TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 68, Issue I, March, 2025

ROBOT VECTOR OPERATING SYSTEM FOR LOCALIZATION AND MAPPING

Alexandra-Cătălina CIOCÎRLAN, Luige VLĂDĂREANU, Ionel-Alexandru GAL Bianca Rodica GHINOIU, Cristina Marilena NIȚU , Marius PANDELEA, Mihaiela ILIESCU

Abstract: *This paper explores the use of the Intel RealSense Depth Camera D455 in the context of localization, autonomous navigation, and mapping of complex environments. We investigated advanced methods for estimating the position and orientation of mobile robots, using depth data to build detailed and accurate maps. The Simultaneous Localization and Mapping (SLAM) algorithm implemented is based on the real-time detection of changes in the environment, ensuring continuous updating of the map. The robot vector, representing its state, is evaluated according to the robot trajectory, significantly influencing the performance of autonomous navigation. The results obtained demonstrate the efficiency of the proposed method in various test scenarios.*

Key words: *Intel RealSense Depth Camera D455, localization, autonomous navigation, mapping, robot vector.*

1. INTRODUCTION

This article evidences results on the control of autonomous mobile robots vectors, a highly relevant research area that has significant implications in autonomous robotics, artificial intelligence (AI), and robotics engineering.

Autonomous mobile robots (AMRs) are complex systems capable of moving and operating independently in an environment, using a range of advanced sensors and algorithms to orient themselves, navigate and perform various tasks.

An essential aspect in controlling autonomous mobile robots is localization and mapping. Simultaneous Localization and Mapping (SLAM) technology is particularly important in this context, allowing robots to create maps of the environment while determining their position in that environment. The use of sensors such as Lidar, RGB-D cameras, and inertial measurement units (IMUs) facilitate this process.

These advanced technologies allow robots to navigate in unfamiliar environments, avoid obstacles, and complete complex missions.

Control algorithms play a crucial role in the operation of autonomous mobile robots. The article discusses the use of advanced control algorithms, including adaptive control and reinforcement learning. These methods allow robots to adapt their trajectory according to changes in the environment and improve their performance over time.

Although significant progress has been made in the field of controlling autonomous mobile robots, there are still many challenges.

AMRs demonstrate outstanding autonomous and flexible navigation capabilities. They use advanced technologies, such as SLAM (Simultaneous Localization and Mapping), to orient themselves and create accurate maps of the environment. SLAM technology allows robots to simultaneously determine their position in space and map the environment, making it easier to avoid obstacles and adapt to changes in the operational environment. This capability is essential for efficient operation in unfamiliar and dynamic spaces.

To highlight the importance of post-processing techniques applied to the depth maps generated by RealSense cameras in improving

Received: 29.01.25; **Similarities**: 05.03.25: **Reviewed**: 20.02./14.02.25: **Accepted**:19.03.25.

the accuracy of depth measurements, we referenced article [1], which emphasizes two essential aspects: temporal filtering, which can significantly reduce depth noise, and spatial filtering, which enables image smoothing without compromising important details. This information is relevant to supporting the methodology used in my paper, providing a solid theoretical framework and arguments based on existing research that validate the proposed approach.

Reference [2] is used to emphasize the relevance of an autonomous navigation system for mobile robots operating in an industrial environment. Citing this reference is essential to highlight how mobile robots can recognize floor markings and navigate along a predefined route, avoiding obstacles and moving autonomously between fixed points.

I used reference [3] as a source because it provides a detailed description of the navigation system, localization, and simultaneous mapping methods employed for managing large-scale urban center maps.

These methods were tested on a real autonomous robot that traversed several kilometers in an urban environment, helping us better understand how SLAM functions in this context.

Because we also focused on the use of the Kalman filter, reference [4] was utilized as it presents an interesting solution involving the use of a global reference system along with a set of point landmarks on a map. This approach addresses the problem of estimating the state vector representing the robot's position and the landmarks.

The tests were conducted according to estimation and filtering theory, providing a solid foundation for solving various localization and simultaneous mapping problems in an unknown environment.

To map an environment, it is essential for the autonomous mobile robot to detect current changes in the explored areas so that the generated map can be used in real-time. The accuracy of the SLAM algorithm is closely tied to the trajectory of the autonomous mobile robot in that environment.

Reference [5] provided insights that we were able to apply in our tests.

2. METHODS AND MATERIALS

Types and characteristics of the materials (subsystems / components / software), as well as associated methods used in this research are further presented.

2.1 Air Vector (drone)

Individualized Tello drone (see figure 1) features that stand out for their ease of use, varied functionality, and ability to provide a pleasant flight experience are:

- dimensions: 98 x 92.5 x 41 mm
- weight: approx. 80 grams (including battery)
- photo resolution: 5 MP (2592 x 1936)
- video resolution: 720p HD at 30 frames per second
- top speed: 8 m/s (18 mph)
- maximum flight altitude: 10 m
- flight time: approximately 13 minutes

Fig. 1. Tello Drone [6]

High-performance depth camera, Intel® RealSense™ D455 (see figure 2), was also used. It is designed to provide accurate color images and depth data for a wide range of applications, such as robotics, augmented reality, and autonomous vehicles.

Important technical features are:

- The *depth accuracy* of the D455 camera is significant, with depth errors below 2% at a distance of 4 meters.
- The *high-resolution RGB* camera provides clear and detailed images, useful for visual recognition and analysis.
- *Resolution* - up to 1280 x 720 pixels- high resolution allows fine detail to be captured in images and depth data.

Fig. 2. Depth Camera D455 [7]

2.2. The Terrestrial Vector (the robot)

The terrestrial vector has integrated tank chassis (see figure 3.) into autonomous mobile robots that have the ability to move and perform missions without direct human intervention. These types of robots can be equipped with cameras, sensors and navigation systems to help them navigate in various unstructured environments.

Fig. 3. Robotic Tank Chassis [8]

The Intel® RealSense™ Depth Camera D455 has been attached to the mobile robot (and resulted the ground robot) - as evidenced in figure 4.

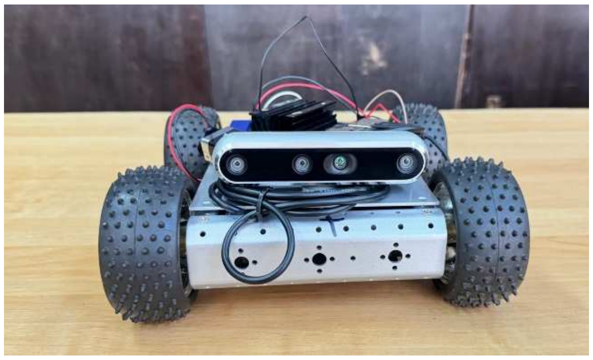
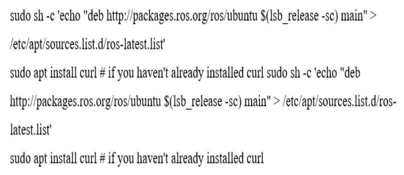


Fig. 4. Depth Camera D455 and Mobile Robot

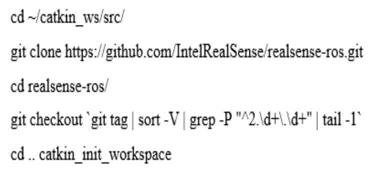
Then followed the ROS installation process using the specific code lines (see figure 5)



```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
sudo apt install curl # if you haven't already installed curl sudo sh -c 'echo "deb
http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-
latest.list'
sudo apt install curl # if you haven't already installed curl
```

Fig. 5. ROS Installation Code Lines

and adding the Intel RealSense™ Depth Camera D455 in ROS (see figure 6)



```
cd ~/catkin_ws/src/
git clone https://github.com/IntelRealSense/realsense-ros.git
cd realsense-ros/
git checkout `git tag | sort -V | grep -P "^2.\d+\.\d+" | tail -1`
cd .. catkin_init_workspace
```

Fig. 6. Installation of Camera D455 in ROS

The next step required for localization and mapping was that of defining the parameters of the D455 depth camera. It involves setting and adjusting various specifications and features to obtain accurate distance measurements and optimize the camera's performance in the desired application. These parameters include, but are not limited to: resolution, frame rate, measurement distance, filtering and processing algorithms, camera calibration. (see figure 7).

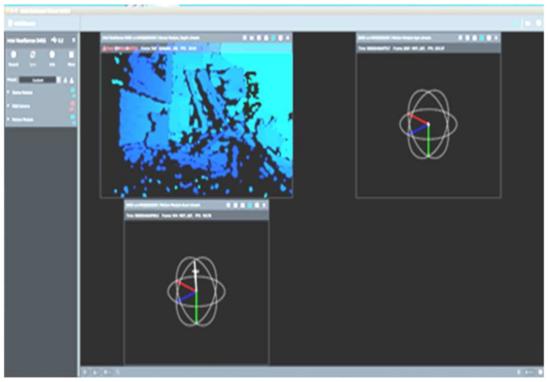


Fig. 7. Camera D455 Parameter Settings

The Intel RealSense D455 camera is an advanced depth camera that offers both depth images and RGB (color) images. Its main functionality and how these images are obtained are essential (see figure 8).

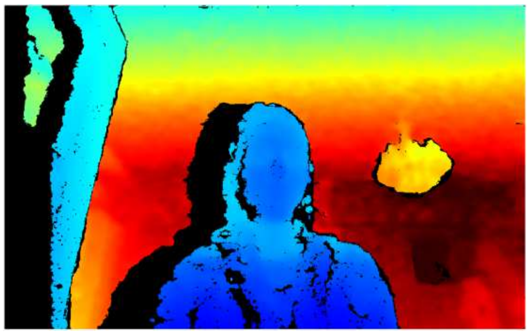


Fig. 8. Capturing RGB images

The block diagram of the command-and-control system of the ground robot and drone is shown in figure 9.

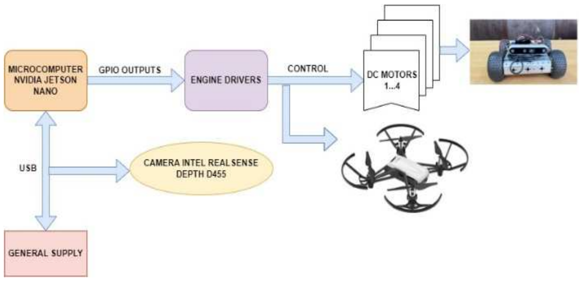

MICROCOMPUTER NVIDIA JETSON NANO
GPIO OUTPUTS
ENGINE DRIVERS
CONTROL
DC MOTORS 1...4
USB
CAMERA INTEL REALSENSE DEPTH D455
GENERAL SUPPLY

Fig. 9. General Robot Vector Scheme

3. ROBOT VECTORS MODELING

The intelligent vector control system of autonomous mobile robots is based on the presence of a leader, which in this case is the DJI TELLO drone. The overall transient state control function of the leading autonomous air vector, in the context of a multi-intelligent vector environment, is defined by the following relationship:

$$\Theta = f_{SCI}(\Delta_{SCI}, \Delta_{STT}, t) \quad (1)$$

The expression of the dynamic mathematical model that defines the leading air vector and includes six variable parameters — three positions and three orientations in the inertial frame of reference — is as follows:

$$\Phi_{VALin} = (x_{in}, y_{in}, z_{in}, \varphi_{in}, \theta_{in}, \psi_{in}) \quad (2)$$

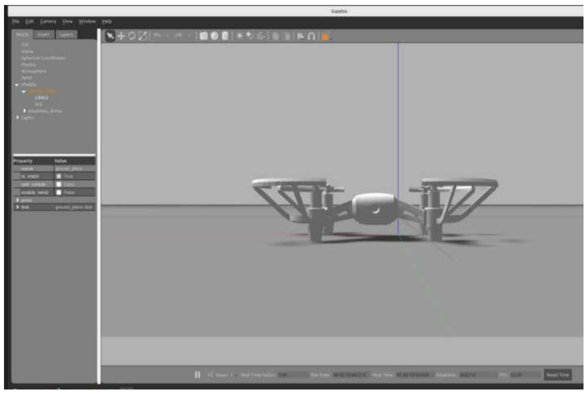
The correlation of the two systems is given by relations system (3):

$$v_{VAL(t)} = \begin{bmatrix} \dot{x}_{in(t)} \\ \dot{y}_{in(t)} \\ \dot{z}_{in(t)} \end{bmatrix} = R \cdot \begin{bmatrix} v_{xpr} \\ v_{ypr} \\ v_{zpr} \end{bmatrix}$$
$$\Omega_{\omega VAL(t)} = \begin{bmatrix} \Omega_{\omega x(t)} \\ \Omega_{\omega y(t)} \\ \Omega_{\omega z(t)} \end{bmatrix} = \begin{bmatrix} \omega_{\varphi_{pr}} \\ \omega_{\theta_{pr}} \\ \omega_{\psi_{pr}} \end{bmatrix} = T \cdot \begin{bmatrix} \dot{\varphi}_{in} \\ \dot{\theta}_{in} \\ \dot{\psi}_{in} \end{bmatrix} \quad (3)$$

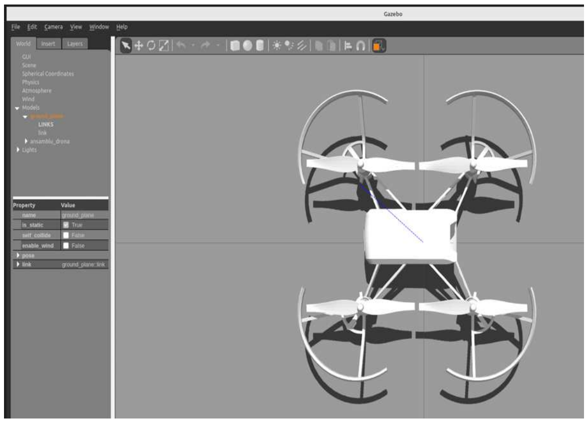
The Gazebo environment is commonly used in ROS because together they provide a complete platform for robotics development and testing. Gazebo enables real-time control of robots in simulation, allowing for running and changing the robot code while they are in the virtual environment.

The Tello drone was simulated in ROS using Gazebo, where a launch file (*.launch) was created to specify all the nodes and parameters required for the package in Gazebo.

Images of how the drone was simulated in the Gazebo are shown in figure 10 (a. front view; b. top view).



a. front view



b. top view

Fig. 10. Drone simulation in the Gazebo

3.2. Gazebo Environment Description

Simulating drones in virtual environments is essential for testing and developing control, navigation, and perception algorithms, with low risk of physical damage or additional costs. Gazebo, integrated with the Robot Operating System (ROS), provides a robust and flexible platform for these simulations.

Gazebo is a high-fidelity robotics simulator that allows the simulation of complex robot dynamics in 3D environments. Gazebo offers advanced features such as:

- *Realistic Physics Simulation*: Uses physics engines such as ODE, Bullet, and DART to recreate motion dynamics and interactions with the environment.
- *3D Visualization:* Provides a graphical interface to view and interact with robots in real-time.
- *ROS Integration:* Enables seamless communication with ROS, facilitating the development and testing of control algorithms.

3.3. Modeling the terrestrial robot vector

Modeling a terrestrial robot vector in the Gazebo involves several steps, from defining the robot model to integrating it into a simulation environment.

The ground robot simulated using Gazebo is shown in figure 11.

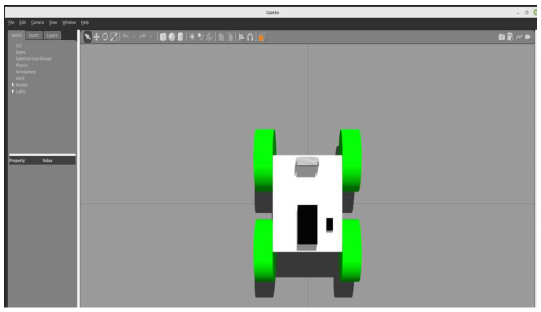


Fig. 11. Robot in the Gazebo

In order to have the simulation in Gazebo, where the robot can be tested, we first need a URDF file to describe it. The URDF file is based on the XML language where there are declared the mass of each element, the inertia, the joints, the links and everything necessary for the simulation of the robot.

The file for describing a robot can become very complex, even if it is a simple model. For this reason, but also to make the URDF file more flexible in case changes occur, XACRO files are better used to define the robot model.

XACRO is not an alternative to the URDF, it is just another way of defining it. In addition, the URDF file can be generated from the XACRO file by executing a single command. For example, if we have created the file "4WD.xacro" and we want to get "4D.urdf", we will execute "rosrun xacro xacro.py 4WD.xacro > 4WD.urdf".

One of the advantages of using XACRO is the ability to declare a constant at the beginning of the file, which can be used in defining several links and, if we want to change one feature value, we can do it only once instead of modifying it for each element. A relevant example for this work being the wheel, which will be defined identically 4 times. Another advantage of XACRO is the ability to calculate mathematical expressions, which are necessary for inertia.

We started the description of the robot by creating a ROS package, which we named "4wd_description" and inside which we created an "urdf" folder containing the file of interest "4wd.xacro". Next, the code written in groups of structures as independent as possible will be explained.

For the file to be considered a XACRO file, the minimum structure is this:

```
<?xml version="1.0"?>
<robot name="4wd"
xmlns:xacro="http://www.ros.org/wiki/xacro">
```

The first line is the declaration of the XML language to which is assigned a version that must be 1.0. This is standard in all XACRO files.

The second line "*tells*" the XACRO system to use the XACRO protocol from the address:

```
<xacro:include filename="$(find
4wd_description)/urdf/4wd_materials.xacro" />
```

By this line another XACRO file has been included that describes a series of colors that can be used in the creation of components. Its content is:

```
<?xml version="1.0"?>
<robot
xmlns:xacro="http://www.ros.org/wiki/xacro">
    <material name="Green">
        <color rgba="0.0 1.0 0.0 1.0"/>
    </material>
    <material name="Blue">
        <color rgba="0.0 0.0 1.0 1.0"/>
    </material>
    <material name="Red">
        <color rgba="1.0 0.0 0.0 1.0"/>
    </material>
    <material name="White">
        <color rgba="1.0 1.0 1.0 1.0"/>
    </material>
    <material name="Yellow">
        <color rgba="1.0 1.0 0.0 1.0"/>
</material>
</robot>
```

This file can be optional, as colors can be given directly in the body of each link. We chose this method for the highest possible simplification of the code. The colors are defined using the RGB system with the values scaled from 0 to 1, instead of 0 to 255. There can also be noticed the first lines needed for any XACRO file, mentioned above.

The next line includes another file, called "4wd.gazebo", which is defined for controlling the robot in the simulator.

```
<xacro:include filename="$(find
4wd_description)/urdf/4wd.gazebo" />
```

The code so far was just the beginning of the file that contained the declaration and initialization part. The next part is the actual construction of each element and the establishment of the parent-child relationships. The first link is the basic one that has a simple description and on which all other links will depend. This is the center of the robot and is especially important in Rviz.

```
<link name="base_link"> </link>
```

It should be noticed that the order in which the links are declared does not matter because the model will be created according to the joints where the parent and child are set.

The following structure describes the robot chassis:

```
<link name="body_link">
        <visual>
<origin xyz="0 0 0.03" rpy="0 0 0" />
          <geometry>
<box size="0.4 ${wheel_separation} 0.09"/>
          </geometry>
<material name="White" />
        </visual>
        <collision>
<origin xyz="0 0 0.03" rpy="0 0 0" />
          <geometry>
<box size="0.4 ${wheel_separation} 0.09"/>
          </geometry>
        </collision>
       <inertial>
<origin xyz="0 0 0.03" rpy="0 0 0" />
          <mass value="${mass}" />
<inertia ixx="${inertia}" ixy="0.0" ixz="0.0"
iyy="${inertia}" iyz="0.0" izz="${inertia}" />
       </inertial>
     </link>
 <joint name="base_to_body" type="fixed">
   <parent link="base_link" />
  <child link="body_link" />
<origin xyz="0 0 ${wheel_radius}" rpy="0 0
0"/>
   </joint>
```

One of the essential elements in the simulation of a robot is "collision" and refers to the way it interacts with the world around it. The element contains a label indicating the geometry of the collision, in the same format as in the case of the visual, for which an origin can also be specified.

Along with collision properties, inertia properties allow for physical simulation of the robot. The element '*inertial*' contains the following labels:

- "origin" - is used to elect the position of the center of mass relative to the origin of the connection.
- "mass"- sets the mass of the connection., it must have a value as close as possible to the real one because it will condition the way the joints move.
- "inertia" - defines the inertia matrix of the bond and will determine how it behaves when

moving or rotating. It was assumed that the material from which the connection is made is homogeneous, and the matrix will have values other than 0, only diagonally. These were calculated based on the primary geometric shapes.

The last element in the structure is the merge that is initialized with a name and a type, after which the parent, the child and the origin are declared.

The robot would not yet be able to move in the simulation because no effort (force or torque) has been applied to the joints, so it is applied "real" motors. For this, transmissions will have to be added to all the "continuous" joints that describe the relationship between the motors and the links, as we did for the front wheel on the right side.

```
    <transmission name="front_right_wheel">
    <type>transmission_interface/SimpleTrans
mission</type>
     <joint name="base_to_front_right_wheel">
    <hardwareInterface>hardware_interface/Eff
ortJointInterface</hardwareInterface>
           </joint>
     <actuator name="front_right_motor">
```

4. RESULTS AND DISCUSSIONS

One of the most important mathematical tools for working with robots is Transformers (TF). ROS provides special packages through which we can use these transformers much more easily to solve various problems. Physical systems, especially those for robots, have many 3D coordinate frames that can change over time. TF is the package that allows the user to keep track of these frames.

In ROS, each node can use the TF packet to transmit a transformation from one frame to another. With all these vertices the transform tree can be created.

The five transformer frames are: "body_link", "battery", "rear_right_wheel", "hcsr04_link" and "front_right_wheel". The last 4 have a common parent, and that is "body_link", which in turn has the "base_link" frame as its parent. The additional information that the tree also provides is:

- "*Recorded at time*" - provides information about the time when the transforms were transmitted in the "Unix time" system.
- "*Broadcaster" - represents the name of the node that transmits data.*
- "*Average rate*" - refers to the average publication rate expressed in Hz.
- "*Buffer length*" - contains how long data has been stored.

The robot's movement was done through the dedicated ROS "teleop_twist_keyboard" package, which allows movement from the keyboard buttons, because it publishes in the subject "/cmd_vel", and the differential control module provided by Gazebo listens to the subject "/cmd_vel".

The simulation of the robot and drone in a virtual environment (see figure 12) is aimed to observe its behavior in various scenarios by varying the speed and changing the shapes and positions of obstacles. For the visualization and verification of these aspects, the Rviz platform was used.

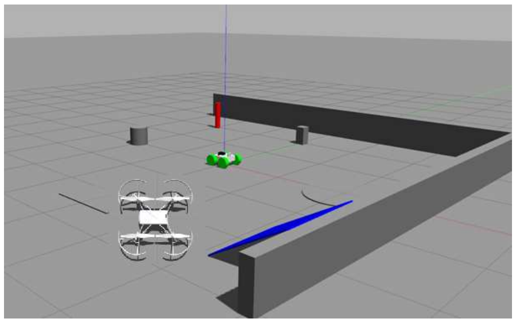


Fig. 12. Robot and drone in the Gazebo

5. CONCLUSION

The article highlights the importance of vector control of autonomous mobile robots and the relevance of this field of research in autonomous robotics, artificial intelligence and robotics engineering.

Advances in localization, mapping, and control algorithm technologies have the potential to transform various industries and significantly improve the capabilities and applicability of autonomous mobile robots.

The simulation of the DJI Tello drone and ground robot in Gazebo, integrated with ROS, provides a robust and efficient test environment for the development and validation of control algorithms. This framework allows researchers and developers to experiment advanced technologies in a safe controlled environment.

6. REFERENCES

[1] Grunnet-Jepsen, A., & Tong, D. (2018). Depth post-processing for intel® realsense™ d400 depth cameras. *New Technologies Group, Intel Corporation*, *3*.

[2] Harapanahalli, S., Mahony, N. O., Hernandez, G. V., Campbell, S., Riordan, D., & Walsh, J. (2019). Autonomous Navigation of mobile robots in factory environment. Procedia Manufacturing, 38, 1524-1531.

[3] Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., & Burgard, W. (2015). Autonomous robot navigation in highly populated pedestrian zones. Journal of Field Robotics, 32(4), 565-589.

[4] Smith, R., Self, M., & Cheeseman, P. (1990). *Estimating uncertain spatial relationships in robotics*. Autonomous robot vehicles, 167-193.

[5] Stachniss, C., Hahnel, D., & Burgard, W. (2004, September). *Exploration with active loop-closing for FastSLAM*. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 2, pp. 1505-1510). IEEE.

[6] https://www.f64.ro/dji-tello/p, accessed on 07.03.2023

[7] https://www.intelrealsense.com/depth-camera-d455/, accessed on 02.03.2023

[8] https://www.robofun.ro/kit-roboti/sasiu-tanc-robotic-metalic-cu-cauciucuri-mari. html, accessed on 02.03.2023

Sistem de operare vector robot pentru localizare și cartografiere

Această lucrare explorează utilizarea camerei Intel RealSense Depth Camera D455 în contextul localizării, navigării autonome și mapării mediilor complexe. Am investigat metode avansate de estimare a poziției și orientării roboților mobili, utilizând date de adâncime pentru a construi hărți detaliate și precise. Algoritmul de Localizare și Cartografiere Simultană (SLAM) implementat se bazează pe detectarea în timp real a modificărilor din mediul înconjurător, asigurând actualizarea continuă a hărții. Vectorul de robot, reprezentând starea acestuia, este evaluat în funcție de traiectoria parcursă, influențând semnificativ performanța navigării autonome. Rezultatele obținute demonstrează eficiența soluției propuse în diverse scenarii de testare.

Cuvinte cheie: *Intel RealSense Depth Camera D455, localizare, navigare autonomă, cartografiere, vector robot.*

Alexandra-Cătălina CIOCÎRLAN, PhD Student, Institute of Solid Mechanics, Robotics and Mechatronics, alexandra.ciocirlan@imsar.ro/katalina.alexandra@yahoo.com, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Luige VLĂDĂREANU, Senior researcher, Dr. Eng., Institute of Solid Mechanics, Robotics and Mechatronics, luige.vladareanu@imsar.ro, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Ionel-Alexandru GAL, Researcher CS II, Dr. Eng., Institute of Solid Mechanics, Robotics and Mechatronics, alexandru.gal@imsar.ro, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Bianca Rodica GHINOIU, PhD Student, Institute of Solid Mechanics, Robotics and Mechatronics, bianca.ghinoiu@imsar.ro, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Cristina MARILENA Nițu, Researcher CS II**,** Dr. Eng., Institute of Solid Mechanics, Robotics and Mechatronics, cristina.nitu@imsar.ro, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Marius PANDELEA, Dr. Eng., Institute of Solid Mechanics, Robotics and Mechatronics, marius.pandelea@imsar.ro, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.

Mihaiela ILIESCU, Senior researcher, Habil. Dr. Eng., **corresponding author**, Institute of Solid Mechanics, Robotics and Mechatronics, mihaiela.iliescu@imsar.ro; iliescumihaiela7@gmail.com, office@imsar.ro, +40 21 312 67 36, 15 Constantin Mille, Bucharest, Romania.