Oscillatory Neural Networks for Obstacle Avoidance on Mobile Surveillance Robot E4

Madeleine Abernot, Thierry Gil, Evgenii Kurylin, Tanguy Hardelin, Alexandre Magueresse, Théophile Gonos, Manuel Jiménez Través, María José Avedillo de Juan, Aida Todri-Sanial

Oscillatory Neural Networks for Obstacle Avoidance on Mobile Surveillance Robot E4 *

Anonymous Authors

Abstract

Neuromorphic computing aims to emulate biological neural functions to overcome the memory bottleneck challenges with the current Von Neumann computing paradigm by enabling efficient and low-power computations. In recent years, there has been a tremendous engineering effort to bring neuromorphic computing for processing at the edge. Oscillatory Neural Networks (ONNs) are brain-inspired neural networks made of oscillators to mimic neuronal brain waves, typically visible on Electroencephalograms (EEG). ONNs provide massive parallelism using coupled oscillators and low power computation using oscillator phase dynamics. In this paper, we present for the first time how to use ONNs to perform obstacle avoidance on a mobile robot. Digitally implemented ONNs on FPGA are used and configured for obstacle avoidance inside the industrial surveillance robot E4 from the company, A.I.Mergence. We show that ONNs can perform real-time obstacle avoidance based on the sensory data from proximity sensors embedded on the E4 robot. The highly parallel architecture of ONNs not only allows fast real-time computation for obstacle avoidance applications but also opens up a novel computing paradigm for edge AI to enable low power and real-time sensing to action computing.

Keywords— A ssociative memory, Neuromorphic computing, Oscillatory Neural Networks, Obstacle avoidance.

1 INTRODUCTION

Edge devices and autonomous robots are widely spread and deployed to respond to various industrial services, manufacturing, and even human services [1]. Autonomous robots are of particular interest for space applications [2], underwater treatment and survey [3], household security monitoring and maintenance [4], to mention a few. However, autonomous robots must treat a large amount of continuous data while operating with limited computational resources and power as they are mainly battery operated. Hence, in recent years, academic and industrial communities have been investigating deploying Artificial Intelligence (AI) on edge devices and robots to allow energy efficient computation and autonomous decision making. But, currently deployed AI on edge is mainly based on Artificial Neural Networks (ANNs), such as Convolutional Neural Networks (CNNs), implemented on CPUs or GPUs which are power-hungry and memory restrained [5].

Neuromorphic computing presents an alternative computing paradigm to address these challenges. A well-known neuromorphic computing solution is the Spiking Neural Network (SNN) [6–8]. SNNs use time between spikes to encode information reproducing brain behavior where information is sent through spikes. Using spikes reduces the mean voltage amplitude, which ultimately reduces the power consumption of systems. In addition, researchers have adapted a biological concept to create a spike-timing-dependent plasticity learning algorithm [9, 10], to enable online learning capability on edge devices.

An alternative energy efficient neuromorphic computing based on oscillatory neural networks (ONNs) has been recently developed where the main computing architecture is analog computing of coupled oscillators phase dynamics [11–13]. ONNs employ oscillators as neurons and analog components, such as resistors or capacitors, as synapses to couple oscillators and create physical interactions between them. Phase dynamics of coupled oscillators have been shown to perform auto-associative-memory applications [13]. In ONNs, information is encoded in the phase difference between oscillators, which also reduces signal voltage amplitude and, consequently, reduces power consumption. Currently, ONNs are being explored from various aspects such as materials [14], devices [15–18], analog circuit design [19, 20] to implementation and applications [21–23].

In this paper, we explore the benefits and costs

*This work was supported by the European Union's Horizon 2020 research and innovation program, EU H2020 NEURONN (www.neuronn.eu) project under Grant 871501.

of the ONN computing paradigm for robotic applications. We investigate how ONNs can be used to perform the obstacle avoidance (OAV) in a real industrial autonomous robot named E4, developed by A.I.Mergence [24]. E4 is a mobile robot developed for office security monitoring purposes. One of the main challenges for autonomous mobile robots remains navigation in their environment. For example, avoiding obstacles is a common and important feature to be ensured by robots to navigate correctly in any environment [25].

E4 robot is equipped with multiple sensors and cameras to navigate, monitor and report on the room security such as fire, water, gas, or intruder. It is equipped with fifteen proximity sensors to detect obstacles and hollows and avoid them. In this work, we propose a solution to use ONNs as Auto-Associative Memory (AAM) to perform OAV based on the proximity sensor values provided by the E4 robot. To do so, we implement two digital ONNs configured for OAV on an FPGA board [26], and integrate it in the E4 robot. The main contributions of this paper are:

1. development of a solution using two cascaded ONNs to perform OAV based on proximity sensor data measurements,

2. system implementation in an FPGA board using fully-digital ONNs, and

3. integration of the FPGA board inside the E4 industrial robot and real-time assessment of ONN OAV versus equivalent software solutions.

First, in Section 2, we detail the ONN computing paradigm, from the biological inspiration to the application-level and learning algorithms. Second, in Section 3 we describe the E4 robot with its architecture and features. In addition, we specify how the OAV feature is implemented inside E4. Then in Section 4, we present our ONN OAV solution for the E4 robot. Finally, in Section 5, we describe ONNs implementation inside the E4 robot, and we report on the resources and timing performances of the proposed solution.

2 OSCILLATORY NEURAL NETWORKS

ONNs are novel neuromorphic computing systems based on coupled oscillators inspired by brain wave oscillations visible on Electroencephalograms (EEG). This section presents the biological inspiration of ONN, its computing principle, applications, and learning algorithm.

2.1 ONN Computing Principle and Biological Inspiration

In ONNs, each neuron is an oscillator and coupled by analog elements to represent synapses, such as resistors or capacitors [27]. The information is represented in the phase difference among oscillators, which reduces the voltage amplitude and ultimately reduces the power consumption of the system. For example, considering a reference oscillator with phase 0^o, if we compute only with binary information, an oscillator in-phase (0^o) with the reference oscillator represents a logic '0', and an oscillator out-of-phase (180^o) with respect to the reference oscillator represents a logic '1'. ONNs use the phase dynamics among coupled oscillators to compute. Once oscillators are initialized with input phase information, the coupling between oscillators allows them to dynamically interact and oscillate until they reach stabilization. The phase differences among oscillators once stabilized, represent the output information. Thus, coupling plays an important role as the memory and steering the interactions among oscillators.

2.2 Auto-Associative Memory Applications

In [12], authors showed that when oscillators are fully coupled, they can perform Auto-Associative Memory (AAM) type of tasks, like in Hopfield Neural Networks (HNNs) [28]. AAM tasks consist of memorizing patterns and retrieving those memorized patterns from corrupted ones. For example, with images, we associate a pixel with each oscillator and the phase of each oscillator represents the color of the pixel. The network is trained to memorize images using specific learning algorithms defined in 2.3. As the first step, the learning algorithm computes the synaptic weights, which determines the couplings among oscillators. Once couplings are set then the next step is inference. During inference, we initialize the network with a corrupted image and let the network oscillate till it stabilizes to one of the trained images, see Fig. 1.

2.3 Learning

The main learning algorithm developed for AAM tasks is the Hebbian learning rule [29]. It is an unsupervised learning rule introduced by Hopfield for HNNs and then adapted to ONNs [12]. Unlike supervised learning algorithms, unsupervised learning algorithms only use input data to compute weights without additional interaction. To learn k patterns X^k, the unsupervised Hebbian learning rule defines the weight w_{ij} between neuron i and neuron j as:

$$w_{ij} = \frac{1}{k} \sum_{k} X_i^k {X_j^k}^T \tag{1}$$

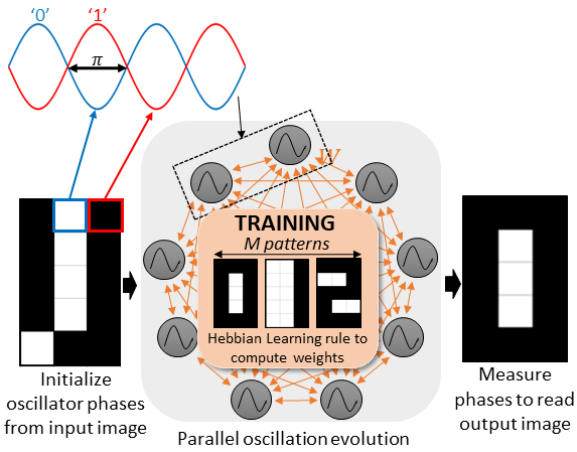

Figure 1: ONN computing paradigm.

with $w_{ij} = 0 \; \forall \; i = j$. Note, Hebbian presents limitations in terms of training images capacity, meaning the maximum number of patterns to be trained while keeping high accuracy. There exist other unsupervised learning rules to counter these limitations [30,31], however, they use more computational resources than Hebbian, though Hebbian capacity is sufficient for this application.

3 E4 ROBOT

E4 robot is developed by A.I.Mergence and it is a surveillance robot developed to ensure office security. It detects human intrusions and environmental risks, such as water, fire, and smoke. Here, we present the E4 robot functionalities and details related to the obstacle avoidance application.

3.1 General Description of E4 Functionalities

E4 robot is a 3-side triangle robot that is 35 cm high, with a 36 cm front side and two 34 cm sides, see Fig. 2. The robot can navigate smoothly in the environment, avoiding obstacles in order to detect anomalies, like water floods or intrusions. Additionally, the robot can interact non-verbally with humans, using sound and luminosity interactions. Available functionalities are:

- 360^o wide human detection
- Sound detection and classification
- 3D environment mapping
- Environment navigation
- Obstacle avoidance
- Flood water detection
- Remote control
- Non verbal human interaction

All these functionalities are enabled by multiple sensors and cameras placed all around the robot, see Fig. 2. The robot analyses the surroundings, sounds and images through sensors, microphones, and smart cameras to detect intrusions. Smart cameras are used to map the environment and ensure good navigation. Fifteen additional Time of Flight (ToF) proximity sensors placed all around the robot ensure correct navigation by detecting obstacles and hollows while avoiding them. Finally, the robot can communicate with the outside world and alert users of an anomaly via a speaker and a LED ring.

The robot's global architecture is shown in Fig. 3. The interface board drives all peripherals, like sensors and cameras, through a CAN communication bus protocol. In addition, specific boards are used for each peripheral. For example, to control LEDs, an additional LED control board is used between the interface board through the CAN bus and the LEDs themselves. In this work, we integrate ONNs implemented on an FPGA board inside the E4 robot to perform the obstacle avoidance application. Thus, we modify the global architecture by connecting an FPGA board to the OAV board to compute the OAV application with ONNs. The following section provides details on how ONN can perform the OAV application in the E4 robot.

3.2 Obstacle Avoidance Application

OAV application uses 15 ToF proximity sensors connected to the OAV board. The 15 ToF sensors are referenced as [32] and can give the distance from $0m$, up to $2m$. They are positioned all around the robot, as shown in Fig. 4. Nine sensors are directed horizontally to detect front objects, and the six others are directed to the ground to detect small ground obstacles and hollows.

OAV board is a Nucleo board referenced as [33].

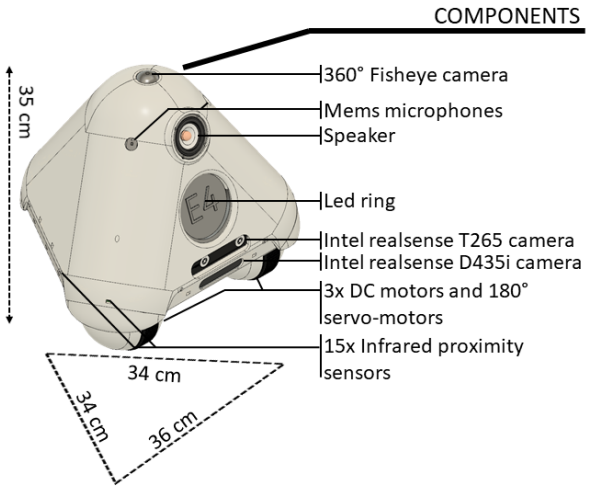

Figure 2: E4 robot functionalities and details on the OAV application.

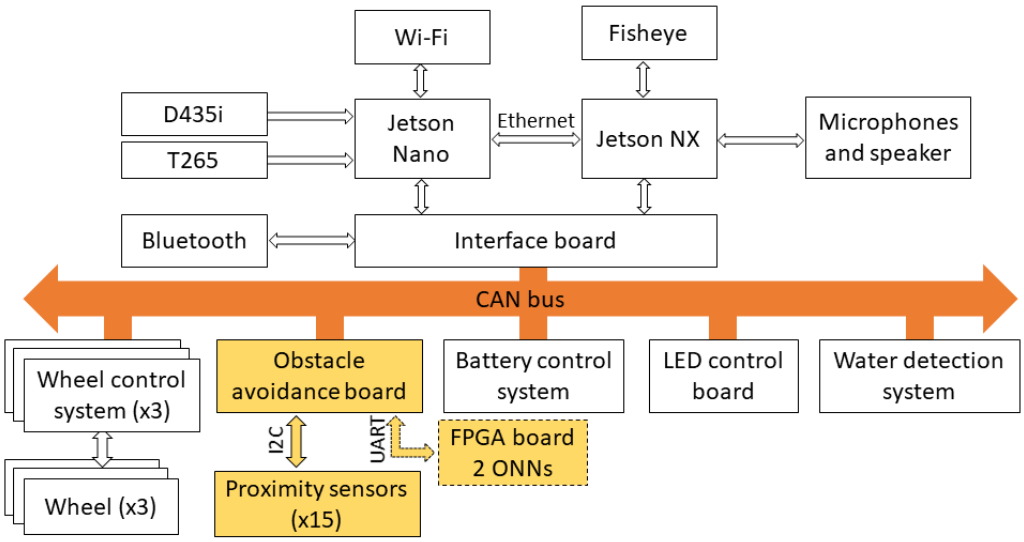

Figure 3: E4 robot global architecture.

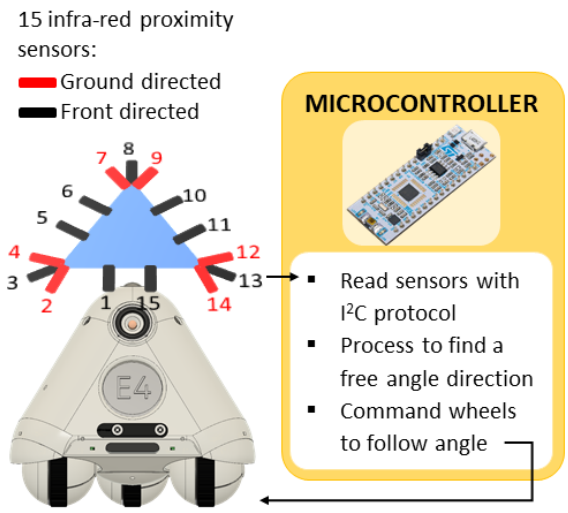

Figure 4: E4 robot components for OAV application.

It is equipped with an STM32 micro-controller. The communication between the OAV board and the proximity sensors is defined by the sensor's manufacturer as the Inter-Integrated Circuit (I^2C) protocol. The I^2C protocol is a serial communication protocol often used in embedded electronics between micro-controller and sensors. Only two wires are necessary for communication, a clock wire to synchronize both sides of the communication and a data wire to transmit data.

The OAV algorithm inside the OAV board first reads the distance measured by each fifteen sensors values using the I^2C protocol. Then, it uses a Braitenberg algorithm [34] to detect if there is an obstacle or a hollow in front of each sensor. From this information, the OAV algorithm deduces directions that are free of obstacles. Finally, one of the available directions is chosen and sent to the wheel control system to avoid any possible obstacles. In this original OAV algorithm version, no information from the previous robot direction is used to define the final direction, however, it can be included to improve the moving flow.

4 ONN FOR OBSTACLE AVOIDANCE APPLICATION

OAV application performs two main functions: 1) detect the obstacles, and 2) avoid them by finding free directions and choosing one. We propose to utilize an ONN for each of these functions, thus in total, two cascaded ONNs to perform the OAV application. The first ONN detects obstacles based on the information gathered from ToF proximity sensors, while the second ONN determines available angle directions from the detected obstacles. This section explains how we encode sensory data into a compatible image for ONN input. Then, we describe the two cascaded ONNs for solving the OAV problem.

4.1 Sensory data encoding

The ToF proximity sensors measure distances between $0cm$ and $2m$ and use the I^2C communication protocol to transmit measured distances to the OAV micro-controller board. We use a thermometer encoding technique to encode each sensor value into a 5x1 column image. The thermometer encoding is adjusted depending on the orientation of the sensor, see Fig. 5. If the sensor is horizontally directed, we directly encode the measured value in a 6-state column (from 0 to 5). However, if the sensor is downward directed (i.e., directed to the ground), we define a default ground value and encode the difference between the measured and the ground value.

Based on these encoded data, we create a 5x9 image by considering the 9 front-directed sensors, one for each column of the image. Also, we add the ground-directed sensor values from sensors closed to each front-directed sensor, see Fig. 5. We fix the maximum value to 5 as each column only has 5 pixels. If the sum is greater than 5, we round it to 5. This method allows us to create areas for each front-directed sensor that take into account values from both front- and ground-directed sensors.

Finally, we obtain an image representing a 360^o *map* of the environment around the robot. Each column represents an area around the robot, and the number of black pixels in each column corresponds to distance with possible obstacles for each area, see Fig. 5. This sensory data encoding is done on the OAV micro-controller board, and the final *map* image is then sent to the FPGA. It allows us to represent an all-around environment of the robot in terms of obstacles and hollows.

4.2 ONN Definition

As aforementioned, the first ONN detects obstacles and hollows, while the second ONN defines free directions (angles) that the robot can take. The first ONN gives as output the black pixels on each column depending on the proximity of the obstacle or hollow. To do so, we train the first ONN with 512 images corresponding to all possible combinations of full-column images representing a wide range of obstacles and hollows. For example, if an obstacle is close enough to be avoided, the ONN will output a black column in the corresponding area of the obstacle. However, if there is no obstacle or if the obstacle is far enough to be avoided later, the ONN will output a white column in the corresponding area. So, ONN identifies if there are obstacles or hollows in the different defined areas, see Fig. 6.

The output of the first ONN contains an image of black and white columns and each row is identical. Thus, the second ONN uses one row as input from the first ONN and outputs patterns corresponding to free directions. We define six possible directions corresponding to the three triangle phases in addition to the three triangle angles. We combine columns corresponding to those six areas and train the second ONN with all possible combinations of those six areas, see Fig. 6. There are 64 possible outputs for the second ONN. These outputs are then sent back to the OAV board, which decides the direction to follow. We define the best direction as the middle of the larger available area.

5 ONN IMPLEMENTATION ON E4 ROBOT

We test our solution with the two cascaded ONNs implemented on an FPGA board integrated into the E4 robot. In Sec. 5.1, we detail how we implement the ONNs on FPGA and how we integrate the FPGA board inside the E4 robot. Then, in Sec. 5.2,

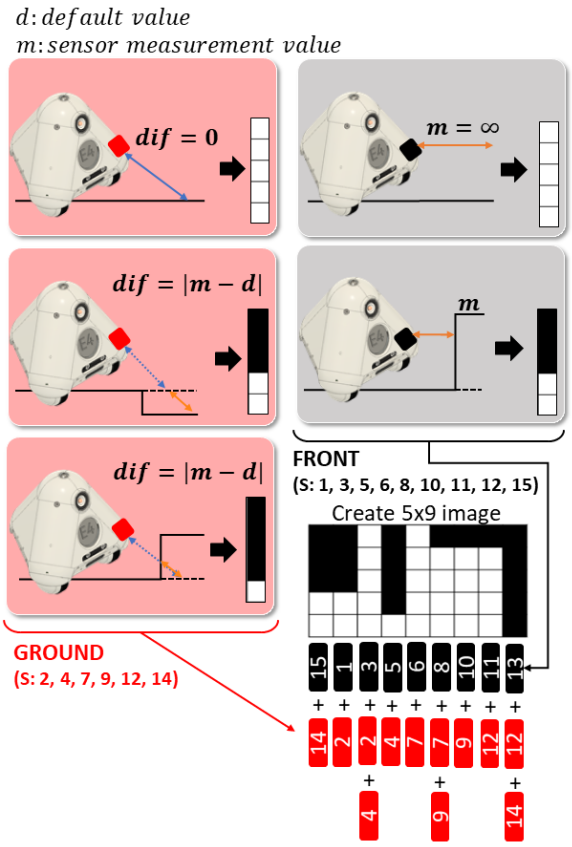

Figure 5: Encoding data from sensors into images, using one sensor per column encoding.

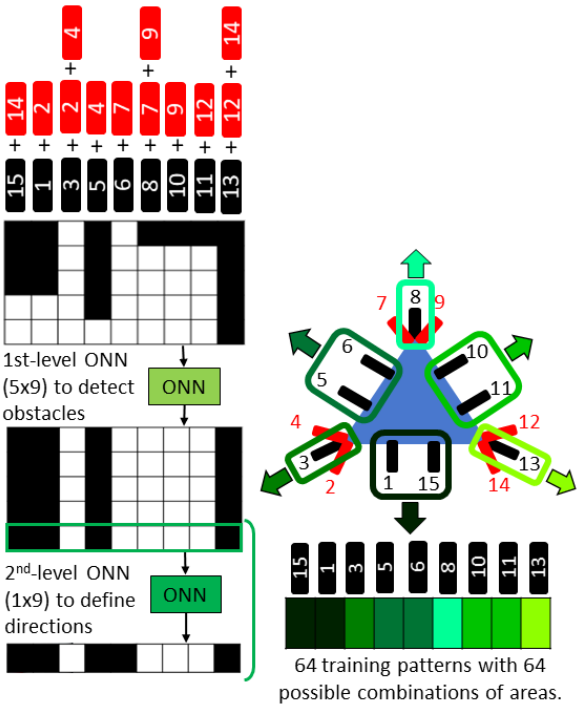

Figure 6: ONNs flow for the obstacle avoidance application.

we present the results obtained with our ONN solution and compare them with existing OAV software algorithms.

5.1 Methods

ONNs are implemented on the CMOD A7 development board from Digilent [35], which contains an Artix-7 FPGA, following the ONN digital design from [26]. Sensory data measurements and their encoding to ONN are performed via the micro-controller on the OAV board. We connect the OAV board with the FPGA board using a Universal Asynchronous Receiver Transmitter (UART) communication protocol, see Fig. 7.

The UART communication is a well-known serial communication protocol used in embedded electronics. It is a particular serial communication as it is asynchronous. It uses only two wires, which is an attractive solution for embedded devices. One wire allows communication from device A to B, while the other wire communicates between device B to A. Its main drawback is the communication latency as each bit is sent serially. We choose UART as the communication protocol, first for its simple implementation, and then because in this OAV application, the latency of sensor measurements is higher than UART latency, see in Section 5.2.

A fully digital ONN design was proposed in [26] for pattern recognition application. It uses digital phase-controlled oscillators as neurons and 5-bit memory registers as synapses to store weights. In this application, we use two cascaded digital ONNs. We initialize the first ONN with sensory data information from the OAV board. Then, we let it compute and wait until ONN stabilizes and its output initializes the second ONN. After, we let the second ONN compute and stabilize, and we send its output to the OAV board. We repeat this process every time new sensory data are available.

The entire process starts with the OAV board, which reads the distance information from the 15 ToF proximity sensors using the I^2C communication bus. The OAV board micro-controller encodes the information into a 5x9 image as described in Section 4.1 and sends the image to the FPGA board through the UART communication interface. The computation of both ONNs, one by one, is done inside the FPGA board. Then, the output of the second ONN, corresponding to obstacle-free areas around the robot, is sent back to the OAV board through the UART communication interface. The OAV board finally chooses one of the best directions from all obstacle-free directions and sends the final direction to the wheel control system to move the robot in the corresponding direction.

To compare our ONN OAV solution with classical software solutions, we develop and implement equivalent algorithms in the micro-controller of the OAV board. We create a first algorithm to detect obstacles from sensory data measurements and a second algorithm to reproduce the behavior of the second ONN and define available directions. We measure the computation time of each algorithm using an oscilloscope and compare it with the computation time of each ONN. Results are presented in Section 5.2.

5.2 Results

Here, we report on our ONN OAV solution's timing and computational resources implemented on the FPGA. In addition, we compare with classical software solutions implemented on the OAV board micro-controller. Performance results are shown in Table 1, and computation time comparisons between ONN and software algorithms are shown in Table. 2.

First, we observe that our ONN OAV solution uses a small number of resources, up to 20% of LUTs, and up to 8% of Flip-Flops of the Artix-7 FPGA. We note that Artix-7 FPGA is a small size and low-power FPGA [36], meaning that the digital ONN can easily be integrated into industrial systems without a significant power overhead. The total initialization and computation time of both ONNs is about 40 μs, which is much shorter than the time needed to read the 15 distance values from the ToF proximity sensors. The measurement time for the 15 sensors has also been optimized by using the continuous reading method.

It is important to point out that the ONN initialization time is higher than the computation time for the second ONN. This is because the ONN initialization process is serial (one bit at a time), thus, the initialization time increases linearly with the number of neurons while the computation time stays constant due to parallel computation. This application shows that only two oscillation cycles are necessary for ONN to stabilize, reaching around 10 μs computation time for both ONNs. Thus, our ONN solution satisfies the real-time constraints given by sensors measurements. ONN computation time is irrelevant in this application as the sensor's measurement limits timing performances. However, ONN can be advantageous for other robotic applications treating sensory data with real-time constraints.

Comparing the ONN computation time with classical software solutions shows that ONNs are as fast as using software algorithms implemented on a micro-controller, see Table 2. We compare only computation times to have a meaningful comparison. It is also important to state that the OAV micro-controller runs at 80 MHz while ONN only oscillates at 187.5 kHz. This indicates that accelerating FPGA frequency for ONNs can surpass the

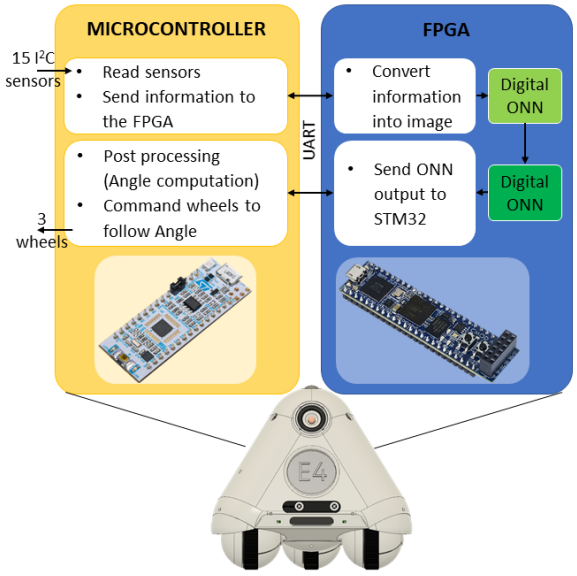

Figure 7: System architecture inside E4 of the ONN on FPGA.

Table 1: Performance of the obstacle avoidance function on the E4 industrial robot.

Demonstrator characteristics	2-ONN solution	
ONNs Performances FPGA - Artix 7: xc7a35	ONN 5x9	ONN 1x9
#Training patterns	512	64
LUTs - 100%: 33 280 (%)	20.07	
Flip-Flops - 100%: 41 600 (%)	7.74	
ONN frequency F_{osc}(KHz)	187.5	
ONN Initialization	15 us	4 us
Computation time	10 us (2 T_{osc})	10 us (2 T_{osc})
Accuracy (%)	100	100
Full system performances (FPGA frequency: 12 MHz)		
15-sensor measurement	27 ms	
UART sensor value transmission	8 ms	
UART direction reception	500 us	
FPS	30	
ONNs Init and Comp. time	44 us (ONN 5x9 + ONN 1x9)	
Robot life time estimation	2h/3h	

micro-controller timing performances for OAV applications.

In addition, timing performances of software solutions implemented on micro-controllers grow linearly with the number of data to treat, while the computing time of ONN remains constant (i.e., only a few cycles to stabilize) when increasing its size. So, if there are many data to treat, ONN computation will be faster than a software solution. Another solution to improve timing performances in case of a large number of sensory data is to use multiple ONNs to compute in parallel. As ONN FPGA resources are low, we can easily duplicate them to reach a fast computation time.

Finally, we show in Table 1 that the global timing performance of our ONN solution is slowed down by the UART serial communication between the OAV board and the FPGA. Another solution for applications with higher real-time requirements will be to consider an complete FPGA implementation with programmable logic for sensor reads. However, such integration inside a complete industrial robot requires anticipation during the system design and development.

6 CONCLUSION

In this paper, we presented a solution to solve the obstacle avoidance (OAV) problem from ToF proximity sensor measurements using two cascaded ONNs configured as an auto-associative memory. We implemented this solution on an FPGA board and integrated it on the industrial E4 robot developed by A.I.Mergence, equipped with 15 proximity sensors. We demonstrate our solution using two cascaded ONNs performing obstacle avoidance on a mobile robot that respects real-time requirements [*A video link will be provided after paper acceptance*]. Only 40 μs are needed to initialize and compute both ONNs, while 27 ms are required to measure the 15 sensor values. We compare the computation time of our two ONNs with equivalent software solutions implemented on the OAV microcontroller and obtain similar performances. However, suppose the software algorithm implemented on the micro-controller needs to treat more sensory data. In that case, the computation time will increase linearly, while with ONN parallel computation, the computation time stays constant. Another advantage of ONN is that it can be easily integrated on edge devices like robots due to its low computational resources, making it easy to duplicate for more parallel computation. To conclude, ONN implementation and integration in an industrial robot to perform obstacle avoidance provides accurate, fast while being a low power solution and requiring little computational resources. This work opens new perspectives to explore ONNs in embedded systems for real-time sensory data treatment.

REFERENCES

[1] Elena Garcia, Maria Antonia Jimenez, Pablo Gonzalez De Santos, and Manuel Armada. The evolution of robotics research. *IEEE Robotics Automation Magazine*, 14(1):90–103, 2007.

[2] Fesq L.M. Volpe R.A. Nesnas, I.A. Autonomy for space robots: Past, present, and future. 2:251–263, 2021.

[3] L. V. Kiselev, A. V. Medvedev, V. B. Kostousov, and A. E. Tarkhanov. Autonomous underwater robot as an ideal platform for marine gravity surveys. In *2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, pages 1–4, 2017.

[4] Chanin Joochim. Autonomous security robot base on slam. In *2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 959–962, 2021.

[5] Hoang-The Pham, Minh-Anh Nguyen, and Chi-Chia Sun. Aiot solution survey and comparison in machine learning on low-cost microcontroller. In *2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 1–2, 2019.

[6] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, 1997.

[7] Hélène Paugam-Moisy and Sander Bohte. *Computing with Spiking Neuron Networks*, pages 335–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[8] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, 06 2018.

[9] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J Neurosci.*, pages 10464–72, Dec 1998.

[10] Rajesh P. N. Rao and Terrence J. Sejnowski. Spike-timing-dependent hebbian plasticity as temporal difference learning. *Neural Comput.*, pages 2221–37, Oct 2001.

Table 2: Computation time comparison between sofware solutions and ONN solution

	Software solution (STM32 @80MHz)	ONNs solution (@187.5 KHz)
Detect obstacles	15 us	10 us (2 osc cycles)
Define direction	5 us	10 us (2 osc cycles)
Full system	20 us	20 us

[11] Tetsuro Endo and Kazuhiro Takeyama. Neural network using oscillators. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 75(5):51–59, 1992.

[12] F.C. Hoppensteadt and E.M. Izhikevich. Pattern recognition via synchronization in phase-locked loop neural networks. *IEEE Transactions on Neural Networks*, 11(3):734–738, May 2000.

[13] Gyorgy Csaba and Wolfgang Porod. Coupled oscillators for computing: A review and perspective. *Applied Physics Reviews*, 7(1), March 2020.

[14] Gabriele Boschetto, Stefania Carapezzi, and Aida Todri-Sanial. Quantum Mechanical Simulations of 2D Materials for Unconventional Computing and Biosensing Applications. In *15th International Conference on Materials Chemistry*, Online Event, Ireland, July 2021.

[15] Stefania Carapezzi, Gabriele Boschetto, Corentin Delacour, Elisabetta Corti, Andrew Plews, Ahmed Nejim, Siegfried Karg, and Aida Todri-Sanial. Advanced design methods from materials and devices to circuits for brain-inspired oscillatory neural networks for edge computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 11(4):586–596, 2021.

[16] Jiadi Zhu, Teng Zhang, Yuchao Yang, and Ru Huang. A comprehensive review on emerging artificial neuromorphic devices. *Applied Physics Reviews*, 7(1), March 2020.

[17] Juan Núñez, María J. Avedillo, Manuel Jiménez, José M. Quintana, Aida Todri-Sanial, Elisabetta Corti, Siegfried Karg, and Bernabé Linares-Barranco. Oscillatory neural networks using vo2 based phase encoded logic. *Frontiers in Neuroscience*, 15, 2021.

[18] Jafar Shamsi, María José Avedillo, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Hardware implementation of differential oscillatory neural networks using vo 2-based oscillators and memristor-bridge circuits. *Frontiers in Neuroscience*, 15, 2021.

[19] Aida Todri-Sanial, Stefania Carapezzi, Corentin Delacour, Madeleine Abernot, Thierry Gil, Elisabetta Corti, Siegfried F. Karg, Juan Núñez, Manuel Jimenez, María J. Avedillo, and Bernabé Linares-Barranco. How frequency injection locking can train oscillatory neural networks to compute in phase. *IEEE transactions on neural networks and learning systems*, 2021.

[20] Corentin Delacour, Stefania Carapezzi, Madeleine Abernot, Gabriele Boschetto, Nadine Azemard, Jeremie Salles, Thierry Gil, and Aida Todri-Sanial. Oscillatory neural networks for edge ai computing. In *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 326–331, 2021.

[21] Dmitri E. Nikonov, Gyorgy Csaba, Wolfgang Porod, Tadashi Shibata, Danny Voils, Dan Hammerstrom, Ian A. Young, and George I. Bourianoff. Coupled-Oscillator Associative Memory Array Operation for Pattern Recognition. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 1:85–93, December 2015.

[22] Thomas Jackson, Samuel Pagliarini, and Lawrence Pileggi. An Oscillatory Neural Network with Programmable Resistive Synapses in 28 Nm CMOS. In *2018 IEEE International Conference on Rebooting Computing (ICRC)*, pages 1–7, McLean, VA, USA, November 2018. IEEE.

[23] Gyorgy Csaba, Arijit Raychowdhury, Suman Datta, and Wolfgang Porod. Computing with Coupled Oscillators: Theory, Devices, and Applications. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

[24] A.I.Mergence. E4 robot.

[25] K. Yojitha, B. Bindu Priya, N. Hari Krishna, D. Yashwanth, and G. Anuradha. A survey on obstacle avoidance and traffic light detection approaches for autonomous vehicle. In *2021 International Conference on Intelligent Technologies (CONIT)*, pages 1–4, 2021.

[26] Madeleine Abernot, Thierry Gil, Manuel Jiménez, Juan Núñez, María J. Avellido, Bernabé Linares-Barranco, Théophile Gonos, Tanguy Hardelin, and Aida Todri-Sanial. Digital implementation of oscillatory neural network for image recognition applications. *Frontiers in Neuroscience*, 15:1095, 2021.

[27] Corentin Delacour and Aida Todri-Sanial. Mapping hebbian learning rules to coupling resistances for oscillatory neural networks. *Frontiers in Neuroscience*, 15:1489, 2021.

[28] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, May 1984.

[29] R. G. M. Morris. D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949. *Brain Research Bulletin*, 50(5):437, 1999.

[30] Yue Wu, Jianqing Hu, Wei Wu, Yong Zhou, and K.L. Du. Storage capacity of the hopfield network associative memory. In *2012 Fifth International Conference on Intelligent Computation Technology and Automation*, pages 330–336, 2012.

[31] Pavel Tolmachev and Jonathan H. Manton. New insights on learning rules for hopfield networks: Memory and objective function minimisation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[32] STMicroelectronics. Vl53l0x proximity sensors.

[33] STMicroelectronics. Nucleo l432-kc.

[34] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. 1984.

[35] Digilent. Cmod a7 development board.

[36] Xilinx. 7 series fpgas data sheet: Overview.