A Mobile Robot Based on Edge AI

Ricardo C. Câmara de M. Santos[1], Mateus Coelho Silva[1], Rodrigo Lucas Santos[1], Emerson Klippel[1], and Ricardo A. R. Oliveira[1]

[1]Departmento de Computação - DECOM
Universidade Federal de Ouro Preto - UFOP
Ouro Preto - MG, Brazil

{ricardocamara03,mateuscoelho.ccom,rodrigolucassantosss,

eklippel,rrabelo}@gmail.com

Abstract. *Recent advancements in technology have enabled the emergence of Industry 4.0, this term is used to describe the integration of various advanced technologies such as artificial intelligence and robotics in industrial settings. The presence of defective products during production incurs additional costs, and traditional manual methods of equipment inspection prove to be inefficient in such a dynamic environment. In this study, we introduce a robot designed specifically for this scenario, capable of performing tasks that require autonomous movement to specific areas of an industrial plant. To achieve this, we employ the concept of Edge AI, applying artificial intelligence on a localized edge computing device. The robot utilizes computer vision through the state-of-the-art YOLOv7 CNN and incorporates feedback control to facilitate its locomotion. The hardware components of this robot include a Jetson Xavier NX, Raspberry Pi 4, a camera, and a LIDAR. Additionally, we conducted a comprehensive performance analysis of the object detection method, measuring metrics such as frames per second (FPS), CPU and GPU consumption, and RAM usage.*

1. INTRODUCTION

The advancements in technology in the 21st century have facilitated the emergence of the fourth industrial revolution, commonly known as industry 4.0. This revolution entails the integration of various digital technologies like artificial intelligence, Internet of Things, and blockchain, as well as other technological advancements such as robotics, digital twins, and cyber-physical systems, all within industrial production environments [Hassoun et al. 2022]. As a result, there is an increasing trend towards enhanced collaboration between robots and human workers in industrial settings, with the aim of improving safety, flexibility, and productivity.

The presence of defective products not only increases costs but also disrupts manufacturing processes. Efficient inspection routines for proactive anomaly and problem detection play a crucial role in the effective maintenance of industrial plants. Traditional manual inspection methods are associated with heavy workloads, risks, and limited efficiency. Moreover, they fail to consistently meet the rising quality standards of modern production means. Consequently, the focus of research worldwide has turned to robots to mitigate these issues [Ebayyeh and Mousavi 2020].

A key component of industry 4.0 is the flexible manufacturing system, an advanced production system that interconnects machines, workstations, and logistical equipment throughout the manufacturing process. This system is designed to handle highly complex manufacturing tasks with significant topological diversity, ensuring timely delivery and minimal manufacturing costs despite frequent changes [Florescu and Barabas 2020]. Accordingly, the need for autonomous movement of the automated inspection agent arises, enabling it to navigate autonomously to inspection sites.

To address this need, we propose an autonomous locomotion method based on Edge AI technology. Our objective is to introduce a robot capable of autonomously moving to specific locations within the production plant. The robot incorporates two sensors: a camera and a LIDAR. The camera captures images, which are then processed using a CNN for object detection [Albawi et al. 2017a]. This enables the robot to identify target positions. The LIDAR, on the other hand, measures the robot's distance to the target position and facilitates its navigation towards the required locations.

2. THEORETICAL REFERENCES

This section provides the essential theoretical framework required for the development of the robot and its autonomous locomotion methodology.

2.1. Autonomous Mobile Robots

An autonomous mobile robot (AMR) is a system specifically designed to operate and navigate in environments that are unpredictable and partially unknown. The primary objective of an AMR is to navigate continuously and avoid collisions with obstacles within a confined environment that is known to the robot [Ishikawa 1991]. The key characteristic of an AMR is its ability to navigate and move without significant human intervention, following a predefined path, whether it is indoors or outdoors.

The foundations of mobile robotics encompass three essential components: locomotion, perception, and navigation [Alatise and Hancke 2020]. In indoor environments, mobile robots typically rely on various sensors, including floor plans, sonar-based localization systems, and inertial measurement units (IMUs). These sensors enable the robot to establish an internal representation of its environment, which is crucial for its proper functioning.

2.2. Convolutional Neural Networks

Machine Learning is a research field within the broader domain of Artificial Intelligence (AI) that focuses on developing techniques capable of extracting meaningful concepts from data samples. One commonly used method in Machine Learning is artificial neural networks (ANNs), which can be understood as non-linear mathematical models used for prediction or content generation based on input data. ANNs consist of interconnected layers of neurons, which serve as the fundamental units of the network. The connections between neurons, known as synapses, transmit the output signal of each neuron to the input of the next layer, allowing information to propagate throughout the network. During the training phase, the synapses are assigned weights that enable the model to appropriately process the information [Krogh 2008].

Deep Learning refers to the utilization of ANNs with a significant number of layers. The interest in having deeper hidden layers has shown superior performance compared to classical methods, particularly in pattern recognition applications [Albawi et al. 2017b]. One widely known type of deep neural network is the Convolutional Neural Network (CNN). CNNs derive their name from the convolution operation, which is a mathematical operation performed between matrices. CNNs excel in various machine learning tasks and exhibit multiple layers that detect hierarchical patterns. These layers include convolutional layers, which employ filters to analyze input images and generate feature maps as output. Pooling layers are used to reduce information from the previous layers, while upsampling layers can be employed to enhance information. CNNs find extensive application in tasks such as object detection, particularly in the analysis of image data [Arulprakash and Aruldoss 2022].

2.3. Edge AI

Current applications of artificial intelligence, such as deep learning, heavily rely on substantial computational power for their execution. This poses a technical challenge when it comes to adapting artificial intelligence applications for embedded devices [Li et al. 2019]. Edge AI offers a novel approach that leverages the concepts of Edge Computing to develop artificial intelligence applications.

Edge computing and artificial intelligence have inherently conflicting requirements. AI typically demands more processing power, whereas edge computing emphasizes hardware miniaturization and increased mobility. Despite this challenge, there is a growing trend of combining these two concepts, particularly in the domain of mobile edge computing [Chen and Ran 2019]. The concept of Edge AI [Wang et al. 2019] aims to address this challenge by adapting AI algorithms and hardware models to enable the development of applications within this context.

3. RELATED WORKS

In the study by Szrek et al. [Szrek et al. 2022], a mobile inspection platform based on autonomous Unmanned Ground Vehicles (UGVs) was proposed. The platform utilized various sensors such as RGB image, sound, and gas sensors. The system employed a laboratory setting with a dedicated test cart for research purposes. The process was divided into inspection planning and execution, controlled by a module based on an STM32 microcontroller. The localization and mapping were achieved using a SLAM algorithm, with communication between modules facilitated by MQTT. This approach relied on predetermined algorithms, sensor fusion, and heavy modules, but did not incorporate artificial intelligence (AI) as in the current work.

Dandurand et al. [Dandurand et al. 2022] introduced an all-weather autonomous inspection robot for electrical substations in severe winter conditions. The system utilized real-time kinematic positioning (RTK) for localization and an IMU for attitude awareness. Although the robot was designed for inspection purposes, it did not rely on AI and employed fixed algorithms for its tasks, distinguishing it from the current approach.

Salimpour et al. [Salimpour et al. 2022] proposed a deep-learning framework for monitoring anomalies and changes in environments at the pixel level using image pairs. The study employed neural network architectures such as SuperPoint and Superglue for

change detection. While they utilized a terrestrial robot with multiple sensors, including RGB camera and a LIDAR, the focus of their work differed from the current study, which emphasizes real-time constraints and hardware profiling.

Cheng et al. [Cheng and Xiang 2020] developed an autonomous trail-type inspection robot for monitoring electrical distribution rooms. The robot was equipped with various sensors, including an optical zoom camera, a thermal imaging camera, and a partial discharge detector. Their system relied on computer vision techniques for recognition tasks, but did not incorporate AI and operated on fixed trails, unlike the current work.

Hercitk et al. [Hercik et al. 2022] presented an autonomous industrial mobile robot designed for cooperation with the production line and logistical tasks. They utilized a commercial robot from Mobile Industrial Robots (MiR) and employed a centralized control station (MiR FLEET) and an I/O module (MiR WISE) for communication. This application had a different purpose and operational approach compared to the current study.

While several applications share similarities with the current work in terms of autonomous navigation and inspection tasks, they differ in various aspects. The proposed approach in this work focuses on a lean system with a minimal sensor set, relying on AI for inspection tasks in an open space. The study also examines real-time constraints and hardware profiling to assess how the Edge platform supports the proposed tasks.

4. METHODOLOGY

In this section, we will provide a comprehensive overview of the functioning of our Autonomous Mobile Robot (AMR). We will delve into its physical characteristics, including its mechanical design, the approach employed for autonomous locomotion, and the hardware components involved. These components are actuators, controllers, sensors, and the crucial Edge AI device that facilitates intelligent decision-making.

4.1. Physical attributes

4.1.1. Mechanics

The robot is enclosed within a metal housing that takes the shape of a parallelepiped, featuring an upper part that can be opened like a trunk. Its physical dimensions are as follows: 22cm width, 34cm length, and 20cm height. To facilitate its movement, the robot is equipped with four omnidirectional wheels positioned at the corners of the housing. Figure 1 shows the robot and provides a visual representation of its dimensions. The camera is positioned at the upper front section of the robot, while the LIDAR is situated in the upper part. The overall weight of the robot is 3.2kg.

The implementation of omnidirectional wheels offers high maneuverability. However, the mechanical structure of these wheels is comparatively complex, necessitating the deployment of separate steering and control systems for each individual wheel. The specific type of omnidirectional wheels employed in the robot is known as mecanum wheels.

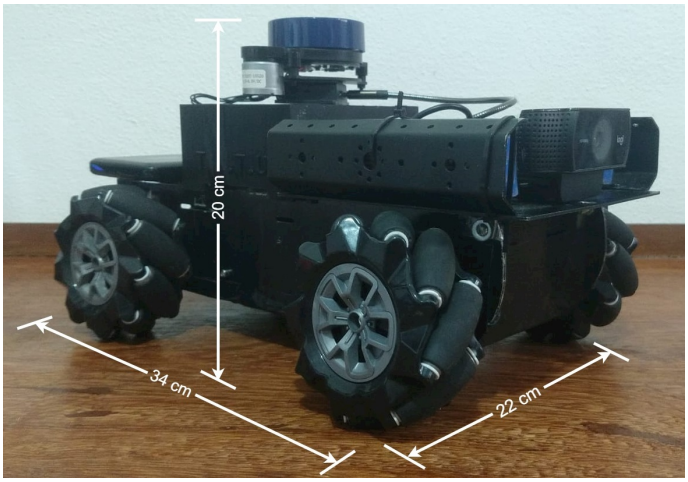

Figure 1. Robot dimensions.

4.1.2. Hardware

The robot utilizes four individual DC motors with a voltage of 12 volts and a speed of 200 rpm, with each motor directly connected to an individual wheel. For motor control, the robot employs a combination of hardware components: the Raspberry Pi 4 Model B, in conjunction with the Stepper Motor HAT V0.1, which serves as the motor driver.

The Raspberry Pi 4 Model B boasts a 64-Bit quad-core processor, capable of running at frequencies of up to 1.5GHz, accompanied by 4GB of RAM. It supports dual-band 2.4/5.0 GHz wireless networking, Bluetooth 5.0/BLE, True Gigabit Ethernet, USB 3.0, and features USB-C power capability. The controller is a composite hardware system, consisting of the Raspberry Pi as the high-level controller and the Stepper Motor HAT as the low-level controller.

The Edge AI device employed is the NVIDIA Jetson Xavier NX development kit, equipped with a 6-core NVIDIA Carmel ARM v8.2 64-bit CPU. It integrates a GPU with 384 NVIDIA CUDA Cores, 48 Tensor Cores, and 2 NVDLA (NVIDIA Deep Learning Accelerator), complemented by 8GB of LPDDR4x RAM. The connectivity options include Gigabit Ethernet and a WiFi/Bluetooth interface (M.2 Key E), along with 4 USB 3.1 ports and HDMI/DP video outputs. The robot incorporates the YDLIDAR X2 as a sensing unit, capable of operating at a range frequency of 3000Hz and a scanning frequency of 5-8Hz, covering a range of 0.12-8m. Additionally, it features a 30FPS (frames per second) Logitech C920 camera with full HD resolution. The Raspberry Pi is powered by a 10000mAh power bank, which supplies a DC5V 2.1A output. The robot also includes two 11.1V Lipo batteries.

The sensors are positioned on the upper external part of the robot and are connected to the Jetson. The Jetson functions as an Edge AI platform, processing the data from the sensors and executing the navigation methodology. A direct Ethernet connection links the Jetson to the controllers (Raspberry Pi + Stepper Motor HAT), enabling the Raspberry Pi to transmit individual power commands to each motor in real-time. Table 1 provides an overview of the functions performed by each hardware component, illustrating the hierarchical relationship between the different levels.

ROLE	DEVICES
Actuators	4 DC Motors
Low Level Controller	Stepper Motor HAT
High Level Controller	Raspberry Pi 4
Edge AI Device	NVIDIA Jetson Xavier NX
Sensing	Câmera + LIDAR

Table 1. Table of hardware roles.

4.1.3. Locomotion Method

The robot's method of locomotion is based on image object detection using a Convolutional Neural Network (CNN), specifically the YOLOv7 model, which allows for the identification of objects in images [Wang et al. 2022]. Once an object is detected, the robot moves towards it until it reaches a close enough proximity to perform a desired task, such as inspection. The objects detected in the image serve as target locations for the robot's positioning.

One notable advantage of using YOLOv7 is its ease of training with images of the desired target locations. In this work, traffic cone images are utilized to simulate these target locations, although other specific objects or machines in an industrial plant could also be used.

The locomotion method follows a state machine logic, depicted in Figure 2. Each state within the state machine is explained in detail, along with the transitions between states.

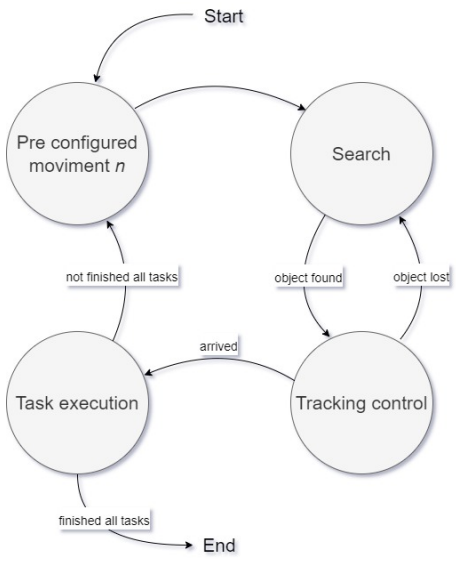

Figure 2. State machine for the locomotion method.

Pre configured movement: Our locomotion method takes as input a set N that contains lists of previously configured sequential movements. This step is currently used by the robot for obstacle avoidance. Since the operating plan is known in advance, a sequence of pre-configured movements can be employed if the robot needs to deviate from an obstacle until the camera can detect the next target location. These movement

sequences consist of predefined movements in the four directions, each executed for a predetermined duration. During this state, the robot executes the list of movements n_i for each entry in the set. If there are no obstacles present at the operation site, the set of previously configured movements will be empty. Once the execution of the movements is completed, the robot transitions to the search state.

Search: In this state, the robot is actively searching for the next target location. It achieves this by rotating around its own axis until it detects the object that represents the following objective location. The rotation is performed in a non-continuous manner, with short pauses between rotations to ensure that the camera image remains clear and avoid blurring, which could make object detection challenging. Once the robot successfully detects the object, it will start moving towards it. This task of moving towards the detected object is carried out in the tracking control stage.

Tracking control: In this state, the robot actively moves towards the target location using a feedback control method. The variable being controlled in this case is the horizontal center of the detected object. The robot compares the horizontal center value of the object with the desired value, which is the center of the image. If the object is positioned more to the right of the image, the robot applies more power to the wheels on the left side to correct its trajectory and move towards the target location and vice versa.

The percentage of power applied to the right side motors RP is defined by Equation 1, and the percentage of total power applied to the left side motors LP is defined by Equation 2. These equations utilize the horizontal distances from the center of the detected object to the left ld and right rd edges of the image. Figure 3 illustrates the cone detection and provides visual representations of the measurements used to calculate the power applied to each motor.

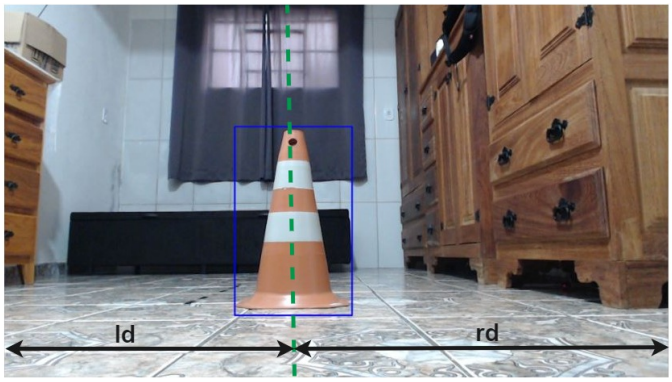

Figure 3. Cone detection and the measures used to define the motors power at tracking control stage.

$$RP = min(200 * (rd/iw), 100) \tag{1}$$

$$LP = min(200 * (ld/iw), 100) \tag{2}$$

The tracking control state utilizes a feedback control model, which is represented in Figure 4. In this model, r represents the reference value, which in this case is the

horizontal center of the image. The error e is calculated as the difference between the horizontal center of the image and the horizontal center of the detected object. The control action u is determined using Equations 1 and 2 to calculate the power applied to each motor. The variable y is updated based on the control action, and Ym represents the new measurement of the variable. Figure 4 provides a visual representation of this feedback control model used in the tracking control state.

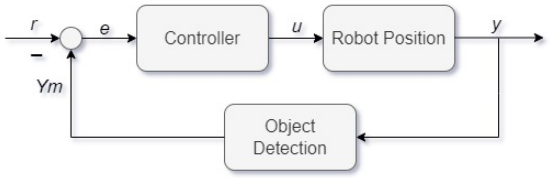

Figure 4. Traditional feedback control model applied to our AMR.

The robot remains in the tracking control state until one of two conditions is satisfied. The first condition is when the robot reaches the target location, which is determined by the LIDAR reading indicating close proximity to the detected object. If this condition is met, the robot transitions to the task execution state. The second condition is when the robot loses sight of the detected object. In such a case, the robot returns to the search state to locate the next target location. These conditions and state transitions are illustrated in Figure 2.

Task execution: The task execution in this work is a generic abstraction that can encompass various tasks performed by the robot in an industrial plant. These tasks may include visual or physical inspections of machines, data reading, material or tool delivery, and other relevant operations. Once the task execution is completed, the robot will conclude its work if all the tasks have been executed. Otherwise, it will transition to the pre-configured movement state, as depicted in Figure 2.

5. EXPERIMENTAL RESULTS

To conduct our tests, we trained YOLOv7 using a dataset of traffic cone images. We collected a total of 584 images from the internet, with 537 images used for training and 47 for validation. The YOLOv7-Tiny model was utilized for the training process, implemented with PyTorch and GPU support. Our application was developed in Python3, utilizing the OpenCV library.

To validate our methodology, we conducted tests in a laboratory setting using four traffic cones as target locations for the robot's locomotion. The robot demonstrated full capability in reaching all the objective locations in sequence. The use of pre-configured movements proved effective in avoiding obstacles, but it requires prior knowledge of the operational site. Occlusion of the detection object can also be overcome using the list of pre-configured motions. However, to avoid relying solely on predetermined movements, the implementation of a navigation method incorporating LIDAR data and a SLAM algorithm, such as SLAM (Simultaneous Localization and Mapping), would be necessary.

For our experiments, we used Ubuntu 18.04 LTS as the Jetson's operating system. In idle state, the average RAM consumption is 19%, CPU consumption is 1%, and GPU consumption is 0%.

To evaluate the performance of our Edge AI device, we conducted profiling and FPS (Frames Per Second) tests. Profiling allowed us to measure the hardware usage of the application, indicating the potential for adding more sensors and processing methods. FPS, on the other hand, provided insight into the speed at which the system processes information from the environment. The measurements for profiling included the average usage of the 6 CPU cores, average RAM usage, and average GPU usage.

We performed two types of implementations for the tests: sequential and parallel. In the sequential implementation, object detection occurs after image capture, while in the parallel implementation, capture and detection happen concurrently and in synchronized threads, ensuring each frame is processed only once. We tested these implementations using three different image capture resolutions: 720x480, 1280x720, and 1920x1080. We used this resolutions because they are standard when dealing with images. The power mode used for the Jetson was the 20 watts mode, utilizing all six cores.

Table 2 displays the average FPS rates for each tested implementation and resolution. Notably, the parallel implementation with a resolution of 720x480 achieved the highest FPS rate. This resolution was the only one where the average detection time was lower than the average capture time. Consequently, it was the only configuration where the detection of the object itself acted as the performance bottleneck. The parallel implementation outperformed the sequential implementation due to the simultaneous capture and detection processes.

On the other hand, the sequential implementation couldn't achieve the same speed since the total FPS rate is influenced by the sum of the capture time and detection time. While the average detection time doesn't vary significantly with increasing resolution, the capture time has a more pronounced effect. This outcome arises from the fact that YOLOv7's detection process operates on a default image size of 640x640. As a result, every image submitted for detection is resized to these dimensions. For the resolutions of 1280x720 and 1920x1080, we observed that the bottleneck of the parallel method shifted to the image capture time, which was comparable to the combined capture and detection times of the sequential method.

	Res 720x480		Res 1280x720		Res 1920x1080	
	SEQ.	PARL	SEQ.	PARL	SEQ.	PARL
FPS	19.99	**25.47**	9.99	9.99	4.99	4.99
Process t. (ms)	39.01	**38.79**	40.69	41.19	42.59	42.41
Capture t. (ms)	10.89	**12.07**	59.22	99.97	157.30	200.09

Table 2. Table of frames per second (FPS), capture time, and process time. The first row includes different resolutions, with 'SEQ.' denoting sequential and 'PARL.' denoting parallel processing.

Figures 5, 6, and 7 illustrate the hardware consumption over a 1-second window with 50 samples taken at intervals of 20 milliseconds. These figures provide a visual representation of the hardware usage during the tests.

Figure 5 presents the profiling results specifically for the 720x480 resolution. It is worth noting that in the sequential implementation, which achieved the highest FPS among all the tests, there is no idle time for the GPU. Conversely, in the parallel implementation, there are periods of GPU idleness where its consumption drops to 0%.

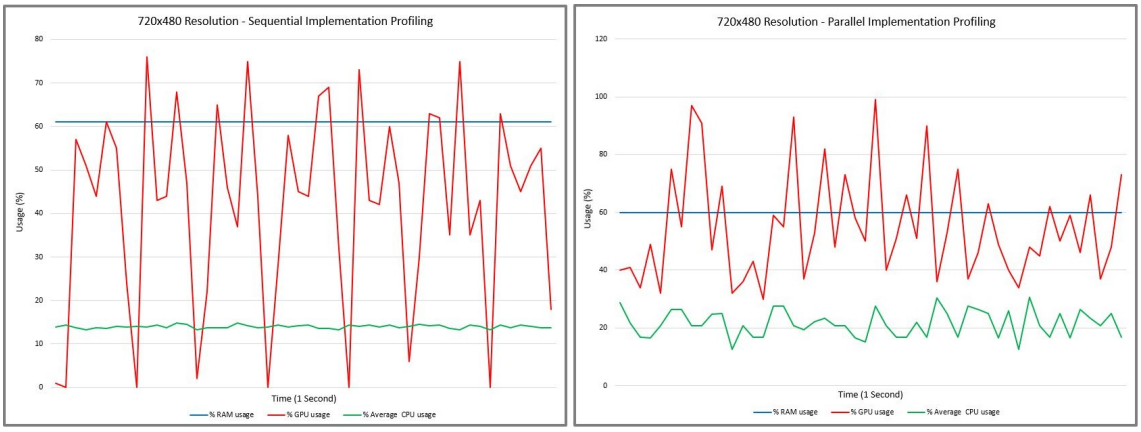

Figure 5. Profiling results for 720x480 resolution.

Additionally, the average CPU consumption was higher in the parallel implementation compared to the sequential implementation.

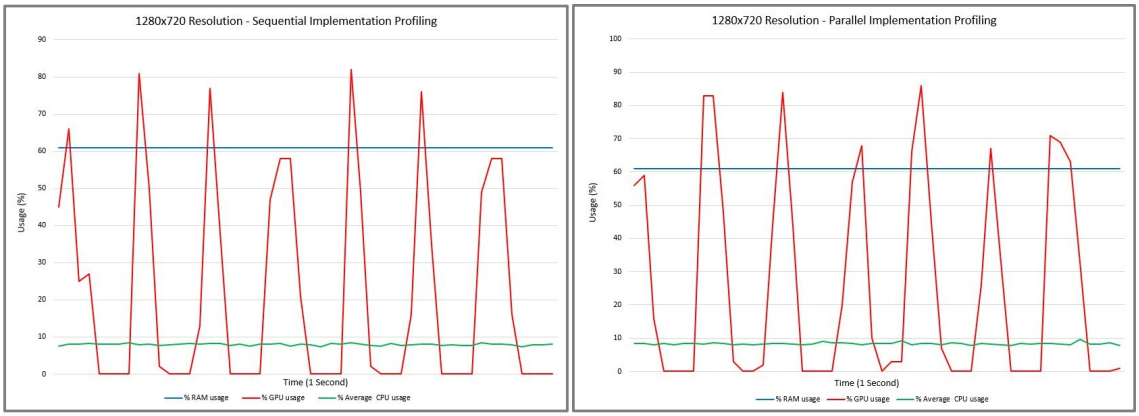

Figure 6. Profiling results for 1280x720 resolution.

In Figure 6, the profiling results for the 1280x720 resolution are depicted. Both the sequential and parallel implementations exhibit instances of GPU idleness, and their average CPU consumption is relatively similar.

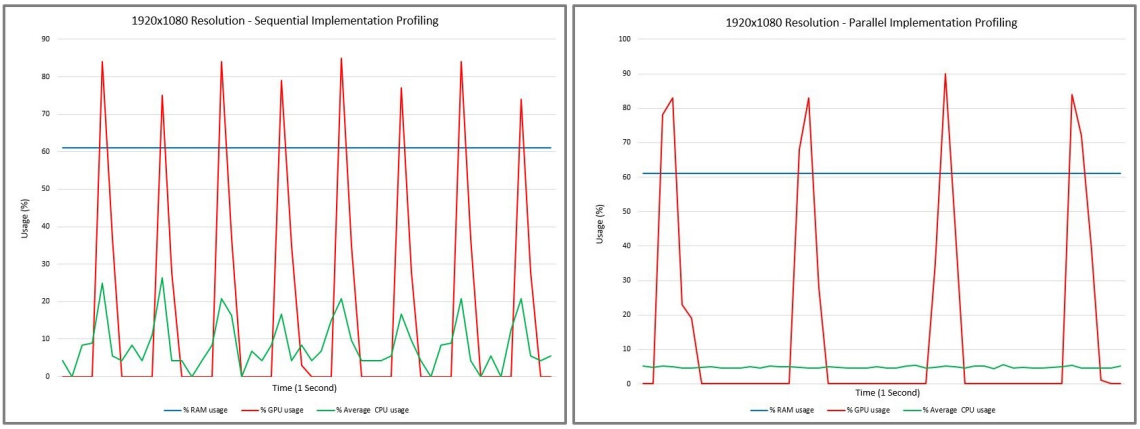

Figure 7. Profiling results for 1920x1080 resolution.

In Figure 7, the profiling results for the 1920x1080 resolution are displayed. Even in this resolution, there are instances of GPU idleness. The parallel implementation showcased the lowest average CPU consumption, which explains why it had the longest capture time compared to other experiments. The sequential implementation exhibited more significant variation in average CPU usage. The average RAM usage remained constant at 61% across all tests.

6. CONCLUSIONS

In this study, we introduce a robot designed for precise movement within an industrial plant, incorporating the concept of Edge AI into its architecture. The robot's navigation is based on a state machine framework, utilizing computer vision and feedback control mechanisms.

The proposed solution uses computer vision techniques to detect specific locations within the industrial environment. This detection process relies on a CNN, specifically YOLOv7. Once the target location is identified, feedback control is employed to guide the robot towards the desired destination. Laboratory testing confirmed the robot's ability to perform movement tasks successfully under controlled conditions.

To assess system performance, a series of tests were conducted to evaluate image capture and object detection across different camera resolutions and implementation approaches, including sequential and parallel methods. The impact of these variations on object detection performance was analyzed.

The highest achieved frame rate was 25 FPS, achieved with the parallel implementation and a resolution of 720x480. This finding indicates that YOLOv7 meets real-time requirements for object detection on the Jetson Xavier NX platform. Additionally, a comprehensive profile of the tests was conducted, examining CPU, RAM, and GPU utilization for each resolution and implementation type.

Future work encompasses several areas. Firstly, validation in real-world environments will provide insights into the robot's performance under more challenging conditions, such as mining scenarios, workshops, and warehouses. Integration of sensor fusion techniques could enable autonomous navigation without relying on explicit movement data inputs. Additionally, the application of reinforcement learning and other AI techniques, as well as the incorporation of probabilistic elements into the state machine using Markov Chains, holds potential for enhancing the robot's overall functionality.

References

Alatise, M. B. and Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017a). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017b). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6.

Arulprakash, E. and Aruldoss, M. (2022). A study on generic object detection with emphasis on future research directions. *Journal of King Saud University-Computer and Information Sciences*, 34(9):7347–7365.

Chen, J. and Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674.

Cheng, M. and Xiang, D. (2020). The design and application of a track-type autonomous inspection robot for electrical distribution room. *Robotica*, 38(2):185–206.

Dandurand, P., Beaudry, J., Hébert, C., Mongenot, P., Bourque, J., and Hovington, S. (2022). All-weather autonomous inspection robot for electrical substations. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 303–308. IEEE.

Ebayyeh, A. A. R. M. A. and Mousavi, A. (2020). A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry. *IEEE Access*, 8:183192–183271.

Florescu, A. and Barabas, S. A. (2020). Modeling and simulation of a flexible manufacturing system—a basic component of industry 4.0. *Applied sciences*, 10(22):8300.

Hassoun, A., Aït-Kaddour, A., Abu-Mahfouz, A. M., Rathod, N. B., Bader, F., Barba, F. J., Biancolillo, A., Cropotova, J., Galanakis, C. M., Jambrak, A. R., et al. (2022). The fourth industrial revolution in the food industry—part i: Industry 4.0 technologies. *Critical Reviews in Food Science and Nutrition*, pages 1–17.

Hercik, R., Byrtus, R., Jaros, R., and Koziorek, J. (2022). Implementation of autonomous mobile robot in smartfactory. *Applied Sciences*, 12(17):8912.

Ishikawa, S. (1991). A method of indoor mobile robot navigation by using fuzzy control. In *Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91*, pages 1013–1018. IEEE.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 26(2):195–197.

Li, E., Zeng, L., Zhou, Z., and Chen, X. (2019). Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457.

Salimpour, S., Queralta, J. P., and Westerlund, T. (2022). Self-calibrating anomaly and change detection for autonomous inspection robots. *arXiv preprint arXiv:2209.02379*.

Szrek, J., Jakubiak, J., and Zimroz, R. (2022). A mobile robot-based system for automatic inspection of belt conveyors in mining industry. *Energies*, 15(1):327.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., and Chen, M. (2019). In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165.