Optimization of a Proportional-Summation-Difference Controller for a Line-Tracing Robot Using Bacterial Memetic Algorithm

Brandon Zahn[1,2(✉)], Ivan Ucherdzhiev[1,3], Julia Szeles[1], Janos Botzheim[1], and Naoyuki Kubota[1]

[1] Graduate School of System Design, Tokyo Metropolitan University, Hachioji, Japan
brandonktm11@gmail.com, ivan_ucherdzhiev@smartcom.bg,
{perecka,botzheim,kubota}@tmu.ac.jp
[2] University of Newcastle, Callaghan, Australia
[3] Technical University of Sofia, Sofia, Bulgaria

Abstract. The smart home of the future will require a universal and dynamic mapping system of a complex labyrinth that devices can use for a wide range of tasks, including item delivery, monitoring and streamlining transportation. In this paper, the core concept of this system is proposed and is demonstrated by a line-tracing robot that has self optimizing properties using an evolutionary algorithm. Tests were performed to find the best controller for the robot and a proportional-summation-difference (PSD) controller was found to be best suited for the robot's hardware specifications. An evolutionary computing algorithm called Bacterial Memetic Algorithm (BMA) was then implemented to optimize the PSD controller's properties to suite the conditions of a track. After several thousand candidate solutions of evolutionary computing and simulated tests, the robot was able to complete a lap in real time with a significant decrease in both lap time and average error.

Keywords: Line-tracing · Proportional-discrete-summation · Evolutionary algorithm · Smart home

1 Introduction

Smart homes of the future will become more complicated and an effective, small-scale, internal transportation system will be required to meet the demands of an efficient top-tier smart home, factory or office. The goal of this paper is to propose a mapping system that can meet these needs of the future by building the foundation for a line-tracing robot that can be used in a modern, internet of things, oriented smart home. The robot will display the basic features of the system by following a single, complex track using a PSD controller that has self optimizing properties.

The robot was designed to follow a target line using an array of six infrared reflectance sensors which served as a digital input to the PSD controller that

N. Kubota et al. (Eds.): ICIRA 2016, Part I, LNAI 9834, pp. 362–372, 2016.
DOI: 10.1007/978-3-319-43506-0_32

drove the robot's motors. Several other controllers were considered such as fuzzy logic and crisp logic but due to the digital input of the sensor array, the PSD controller was superior in performance. An evolutionary algorithm called Bacterial Memetic Algorithm (BMA) was used to optimize the properties of the controller to minimize the lap time and average error of the robot around a track. A software simulation was developed to simulate the robot on any given track that the user required. Several thousand evaluations were performed using this algorithm together with the simulator to generate the most optimal coefficients for the PSD controller which best suited the conditions of the track. Once the simulation was complete, the most efficient coefficients were found, they were updated and the robot was able to complete a lap with approximately 10 % greater speed and an increase in line-tracing accuracy than when the initial PSD controller's coefficients were used.

Future work is planned to further develop the mapping system. The robot should be able to perform tasks such as monitoring using a camera, reflectance sensors, a gyroscope and other peripherals, while constantly being connected to a base station via a Wi-Fi connection. Some more examples of planned work include a manual control override from a user and to incorporate an intelligent control system so that the robot can efficiently find its way through a maze-like labyrinth.

There is a lot of research that is investigating possible ways on how to make smart homes more efficient and comfortable through the use of many varying techniques. Inhabitant and object tracking in smart homes is a large topic in most research that was investigated. The existence and location of an inhabitant and the number of inhabitants existing at the same place are major challenges that are being discovered for the advancement of smart home research. As a result, many different solutions have been proposed to accurately monitor and track inhabitants of a smart home in a noninvasive way [7].

Currently, there are various techniques for monitoring and sensing in smart homes. Some smart homes use RFID chips to easily detect various objects as they do not have to be within line of sight of a sensor. RFID chips are very cheap and can be easily integrated into an existing system. There is a requirement that the object being detected must have a RFID chip installed or attached to it. This causes a problem when an object does not have a RFID chip attached as it cannot be detected. There has been some development using this technique, but there are many other possible ways for monitoring [4,12]. Objects can also be detected with voice or sound sensors. There are some objects which cannot be detected by voice because they do not have one and sound sensors may be prone to large amounts of interference, making it difficult to monitor a target object. Another alternative is using temperature sensors. Objects which have similar or near equivalent temperatures or environments that have a small temperature range can make monitoring via temperature very problematic.

Detecting inhabitants and various objects in smart homes can also be achieved through the use of cameras. To monitor the majority of a smart home's contents, a lot of individual cameras are required. The cameras need to be placed

in the right position and at the right angle for maximum efficiency, making the installation process tedious [14]. If too many cameras are in operation, it can often cause distress to the inhabitants as it is essentially an invasion of privacy even though they would be only used to better their living conditions and add more functionality to a smart home [10]. In this example, a line-tracer using the mapping system being proposed can fulfill the requirements of smart home monitoring [13]. It will only require one camera and can follow a given labyrinth to detect inhabitants or objects so it is not necessary to install many cameras in the home. The line-tracer can be equipped with various sensors to detect more than just the actual objects, but gain information on the environment's condition as well due to it's powerful hardware and the numerous peripherals available for use. It is also possible to control the robot from an external source for more accurate control [8].

A controlling method of the behavior of a mobile robot in a dynamic environment has also been researched that is useful for the advancement of the mapping system. In order for a robot to have sufficient control in a changing environment, it should control the computational cost according to the state of the environment, and it must take a feasible behavioral alteration within a finite time [9]. Another method of controlling the behavior of a mobile robot that can be implemented into the system is perception-based robotics such as logical inference and fuzzy computing [6].

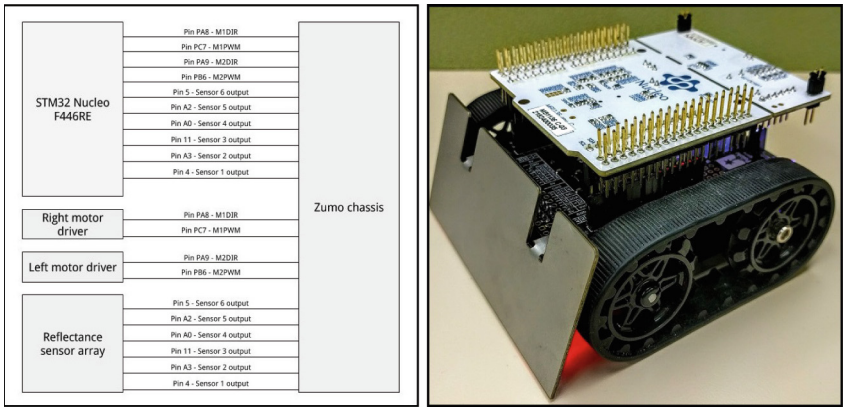
This paper covers concept introduction, the specifications of the robot used, a proposed solution method including the controllers considered and the testing method, the experimental results and analysis and finally, conclusions and planned future work.

2 Robot Specifications

The robot was made with the following parts: STM32 Nucleo F466RE development board [11], Zumo reflectance infrared sensor array and Zumo robot V1.2. The pin configuration of these devices can be seen in Fig. 1a and an image of the complete robot can be seen in Fig. 1b. The STM32 Nucleo F466RE development board was chosen because of it's powerful hardware and numerous peripheral pins available. This development board was also designed to be connected with Arduino devices so it could be attached to the Zumo robot.

Some features of the development board include a 32-bit STM32 microcontroller with a clock frequency up to 180 MHz and 225 DMIPS, 512 kB of flash memory and 128 kB RAM, up to 17 internal timers, up to 114 I/O ports with interrupt capability, three audio to digital converters and up to 20 communication interfaces. This board gives the ability for many additional features to be integrated into the robot.

The operation of the robot in real time is as follows: the digital input from the six IR sensors is used to determine the current location and orientation of the robot with respect to the black target line. Any sensor that does not detect a reflection of IR light, will be set to HIGH, meaning the sensor is over a black surface. This input is then stored in a char variable in the program. Please refer



(a) Block diagram of the pin config (b) An image of the complete robot

Fig. 1. Robot details

Table 1. PSD lookup table

Binary input	Hexadecimal input	Output
0000 0001	0×01	error $= 5$
0000 0011	0×03	error $= 4$
0000 0010	0×02	error $= 3$
0000 0110	0×06	error $= 2$
0000 0100	0×04	error $= 1$
0000 1100	$0 \times 0C$	error $= 0$
0000 1000	0×08	error $= -1$
0001 1000	0×18	error $= -2$
0001 0000	0×10	error $= -3$
0011 0000	0×30	error $= -4$
0010 0000	0×20	error $= -5$
0011 1111	$0 \times 3F$	interrupt flag

to Table 1 for the lookup table that shows the valid inputs used by the PSD controller. If, for example, the error input was -5, the PSD controller would make a difference between the left and right tracks to be 100 %, meaning the left motor would have 0 % velocity of the maximum speed, while the right motor would have 100 % of the maximum speed. This would result in an immediate 90 degree turn to the left. When the robot is removed from the track, an interrupt flag is called and will then stop the motors and timers of the robot. The program will wait for an input from the user via a serial connection and then output the lap time and average error around the track to a terminal. Finally, the robot's program is terminated.

3 Proposed Solution Method to Problem Statement

The problem statement is as follows: The target objective for experiment and analysis is to optimize the PSD controller in order to minimize the lap time of the robot and increase line-tracing accuracy.

In order to meet the requirements of the problem statement, the proposed solution method consists of developing a software simulation of the robot to simulate the tests that would be performed in real time. Initial coefficients are selected and the robot is tested in the simulator. Once this preliminary test is completed, the lap time is sent from the simulator to the BMA which then performs calculations to find the best coefficients for a minimization in lap time. Once the optimal solution has been found, tests in real time with the optimal coefficients are performed to check the validity of the simulation results.

3.1 The Controllers and Logic of Line-Tracing

Fuzzy logic is a problem solving control system methodology used in a wide range of applications for data acquisition and system control and provides a simple way to arrive at a definite conclusion based upon vague, ambiguous or noisy analog inputs. Fuzzy logic mimics the thought process of humans and how a person would make a decision. For this reason, it is considered to be one of the three main branches of computational intelligence, along with evolutionary computation and neural networks. Fuzzy logic incorporates a simple rule based IF X AND Y THEN Z approach to solving a problem rather than attempting to model a system mathematically [5]. Fuzzy logic uses a membership function that gives a variable a certain membership value within a set. This means an input can be in several independent sets but will have varying membership values for each.

Fuzzy logic and crisp logic were tested with the robot's hardware setup and several problems were encountered with the logic operating at an acceptable level. This is due to the input of these systems being of a discrete type. Fuzzy logic is best applied when an analog input is being used and a decision is made on how to indirectly control a target output. In the case of the robot's hardware configuration, discrete inputs from the sensor array were being transformed into analog outputs to the motor drivers. For this reason, a PSD controller was the most suitable controller for the task.

The PSD controller was derived from the proportional-integral-derivative (PID) controller which is a control loop feedback mechanism that continuously calculates an error value as the difference between a measured variable and a target set point. The controller attempts to minimize the error over time by adjustment of a control variable to a new value determined by the formula

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}. \tag{1}$$

The coefficients K_P, K_I, K_D are non negative and influence the behavior of the PID controller [3]. Since the PID controller needed to be discrete to work with the robot's hardware specifications, it was converted into a PSD controller represented by the formula

$$u[n] = K_P e[n] + K_S \sum_{n=1}^{\infty} e[n] + K_D \frac{e[n] - e[n-1]}{n - (n-1)}. \tag{2}$$

Where $e[n]$ and $e[n-1]$ inputs represent the error and previous error respectively from the target set point, which are the two center sensors in the sensor array. A dozen preliminary tests were performed to find suitable initial values for the coefficients K_P, K_S, K_D of the PSD controller. It was then used to drive the left and right motors of the robot to guide it along the target line by changing the difference in velocities of the motors.

3.2 Bacterial Memetic Algorithm

Bacterial Memetic Algorithm (BMA) [1] is applied for finding the optimal parameter setting of the controller's K_P, K_S, K_D parameters. BMA is a population based stochastic optimization technique which effectively combines global and local search in order to find a good quasi-optimal solution for the given problem. In the global search, BMA applies the bacterial operators, the bacterial mutation and the gene transfer operation. The role of the bacterial mutation is the optimization of the bacteria's chromosome. The gene transfer allows the information's transfer in the population among the different bacteria. As a local search technique, the Levenberg-Marquardt method is applied by a certain probability for each individual [1].

In the case of evolutionary and memetic algorithms, the encoding method and the evaluation of the individuals, or bacteria, have to be discussed. The bacterium consists of the three controller parameters to be optimized, K_P, K_S, K_D. The evaluation of the bacteria is calculated by the simulator program explained in Sect. 3.3.

The operation of the BMA starts with the generation of a random initial population containing N_{ind} individuals. Next, until a stopping criterion is fulfilled, which is usually the number of generations N_{gen}, we apply the bacterial mutation, local search, and gene transfer operators. Bacterial mutation creates N_{clones} number of clones of an individual, which are then subjected to random changes in their genes. The number of genes that are modified with this mutation is a parameter of the algorithm by the name l_{bm}. After the bacterial mutation, for each individual the Levenberg-Marquardt algorithm is used by a certain probability LM_{prob}, until a certain number of iterations LM_{iter}. The two other parameters of this step are the initial bravery factor γ_{init} and the parameter for the terminal condition τ. The last operator in a generation is the horizontal gene transfer. It means copying genes from better individuals to worse individuals. For this reason, the population is split into two halves, according to the

cost values. The number of gene transfers in one generation N_{inf}, as well as the number of genes l_{gt}, that get transferred with each operation, are determined by the parameters of the algorithm. In the experiment, the parameter settings for BMA is presented in Table 2.

Bacterial memetic algorithm has been successfully applied to a wide range of problems. More details about the algorithm can be found in [1,2].

3.3 Robot Simulator

As the BMA creates several thousand candidate solutions of the robot's PSD controller, a software simulation of the robot was required to perform thousands of evaluations within a reasonable time frame. The simulator was developed in the C# environment using Visual Studio. In the simulation, the user can load a top-view image of the actual track as a .jpg or .png file. In order to make it easier to see, the loaded image is resized to fit the screen of the program. The track contrast is enhanced by coloring all the pixels that have a RGB value less than a predefined threshold pure black, as they are considered to be the line of the track. All the pixels in the image that have a RGB value greater than a predefined threshold are colored pure white and then the whole image is redrawn to get a clear track with no interference.

The line-tracer is placed in a picture box and the user can place the starting line and the line-tracer anywhere on the track. Then the user can define the orientation of the robot related to the track. The scale of the robot related to the size of the track is then calculated and corrected by measuring the size of the robot in pixels and the width of the track line in pixels to a specific ratio that matches the real ratio between the track line and the robot's size. The line-tracing animation is done by using a forward kinematics model. The image of the track is a static frame and the image of the robot is a dynamic frame. The frame details are shown in Fig. 2.

$$T_i = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & X_{Ti} \\ -\sin(\theta) & \cos(\theta) & 0 & Y_{Ti} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{Tr} \\ Y_{Tr} \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

A rotation matrix shown in Eq. 3 is used to rotate the robot's frame with reference to the track's frame where X_{Ti} is the x coordinate of the center point of the robot in the track frame, Y_{Ti} is the y coordinate of the center point of the robot in the track frame, θ is the rotation angle between the track's frame and the robot's frame, X_{Tr} is the x coordinate of a point in robot's frame, Y_{Tr} is the y coordinate of a point in robot's frame and T_i is the result vector which contains the coordinates of the robot's frame point with respect to the track's frame. With this kinematic model, the simulation software recalculates the robot sensor's coordinates every millisecond. The PSD controller checks the color of the pixels under the sensors and recalculates the linear and angular velocity of the robot depending on the position and orientation of the robot with respect

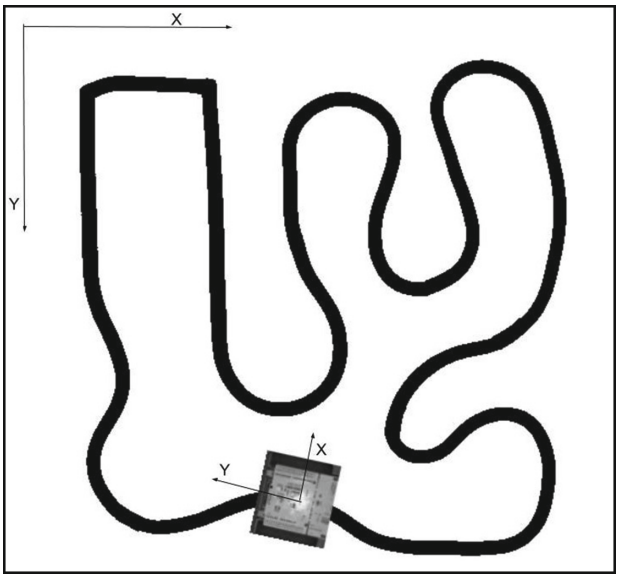


Fig. 2. A diagram of the robot simulation software setup

to the line. After every calculation of the velocities, the simulation software updates the position of the robot image and starts the whole process again from the beginning. Once the robot completes a full valid lap of the track, the lap time is then sent to the BMA to begin a new evaluation. If the robot looses the line, an invalid lap time is sent to the BMA.

4 Experimental Results and Analysis

The track used in both the simulation and real time tests had a total length of 4.8 meters, a thickness of 2 cm and included curves to test all lookup cases for the PSD controller. A comparison can be seen in Fig. 4a between the real world test conditions and Fig. 4b showing the simulator configuration. During the simulated tests, a decrease in lap time from 13.2 s to 10.4 s was achieved over 20 generations in the BMA, which is approximately a 20 % decrease from the initial generation's lap time. Table 2 shows the parameters that were used in the BMA for the optimization process. 20 generations, 4 clones and 10 individuals were used in the simulation. Figure 3 shows how the target error has been minimized.

Real time tests before and after applying the optimized PSD controller show a similar result in Table 3 with an approximate decrease in lap time of 9.7 % and showed a 0.856 % increase in line-tracing accuracy. There are several factors that may have affected the tests in real time, such as friction between the tracks of the robot and the surface of the track and interference like dust and dirt blurring the track for the robot. Also due to an inconsistent power supply from four 1.5 V batteries, the robot's maximum speed has likely been reduced. Inaccuracies in the software simulation could also be reason for slightly differing results.

Table 2. BMA parameters used in the evolutionary algorithm

Parameter	N_{gen}	N_{ind}	N_{clones}	l_{bm}	N_{inf}	l_{gt}	LM_{prob}	LM_{iter}	γ_{init}	τ
Value	20	10	4	1	3	1	10 %	8	1.0	0.0001

Table 3. Results of the real time test

Parameter	Before optimization	After optimization
Average Lap Time (sec)	10.951	9.896
Average Error (%)	8.56	7.704

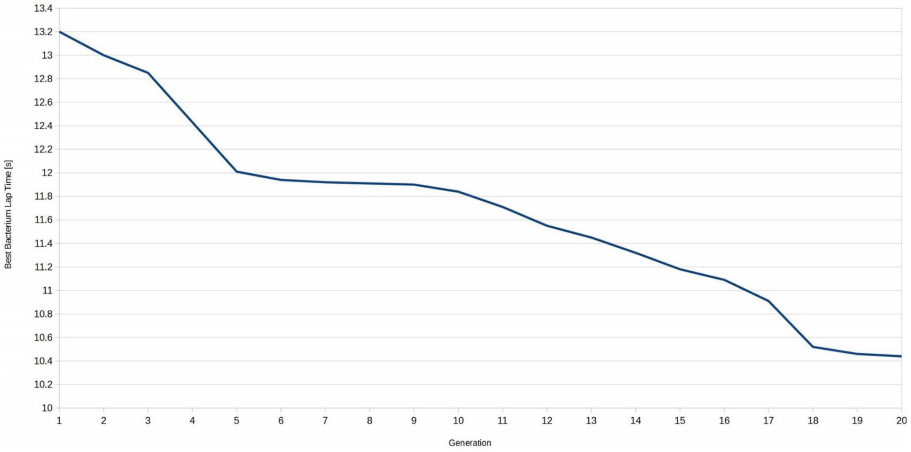
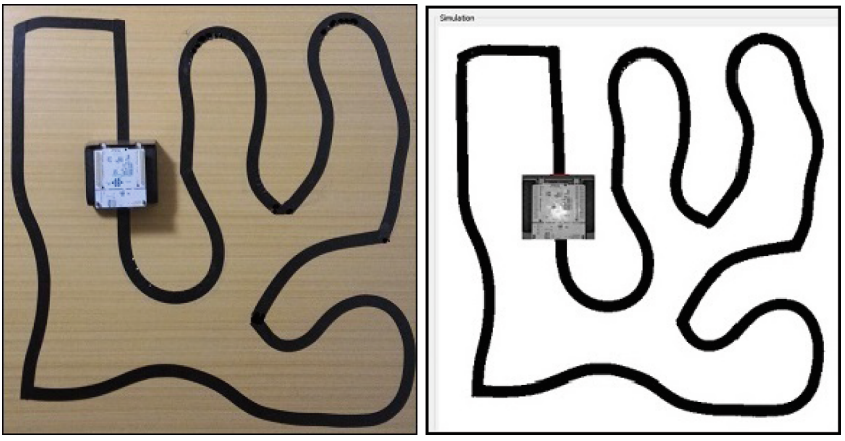


Fig. 3. A line graph showing the bacterium lap time over number of generations



(a) Image of the real time tests (b) Image of the simulation tests

Fig. 4. A comparison between the real time tests and the simulated tests

5 Conclusion

From the results, the conclusion can be made that the optimization of the PSD controller has been successful. This is a good foundation to start developing more features of the robot and increase the number of tasks that the robot can perform. By doing this, the mapping system that the robot will be using will also be further developed to include many important features that a universal, dynamic mapping system requires to be successful. There are many tasks that require completion to fully realize the proposed mapping system.

Avoiding obstacles blocking the pathway is a priority task and is an important capability for real world applications so the ability to follow a line with obstacle avoidance capabilities will be investigated. The robot should also be able to navigate more complex tracks with broken lines seen in a labyrinth that consists of intermittent broken lines and interference. The current method for optimizing the controller's coefficients requires an external software simulation and evolutionary algorithm but it should internally optimize the controlling software in real time without requiring external inputs so the ability to optimize the controller's properties internally and automatically in real time will be investigated. A manual controller will also be developed to allow a user to manually control the robot from a computer or smart phone through a Wi-Fi connection. It will also be possible to allow an external source, such as a smart home, to control the robot. Line mapping and memorizing is a key feature for a universal mapping system. Another key task is to memorize and store the location of a line and it's features. If it is a maze-like labyrinth, the robot will know the route to reach a destination after it has analyzed and memorized the whole course. If multiple routes to a target destination are present, then an algorithm to find the fastest route will be required so an intelligent control system that can efficiently find the best route through a labyrinth will be investigated. Instead of using a discrete input, the robot will be modified to allow for an analog input to be used so that fuzzy logic may be more easily used in controlling the robot, which will increase the accuracy of the robot's control at the cost of a more resource demanding software. Finally, through a Wi-Fi connection, the robot should be able to stream a video feed from a camera attached to the robot to a user or the smart home.

Due to the powerful hardware used, there are many opportunities available for expansion of active peripherals and sensors. This will give the robot the flexibility and capability to fulfill many different tasks and play an integral role in the smart home of the future.

References

1. Botzheim, J., Cabrita, C., Kóczy, L.T., Ruano, A.E.: Fuzzy rule extraction by bacterial memetic algorithms. Int. J. Intell. Syst. **24**(3), 312–339 (2009)
2. Botzheim, J., Toda, Y., Kubota, N.: Bacterial memetic algorithm for offline path planning of mobile robots. Memet. Comput. **4**(1), 73–86 (2012)

3. Hagglund, T.: PID controllers: theory, design and tuning. In: ISA: The Instrumentation, Systems, and Automation Society, Upper Saddle River, New Jersey (1995)
4. Jin, P.Y., Bann, L.L., Kit, J.L.W.: RFID-enabled elderly movement tracking system in smart homes (2014)
5. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic. Theory and Applications. Prentice Hall, Upper Saddle River (1995)
6. Kubota, N.: Perception-based robotics based on perceiving-acting cycle with modular neural networks. In: Proceedings of the 2002 International Joint Conference on Neural Networks, pp. 477–482, May 2002
7. Lyman, G.: Human tracking methods comparsion for smart house (2014)
8. Mowad, M., Fathy, A., Hafez, A.: Smart home automated control system using android application and microcontroller (2014)
9. Nojima, Y., Kubota, N., Kojima, F., Fukuda, T.: Control of behavior dimension for mobile robots. In: Proceedings of the Fourth Asian Fuzzy System Symposium, pp. 652–657, May 2000
10. Robels, R.J., Kim, T.H.: Systems and methods in smart home technology: a review (2010)
11. STMicroelectronics: Nucleo-F446RE Data Sheet (2015)
12. Tjiharjadi, S.: Design of an integrated smart home control (2013)
13. Valtonen, M., Vuorela, T.: Capacitive user tracking methods for smart environments (2012)
14. West, G., Newman, C., Greenhill, S.: Using a camera to implement virtual sensors in a smart house (2003)