A. Brooks,·«Natural Decomposition of Free space for Path Planning», IEEE Conference on Robotics and A.utomation, St. Louis., MO, March 1985.

54. Samuel, Arthur (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development (англ.). 3 (3): 210—229. CiteSeerX 10.1.1.368.2254. doi:10.1147/rd.33.0210. S2CID 2126705.

55. Milner, Peter M. (1993). The Mind and Donald O. Hebb. Scientific American (англ.). 268 (1): 124—129. Bibcode:1993SciAm.268a.124M. doi:10.1038/scientificamerican0193-124. ISSN 0036-8733. JSTOR 24941344. PMID 8418480.

56.Russell, Stuart; Norvig, Peter (2003) [1995]. Artificial Intelligence: A Modern Approach (вид. 2nd). Prentice Hall. ISBN 978-0137903955.