The 10h IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2019
5-7 June, 2019, Leeds, United Kingdom

Performance Evaluation of Real-Time System for Vision-Based Navigation of Small Autonomous Mobile Robots

Yehor Boltov[1], Inna Skarga-Bandurova[1,2], Igor Kotsiuba[2], Mykhailo Hrushka[1], Gennady Krivoulya[3], Rostyslav Siriak[1]
[1]Volodymyr Dahl East Ukrainian National University, Severodonetsk, Ukraine, pritexnet@gmail.com, skarga-bandurova@snu.edu.ua, mikkgru@yahoo.com, hashem.r@gmail.com
[2]G.E. Pukhov Institute for Modeling in Energy Engineering National Academy of Sciences of Ukraine, Kyiv, Ukraine, i.kotsiuba@gmail.com
[3] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, krivoulya@kture.kharkov.ua

Abstract—**Real-time operation systems (RTOS) have become very important to the development of autonomous mobile robots. The choice of RTOS has tremendous sway with processor utilization, response time, and real-time jitter. In this paper, we present experimental trials and analyze the feasibility of RTOS on a single-board computer for image recognition and vision-based navigation of small autonomous robot. Several real-time (RT) patches Linux frequently used not only in robotics are implemented and tested on the Raspberry Pi2 equipped with a native camera board. To study the speed of image recognition (classification) OpenCV library was used. Test results show that the RT Patch Linux can produce higher throughput compared to Xenomai, but it can be seen that RT systems almost did not affect the speed of static images recognition systems almost did not affect the speed of image recognition.**

***Keywords*— real-time operation system; RTOS; mobile robot; Raspberry Pi; patch; Linux; classification.**

I. Introduction

Single-board computers are gaining increasing popularity due to their size, with the result that they are widely used in robotics. But for the short form factor, you have to pay weak technical characteristics. Vision-based navigation is one of the crucial tasks that robotics address. Many factors are imposing practical limitations on a robot's ability to see, learn and explore the environment. For this reason, navigation in unknown or partially unknown environments remains one of the biggest challenges in today's autonomous mobile robots implemented on single board computers [1]. This task requires a sufficiently large number of capacities, and to solve it in a single-board computer we should use all possible potential. To date, there are various ways of improving computer performance for a particular task. One of the possible solutions is to utilize of real-time (RT) systems, where compliance with specific time limits is very important, but, along with this, the overall system performance may decrease. Thus, the developer is faced with the hard choices of using real-time systems to dealing with performance or accuracy in image recognition and vision-based navigation. Objectives of this research are to study and use a real-time operating system (RTOS), Linux and their patches on a Raspberry Pi single board computer and to perform its further implementation in a small autonomous mobile robot equipped with the camera module and IR sensor.

The purpose of the article is to investigate the feasibility of using real-time operating systems (RTOS) on a single-board computer for image recognition and vision-based navigation of small autonomous robot. Achieving this goal requires the following tasks:

- testing and estimating the delay of real-time systems to improve recognition efficiency during cyclic (continuous) operation;
- assessment of the quality and speed of classification of images.

The main contribution of this work is adding knowledge, experience, and insight about RT-patched Linux as an RTOS for vision-based navigation of small autonomous robots, implemented on Raspberry Pi single board computer, comparing time-delays in RT-systems, their influence on image recognition and further usage raw sensory data for dynamic robot navigation; to develop, implement and producing a small autonomous system based on RTOS.

The rest of the paper is organized as follows. Section II discusses state-of-the art in the areas of dynamic navigation, machine vision and real-time image recognition, and real-time operating systems evaluation for mobile robots. Section III shows system design and testing environment. Section IV presents obtained results. Finally, Section IV provides a discussion of the results.

II. State-of-the-art

The current task joins three areas related to the real-time systems; they are dynamic navigation for mobile robots, machine vision and real-time image recognition, and real-time operating systems evaluation.

Dynamic navigation for mobile robots is explored in a large number of publications. Thus, Habib [1] presented a state-of-the-art in map building and localization for mobile robots navigating within an unknown environment. The deep reinforcement learning technique for robot navigation in unknown rough terrain is proposed in [2]. They used elevation maps and high dimensional sensory data from depth images to perform robot navigation through rough terrain. They are also categorized existing work on dynamic navigation into two groups that focus on classification technique and learning technique.

Paper [3] includes a comprehensive classification of vision systems for ground mobile robots. The modern sensing systems of intelligent robots are discussed in [4]. A real-time image recognition system for tiny autonomous mobile robots is discussed in [5]. Mahlknecht et al. proposed an object recognition algorithm based on a combination of edge and color detection and used a fixed model for each recognizing object. Objects classification technique for mobile robots is presented in [6]. Mobile robot navigation using a neural network for image classification is also discussed in [7, 8, 9]. Visual image processing technique based on OpenCV for a mobile robot is presented in [10]. A framework for image processing on a Raspberry Pi platform is proposed in [11]. As it mentioned in [12], evolutionary adaptation is one of the most important functions for the mobile robot navigating in the unknown, unstructured environment. At the same time, one of the most challenging tasks in navigation is real-time communication and control. In this context, researchers turn to the subject of measuring and evaluating real-time performance in robotic applications. Thus, the real-time performance of UDP based communications in Linux on multicore embedded devices is evaluated in [13]. Results of analysis and benchmarking performance of real-time patch Linux and Xenomai are presented in [14]. Marieska et al. compare both RTOS based on three performance metrics: processing time, jitter, and throughput. The performance of RTOS on a single board computer for a wheeled mobile robot with the ultrasonic sensor is analyzed in [15]. Achieved results showed that RTOS with Qt-based program enables the robot to respond less than 1 second. In this paper, we try to extend the awareness of RTOS performance on a single board computer for classification tasks and vision-based navigation of small autonomous mobile robots.

III. System Overview and Experiment Design

A. The Overall Designed System

The system design of RTOS on single board computer equipped with camera module and IR sensor is based on three main components: Single Board Control Module, Sensor Module, and Servo Controller module. The system and its components are shown in Fig. 1.

The central system controller of an autonomous mobile robot is the control module consisted of a Raspberry Pi Single Board Computer based on Bcm2836 processor with Linux. It is a primary control module for all connected components.

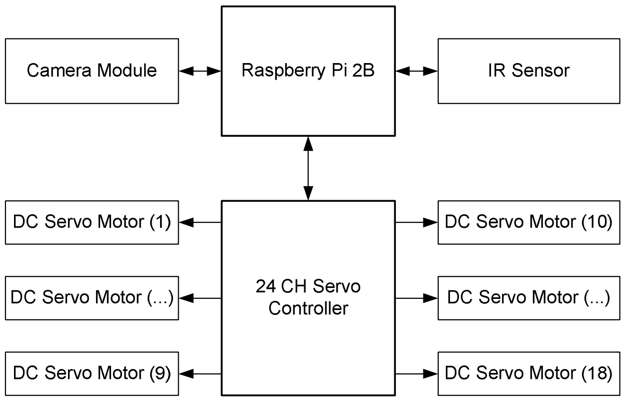

Figure 1. The block diagram of the overall designed system

To implement the functions of vision-based dynamic navigation, a neural network is included in the system structure (see Fig. 2).

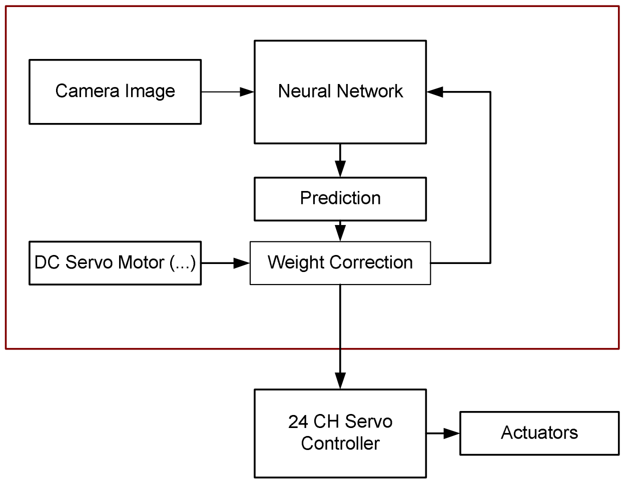

Figure 2. The block diagram of the vision-based navigation module

The use of a neural network and various software modules provides a significant numerical load on the system. Since it is planned to perform operations cyclically, the detection of an obstacle without delays is a very important task. Thus, if the mobile robot does not detect an obstacle in time, a collision will occur and this may have negative consequences for the robot.

As it mentioned in [5], real-time actions in an object recognition system that employs a video camera can be conducted when the process of transferring and recognizing an object takes no longer than the inverse of the frame rate. If this condition cannot be guaranteed, some images will be lost, or if stored in RAM the process of recognition will delay at each frame for additional time. In this case, the upper bound delay (Tud) can be defined by the frame rate (FR) as follow:

$$T_{ud} = \frac{1}{F_R}.$$

The challenge with Linux-based single board computers is that Linux actually is not a tough real-time operating system, and it does not guarantee the completion of tasks on time [16, 17]. The kernel may suspend the task execution while it is working and it may remain suspended for any long time (for example, while servicing an interrupt).

The basic idea of creating a standard, tight Linux environment in real time is that a small, high-quality real-time kernel works between hardware and standard Linux. Real-time tasks are performed by this real-time kernel (from execution to the end) and normal Linux processes are terminated during this period. In real-time, the real-time kernel scheduler views the standard Linux kernel as a simple process that, when given the opportunity to run, runs its own scheduler of normal Linux processes. But since the real-time kernel runs at a higher priority, normal Linux processes can be preempted at any time in real-time tasks. Interrupt control is another factor that drives the real-time kernel [17]. Therefore, it is proposed to upgrade the OS of a single-board computer using a real-time system. The performance criteria for the system are jitter and time to recognize the object (familiar to the neural network).

B. *Experimental Units and testing Scenario*

As the research objects, the most popular RT systems for Linux are selected, namely the patch of the kernel PREEMPT_RT and the Xenomai framework (also with the kernel patch) [18]. Also, for comparison, the results of the performance of the standard kernel are present.

The experiment plan contains the following steps:

1) Preparing a Linux image with a standard kernel;

2) Preparation of images with patch PREEMPT_RT and Xenomai;

3) Compiling and Compiling and configuring the OpenCV library and the GoogLeNet deep learning neural network;

4) Testing cyclic delays using the cyclictest utility;

5) Testing the speed of classification of prototypes (images) using OpenCV and GoogLeNet (Fig. 3).

All studies were conducted on a single-board computer Raspberry PI 2 equipped with a quad-core ARM processor, with the technical characteristics given in Table 1.

TABLE I. SPECIFICATION OF RASPBERRY PI 2 MODEL B

Processor	Bcm2836 quad-core ARM Cortex A7
CPU	900 MHz
RAM	1 GB
GPU	Dual-core Videocore IV

The Raspberry Pi 2 (released in February 2015) is equipped with 1 GB of RAM, a 900 Mhz quad-core ARM Cortex A7 CPU, 10/100 Ethernet, and retails for $35.00. It supports both the Linux and Windows operating systems. A single 5-Volt DC 2-Amp power supply is sufficient to power the device, and the unit consumes up to 4 watts of power.

Figure 3. Examples of testing samples

At the moment, the latest official version of the Linux kernel for Raspberry was 4.14, so all subsequent research is done on this version of the kernel [19]. To assess the quality and speed of recognition, two random images (colored) with an open license to use were prepared (see Fig. 3). The images have the format "jpg" and resolution "5000×2909" and "4256×2832", respectively.

IV. TESTING RESULTS

For testing, the Cyclictest utility from the RT-Tests kit [20] was used. This is one of the most commonly used tools for assessing the relative performance of RTOS systems. Cyclictest measures accurately and repeatedly the difference between the designated wake-up time of the stream and the time it wakes up to provide statistics on system delays. It can measure delays in real-time systems caused by hardware, firmware, and the operating system.

A. *RTOS Timeouts*

To measure delays, Cyclictest starts a non-real-time mainstream (scheduling class SCHED_OTHER), after which the mainstream starts a certain number of real-time measurement flows (scheduling class SCHED_FIFO). Measured flows periodically wake up at a certain interval using a timer with expiration (cyclical signal). Subsequently, the difference between the programmed and effective wake-up time is calculated and transmitted to the main thread through shared memory. The main thread tracks the delay value and prints the minimum, maximum, and average delays.

The command for testing is as follows:

cyclictest -l·1000000 -m --policy=fifo -Sp99 -i250 -h500,

where l denotes a number of cycles, m is lock process memory pages, policy is a real-time policy, p is a real time priority; i is an interval in microseconds, and h is a maximum tracking time in microseconds (used to get histogram data).

Results of testing standard kernel (4.14.74-v7+) are displayed in Fig. 4.

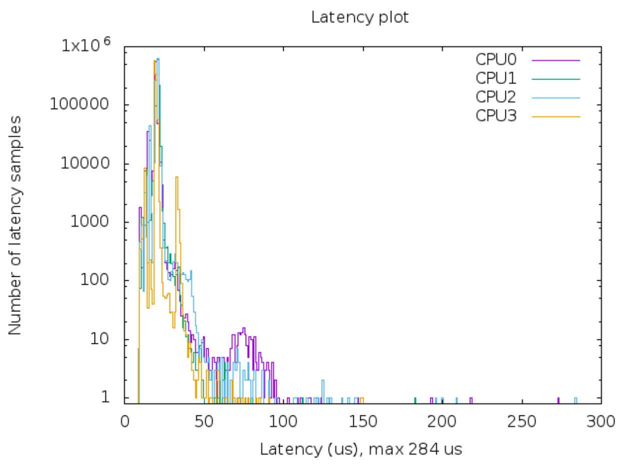

Figure 4. Delay Histogram with standard Linux Kernel

Results of testing kernel delay with PREEMPT_RT patch (4.14.74-rt44-v7+) are displayed in Fig. 5.

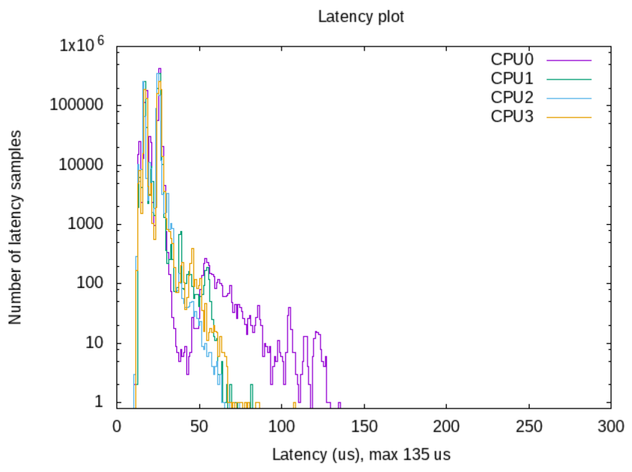

Figure 5. Kernel Delay Histogram with PREEMPT_RT patch

Results of testing kernel delay with Xenomai patch (4.14.37-v7+) are displayed in Fig. 6. The figures show the jitter for each processor core (4 cores of the Rpi2 processor, which operate in parallel) and the associated number of tests with it. The greater the maximum delay (Latency (us), max) on any of the cores of the system, the worse.

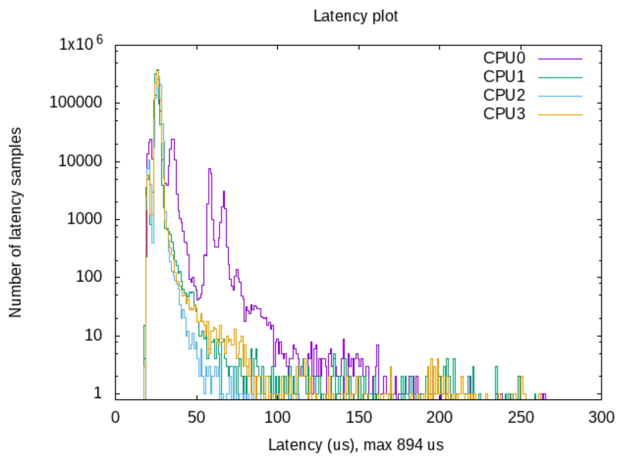

Figure 6. Kernel Delay Histogram with Xenomai patch

B. The Results of Testing the Speed of Image Recognition

To study the speed of recognition (classification) we used OpenCV library, which allows the use of pre-prepared neural networks with popular systems of deep learning [21]. The results of recognition speed, as well as the quality of classification (is demonstrative) on various systems are shown in Table. 2, 3.

TABLE II. RESULTS FOR THE FIRST SAMPLE

Standard kernel	PREEMPT_RT	Xenomai
1.2992 sec	1.1953 sec	1.2734 sec
1. label	castle, probability: 0.89271	
2. label	cliff, probability: 0.049932	
3. label	breakwater, probability: 0.0084649	
4. label	promontory, probability: 0.0063658	
5. label	lakeside, probability: 0.0043311	

TABLE III. RESULTS FOR THE SECOND SAMPLE

Standard kernel	PREEMPT_RT	Xenomai
1.2949 сек	1.1868 sec	1.3001 sec
1. label	white wolf, probability: 0.6342	
2. label	Arctic fox, probability: 0.32954	
3. label	timber wolf, probability: 0.026111	
4. label	kuvasz, probability: 0.0076301	
5. label	ice bear, probability: 0.0015258	

The tables show the total time required by the current configuration system to identify familiar objects. At the same time, the system configuration does not affect the accuracy of recognition.

As can be seen from the tables, for our data RTOS systems have a negligible effect on the rate of classification, while the quality of classification remains almost unchanged.

V. Conclusion

As a result of the study, it can be seen that RT systems almost did not affect the speed of image recognition. Slight fluctuations in performance are due to the peculiarity of core patches and the displacement of secondary processes, but, as a rule, in the robotics system, the system is not limited to static recognition, but is used for cyclic work. Judging by the histograms, real-time systems have low cycles of delay, indicating a sufficient level of resource planning, so these systems can be used in downloaded cyclic processes, including in cyclic image recognition. The kernel delay using the Xenomai framework turned out to be less good than expected, possibly due to the kernel configuration or a defect in the testing process. Therefore, in order to increase the cyclic accuracy of the recognition task on single-board computers, it is possible to recommend the use of the PREEMPT_RT patch, which has the lowest cyclic delay and gives little advantage over the recognition speed.

Regarding the quality of the classification, we can assume that the OpenCV library and the prepared neural network managed to do well. On the first sample, the neural network really found the castle (probability 89%) and the rock (4%), and the second white wolf (63%).

References

[1] M. K. Habib. "Real Time Mapping and Dynamic Navigation for Mobile Robots," International Journal of Advanced Robotic Systems, (September 2007). doi:10.5772/5681.

[2] K. Zhang, F. Niroui, M. Ficocelli, and G. Nejat. "Robot Navigation of Environments with Unknown Rough Terrain Using Deep Reinforcement Learning" Available: http://asblab.mie.utoronto.ca/sites/default/files/SSRR18_0040_FI.pdf

[3] J. Martinez-Gomez, A. Fernandez-Caballero, I. Garcia-Varea, L. Rodriguez, and C. Romero-Gonzalez. "A Taxonomy of Vision Systems for Ground Mobile Robots."International Journal of Advanced Robotic Systems, (July 2014). doi:10.5772/58900.

[4] Y.P. Kondratenko, O.S. Gerasin, A.M. Topalov. "Modern Sensing Systems of Intelligent Robots Based on Multi-Component Slip Displacement Sensors." Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, September 24 – 26, Volume 2, pp. 902–907, 2015 DOI: 10.1109/IDAACS.2015.7341434

[5] S. Mahlknecht, R. Oberhammer, G. Novak "Image recognition for autonomous mobile robots"

[6] Ch. Xu , T.Cao, Z. Li and X. Xiao. "Objects Classification for Mobile Robots Using Hierarchic Selective Search Method" ICIEA 2016 MATEC Web of Conferences 68 , 03001 (2016) DOI: 10.1051/matecconf/2016603001

[7] T. S. Jin, J. M. Lee. "Mobile robot navigation by image classification using a neural network" IFAC Proceedings Volumes Volume 37, Issue 12, August–September 2004, pp.203-208

[8] L. Ran, Y. Zhang, Q. Zhang, T. Yang. "Convolutional Neural Network-Based Robot Navigation Using Uncalibrated Spherical Images." Sensors (Basel, Switzerland), Vol. 17(6), 1341, 2017. doi:10.3390/s17061341

[9] G. Măceşanu F. Moldoveanu "Computer Vision Based Mobile Robot Navigation in Unknown Environments" Bulletin of the Transilvania University of Braşov. Vol. 3 (52), 2010 Series I: Engineering Sciences

[10] S.-J. Zhang, Lei-Zhang, and Ri Gao. "Research on Visual Image Processing of Mobile Robot Based on OpenCV." Journal of Computers Vol. 28, No. 5, 2017, pp. 255-275 doi:10.3966/199115992017102805023

[11] K. Horak, L. Zalud "Image Processing on Raspberry Pi for Mobile Robotics" Available: http://midas.uamt.feec.vutbr.cz/data/vision_Horak_paper_11_IJSPS_2016.pdf

[12] A.N. Tkachenko, N.M. Brovinskaya, Y.P. Kondratenko Evolutionary adaptation of control processes in robots operating in non-stationary environments // Mechanism and Machine Theory. – Printed in Great Britain, 1983. – Vol. 18, No. 4. – pp. 275-278. DOI: 10.1016/0094-114X(83)90118-0

[13] S. V. Gutiérrez, L. U. S. Juan, I. Z. Ugarte, V. M. Vilches. "Real-time Linux communications: an evaluation of the Linux communication stack for real-time robotic applications." Available: https://arxiv.org/pdf/1808.10821.pdf

[14] Marieska, M. Kistijantoro, A.I.M. Subair. "Analysis and benchmarking performance of Real Time Patch Linux and Xenomai in serving a real time application." Proceedings of the 2011 International Conference on Electrical Engineering and Informatics.doi:10.1109/iceei.2011.6021563

[15] W. Atmadja, B. Christian, L. Kristofel "Real Time Operating System on Embedded Linux with Ultrasonic Sensor for Mobile Robot" IAICT 2014, Bali 28-30 August 2014 pp. 22-25.

[16] R.Love. "Linux kernel. Description of the development process," 2013 – 496 p.

[17] Abbott. "Linux for Embedded and Real-time Applications," 4th Edition, 2017 – 304 p.

[18] Xenomai Wiki. Available: https://gitlab.denx.de/Xenomai/xenomai/wikis/home (20.02.2019).

[19] Raspberry PI official site. Available: https://raspberrypi.org/downloads/raspbian/ (20.12.2018).

[20] Latency of Raspberry Pi 3 on Standard and Real-Time Linux 4.9 Kernel Available: https://metebalci.com/blog/latency-of-raspberry-pi-3-on-standard-and-real-time-linux-4.9-kernel/

[21] Pajankar. "Raspberry Pi Computer Vision Programming," 2015 – 168.