

Performance Evaluation of Reinforcement Learning Algorithms for Navigation in Unstructured Environments

Francesco Cancelliere¹, Giuseppe Sutura², Dario Calogero Guastella², Simone Palazzo², Giovanni Muscato², and Concetto Spampinato²

¹Dipartimento di Ingegneria Elettrica, Elettronica ed Informatica, University of Catania, e-mail: name.surname@phd.unict.it

²Dipartimento di Ingegneria Elettrica, Elettronica ed Informatica, University of Catania, e-mail: name.surname@unict.it

KEYWORDS

unmanned ground vehicle, reinforcement learning, autonomous navigation, outdoor unstructured environments.

ABSTRACT

Autonomous navigation in unstructured outdoor environments presents significant challenges due to the complexity and variability of terrain and obstacles. Traditional navigation methods struggle with adaptability, while learning-based approaches, particularly deep Reinforcement Learning (RL), have shown promise but faces difficulties in generalization and data efficiency. To address these challenges, we tested the performance of various deep RL algorithms for point-goal navigation, on MIDGARD, our photorealistic simulation platform built on Unreal Engine. We compare PPO, A2C, SAC, and TD3, evaluating their effectiveness based on success rates and reward progression. Our results indicate that PPO outperforms other algorithms, achieving the highest success rate, while off-policy methods struggle due to inefficient exploration and policy updates.

INTRODUCTION

Navigating autonomous ground robots in complex outdoor and unstructured environments is inherently difficult. Classical methods rely on hard-set rules that are difficult to adapt to different environments. They usually employ costmaps built from heterogeneous sensors, such as cameras and LiDARs, and rely on drift-prone proprioceptive sensors, which skew the state estimations.

Learning-based methods have proven to be effective for navigation in structured environments. Most of these methods are usually integrated within the classical navigation stack, replacing parts of it, from mapping, to costmap generation, and localization (Xiao et al., 2022). End-to-end models, instead of a modular approach, have achieved good results on navigation tasks in visually realistic simulations (Wijmans et al., 2020), but still lack good results in the real world. Although end-to-end approaches have

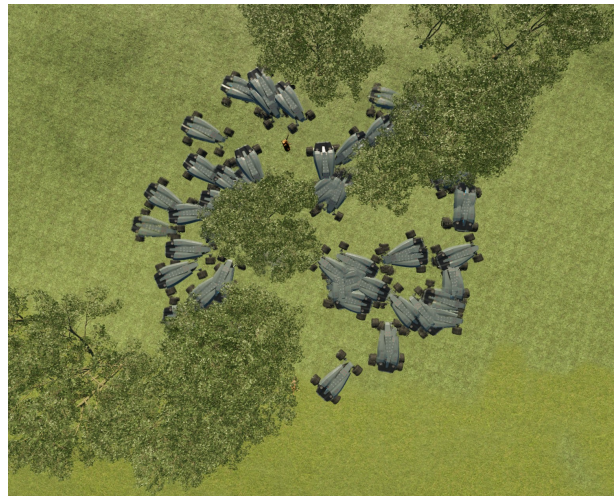


Figure 1: MIDGARD Simulator multi-agent training in Forest Environment

shown promise in simulation settings, their performance often degrades in off-distribution conditions. A key obstacle is the need for extensive data across diverse conditions -both for the environment and the agent- to effectively train robust deep learning models. Real world training is further complicated by high cost, in terms of training time and physical deployment, risk to vehicles, and limited variability in training scenarios, emphasizing the importance of simulation.

In recent years, deep Reinforcement Learning (RL) has been posed as the next logical step for learning navigation tasks. The enticing idea at the core of the biologically inspired algorithms seems the closest to how humans learn: interacting with the world and exploring new actions and ideas (Tang et al., 2024). Among the issues for training with Reinforcement Learning, the most critical is its difficulty to generalize, given the sampling inefficiency of the training algorithms. This is further exacerbated by the gap between the real world and simulation distributions, creating a bottleneck for deployment in real scenarios. To mitigate this issue, one option is to train by using simulation engines capable of creating photorealistic environments,

thus reducing the sim-to-real distribution shift. Another detrimental effect of RL algorithms’ sampling inefficiency is the difficulty of learning from scratch a robust policy for highly complex tasks. A common method to address this issue is to use Imitation Learning and Reinforcement Learning in conjunction. The former is used to build up a policy from an expert agent, and then the latter for fine-tuning the downstream task. This method has been proven to be successful in learning complex (Lee et al., 2020) and vision-guided (Xing et al., 2024) tasks.

The main objective of this paper is to evaluate the performance of an expert agent that has access to privileged information about the world. This policy can then be used to distill a task-specific policy relying on noisy high-dimensional sensory data, such as, e.g., RGB images. The expert agent is trained to navigate in a forest-like environment with well-established RL algorithms, using MIDGARD (Vecchio et al., 2024), our photorealistic simulator.

SIMULATOR

MIDGARD is a simulation platform for autonomous robot navigation in outdoor photorealistic environments. It supports ready-made and procedurally generated scenes with adjustable complexity. It provides an HTTP endpoint compatible with ROS (Macenski et al., 2022) and Gymnasium (Towers et al., 2024) for easy integration with reinforcement and imitation learning algorithms.

Built on Unreal Engine 5, MIDGARD offers high-fidelity simulations, ensuring diverse input data for reinforcement learning and imitation learning tasks. It also supports parallel training with multiple agents, reducing

training time. Agents can operate independently or share information through a distributed policy.

The platform includes four scene types: Meadow (Fig. 2a), Forest (Fig. 2b), Volcanic (Fig. 2c), and Arctic (Fig. 2d).

Scene generation allows obstacle selection and density control, facilitating curriculum learning by gradually increasing difficulty. Customization options include rendering parameters, camera types, time of day, and weather conditions.

The simulated sensors available on MIDGARD are:

- **Vision Sensors:** On-board cameras capturing RGB, depth, and semantic segmentation data.
- **Low-Level Sensors:** Elevation maps, distances, inertial measurements, and collision detection.

METHODOLOGY

The main task of this paper is to develop an agent able to perform point-goal navigation in an unstructured environment 1, where goals are specified as a point relative to the agent, with the assumption of it always being reachable. The agent has to be able to efficiently navigate to a target, with obstacles along the way.

This reinforcement learning problem is posed as a Markov Decision Process, which is defined as a tuple:

$$M = (S, A, P, R, \gamma)$$

where:

- S is the set of states,
- A is the set of actions,
- $P(s'|s, a)$ is the transition probability function, which defines the probability of transitioning to state s' given that action a was taken in state s ,
- $R(s, a)$ is the reward function, which specifies the expected reward for taking action a in state s ,
- γ is the discount factor, where $0 \leq \gamma \leq 1$, controlling the importance of future rewards.

The state space S is the set of all possible states an agent can be in. For this task, it is defined as the distance and angle from the target, which is calculated with the Euclidean distance between the agent’s and the target’s position referred to a common world frame, given by:

$$d = \|\mathbf{p}_{\text{agent}} - \mathbf{p}_{\text{target}}\|$$

and the angular difference between the current forward direction of the robot and the robot-target joining segment. The state space is augmented with the distance and angle of the agent relative to every obstacle, since the agent needs to not collide with them. Following the privileged information paradigm as in (Chen et al., 2020), the exact obstacles’ positions are known to the agent.

The action space A represents the set of all possible actions the agent can take. Since the simulated robot



Figure 2: Pre-built different types of scenes for procedural generation.

platform follows a non-holonomic skid-steer kinematic model, the action space is defined as:

$$A = \begin{cases} 0 \leq v \leq 1, \\ -1 \leq \omega \leq 1. \end{cases}$$

Here, v denotes the linear velocity, normalized between 0 and 1, where 0 corresponds to the robot being stationary, and 1 represents its maximum forward velocity. Similarly, ω represents the angular velocity, constrained between -1 and 1, where -1 and 1 correspond to the maximum rotational speeds either clockwise or counterclockwise. Since the task is currently formulated so as not to depend on higher fidelity simulation details, such as wheel-ground interactions, or suspension simulation, as suggested in Truong et al. (2023), the agent is teleported at each timestep using the non-holonomic dynamic equations to reduce computational load without sacrificing the agent’s accuracy:

$$\Delta\theta = \omega \cdot \Delta t$$

$$\Delta x = v \cdot \cos(\theta) \cdot \Delta t$$

$$\Delta y = v \cdot \sin(\theta) \cdot \Delta t$$

Reward design is pivotal for practical policy training. A poorly chosen reward signal can limit the convergence, leaving the policy trapped in a sub-optimal region or failing altogether. Unfortunately, there is no universal heuristic for defining rewards, as the process must be tailored to each specific task. Best practices often recommend using dense reward functions, ideally augmented by a potential function to translate sparse rewards into more informative signals. For this task, the reward function is:

$$R = \begin{cases} +5, & \text{Goal Reached} \\ -2, & \text{Collision} \\ -2, & \text{Out-of-bounds} \\ 0.1 (d_t - d_{t-1}) - 0.01 & \text{Otherwise} \end{cases}$$

This reward function provides a strong incentive for the agent to reach the goal by giving a positive reward upon success. It heavily penalizes collisions and going out of bounds, both of which end the episode and yield a negative reward. At each timestep, the agent receives a shaping reward proportional to how much it reduces its distance to the goal, encouraging continuous forward progress. A fixed negative reward at each timestep promotes the agent to not take unnecessarily long trajectories.

In order to simplify learning, we suppose that the transition probability function is deterministic, meaning that given an action, the probability of transitioning from state s to state s' is equal to 1.

The policy follows an intuitive and effective rationale. The model’s input consists of the current distance and

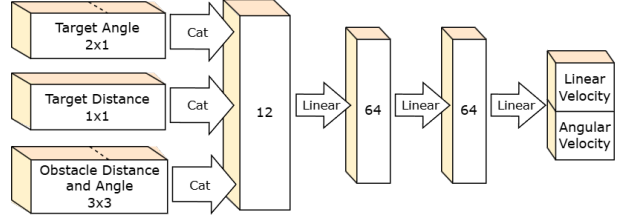


Figure 3: Network’s Architecture

Parameter	Value
Steps per Episode	2000
Obstacle Density	20%
Min Target Distance	10 m
Max Target Distance	20 m
Number of Agents	200

Table 1: Simulation Parameters

angle to the target, which is encoded with the sine and cosine values of the angle α :

$$\sin(\alpha) = \frac{\mathbf{v}_{\text{agent}} \times \mathbf{v}_{\text{seg}}}{\|\mathbf{v}_{\text{agent}}\| \|\mathbf{v}_{\text{seg}}\|}$$

$$\cos(\alpha) = \frac{\mathbf{v}_{\text{agent}} \cdot \mathbf{v}_{\text{seg}}}{\|\mathbf{v}_{\text{agent}}\| \|\mathbf{v}_{\text{seg}}\|}$$

where $\mathbf{v}_{\text{seg}} = \mathbf{p}_{\text{agent}} - \mathbf{p}_{\text{target}}$.

The input also includes the distances and angles to the three nearest obstacles. This input is processed by a fully connected network with two hidden layers, each containing 64 nodes. The network ultimately outputs the linear and angular velocities. The complete architecture is illustrated in Fig. 3.

EXPERIMENTAL RESULTS

The implementations of the algorithms considered for the evaluation are those available within the stable-baselines3 library (Raffin et al., 2021). Specifically, the algorithms are tested:

- On-Policy:
 - PPO (Schulman et al., 2017)
 - A2C (Mnih et al., 2016)
- Off-policy:
 - SAC (Haarnoja et al., 2018)
 - TD3 (Cimurs et al., 2021)

The configuration for each simulation run is detailed in Table 1. Obstacle density has been chosen so that the agent has to avoid some obstacle in its path, but not be so easy that it only relies on the target’s distance and angle information. Higher obstacle density does not let the agent learn, because through initial exploration it is not able to reach the target, and not correlate that reaching the goal means higher reward.

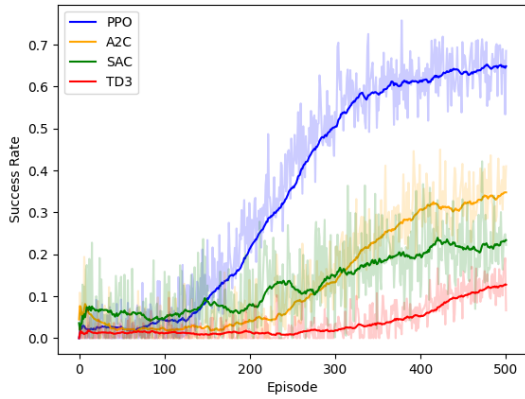


Figure 4: Success Rate per Episode

In single-agent reinforcement learning environments, whenever an agent terminates an episode (due to reaching the target or collision) the whole environment is reset, and it is good practice to change the position of every obstacle, preventing the policy from overfitting to a specific obstacle arrangement. However, in multi-agent simulation this is not feasible, as an agent can terminate whilst others are still in the middle of a run. To prevent overfitting, every N steps the environment is totally reset, even if not every agent has terminated. In the results, we report the values at the end of every environment episode.

We report the success rate, defined as the proportion of agents that reached the target out of the total agents that completed the episode, along with the average sum of undiscounted rewards per episode. The success rate curves (Fig. 4) show that PPO is the most suited, steadily increasing over time and stabilizing at around 65%. This suggests that PPO is well-suited for this kind of problem, likely due to its ability to balance exploration and stability during policy updates. A2C also performs reasonably well, reaching a success rate of about 30%, but its learning appears slower and less stable compared to PPO. Meanwhile, SAC and TD3 struggle to improve significantly, plateauing at much lower success rates of around 20% and 12%, respectively. The reward curves (Fig. 5) closely follow the trends seen in the success rate, which makes sense given that the reward function is directly tied to how well the agent reaches the goal. Initially, all algorithms experience low, often negative rewards due to inefficient or random movement, leading to frequent penalties for collisions or taking unnecessarily long paths. Over time, as they learn more effective strategies, their rewards improve. One reason PPO excels in this setting is likely due to its on-policy nature. Unlike off-policy methods like SAC and TD3, which rely on replay buffers and can struggle with non-stationary data distributions, PPO updates its policy directly using recent experiences, ensuring a more stable learning process. A2C, also an on-policy method, performs decently but lacks the robustness of PPO’s clipped objective function, which prevents overly aggressive updates that can destabilize learning. SAC and

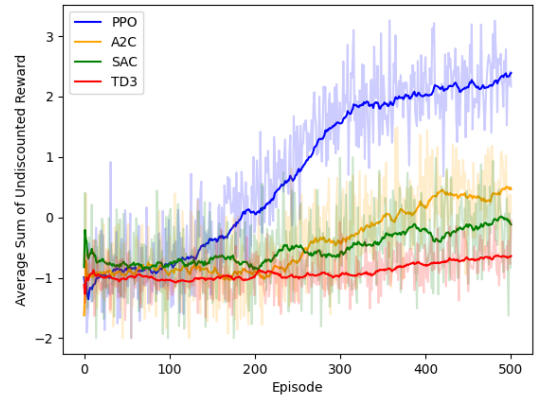


Figure 5: Average Sum of Undiscounted Reward per Episode

TD3, despite being strong algorithms in other continuous control tasks, may require more careful hyperparameter tuning or additional exploration strategies to be effective here. Looking at the early stages of training, all algorithms start with near-zero success rates and negative rewards, which is expected. At first, they act almost randomly, accumulating penalties as they collide with obstacles or take inefficient paths. As training progresses, PPO and A2C manage to escape this phase, whereas SAC and TD3 remain stuck in suboptimal behaviors for much longer. This suggests that these algorithms might be struggling with sample efficiency or are not exploring the environment effectively.

CONCLUSIONS AND FURTHER DEVELOPMENTS

In this paper, we have tested different Reinforcement Learning algorithms in MIDGARD, a simulator for navigation in unstructured environments.

Among the tested algorithms, PPO performs the best, whilst still not reaching an optimal policy, which is inherently difficult for Reinforcement Learning algorithms. Probably further parameter tuning or reward shaping could improve its performance.

A2C reaches discrete results, but it is slower and less accurate than PPO.

To improve SAC and TD3’s performance, further hyperparameter tuning, reward function modifications, or changes in exploration strategies could be explored. Alternatively, testing these algorithms in different environments might help clarify whether their struggles are inherent to the task or specific to the current setup.

These insights have provided a good baseline from which a more complex policy could be developed, leveraging sensors such as cameras and LiDAR.

Table 2: Hyperparameters for PPO

Hyperparameter	Value
Optimizer	Non-Stationary Adam
Learning Rate	3×10^{-4}
Clip Range	0.2
Discount Factor (γ)	0.99
GAE Parameter (λ)	0.95
Batch Size	256
Mini-batch Size	64
Entropy Coefficient	0.01
Value Function Coefficient	0.5
Max Gradient Norm	0.5
Steps per Update	2048

Table 3: Hyperparameters for A2C

Hyperparameter	Value
Optimizer	Adam
Learning Rate	7×10^{-4}
Discount Factor (γ)	0.99
GAE Parameter (λ)	1.0
Entropy Coefficient	0.01
Value Function Coefficient	0.5
Max Gradient Norm	0.5

Table 4: Hyperparameters for SAC

Hyperparameter	Value
Optimizer	Adam
Learning Rate	3×10^{-4}
Discount Factor (γ)	0.99
Replay Buffer Size	10^6
Batch Size	128
Entropy Coefficient (α)	Automatic tuning
Target Update Interval	1
Soft Update Coefficient (τ)	0.005

Table 5: Hyperparameters for TD3

Hyperparameter	Value
Optimizer	Adam
Learning Rate	3×10^{-4}
Discount Factor (γ)	0.99
Replay Buffer Size	10^6
Batch Size	128
Target Update Interval	2
Soft Update Coefficient (τ)	0.005
Policy Noise	0.2
Noise Clipping	0.5
Exploration Noise	0.1

APPENDICES

1 ACKNOWLEDGMENTS

The work of S. Palazzo has been supported by "PIAno di inCENTivi per la Ricerca di Ateneo 2024/2026 - Linea di Intervento 1 - Progetti di ricerca collaborativa", University of Catania. The work of F. Cancelliere, G. Sutera, D.C. Guastella, G. Muscato, and C. Spampinato has been supported by PNRR MUR project PE0000013-FAIR.

References

- Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Kr henb hl. Learning by cheating. In *Conference on robot learning*, pages 66–75. PMLR, 2020.
- R. Cimurs, I. H. Suh, and J. H. Lee. Goal-driven autonomous exploration through deep reinforcement learning, 2021. URL <https://arxiv.org/abs/2103.07119>.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL <https://arxiv.org/abs/1801.01290>.
- Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL <https://www.science.org/doi/abs/10.1126/scirobotics.abc5986>.

<https://www.science.org/doi/abs/10.1126/scirobotics.abc5986>.

- Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66), 2022. doi: 10.1126/scirobotics.abm6074. URL <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.
- Volodymyr Mnih, Adri  Puigdom nech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. URL <https://arxiv.org/abs/1602.01783>.
- A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL <http://jmlr.org/papers/v22/20-1364.html>.
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Mart n-Mart n, and Peter Stone. Deep reinforcement learning for robotics: A sur-

vey of real-world successes, 2024. URL <https://arxiv.org/abs/2408.03539>.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL <https://arxiv.org/abs/2407.17032>.

Joanne Truong, Max Rudolph, Naoki Harrison Yokoyama, Sonia Chernova, Dhruv Batra, and Akshara Rai. Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 859–870. PMLR, 14–18 Dec 2023. URL <https://proceedings.mlr.press/v205/truong23a.html>.

Giuseppe Vecchio, Riccardo E. Sarpietro, Francesco Cancelliere, Simone Palazzo, Dario C. Guastella, Alessandro Strano, Ignacio Carlucho, Giovanni Muscato, Stefano V. Albrecht, and Concetto Spampinato. Midgard: A robot navigation simulator for outdoor unstructured environments. In *European Robotics Forum 2024*, pages 120–125, Cham, 2024. Springer Nature Switzerland.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, 2020. URL <https://arxiv.org/abs/1911.00357>.

Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. Motion planning and control for mobile robot navigation using machine learning: a survey, 2022. URL <https://arxiv.org/abs/2011.13112>.

Jiaxu Xing, Angel Romero, Leonard Bauersfeld, and Davide Scaramuzza. Bootstrapping reinforcement learning with imitation for vision-based agile flight. *8th Conference on Robot Learning (CoRL)*, 2024.

AUTHOR BIOGRAPHIES

FRANCESCO CANCELLIERE received his M.Sc. in Automation Engineering and Control of Complex Systems in 2021 and his B.Sc. in Electronics Engineering in 2020. He is currently pursuing his Ph.D. at the University of Catania, Italy. His research focuses on the autonomous navigation of robotic platforms in unstructured environments, and he has participated in multiple research projects related to this field.

GIUSEPPE SUTERA received his Ph.D. at the University of Catania, Italy, in 2022. Since June 2023, he has been an Assistant Professor at the Department of Electrical, Electronic, and Computer Engineering at the University of Catania. He collaborates with the Robotic Systems Group and focuses on developing cooperative ground and aerial mobile robotic platforms.

DARIO GUASTELLA received his Ph.D. at the University of Catania, Italy, in 2019. Since January 2022, he has been an Assistant Professor at the Department of Electrical, Electronic, and Computer Engineering at the University of Catania. His research is conducted within the Robotic Systems Group and focuses on cooperative mobile robots (both ground and aerial vehicles), terrain traversability analysis, and Artificial Intelligence for autonomous navigation.

SIMONE PALAZZO is an Assistant Professor at the University of Catania, Italy. His research interests include machine learning, artificial intelligence, and pattern recognition, with a particular focus on computer vision, medical imaging, explainable AI, and bio-inspired methods for machine learning. In addition to participating in large-scale research projects, he is the principal investigator for the University of Catania in the Italian RESILIENT project (PRIN 2022 PNRR call) and in the EU project ECS4DRES.

CONCETTO SPAMPINATO earned his Laurea degree and completed his Ph.D. in Computer Engineering at the University of Catania in 2004 and 2008, respectively. He is currently an Associate Professor at the same institution. He spent a research term at the University of Edinburgh during 2008–2009, focusing on object detection and recognition. Since October 2016, he has been a Courtesy Faculty member at the Center for Research in Computer Vision at the University of Central Florida. His research focuses on learning-based computer vision and pattern recognition, particularly using deep learning techniques. In 2014, he founded the Pattern Recognition and Computer Vision Laboratory at the University of Catania. He has authored over 200 publications and led various EU, national, and regional projects on foundational AI and its applications.

GIOVANNI MUSCATO received the Electrical Engineering degree from the University of Catania, Italy, in 1988. He then joined the Centro di Studi sui Sistemi in Turin, Italy. In 1990, he joined the DIEEI at the University of Catania, where he is currently a Full-Time Professor of robotics and automatic control

and, since 2018, the Director of the Department. His research focuses on service robotics and the cooperation between ground and flying robots. He coordinated the EC project Robovolc and is the local coordinator of several national and European projects in robotics. He has authored over 300 papers in scientific journals and conference proceedings, as well as three books on control and robotics. He serves on the Board of Trustees of the Climbing and Walking Robots (CLAWAR) Association and is a Senior Member of the IEEE. His web page can be found at: <http://www.muscato.eu>.