2024 6th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA) | 979-8-3315-3217-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SUMMA64428.2024.10803743

Autonomous Navigation of a Mobile Robot Using Overhead Camera and Computer Vision Methods for Time-Critical Tasks

Anatoly Tselishchev
Department of Applied Mathematics
Lipetsk State Technical University
Lipetsk, Russia
anatolysamaris@gmail.com

Diana Ustinova
Department of Applied Mathematics and Cybernetics
Petrozavodsk State University
Petrozavodsk, Russia
Bibedaa@yandex.ru

Dmitry Savinov
Institute of Advanced Technologies "School of X"
Don State Technical University
Rostov-on-Don, Russia
savinov.dima44@yandex.ru

***Abstract*—This paper describes the implementation of an intelligent navigation system for the mobile robot GFS-X equipped with a manipulator, aimed at solving a range of tasks on the arena within a limited time frame. The approach to route planning and adjustment using a map obtained from an overhead camera is examined. YOLOv11 is utilized to gather information about the location of objects and their identification.**

***Index Terms*—yolo, autonomous navigation, mobile robot, computer vision, overhead camera, object detection, image processing**

I. Introduction

Robotic systems are becoming a popular solution for performing tasks in hazardous environments and optimizing performance in dynamically changing conditions. This necessitates ongoing research and development in the field of intelligent control systems for robots to accomplish designated tasks. One of the most informative sensors available in robotic systems is the camera, which can be mounted on mobile robots or integrated into video systems with distributed cameras, facilitating the use of real-time computer vision and image processing algorithms. In this work, we utilize an overhead camera positioned above the test arena to construct an optimal route to the target and to rapidly adjust the route in response to significant changes in the surrounding environment, as well as an onboard camera to verify the successful completion of the task. Consequently, the intelligent system generates control commands and monitors the execution of tasks.

II. Task definition

The robot was assigned several tasks that it needed to complete in a maze within a limited timeframe of 2 minutes:

- Pressing the green button;
- Transferring cubes to the basket corresponding to robot's team;
- Moving a ball to the basket corresponding to robot's team.

Points are awarded for the successful completion of each task.

Additionally, there are actions that the robot is prohibited from undertaking:

- Exiting the boundaries of the arena;
- Colliding with the wall;
- Colliding with the opposing robot.

Points are deducted for these violations.

Consequently, it is essential to develop a strategy for task execution, outline optimal routes for the robot, and ensure efficient manipulation of objects.

The following sensors are available for task execution: two cameras (one with an overhead view and one mounted on the robot's body), an ultrasonic sensor, and three infrared sensors for distance measurement.

A manipulator mounted on the robot's body is utilized for object interaction.

III. Overhead camera navigation

The fundamental algorithm for executing tasks is object detection on the arena using the state-of-the-art architecture of the convolutional neural network [1] YOLOv11 [2]. By obtaining information about the detected objects and their positions, we can perform a variety of useful actions for robot control and trajectory planning.

A. Camera calibration

The upper camera is a wide-angle camera with a field of view of 102 degrees. One of the characteristics of such cameras is the presence of distortions known as the "fisheye effect." [3]

979-8-3315-3217-8/24/$31.00 ©2024 IEEE

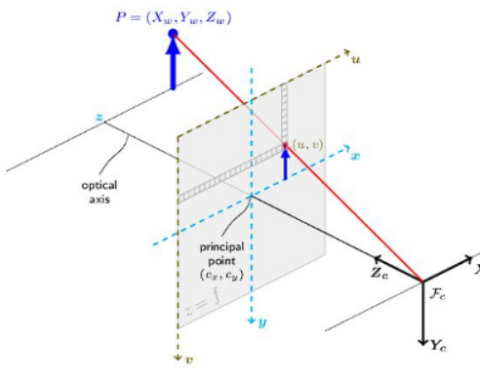

Fig. 1. OpenCV camera model.

These distortions can significantly impair the accuracy of calculations based on the positions of detected objects; therefore, prior to the utilization of the neural network, the frames undergo a calibration procedure [4].

The calibration is performed using a specific formula that takes into account the camera parameters. In particular, the formula may be expressed as follows:

$$K = \begin{pmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{pmatrix}$$

- fx and fy are the focal lengths of the camera along the x and y axes, respectively, which determine the scale of the image. - cx and cy are the coordinates of the image center, indicating the location of the principal point in the image (typically the center of the frame).

$$\text{distortion} = \begin{pmatrix} k1 & k2 & p1 & p2 & k3 \end{pmatrix}$$

- $k1, k2, k3$ are the radial distortion coefficients that account for distortions occurring at the edges of the image (e.g., "pincushion" or "barrel" distortion). - $p1, p2$ are the tangential distortion coefficients that consider the shift of the optical axis, causing distortions in the image when the optical system is not perfectly aligned.

B. Creating a dataset

The dataset [5] was created based on calibrated images from the upper camera. To increase the training sample and enhance the model's ability to generalize features, augmentation techniques such as noise and shear were applied. A total of approximately 500 images were initially captured, which increased to 1100 after augmentation.

The annotation of the dataset using objects that are directly present during task execution ensures the highest possible accuracy for the detector model.

The model was trained for 300 epochs, achieving a mAP50 metric value of 0.88, which is a good indicator of the model's accuracy, considering the impact of distortions in the images.

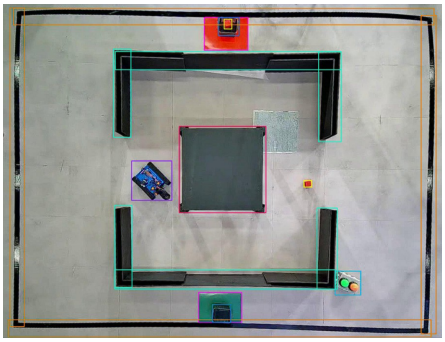

Fig. 2. Example of a labeled frame.

C. Route planning

To effectively plan [6] the route, a graph was developed, representing an abstraction of the space in which navigation [7] takes place. The simplest approach to constructing such a graph is to use each pixel of the field as a vertex. However, considering the large number of pixels in an image, such a grid graph would require significant amounts of memory and would include redundant information.

Instead, a more practical solution is to select only key points in the space as the vertices of the graph [8]. The graph is marked based on distances from static objects, such as walls, boundaries, and blind spots. This results in a graph consisting of no more than 50 vertices.

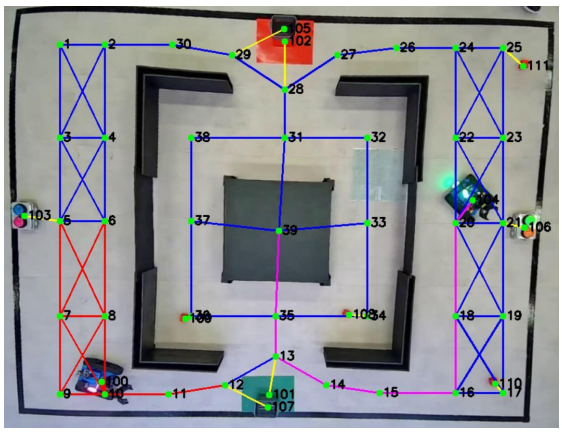

Fig. 3. Graph map initialization.

In the considered space, there are also dynamic objects that need to be taken into account in the graph structure. These dynamic vertices are connected to static ones, ensuring minimal changes in the graph when objects move within the field.

To solve the problem of finding the shortest path in the graph, the A^* algorithm was used. This algorithm requires a weighted graph, which is why all edges of the graph were assigned a weight of 1. The A^* algorithm has good asymptotic complexity, $O(|E|log(V))$, where E is the number of edges

and V is the number of vertices. This makes it efficient for finding optimal routes in various scenarios.

The considered space also contains an enemy robot, and colliding with it is unacceptable. Therefore, all edges directly connecting vertices located near this robot were assigned a high weight. This allows the A^* algorithm to avoid routes that intersect or approach the enemy object, thus ensuring safe navigation and successful task execution.

D. Calculating the robot's rotation

Since the robot does not have instruments to measure its position, the upper camera was utilized for the approximate calculation of the robot's rotation angle relative to the camera [9]. This enables the division of the route into sub-tasks such as "Move forward by N centimeters" and "Perform a turn of N degrees."

Fig. 4. Example of robot's angle approximation.

To solve this problem, we will find the bounding box containing our robot, and then we will determine its center. Our robot has a LED of a specific color located at the rear of the robot's body, perpendicular to the robot's movement. Using thresholding and contour detection, we find the centroid of the LED. Then we will find the vector from the centroid of the LED to the center of the robot. By calculating the angle between this vector and the vector along the x-axis of the image, we will obtain an approximate value of the robot's rotation angle relative to the camera, which allows us to control the robot's turning angle.

IV. Robot Movement

For the task of moving the robot, the main types of commands were identified, which are generated on the server (a laptop in the local network connected to the camera and performing the necessary computations) and sent to the Raspberry Pi single-board computer located within the robot:

- Forward movement for a specified distance. This command facilitates the robot's movement over a predetermined distance. Carefully selected coefficients are employed to enable smooth acceleration and deceleration, thereby reducing the error along the perpendicular axis and enhancing accuracy.
- Rotation by a specified angle. This command is used to alter the robot's direction of movement by a defined angle. The angle is determined using data from the overhead camera, and iterative adjustments ensure high precision in the rotation.
- Movement along a wall for a specified distance. This command allows the robot to traverse parallel to the wall, which increases the speed and confidence of navigation. The robot utilizes the wall as a reference, thereby reducing the risk of collisions, while a versatile regulation method ensures stable movement.

A. Movement regulation

The control of the robot's movement is complicated by the absence of odometry and limited data: there are no encoders or IMU, and delays from the overhead camera hinder timely adjustments. The robot relies on ultrasonic and infrared sensors. Standard PID [10] controllers under such conditions lead to oscillatory motion: as the robot corrects the error, it alters its direction vector, causing it to move either towards or away from the wall [11], rather than parallel to it. This increases the risk of collision and the loss of signal from the ultrasonic sensor.

To address these issues, an unconventional solution was implemented — the use of a negative integral controller with buffering. This approach involves the controller accumulating error as the robot moves. When the robot approaches the wall [12] and the error approaches zero, the accumulated negative error starts to exert a significant effect in the opposite direction, thereby aligning the robot along the wall.

Unlike a classic integral controller, which only increases its impact with accumulated error, the negative integral controller allows for preemptive compensation of deviations. This prevents oscillatory motion of the robot, enabling it to move parallel to the wall.

An iteration of the calculation of the control input to the motor is demonstrated. Firstly we calculate PI-component of the regulator:

$$PI_{output} = K_p \cdot e(t) - K_i \cdot \sum_{i=0}^{239} e_{buffer}[i] \tag{1}$$

where PI_{output} is output value of PI-regulator, K_p is a proportional coefficient, $e(t)$ is a current error, K_i is an integral coefficient, e_{buffer} is an array of errors where the last 240 error values are stored.

Then we normalize the value of PI_{output} using (2) and interpolate in the range between -20 and 20 using (3).

$$PI_{norm} = \max(-100, \min(100, PI_{output})) \tag{2}$$

where PI_{norm} is a value between -100 and 100.

$$PI_{interp} = \frac{(PI_{norm} + 100)}{200} \cdot 40 - 20 \tag{3}$$

where PI_{interp} is interpolated PI-regulator value between -20 and 20.

The full formula of the control input to the motor is:

$$Motor\_pwm = base\_velocity + PI_{interp} \quad (4)$$

where $base\_velocity$ is the velocity that is when motor has no regulations.

B. Manipulator

For the successful execution of object manipulation, it is essential for the robot to maintain a specified distance from the object and to be correctly oriented towards it. This is achieved through the regulation of the rotation angle via cameras and the previously described motion commands. The manipulator control system ensures the execution of sequential commands determined during experiments, thereby guaranteeing successful grasping and manipulation of objects. This approach provides high precision in manipulations even when sensor data regarding the robot's position and state is limited.

V. Conclusion

Despite the limited set of sensors, it is possible to perform tasks with a mobile robot based on an overhead camera that provides for all strategic and tactical levels of the robot's operation, while identifying an appropriate controller to ensure stable movement of the robot in accordance with the commands it receives. Although the testing of the intelligent system was conducted in a gaming arena environment, such a solution can be applied in real-world scenarios with more complex conditions and a greater volume of input data.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 10.1109/CVPR.2016.91, 2016.

[2] Juan Terven, Diana-Margarita Cordova-Esparza, Julio-Alejandro Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction," 2023, 1680-1716, doi: 10.3390/make5040083.

[3] Christian Bräuer-Burchardt, Klaus Voss, "A new algorithm to correct fish-eye and strong wide-angle lens distortion from single images," 2001 IEEE Trans Image Process, 1, 225-228, 10.1109/ICIP.2001.958994.

[4] Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.

[5] Cristian Rocco, "Synthetic Dataset Creation For Computer Vision Application: Pipeline Proposal," 2021, 10.13140/RG.2.2.12115.25126.

[6] M. Elbanhawi, M. Simic, "Sampling-Based Robot Motion Planning: A Review," in IEEE Access, vol. 2, pp. 56-77, 2014, doi: 10.1109/ACCESS.2014.2302442.

[7] Edwin Olson, "AprilTag: A robust and flexible visual fiducial system," 2011, IEEE International Conference on Robotics and Automation, 3400 - 3407, 10.1109/ICRA.2011.5979561.

[8] Fareh R, Baziyad M, Rahman MH, Rabie T, Bettayeb M., "Investigating Reduced Path Planning Strategy for Differential Wheeled Mobile Robot," Robotica, 2020, 38(2):235-255, doi:10.1017/S0263574719000572.

[9] Devi Parikh, Gavin Jancke, "Localization and Segmentation of A 2D High Capacity Color Barcode," 2008, 1 - 6. 10.1109/WACV.2008.4544033.

[10] Hendril Purnama, Tole Sutikno, Srinivasan Alavandar, Nuryono Satya Widodo. (2019). Efficient PID Controller based Hexapod Wall Following Robot. 10.23919/EECSI48112.2019.8976964.

[11] Farkh Rihem, Khaled Aljaloud, "Vision Navigation Based PID Control for Line Tracking Robot," 2023, Intelligent Automation and Soft Computing, 35, 901-911, 10.32604/iasc.2023.027614.

[12] Heru Suwoyo, Ferryawan Kristanto, "Performance of a Wall-Following Robot Controlled by a PID-BA using Bat Algorithm Approach," 2022, International Journal of Engineering Continuity, 1, 56-71, 10.58291/ijec.v1i1.39.