Enhancing VSSS Pose Control with Hardware Acceleration and Bio-Inspired Algorithm Optimization

M. A. Pastrana *, Matheus de S. Luiz*, Filipe R. Barbosa†, Roberto de Souza Baptista ‡, Daniel M. Muñoz*‡
*Department of Mechanical Engineering, Mechatronics Graduate Program
†Faculty of Science and Technology in Engineering, Aerospace Engineering Undergraduate Program
‡Faculty of Science and Technology in Engineering, Electronics Engineering Undergraduate Program
University of Brasilia, Brasilia, DF, Brazil
Email: mario.pastrana@ieee.org, matheusluiz@ieee.org, 232002181@aluno.unb.br ,
baptista@ieee.org, damuz@unb.br

***Abstract*—This paper presents a novel approach to enhancing pose control for Very Small Size Soccer (VSSS) robots through the integration of hardware acceleration and bio-inspired optimization algorithms. The proposed system leverages Field-Programmable Gate Array (FPGA) technology to implement a PID controller with varying levels of floating-point precision (11, 16, 23, and 27 bits), enabling efficient and high-performance control. Hardware-in-the-Loop (HIL) simulations, conducted using the CoppeliaSim platform, validate the system's effectiveness, demonstrating significant improvements in control accuracy as precision increases. Two bio-inspired algorithms—Differential Evolution (DE) and Moth Flame Optimization (MFO)—are employed to fine-tune the PID parameters. Both algorithms demonstrate strong convergence, with DE achieving a fitness function value of 0.4172 and MFO reaching 0.3736. The results highlight the potential of combining FPGA-based hardware acceleration with bio-inspired optimization to address the dynamic and demanding challenges of the VSSS competition robot.**

***Index Terms*—bio-inspired algorithm, FPGA, HIL, VSSS.**

I. Introduction

The VSSS (Very Small Size Soccer) competition represents a dynamic and challenging environment for robotics, where precision, speed, and adaptability are paramount [1]. One of the critical aspects of success in VSSS is effective pose control, which ensures that robots can accurately navigate the field, respond to opponents, and execute strategies in real time [2]. However, achieving high-performance pose control in such a fast-paced and unpredictable setting requires innovative approaches that combine computational efficiency with advanced control strategies.

In recent years, hardware acceleration has emerged as a powerful tool for enhancing the performance of robotic systems [3], [4]. By offloading computationally intensive tasks to specialized hardware, such as FPGAs (Field-Programmable Gate Arrays) that can achieve significant improvements in speed and efficiency resources consumptions [5]. Although enhanced control of robotic platforms can be achieved by implementing controllers in FPGAs, few studies have explored this approach. One notable example is the work of Li *et al.* [6], which demonstrated the implementation of speed control for a VSSS competition robot using an FPGA.

Complementing hardware acceleration, bio-inspired algorithms provide a powerful framework for addressing complex control challenges [7]. Inspired by natural systems, such as swarm intelligence [8] and genetic mutation [9], these algorithms are commonly applied to tasks like path planning [10] and vision-based approaches [11] in the context of the VSSS competition. Moreover, these algorithms can also enhance position control, as demonstrated by Cheng *et al.* [12], which utilized the Particle Swarm Optimization (PSO) algorithm and the Genetic Algorithm (GA) to optimize a fuzzy controller for a VSSS robot.

To validate and refine these advanced control strategies without the need for a physical platform, Hardware-in-the-Loop (HIL) simulation serves as a critical testing tool [13]. HIL allows for the integration of real hardware components, such as microcontrollers or sensors, into a simulated environment, enabling rigorous testing of pose control algorithms under realistic conditions [14], [15]. This approach not only reduces development time and costs but also provides a safe and controlled setting to identify and address potential issues before deploying the system in the actual competition. By leveraging HIL, we can ensure that the combined power of hardware acceleration and bio-inspired algorithms is fully optimized for real-world performance.

Due to the limited research on enhancing pose controllers for the VSSS competition through FPGA-based solutions and bio-inspired algorithms, this paper addresses this gap by exploring the integration of hardware acceleration using FPGAs. Specifically, it leverages floating-point arithmetic with varying precision levels (11, 16, 23, and 27 bits) and evaluates two bio-inspired algorithms: one swarm-based method, the Moth Flame Optimization (MFO), and one evolutionary-based approach, the Differential Evolution (DE) algorithm. These

979-8-3315-6017-1/25/$31.00©2025 IEEE

techniques are applied to improve pose control in the VSSS context. Furthermore, the system is rigorously tested and validated using HIL simulation with CoppeliaSim software, eliminating the need for a physical robot platform while ensuring robust performance evaluation.

II. Theoretical Background

This section outlines the theoretical foundation for the paper's key components: the VSSS competition robot, FPGA-based hardware acceleration, bio-inspired algorithms (MFO and DE), and HIL simulation for system validation.

A. Very Small Size Soccer kinematics robot

The Very Small Size Soccer (VSSS) competition is a dynamic and challenging robotic soccer event where small, autonomous robots compete in a fast-paced environment. The robots, constrained to a size of 7.5 cm × 7.5 cm × 7.5 cm and a weight of 500 grams, must navigate a small field, avoid opponents, and score goals while operating autonomously. Figure 1 illustrates the kinematic model of a wheeled mobile robot used in the VSSS, featuring two coaxial rear driving wheels. In this model, P denotes the geometric center of the line connecting the left and right wheels, L represents the distance between the wheels, and d is the wheel diameter. The velocities of the left and right wheels are given by V_l and V_r, respectively. The robot's pose is defined by the vector $q = (x, y, \theta)^T$ in the global coordinate system XOY, where (x, y) are the coordinates of the center point P, and θ is the angle between the robot's direction of motion and the horizontal axis [16].

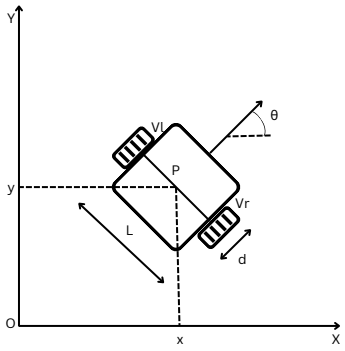


Fig. 1: Differential robot

To calculate the values of V_l and V_r based on the kinematic model, the differential drive robot equations are used as shown below:

$$\dot{x} = \frac{R}{2} \times (V_r + V_l) \times \cos\theta, \tag{1}$$

$$\dot{y} = \frac{R}{2} \times (V_r + V_l) \times \sin\theta, \tag{2}$$

$$\dot{\phi} = \frac{R}{L} \times (V_r - V_l), \tag{3}$$

where $R = \frac{d}{2}$, $\dot{x}$ represents the velocity in the X-direction, $\dot{y}$ represents the velocity in the Y-direction, and ϕ is the angular velocity of the robot. Furthermore, the kinematic equations from [16] were used and are presented below:

$$\dot{x} = V \times \cos\theta, \tag{4}$$

$$\dot{y} = V \times \sin\theta, \tag{5}$$

$$\dot{\phi} = W, \tag{6}$$

where V is the linear velocity of the robot, and W is the angular velocity. Using these equations, V_l and V_r are calculated and presented below:

$$V_l = (2 \times V - W \times L)/d, \tag{7}$$

$$V_r = (2 \times V + W \times L)/d. \tag{8}$$

B. Hardware Reconfiguration

Field Programmable Gate Arrays (FPGAs) are reconfigurable hardware devices whose internal architecture is determined by their granularity, which refers to the smallest functional unit within the device, known as Configurable Logic Blocks (CLBs). In fine-grained architectures, CLBs primarily consist of Look-Up Table (LUT)-based memories for implementing combinational functions and Flip-Flops (FFs) for sequential logic. FPGAs can contain thousands of CLBs, interconnected via a programmable network of connection and routing blocks, enabling the creation of complex digital circuits [17]. Beyond CLBs and FFs, FPGAs include other fundamental components: (a) input/output blocks, (b) digital signal processing (DSP) blocks, (c) RAM memory blocks, and (d) specialized circuits such as PLLs and ADCs. Modern FPGAs are capable of operating at frequencies reaching up to 1 GHz.

This paper employed the AMD-Xilinx SoC FPGA Zynq 7020. The Zynq 7020 features two primary sections: (a) the Processing System (PS), which includes a dual-core ARM Cortex A9 processor running at 650 MHz, and (b) the Programmable Logic (PL) section, which houses the FPGA fabric. Communication between the PS and PL is facilitated through the AXI on-chip protocol.

C. Differential Evolution Algorithm

Differential Evolution (DE) is a stochastic optimization algorithm inspired by biological evolution principles [9], [18]. It begins with a population of M candidate solutions (individuals) initialized randomly and iteratively refined through three key operations: mutation, crossover, and selection.

During the mutation phase, a new individual is generated by adding a scaled difference between two randomly selected individuals to a third. The mutation equation is given by:

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3}), \tag{9}$$

where v_i is the mutated individual, x_{r1}, x_{r2}, and x_{r3} are three randomly selected individuals, and F is the mutation factor that scales the difference between individuals. The crossover operation combines features of the original individual x and the mutant individual v to create a test candidate u, as defined by:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r_{i,j} \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}, & \text{otherwise}, \end{cases} \tag{10}$$

where CR is the crossover rate, determining the probability of incorporating features from v into u. Finally, the selection phase evaluates the fitness of the test candidate u against the original individual x, with the fitter solution advancing to the next generation.

D. Moth Flame Optimization Algorithm

The Moth Flame Optimization (MFO) algorithm is a swarm intelligence-based optimization technique inspired by the navigation behavior of moths around light sources [19]. The algorithm starts with a population of moths initialized randomly, and their positions are updated iteratively based on their attraction to flames, which represent potential solutions.

The position update for each moth relative to a flame is defined by:

$$M_i = S(M_i, F_j), \tag{11}$$

where M_i represents the i-th moth, F_j represents the j-th flame, and S is a spiral function given by:

$$S(M_i, F_j) = D_i \times e^{bt} \times \cos(2 \times \pi \times t) + F_j, \tag{12}$$

where $D_i = |F_j - M_i|$ is the distance between the i-th moth and the j-th flame, b is a constant defining the spiral's shape, and t is a random number in the range $[-1, 1]$. The number of flames decreases adaptively over iterations, as described by:

$$OF = \text{round}\left(N - l \times \frac{N-1}{T}\right), \tag{13}$$

where N is the maximum number of flames, l is the current iteration, and T is the maximum number of iterations. FO has demonstrated successful applications in a wide range of domains, notably in engineering design, image processing, and related fields [20].

III. Methodology

This section details the development and optimization of the VSSS pose control system, focusing specifically on the orientation controller, including the hardware design of the controller on the Zynq 7020 FPGA, a CoppeliaSim-based robot simulator for HIL testing, and a bio-inspired optimization framework to fine-tune controller parameters for robust performance.

A. Orientation Controller on Hardware

The orientation control employs a PID-based angular velocity (W) controller on the Zynq 7020 SoC FPGA and a proportional (P) controller (implemented in software) for linear velocity (V), following established methodologies [21]. The architecture for the angular velocity controller involved a fully hardware-based with 11-bit, 16-bit, and 23-bit floating-point arithmetic precision [22], [23]. Specifically, the 11-bit precision uses a 5-bit mantissa and 5-bit exponent, the 16-bit precision uses a 10-bit mantissa and 5-bit exponent, and the 23-bit precision uses a 14-bit mantissa and 8-bit exponent. The proposed hardware design, illustrated in Fig. 2, includes a PID controller employing six adder blocks and seven multiplier blocks.

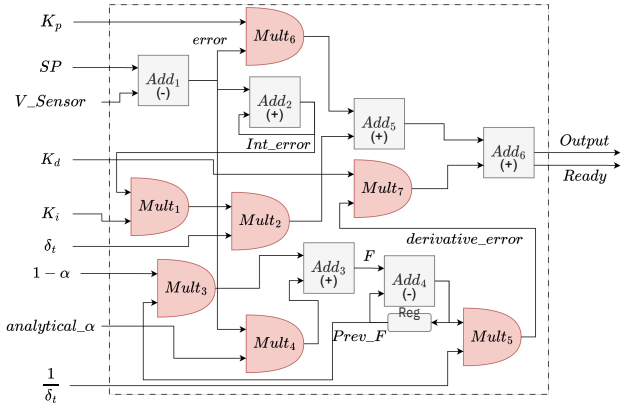


Fig. 2: Full hardware angular velocity controller.

The input *V_Sensor* represents the orientation of the robot, which is obtained from CoppeliaSim and transmitted to the SoC FPGA. The input *SP* (Setpoint) corresponds to the target orientation. The inputs *Kd*, *Ki*, and *Kp* are the PID controller constants. The inputs α and $1-\alpha$ represent the filter coefficients used in the derivative term calculation. Additionally, the input δ_t denotes the sample time. The *Output* is the control signal generated by the PID controller, which, for this application, corresponds to the angular velocity W in Eqs. (7) and (8).

B. Simulator Robot Environment and HIL communication

The robot simulator for the VSSS competition was developed using the CoppeliaSim robot simulation platform and it's present by Luiz *et al* in [1]. The robot model was designed and imported into the simulator, along with a red ball representing the target point that the robot must reach. The orientation of the robot is determined relative to the horizontal axis of the simulator, with an initial orientation of zero degrees. The angle to the target point (ball) is calculated using the arctangent function (atan2), based on the coordinates of both the robot and the target. The simulator accepts wheel speeds as inputs and outputs three critical variables for the pose controller: the robot's current position, its current orientation, and the relative

angle between the robot and the target point, along with the target's position.

Subsequently, Fig. 3 illustrates the HIL communication flow diagram. In this setup, the angular velocity control and the Comblock that use Universal Direct Memory Access (UDMA) [24] are implemented in the Programmable Logic (PL) section of the FPGA. On the other hand, FreeRTOS is utilized in the Processing System (PS) of the ARM core to facilitate communication between the Pynq Z2 board and the computer via Ethernet communication. The data from the controller is transmitted to a Python program, which sends the calculated speed values to the robot within the simulator. Conversely, the orientation values from the simulator are sent back to the angular velocity controller in the PL section for further processing.

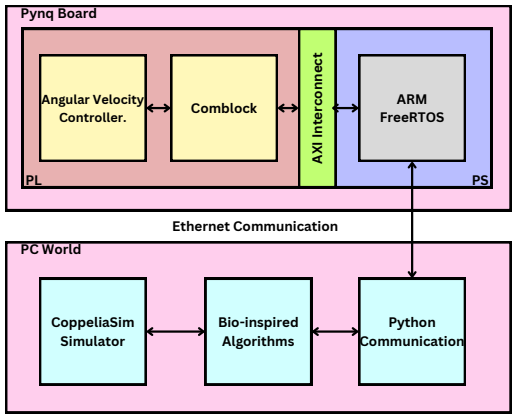


Fig. 3: Flow communication HIL

By utilizing the HIL flow chart, the tuning of the angular velocity control can be efficiently carried out using bio-inspired algorithms. These algorithms generate potential tuning solutions for the controller, which are then transmitted to the controller and simulated. This iterative process continues until the bio-inspired algorithm converges or reaches a stopping condition. The specific parameters employed in the bio-inspired algorithms are detailed in the following subsection.

C. Bio-inspired algorithm Parameters

To train the PID controller, the framework depicted in Fig. 3 was employed, alongside the bio-inspired algorithms DE and MFO. While parameters for bio-inspired algorithms can be fine-tuned, as demonstrated by Nobile *et al.*, who optimized PSO parameters such as coefficients C_1 and C_2 and the inertia factor [25], this study adopts fixed parameter values. The choice of 7 agents was guided by empirical experiments. The parameters for the MFO algorithm were set according to their standard configurations, while the parameters for DE (Differential Evolution) were adopted from previous studies [19], [26], with the crossover rate (Cr) and scaling factor (F) set to 0.75 and 1.25, respectively. The PID controller model involves a three-dimensional parameter space defined by K_p, K_i, and K_d. The search space for these parameters was constrained to a maximum of 0.8 and a minimum of 0.0001. Each algorithm was executed with maximum of iteration of 20. To ensure the reliability and robustness of the optimization results, the entire process was repeated five times, ultimately producing two optimized controllers.

The fitness function is designed to fine-tune the angular velocity controller for enhanced system performance, ensuring efficient operation while minimizing error and achieving stability. To address the challenge the Mean Square Error (MSE) of angles and the euclidean distance is used and the complete fitness function is defined as follows:

$$fitness = \frac{1}{n} \times \sum_{i=1}^{n} \left|\theta_i - \hat{\theta}_i\right| + \sqrt{(x-\hat{x})^2 + (y-\hat{y})}, \quad (14)$$

where, θ represents the robot's current orientation, $\hat{\theta}$ denotes the ideal or target orientation, x and y correspond to the robot's current position coordinates, and $\hat{x}$ and $\hat{y}$ represent the target position coordinates. The following section describes the hardware architecture designed for implementing the optimized angular velocity control system.

IV. Results

This section presents the experimental results of the proposed angular velocity control, including FPGA-based controller simulations, hardware resource utilization, and performance evaluation using bio-inspired optimization algorithms. The findings demonstrate the effectiveness of the system in achieving high precision and efficient control for VSSS robots.

A. Orientation Controller simulated on FPGA

To validate the proposed orientation controller, a test-bench file and an input file containing 600 random values, along with random values for K_p, K_i, and K_d, were generated using Python software. The output corresponding to these inputs, which utilized 64-bit floating-point precision, was saved in decimal format to facilitate comparison with the simulation results of orientation controllers using 11, 16, 23, and 27-bit floating-point precision. Table I presents the MSE between the Python-generated file and the simulation results of the controllers.

TABLE I: MSE comparison between 64-bits Python precision and the proposed angular velocity controller

	11 bits	**16 bits**	**23 bits**	**27 bits**
MSE	52.6229	0.0040	2.6193×10^{-5}	2.6534×10^{-7}

Initially, it can be observed that the MSE is relatively high when using 11 bits, but it decreases as the precision size increases, reaching values as low as $\times 10^{-7}$ in the case of 27 bits.

B. Orientation Controller Implemented on FPGA

The implementation of the different controllers using 11, 16, 23, and 27-bit floating-point precision was carried out

using Vivado software . The consumption of LUTs, FF, DSPs, and energy usage can be observed in Table II. From this, it is evident that the 11-bit controller consumes the fewest resources; however, it also exhibits the highest MSE. For this reason, it is discarded for implementation in the HIL system. On the other hand, the 16, 23, and 27-bit controllers utilize the same number of DSPs but vary in LUT consumption (2.31%, 3.27%, and 4.72%, respectively), FF usage (0.32%, 0.43%, and 0.51%, respectively), and energy consumption (0.005 W, 0.007 W, and 0.011 W, respectively).

TABLE II: Hardware consumption of the PID controller on the Zynq7020 device.

	LUTs	DSP	FF	Power (W)
PID 11 Bits	958 1.80%	0 0%	240 0.23%	0.003
PID 16 Bits	1230 2.31%	7 3.18%	340 0.32%	0.005
PID 23 Bits	1741 3.27%	7 3.18%	462 0.43%	0.007
PID 27 Bits	2510 4.72%	7 3.18%	542 0.51%	0.011

Figure 4 shows the layout of each of the controllers implemented on the Zynq 7020. The 11-bit controller is highlighted in yellow, the 16-bit controller in red, the 23-bit controller in blue, and the 27-bit controller in purple. The cyan-highlighted elements represent supplementary components synthesized by Vivado in hardware to facilitate communication between the FPGA and the CoppeliaSim simulator

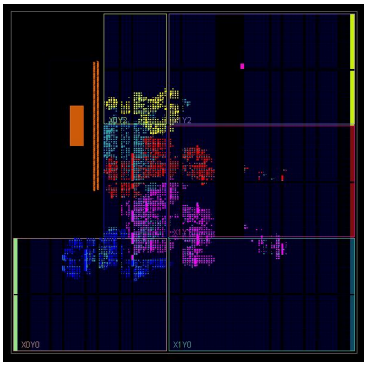
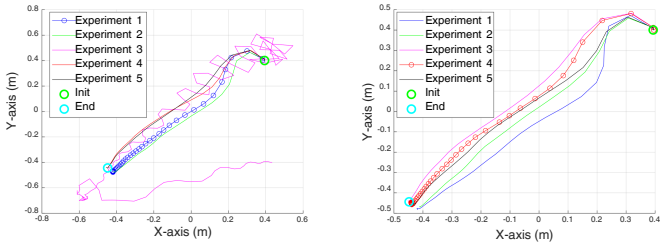


Fig. 4: Layout for each PID controller.

Based on these resource consumption results, along with the simulation precision results from the previous section, the 23-bit controller is chosen for HIL implementation. This controller offers good precision on the order of $\times 10^{-5}$ while consuming fewer resources and less energy compared to the 27-bit controller.

C. HIL robot simulator results

The results of the Hardware-in-the-Loop (HIL) implementation are presented in the YouTube playlist for the MFO [27] and DE [28] algorithms. In the training video, it is evident that during the initial iterations, the bio-inspired algorithms exhibit suboptimal performance, causing the robot to fail in reaching the target point. However, as the iterations progress, the algorithms converge to better solutions for the orientation controller, ultimately achieving a controller capable of guiding the robot to the desired position in the final iterations.



(a) DE algorithm. (b) MFO algorithm

Fig. 5: Results of the bio-inspired experiments

Figure 5 illustrates the performance of the controller in all experiments, with the best-performing experiment for each algorithm highlighted with circles. The DE algorithm achieved its best result in Experiment 1, with a fitness function value of 0.4172 and controller constants $K_p = 0.3629$, $K_i = 0.1891$, and $K_d = 0.0001$, completing in 165 minutes. For the MFO algorithm, the best result was obtained in Experiment 4, with a fitness value of 0.3736 and constants $K_p = 0.3278$, $K_i = 0.0774$, and $K_d = 0.0003$, requiring 80 minutes.

V. Conclusion

This paper presented a comprehensive approach to enhancing pose control for Very Small Size Soccer (VSSS) robots by integrating hardware acceleration and bio-inspired optimization algorithms. The proposed system leveraged the computational power of FPGA-based hardware to implement a PID controller with varying levels of floating-point precision (11, 16, 23, and 27 bits). Through rigorous Hardware-in-the-Loop (HIL) simulations using CoppeliaSim, the system was validated, demonstrating significant improvements in control performance as the precision of the FPGA implementation increased. The 23-bit precision controller was identified as the optimal choice, offering a balance between high precision (MSE on the order of 10^{-5}) and efficient resource utilization.

The bio-inspired algorithms, Differential Evolution (DE) and Moth Flame Optimization (MFO), were employed to fine-tune the PID controller parameters. Both algorithms demonstrated their ability to converge to effective solutions, with DE achieving a fitness function value of 0.4172 and MFO reaching 0.3736. The iterative optimization process highlighted the adaptability of these algorithms in refining the controller's performance, particularly in the later iterations where the robot successfully reached the target point.

The results underscore the potential of combining hardware acceleration with bio-inspired optimization techniques to address the dynamic and demanding challenges of the VSSS competition. The FPGA-based implementation not only

https://www.amd.com/es/products/software/adaptive-socs-and-fpgas/vivado.html

provided a robust and efficient platform for real-time control but also demonstrated the feasibility of using HIL simulations to validate and optimize control strategies without the need for physical hardware.

A. Limitations and Future Works

Despite the promising results, this study has certain limitations. The current implementation is limited to a single robot in a simulated environment, and real-world scenarios involving multiple robots and dynamic obstacles may present additional challenges. Future work could focus on extending the system to support multi-robot coordination and collaboration in the VSSS competition, incorporating advanced path-planning and obstacle-avoidance strategies.

ACKNOWLEDGMENT

The authors would like to thank the International Center for Theoretical Physics (ICTP) for its UDMA tool used in the HIL system. This research was partially supported by the Coordination for the Improvement of Higher Education Personnel of Brazil CAPES (88887.775422/2022-00 and 88881.934265/2024-01).

REFERENCES

[1] M. d. S. Luiz, M. A. Pastrana, G. A. de Oliveira e Aguiar, R. B. Costa, and L. F. R. e. Silva, "Behavior-based control with learning from demonstration for path following applied to mobile robots soccer," in *2024 Latin American Robotics Symposium (LARS)*, 2024, pp. 1–6.

[2] T. P. Baldão, M. R. O. A. Maximo, and C. de Azevedo Castro Cesar, "Decision-making for 5x5 very small size soccer teams," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, 2020, pp. 1–6.

[3] Y. Al-Ameri, M. Nguyen, and T. Westerlund, "Fpga-based hardware acceleration for deep learning in mobile robotics," in *2024 IEEE Nordic Circuits and Systems Conference (NorCAS)*, 2024, pp. 1–7.

[4] K. Gac, G. Karpiel, and M. Petko, "Fpga based hardware accelerator for calculations of the parallel robot inverse kinematics," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, 2012, pp. 1–4.

[5] B. Betkaoui, D. B. Thomas, and W. Luk, "Comparing performance and energy efficiency of fpgas and gpus for high productivity computing," in *2010 International Conference on Field-Programmable Technology*, 2010, pp. 94–101.

[6] C. Li, Y. Jiang, Z. Wu, and T. Watanabe, "A multiprocessor system for a small size soccer robot control system," in *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, 2008, pp. 115–118.

[7] A. Ali, Y. Hafeez, S. Muzammil Hussainn, and M. U. Nazir, "Bio-inspired communication: A review on solution of complex problems for highly configurable systems," in *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, 2020, pp. 1–6.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[9] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[10] M. Saska, M. Macas, L. Preucil, and L. Lhotska, "Robot path planning using particle swarm optimization of ferguson splines," in *2006 IEEE Conference on Emerging Technologies and Factory Automation*, 2006, pp. 833–839.

[11] N. Setyawan, N. Mardiyah, K. Hidayat, Nurhadi, and Z. Has, "Object detection of omnidirectional vision using pso-neural network for soccer robot," in *2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2018, pp. 117–121.

[12] S.-W. Cheng, J.-S. Chiou, Y.-C. Hu, K.-Y. Wang, and M.-Y. Shieh, "Application of the ga-pso with the fuzzy controller to the robot soccer," in *Proceedings of SICE Annual Conference 2010*, 2010, pp. 2083–2087.

[13] P. Sarhadi and S. Yousefpour, "State of the art: hardware in the loop modeling and simulation with its applications in design, development and implementation of system and control software," *International Journal of Dynamics and Control*, vol. 3, no. 4, pp. 470–479, Dec 2015. [Online]. Available: https://doi.org/10.1007/s40435-014-0108-3

[14] M. A. Pastrana, L. H. Oliveira, M. S. Santana, V. C. Oliveira, J. Mendoza-Peñaloza, and D. M. Muñoz, "Hardware implementation of a gmdh controller for mobile robot obstacle following/avoidance," in *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, 2023, pp. 206–211.

[15] M. A. Pastrana, L. H. Oliveira, J. Bautista, J. Pinto-Lopera, J. Mendoza-Peñaloza, and D. M. Muñoz, "Sub-optimal perceptron controller using woa algorithm for farming watering mobile robot," in *2024 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2024, pp. 1–6.

[16] H. Chen, Z. Wang, X. Liu, W. Ma, Q. Wang, W. Zhang, and T. Zhang, "A hybrid motion planning algorithm for multi-mobile robot formation planning," *Robotics*, vol. 12, no. 4, 2023. [Online]. Available: https://www.mdpi.com/2218-6581/12/4/112

[17] K. Masselos and N. S. Voros, *Introduction to Reconfigurable Hardware*. Boston, MA: Springer US, 2005, pp. 15–26.

[18] M. F. Tasgetiren, O. Bulut, and M. M. Fadiloğlu, "A differential evolution algorithm for the economic lot scheduling problem," in *2011 IEEE Symposium on Differential Evolution (SDE)*, 2011, pp. 1–6.

[19] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[20] S. K. Sahoo, A. K. Saha, A. E. Ezugwu, J. O. Agushaka, B. Abuhaija, A. R. Alsoud, and L. Abualigah, "Moth flame optimization: Theory, modifications, hybridizations, and applications," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 391–426, Jan 2023. [Online]. Available: https://doi.org/10.1007/s11831-022-09801-z

[21] W. Serralheiro, "A motion control scheme for a WMR based on input-output feedback linearization and PID," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, 2019, pp. 222–227.

[22] D. M. Muñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "FPGA based floating-point library for cordic algorithms," in *2010 VI Southern Programmable Logic Conference (SPL)*, 2010, pp. 55–60.

[23] D. Muñoz, D. Sanchez, C. Llanos, and M. Ayala-Rincon, "Tradeoff of FPGA design of a floating-point library for arithmetic operators," vol. 5, 03 2010.

[24] M. L. Crespo, F. Foulon, A. Cicuttin, M. Bogovac, C. Onime, C. Sisterna, R. Melo, W. Florian Samayoa, L. G. García Ordóñez, R. Molina, and B. Valinoti, "Remote laboratory for E-Learning of systems on chip and their applications to nuclear and scientific instrumentation," *Electronics*, vol. 10, no. 18, 2021.

[25] M. S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, G. Mauri, and G. Pasi, "Fuzzy self-tuning PSO: A settings-free algorithm for global optimization," *Swarm and Evolutionary Computation*, vol. 39, pp. 70–85, 2018.

[26] M. Arafa, E. A. Sallam, and M. M. Fahmy, "An enhanced differential evolution optimization algorithm," in *2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, 2014, pp. 216–225.

[27] M. Pastrana. (2025) The playlist for the tuning the orientation controller using the MFO algorithm. Accessed on February 28, 2025. [Online]. Available: https://www.youtube.com/playlist?list=PLPnSV069ZnAJDGYxc-8qK_a1ffl0dz1zL

[28] ——. (2025) The playlist for the tuning the orientation controller using the DE algorithm. Accessed on February 28, 2025. [Online]. Available: https://www.youtube.com/playlist?list=PLPnSV069ZnALDO2oHKTn4n6Jq6vndYD9i