

# The Cambridge RoboMaster: An Agile Multi-Robot Research Platform

Jan Blumenkamp, Ajay Shankar,  
Matteo Bettini, Joshua Bird, and Amanda Prorok

Department of Computer Science and Technology  
University of Cambridge  
United Kingdom  
{jb2270, as3233, mb2389, jymb2, asp45}@cst.cam.ac.uk

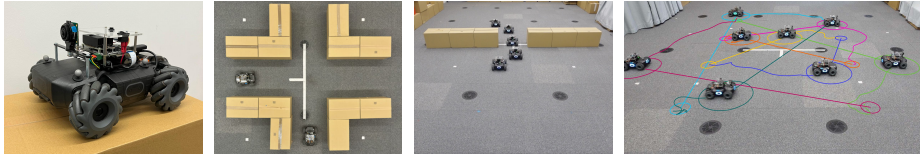
**Abstract.** Compact robotic platforms with powerful compute and actuation capabilities are key enablers for practical, real-world deployments of multi-agent research. This article introduces a tightly integrated hardware, control, and simulation software stack on a fleet of holonomic ground robot platforms designed with this motivation. Our robots, a fleet of customised DJI Robomaster S1 vehicles, offer a balance between small robots that do not possess sufficient compute or actuation capabilities and larger robots that are unsuitable for indoor multi-robot tests. They run a modular ROS2-based optimal estimation and control stack for full onboard autonomy, contain ad-hoc peer-to-peer communication infrastructure, and can zero-shot run multi-agent reinforcement learning (MARL) policies trained in our vectorized multi-agent simulation framework. We present an in-depth review of other platforms currently available, showcase new experimental validation of our system’s capabilities, and introduce case studies that highlight the versatility and reliability of our system as a testbed for a wide range of research demonstrations. Our system as well as supplementary material is available online.<sup>1</sup>

## 1 Introduction

Multi-agent robotics research necessitates robotic platforms that can be used as deployment testbeds for rapid development and evaluations [36, 18, 54, 6, 45, 8, 38, 42, 56, 46, 52, 31, 58, 7, 15, 12, 14, 30, 57]. The choices for such platforms are usually informed by several factors such as the configuration space of the problem, the number of agents involved, and the computational sophistication expected from each agent. For instance, small-scale robots, such as the Turtlebot [53], will frequently be limited in their agility of motion and are restricted in their onboard computing capabilities. On the other hand, while larger robots (Jackal [20] or AutoRally [55]) have greater manoeuvrability and can carry more compute payloads, operating multiple of them in smaller research spaces can be cost- and space- prohibitive. Successfully testing and developing multi-robot solutions requires platforms that strike a fine balance between these objectives.

---

<sup>1</sup> <https://proroklab.github.io/cambridge-robomaster>

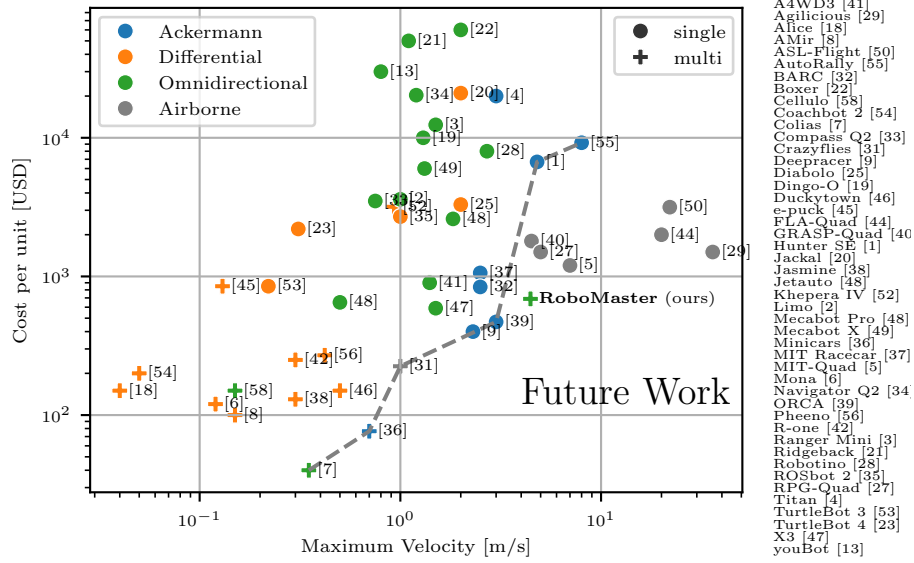


**Fig. 1.** Shown from left to right: 1) A close-up of one RoboMaster equipped with an NVidia Jetson Orin NX and a forward-facing camera 2) Two RoboMasters in a SLAM-based collision avoidance scenario 3) Five RoboMasters moving through a narrow constriction after breaking formation [15] 4) Eight RoboMasters in a multi-robot navigation scenario trained in VMAS [10].

In this article, we present an agile and affordable system and platform, shown in Fig. 1 as a multi-robot team setup. Our system, an omnidirectional indoor ground robot based on the DJI Robomaster S1, is *affordable* (starting at USD 689 in the base configuration to USD 1633 with all sensors and an Jetson Orin NX), *agile* (we demonstrate indoor trajectory tracking capabilities of up to 4.45 m/s and accelerations of up to 5 m/s<sup>2</sup>), *versatile* (we show experiments from classical control to zero-shot deployment of MARL policies trained in our simulator VMAS [10]), and is easily *replicable* (our entire stack is open-sourced, requiring minimal construction time). These have been made possible due to our retrofitting of the onboard compute module and careful reverse engineering of the main communication interface. We present two further contributions in this work. First, through an expansive review of related systems and literature, we contrast our solution against others’ and position our system at the frontier of low-cost agile research platforms. Second, we describe enhancements to our prior work (VMAS [10]) that enable rapid deployment of control policies learnt in simulation onto this platform. We present evaluations and case studies that corroborate the claims presented above, and showcase the wide range of research avenues that this system makes accessible.

## 2 Related Work

In the last decade, a wide range of (multi-) robot platforms have been introduced to the research community. Here we present an in-depth literature review of commercial as well as research platforms. Our classification scheme will differentiate between them based on their dynamics (Ackermann, differential, and omnidirectional drive, as well as airborne/multirotor platforms), and also based on their specific use cases in single- or multi-agent research. Since we are primarily interested in multi-robot research, we investigate the tradeoff between cost and speed of these platforms, as agility is an attribute we require in the team (note that we use ‘speed’ and ‘agility’ interchangeably – this is a justified simplification for indoor robots operating in confined spaces), and cost limits the size of the team.



**Fig. 2.** We compare a variety of different research and commercial robotic platforms concerning their tradeoff between cost per unit and maximum velocity. We differentiate between Ackermann, Differential, Airborne and Omnidirectional platforms as well as between platforms specifically targetting multi-agent research. A clear correlation can be seen between speed and cost, as well as a trend for multi-agent platforms to be low-cost, but as a consequence slow. We draw the Pareto front as a dashed line, with our proposed platform pushing the limits on the cost and speed tradeoff.

Fig. 2 visualises our survey in a 2D plot, and Tab. 1 provides detailed statistics specifically for omnidirectional drive robots. We find that the majority of multi-agent research platforms utilize differential drive [45, 38, 6, 46, 56, 42, 52, 8, 18], thus positioning them in the bottom left corner of Fig. 2. The reason is that such platforms tend to be designed to minimize cost, with tens of such agents in mind. The slowest of these platforms is Alice [18] at 4 cm/s, and the fastest is the Khepera IV [52] with 0.93 m/s. Three ground-based platforms are specifically geared towards multi-agent research but use dynamics other than differential drive, specifically the Minicar [36], Cellulo [58], and Colias [7]. More recently, slightly faster commercial differential drive robots have become popular [20, 23, 53, 25, 35], ranging from 0.22 m/s for the Turtlebot 3 [53] to 2 m/s for the Diabolo [25].

From the bottom left, we can see two clusters emerging towards the top and the right. The omnidirectional drive robots [58, 7, 48, 33, 13, 21, 34, 19, 49, 47, 3, 48, 22, 28, 41, 2] tend to move faster than differential drive, ranging from 0.15 m/s (Cellulo [58]) to 2.7 m/s (Festo Robotino [28]), but are also significantly more expensive, with a mean cost of USD 13k.

Aerial platforms (such as multirotors) on the other hand are moderately priced at a mean of USD 1600, and can move at much higher speeds of up to

**Table 1.** We list critical specifications for omnidirectional robots from related work. ‘Multi-Agent’ indicates whether the platform is introduced as a multi-agent platform, and ‘Commercial’ indicates whether the platform can be obtained commercially.

| Brand/Uni   | Product                  | Vel<br>[m/s] | Cost<br>[USD] | Mass<br>[kg] | L x B x H<br>[mm] | Multi<br>Agent | Commercial |
|-------------|--------------------------|--------------|---------------|--------------|-------------------|----------------|------------|
| EPFL        | Cellulo [58]             | 0.15         | 150.0         | 0.18         | 73 × 80 × 80      | ✓              |            |
| Lincoln Uni | Colias [7]               | 0.35         | 40.0          | 0.01         | 40 × 40 × 40      | ✓              |            |
| Hiwonder    | Jetauto [48]             | 0.50         | 650.0         | 4.00         | 324 × 659 × 260   |                | ✓          |
| Hangfa      | Compass Q2 [33]          | 0.75         | 3,500.0       | 16.00        | 430 × 330 × 115   |                | ✓          |
| KUKA        | youBot [13]              | 0.80         | 30,000.0      | 20.00        | 580 × 380 × 140   |                | ✓          |
| Agilex      | Limo [2]                 | 1.00         | 3,600.0       | 4.80         | 251 × 220 × 322   |                | ✓          |
| Clearpath   | Ridgeback [21]           | 1.10         | 50,000.0      | 135.00       | 790 × 960 × 310   |                | ✓          |
| Hangfa      | Navigator Q2 [34]        | 1.20         | 20,300.0      | 35.00        | 520 × 480 × 240   |                | ✓          |
| Clearpath   | Dingo-O [19]             | 1.30         | 10,000.0      | 13.00        | 680 × 510 × 111   |                | ✓          |
| RoboWorks   | Mecabot X [49]           | 1.32         | 6,000.0       | 20.50        | 630 × 581 × 203   |                | ✓          |
| Lynxmotion  | A4WD3 [41]               | 1.40         | 900.0         | 5.55         | 372 × 374 × 152   |                | ✓          |
| RDK         | X3 [47]                  | 1.50         | 590.0         | 1.93         | 181 × 236 × 185   |                | ✓          |
| Agilex      | Ranger Mini [3]          | 1.50         | 12,400.0      | 64.50        | 500 × 738 × 338   |                | ✓          |
| RoboWorks   | Mecabot Pro [48]         | 1.83         | 2,600.0       | 10.80        | 541 × 581 × 225   |                | ✓          |
| Clearpath   | Boxer [22]               | 2.00         | 60,000.0      | 127.00       | 550 × 750 × 340   |                | ✓          |
| Festo       | Robotino [28]            | 2.70         | 8,000.0       | 22.80        | 450 × 450 × 325   |                | ✓          |
| Cambridge   | <b>RoboMaster</b> (ours) | 4.45         | 689.0         | 3.00         | 240 × 320 × 200   | ✓              | ✓          |

36 m/s (Agilicious [29]), but come with other disadvantages compared to ground robots, such as operational overheads like short runtime (minutes as opposed to hours) or more challenging infrastructure needs (recovery and safety mechanisms and the availability of accurate indoor tracking). These issues become particularly noticeable and more pronounced when developing multi-agent systems – the feasibility of an experiment depends on all robots concurrently operating without faults and risks. In a multi-robot system, individual robots requiring additional operator attention and overhead can thus pose a significant barrier to rapid deployment and testing.

To the right of the plot, we observe a cluster of Ackermann drive robots [36, 9, 37, 32, 4, 39, 1, 55]. These platforms move from top speeds ranging from 0.7 m/s for the Minicar [36] (USD 76.5) to 8 m/s for the AutoRally [55] (ca. USD 10k), with an average of USD 4850. While some of these platforms are agile, they tend to be expensive and not suitable for constrained indoor spaces.

Our solution sits at a frontier beyond which agility (speed) is attained only by aerial platforms, and a lower cost is only attained by sacrificing agility. Furthermore, as we will describe in later sections, our software stack supports a wide range of relevant research problems, and has a proven track record throughout six previous publications over the last three years [15, 12, 14, 30, 57, 16].

### 3 Technical Details

This section details the background and design decisions made when developing our robot platform. We also cover all the supporting network, control and simulation infrastructure that complement it, and thus enable rapid experimentation.

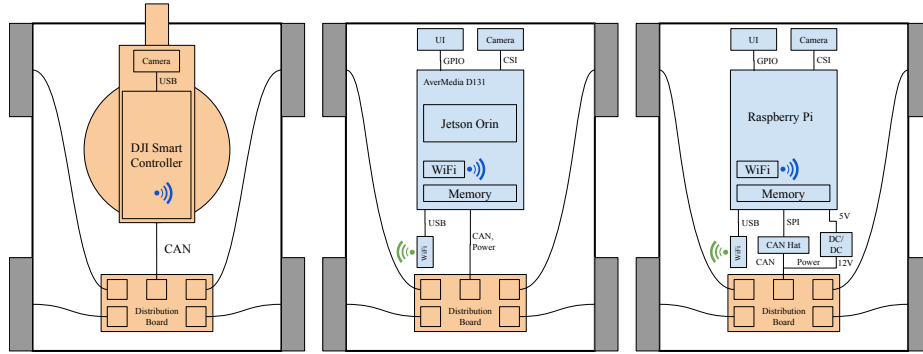
### 3.1 Platform

We selected the DJI RoboMaster S1 as our platform’s base, leveraging its design inspired by the RoboMaster competition—a DJI initiative to foster technological exchange and innovation among regional universities [24]. The S1, an educational robot, integrates a tank-like structure with a spring-dampened beam axle front suspension and a four-wheel omnidirectional drive system, and is designed for indoor use. It features a gimbal-mounted blaster toy gun and an onboard computer. DJI specifies the base platform’s maximum speeds as 3.5 m/s forward, 2.5 m/s backward, 2.8 m/s laterally, and 600 deg/s rotationally, with a weight of 3.3 kg (note that as part of our contribution, we side-step these manufacturer-imposed restrictions). Each wheel has a diameter of 100 mm, is powered by a brushless motor with integrated gears and a driver and supports closed-loop speed control up to 1000 RPM and a maximum torque of 0.25 Nm. This gives the platform a theoretical maximum speed of  $(\pi \cdot 0.1 \text{ m} \cdot 1000 \text{ RPM}) / (60 \text{ s/M}) = 5.23 \text{ m/s}$ . The system is powered by a 25.91 Wh smart battery, offering 2.4 Ah capacity. [26].

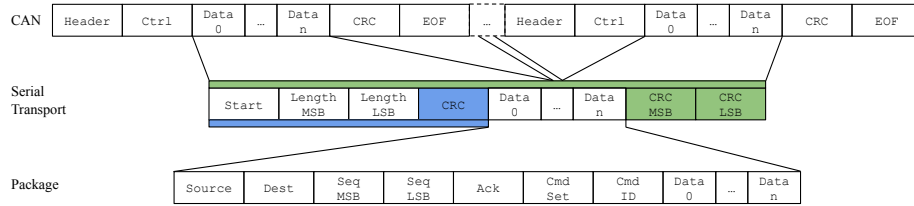
The onboard computer interfaces with a sub-controller, managing all sensors and wheels. We replaced the onboard computer, gimbal, and blaster gun with a custom computing solution. We laser-cut components from acrylic to mount our custom expansion in place of the turret. We provide a detailed overview of the compute configurations in Fig. 3. All our design files are also part of our open-source release.<sup>2</sup>

### 3.2 Compute

The stock onboard compute module on the RoboMasters is impractical for running custom algorithms or implementing additional interfaces to cameras, sensors, networks, etc. We therefore upgraded the onboard computer to a more powerful and versatile single board computer (SBC), focusing on the NVidia Jetson Orin family for its recency and lifecycle availability until January 2030<sup>3</sup>. While the Jetson Orin NX 16GB was selected for its robust processing capabilities and suitability for our research, our design easily admits other cost-effective single-board computers such as the Raspberry Pi. The Jetson Orin series necessitates a carrier board. Our choice, the AverMedia D131 carrier board, was influenced by the availability of a CAN interface – essential for communicating with the RoboMaster sub-controller – and an optimal balance of interface availability and cost. It accommodates two PCIe M.2 slots, where we installed a 256 GB SSD and an Intel 9260 WiFi module. In an alternative Raspberry Pi setup, we used the built-in ports and interfaces and added an additional CAN shield for communication with the RoboMaster base. We provide a detailed bill of material in Tab. 2.



**Fig. 3.** We modify the RoboMaster S1 Platform by removing the USB Camera, Smart Controller, Blaster and Turret and by replacing them with either a Jetson Orin on an AverMedia D131 carrier board or a Raspberry Pi. The platform is equipped with two WiFi modules, one for infrastructure communication (dark blue) and one specifically chosen for ad-hoc peer-to-peer communication (green).



**Fig. 4.** The CAN-based protocol used in the RoboMasters. DJI uses CAN as transport layer to allow multiple connected devices on the bus to broadcast information. Due to the limited length of CAN frames, a serial transport protocol is built on top, with a higher-level package definition permitting publisher/subscriber-type communication architectures.

### 3.3 CAN Communication

DJI offers an API for the RoboMaster EP, the educational counterpart to the RoboMaster S1, which enables control via a secondary computer (e.g., a Raspberry Pi) through USB. However, this platform is not listed for sale on the DJI store, is unavailable in many parts of the world (including the UK), and at around USD 1500, is significantly costlier than the S1 model. The API, which DJI has made open-source<sup>4</sup>, served as the foundation for our reverse engineering of the communication protocol between the main computer and the sub-controller. DJI employs a Controller Area Network (CAN) for hardware layer transport but overlays it with a simplified serial package-based protocol. This design likely

<sup>2</sup> <https://github.com/proroklab/cambridge-robomaster>

<sup>3</sup> <https://developer.nvidia.com/embedded/lifecycle>, accessed 24/02/22

<sup>4</sup> <https://github.com/dji-sdk/RoboMaster-SDK>, accessed 03/19/24

**Table 2.** We provide a cost breakdown for six different compute and sensing configurations for our platform at the time of paper writing. Optional equipment, such as sensors, are put in parenthesis. All costs are stated in USD.

| Item                              | Raspberry Pi | Orin Nano | Orin NX |
|-----------------------------------|--------------|-----------|---------|
| Robot Base: RoboMaster S1         | 549          | 549       | 549     |
| Compute                           | 55           | 255       | 670     |
| Carrier: AverMedia D131           |              | 185       | 185     |
| WiFi: Intel 9260                  |              | 15        | 15      |
| Mem: Micron MTFDKBA256TFK         |              | 65        | 65      |
| Mem: SD Card 256 GB               | 30           |           |         |
| Infrastructure: DC/DC Converter   | 20           |           |         |
| Infrastructure: Waveshare CAN Hat | 15           |           |         |
| Infrastructure: Acrylics          | 10           | 10        | 10      |
| Infrastructure: Mounting          | 10           | 10        | 10      |
| Infrastructure: OLED Display      | (10)         | (10)      | (10)    |
| WiFi: Netgear A6210               | (50)         | (50)      | (50)    |
| Sensing: Pi HQ Camera             | (50)         | (50)      | (50)    |
| Sensing: Fisheye Lens             | (20)         | (20)      | (20)    |
| Total Base                        | <b>689</b>   | 1089      | 1504    |
| Total Optional                    | (819)        | (1219)    | (1633)  |

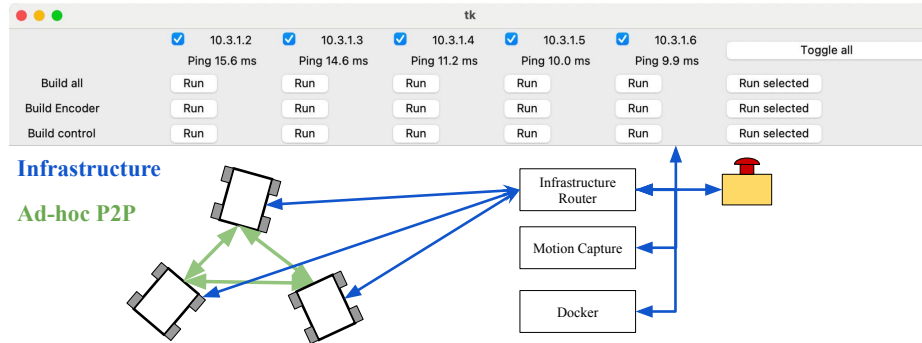
addresses the limitations of CAN’s 8-byte data size per packet while facilitating multi-node network connectivity, leveraging CAN’s collision and arbitration management capabilities. We outline the protocol’s structure in Fig. 4, noting its use of a higher-level transport layer to specify node source and destination IDs, sequence counters for package tracking, various command sets and IDs, and acknowledgements, enabling subscription to specific package IDs for continuous data stream over the network. We release our C++ implementation of the messaging protocol as a stand-alone repository.<sup>5</sup>

### 3.4 Infrastructure

Here we describe components of our supporting infrastructure, each of which is a product of over 4 years of refining and development. The high reliability of these subsystems has been vital to our work in deploying multi-agent research. We provide a high-level overview of the infrastructure in Fig. 5

**Wireless Communication.** Our system employs dual network interfaces to facilitate distinct communication channels: a backbone *infrastructure* network for all participating systems, and, an *ad-hoc* network exclusively for inter-robot communication. For infrastructure-related communication, we utilize an Intel 9260 PCIe module for the Jetson and the integrated WiFi module for the Raspberry Pi that connects to a central WiFi router (described below). This setup integrates with broader infrastructure components, such as desktop systems operating the Multi-Robot Manager UI, a motion capture system, and essential hardware like an emergency stop button. This network also allows for SSH access into the robots, file sharing, and centralized ROS messaging. In scenarios

<sup>5</sup> [https://github.com/proroklab/robomaster\\_sdk\\_can](https://github.com/proroklab/robomaster_sdk_can)



**Fig. 5.** We utilize two wireless networks, one for infrastructure (blue) and one for ad-hoc peer-to-peer communication between agents. The infrastructure network is used to communicate with fixed infrastructure. The multi-agent user interface to control multiple agents efficiently is displayed in the top. The ad-hoc network is used for experiments requiring decentralized communication.

requiring only centralized control, a desktop machine executes our entire control stack (*Freyja*, described later) for each robot, and transmits low-level wheel commands to each robot via ROS2.

For direct robot-to-robot communication, we use Netgear A6210 USB3.0 modules to establish an adhoc multicast network. This configuration does not rely on external infrastructure, enabling seamless peer-to-peer interactions.

**Network.** The deployment of an enterprise-grade router proved crucial for centralized robot operation. We opted for a Mikrotik Router, selected for its superior configurability and flexible antenna setup. We employ a structured IP assignment strategy and bridge appropriate subnetworks that contain different categories of devices (robots, computers, guest devices, etc).

Our router and ad-hoc setups typically operate at a broadcast channel capacity of 26 Mb/s, and we typically observe around 1 Mb/s network load when operating 5 robots concurrently in a centralized mode (transmitting wheel velocity commands to each robot at 50 Hz). For details on our Ad-hoc peer-to-peer network, we refer the reader to [15] and [16].

**Emergency Stop.** Crash prevention is a priority in high-velocity multi-robot experiments, and can be particularly complex to implement in a fully decentralized framework. Inevitable system latencies in re-establishing SSH sessions and the finite interval between terminating a control process and robot immobilization necessitate a more reliable stop mechanism. Consequently, we integrate an emergency stop service into our RoboMaster ROS driver, accessible via the infrastructure network, that immediately halts velocity commands at the lowest level. Additionally, we developed a stand-alone physical emergency stop button, powered by a Raspberry Pi Zero, that connects to the infrastructure network and features color LEDs to display the connectivity of the robots.



**On-Robot User Interface.** Incorporating on-robot displays has proven advantageous, especially for operations outside the conventional laboratory setting where an infrastructure network may be absent. These displays, compact and economical at  $128 \times 64$  pixels, present vital statistics including CPU and memory usage, CPU temperature, network details (such as IP addresses and interface connectivity), battery status, and the robot’s hostname. An integrated button activates the emergency stop service if the robot is in motion. This feature is particularly useful in field deployments, offering a direct control and monitoring mechanism when network access is unavailable.

**ROS2 Drivers** We implemented a custom ROS2 Driver that accepts both high-level velocity commands in the robot’s body frame as well as low-level wheel speed commands. We use `systemd` to start these drivers automatically during the system boot to make the experimental setup seamless.

**Decentralized Deployment** For deploying code across multiple robots, we utilize the infrastructure network, employing `rsync` over SSH to synchronize updates from a central workstation to each robot. We extensively leverage Docker containers to remove experimental cross-interference among different users and setups. This approach is particularly beneficial on the NVidia Jetson platform, for which an extensive library of pre-configured containers is available. These containers include environments for ROS, ROS2, PyTorch, and various other learning frameworks, either standalone or in combination<sup>6</sup>. Docker is also configured to manage local image repositories within the infrastructure network, facilitating efficient image sharing among robots.

**Multi-Robot User Interface** We developed a configurable Python-based User Interface (UI) operated from a workstation linked to the infrastructure network. This UI facilitates connections to multiple nodes through SSH and supports scripting through a configuration file. It is designed to automate the parallel execution of repetitive commands that normally need to be executed in different terminals on different devices, thus significantly reducing the manual overhead associated with logging into individual robots, sourcing workspaces, and running launch files, especially when identical commands are required on multiple robots. A snapshot of the UI is shown in Fig. 5.

### 3.5 Sensors

Our platform is equipped with an array of exteroceptive and proprioceptive sensors, supported by ROS2 drivers for seamless integration.

**Proprioceptive** The DJI RoboMaster features an onboard suite of sensors including a 3D attitude sensor, wheel encoders, body velocity sensors, and battery level sensors, all accessible through the CAN interface.

**Exteroceptive** The D131 carrier board accommodates two camera CSI inputs. We utilize this capability by attaching a Raspberry Pi HQ camera, equipped with a fisheye lens, delivering a rectified image with a  $120^\circ$  field of view

<sup>6</sup> <https://github.com/dusty-nv/jetson-containers>, accessed on 20/03/2024.

(FOV) using OpenCV. Additionally, the RoboMaster is outfitted with physical bumper sensors for enhanced interaction with its environment.

### 3.6 Control: Freyja

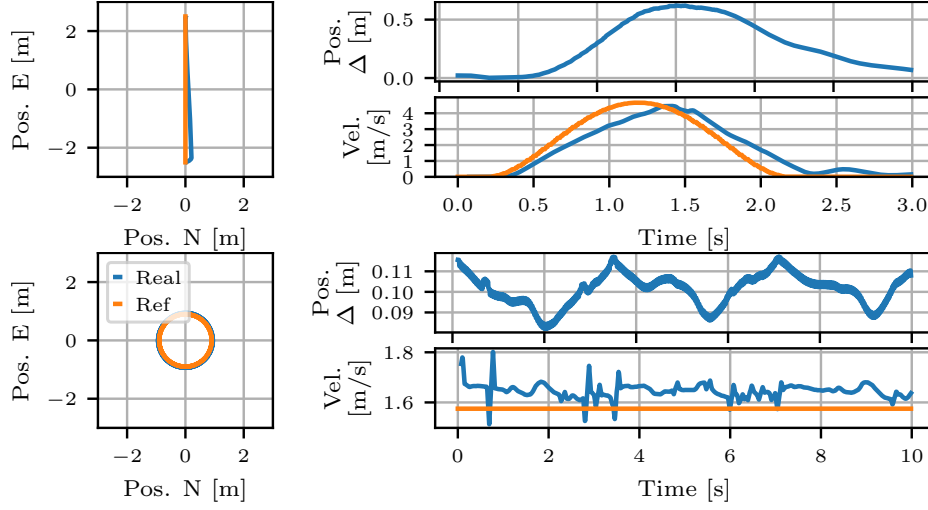
A necessary step in successful multi-robot deployments is the dependability of individuals to track their target trajectories accurately. Towards this end, our on-board control stack, *Freyja* [51], comprises of an optimal feedback controller and a library of state filters developed entirely in C++ over the ROS2 middleware. The model-based controller is a linear quadratic regulator (LQR) that regulates the position and velocity of the robot along a trajectory by commanding wheel speeds for the low-level driver. Denoting  $\mathbf{x} \equiv [\vec{p}, \vec{v}]^\top$  as the state variable (with  $\vec{p}$  and  $\vec{v}$  as the world-frame position and velocity vectors), we can write a dynamic model for the system as  $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ , where  $\mathbf{A}$  and  $\mathbf{B}$  are system matrices, and  $\mathbf{u}$  is the applied control input. Then, for some reference state  $\mathbf{x}_{\text{ref}}$ , it is possible to design a matrix  $\mathbf{K}$  for the control law,  $\mathbf{u} = -\mathbf{K}(\mathbf{x} - \mathbf{x}_{\text{ref}})$  that stabilizes the system to that (possibly time-varying) reference state.

### 3.7 Simulation: VMAS

Simulation is essential in multi-robot research as it enables safety and efficiency in data collection for machine learning pipelines. State-of-the-art simulators employ GPU vectorization to massively parallelize this process [43], leveraging the Single Instruction Multiple Data paradigm. Towards this end, we provide a model for the RoboMaster platform in the Vectorized Multi-Agent Simulator (VMAS) [10]. VMAS is a simulator consisting of a PyTorch differentiable physics engine and a set of challenging multi-robot tasks. Thanks to this, it is possible to rapidly train and deploy multi-robot policies from the simulator to the real world in a zero-shot manner, as demonstrated in previous work [12] (video here).

To enable this deployment pipeline, we integrated several new features into the simulator. Firstly, to generate the  $\vec{v}_{\text{ref}}$  input required by Freyja, we implement a PID controller layer that regulates the input forces required by VMAS based on these reference velocities. This layer allows us to train neural network policies that output desired velocities  $\vec{v}_{\text{ref}}$  and directly use them on our physical platform, without any change. Secondly, a key to the real-world deployment was an iterative fine-tuning process for some of the physics parameters in VMAS. These are parameters that depend on the robot platform and its interaction with the environment, such as friction coefficients, drag, control ranges, state boundaries, and robot geometry. This tuning process can be done once for each robot-environment pair and can be then utilized for any subsequent deployments.

A model for the RoboMaster platform in VMAS enables applications beyond prototyping in simulation and deploying in the real world. For instance, thanks to the accelerated computing available on our platform, policies trained in VMAS on approximated representations of the environment can be fine-tuned on the robot using data collected in the real world.



**Fig. 6.** We demonstrate the efficiency of our platform combined with the Freyja controller suite in two single-agent trajectory tracking experiments, in a straight line trajectory (top) and a circular trajectory (bottom). We report the reference position and velocity (orange) and measured position and velocity (blue) as well as the tracking error dp over time. The position error increases and decreases since the reference acceleration is set to a higher value than the platform’s physical limit.

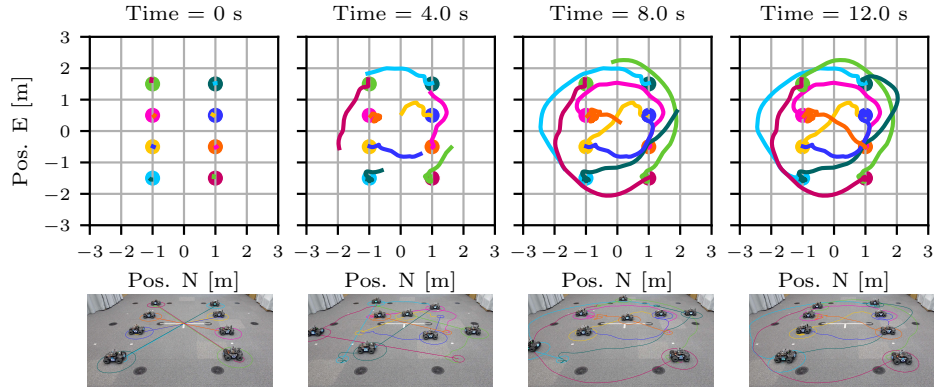
## 4 Evaluations

We now demonstrate the characteristics of our platform and the supporting suite of software framework through a wide range of experiments. These explore the versatile and diverse nature of studies that this platform enables, such as,

- running classical model-based control (Freyja) for agile trajectory tracking,
- zero-shot sim-to-real transfer of multi-agent navigation tasks learnt in VMAS,
- running distributed visual SLAM onboard to track relative position and thus avoid collisions, and,
- utilizing large neural networks based on the DinoV2-s architecture to estimate relative poses and then track trajectories.

Finally, we summarize prior work from our research group that utilizes this platform for real-world deployments of multi-robot systems research.

**Classical Trajectory Tracking.** We first demonstrate the agility of our platform in a single-robot experiment in which we benchmark the tracking performance of Freyja as well as the velocity of our robot platform at high speeds. We demonstrate this on a straight line trajectory and a circle trajectory in Fig. 6. The straight line trajectory has a length of ca. 5 m, and the circle trajectory has a diameter of 1.5 m. We show that the robot reaches a top velocity of 4.45 m/s, thus exceeding the maximum rated speed of the DJI RoboMaster platform by 1.15 m/s, and accelerations of up to 5 m/s<sup>2</sup> in the straight line trajectory with a

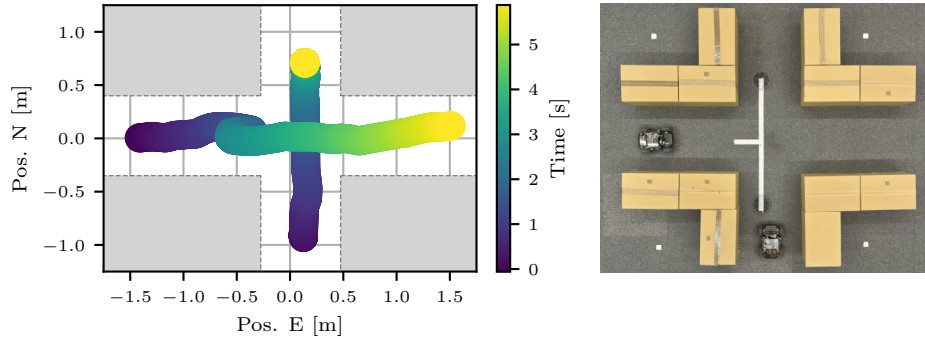


**Fig. 7.** We showcase our sim-to-real capabilities on a MARL position swapping policy trained entirely in VMAS [10]. Eight agents (color-coded) start on opposite sides of a  $4\text{ m} \times 2\text{ m}$  rectangle and attempt to swap positions with agents on the respective other side, which leads to multiple conflicts. The lines indicate their trajectories up to a particular time point, and the circles indicate their corresponding goals. We show snapshots from the real-world setup at the bottom.

maximum error of 0.5 m, and maintains a velocity of 1.7 m/s on the circle trajectory with an average error of 0.1 m. In the line trajectory, we are constrained by the size of the laboratory, which prohibits the robots from going faster. The platform is capable of at least 600 deg/s in angular velocity, however, being a holonomic system, we do not stress this any further.

**Sim-to-real Multi-Agent Navigation.** We now demonstrate zero-shot deployment of MARL policies for a multi-robot navigation task trained in VMAS [10] using the BenchMARL library [11] and TorchRL [17], following the pipeline described in Sec. 3.7. Agents are rewarded for navigating to their goal while avoiding collisions with each other, to do so they use the GPPO model ([12]), which leverages a Graph Neural Network (GNN) in the policy for inter-agent communication. The GNN utilizes a 5-layer MLP with a total of 2300 trainable parameters, which can be evaluated on the Jetson Orin NX in 0.5 ms. The results, reported in Fig. 7, show that eight agents are able to seamlessly execute the learned policy collision-free in the previously unseen real-world environment.

**Distributed Visual SLAM.** In this demonstration, we showcase the system’s ability to run a decentralized visual SLAM system locally onboard using the optionally attached Raspberry Pi HQ cameras. Our collaborative SLAM system enables the agents to share a *unified* map of the world, thus facilitating relative positioning even when their views do not overlap, and even when the other agent is not explicitly detected in view. To validate the quality of this shared map, we use a nonlinear model predictive controller to avoid collisions with both peers and static obstacles. Fig. 8 tests a collision avoidance scenario at an intersection,



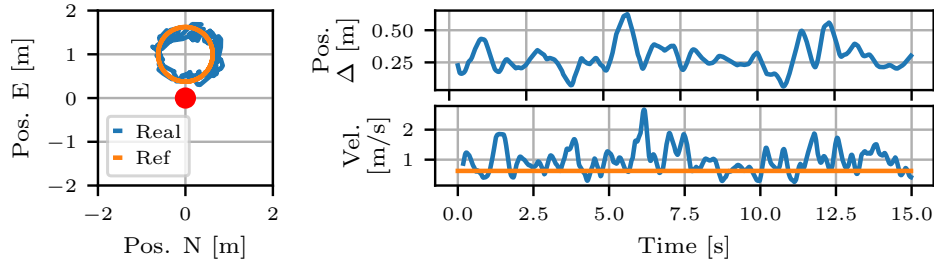
**Fig. 8.** We demonstrate multi-agent collision avoidance, facilitated by a distributed visual SLAM system running locally on the robots. Two robots are set  $90^\circ$  to each other in an intersection environment (right). The agent travelling along the E axis is given a goal pose on the other side of the intersection, and successfully avoids a collision when the agent travelling along the N axis is pushed through the intersection. The trajectories generated by the SLAM system are presented on the left chart.

where two robots (travelling along the horizontal and vertical axes) would nominally collide at the intersection. However, they are able to localize each other close to the origin when common features are available in the environment, and the robot travelling along the vertical axis is able to slow down to avoid colliding. We are including additional details of this setup in a follow-up work.

**Multi-robot control using CoViS-Net.** In this experiment, we deploy CoViS-Net [16], a pre-trained neural network to estimate relative poses between two camera frames, on two different robots. The camera images of two robots are encoded, and the features are broadcasted over the ad-hoc network to other robots within the communication range, based on which the relative pose can be predicted, which is then used to track a trajectory using Freyja. In this setup, we use two robots, one that serves as origin for the reference coordinate frame of the other robot, which is commanded to move along a shifted circle trajectory. The result, visualized in Fig. 9, shows that we are able to move with a velocity of about 1 m/s on the circle, with an average estimated error of 0.25 m. Our model uses the DinoV2-s base neural network with 21M parameters, which we are able to run in 20 ms on the Jetson Orin NX after optimising with NVidia TensorRT.

**Case Studies.** Our system has reached this level of maturity over the course of several iterations of research and development in the last four years. Apart from the capabilities demonstrated above, this framework has featured as a primary research platform in several prior multi-agent works from our group:

- *Multi-agent passage scenario*, where five RoboMasters navigate through a narrow passage, coordinated by onboard decentralized homogeneous GNN [15];
- *Real-world HetGPPO*, where heterogeneous GNNs trained in VMAS deployed on RoboMasters demonstrate superior resilience to real-world noise [12];



**Fig. 9.** The decentralized trajectory tracking tracks a reference trajectory relative to a stationary robot (red dot), based on the relative poses estimated by a CoViS-Net [16]. We run this neural network on board, in real-time. The position error and velocity are estimated from the predicted poses.

- *Visual Navigation*, where a RoboMaster navigates an environment guided entirely using visual sensors and a policy trained in simulations [14];
- *CBFs for multi-agent control*, where four RoboMasters use a control barrier function based GNN strategy to navigate [30]; and,
- *Single-agent search & navigation*, where LQR, CBF and RRT\* are combined for safe and optimal single-robot motion planning [57].
- *Deployment of multi-agent foundation models*, where up to four RoboMaster estimate the relative pose to perform decentralized formation control in real-world scenarios [16].

## 5 Conclusion

This article introduces a versatile omnidirectional ground robot system that is based on extensive modification and enhancement of the original DJI RoboMaster S1 platform. We significantly extend the capabilities of the base platform by providing multiple options for more flexible and powerful compute solutions, on-board sensors, model-based control, and sim-to-real capabilities for distributed RL policies. After an intensive research and development pipeline over four years, our full platform (hardware and software) now serves as a highly reliable testbed well-suited for a wide range of experiments. We showcase its capabilities through four new evaluations with fleet sizes of up to eight robots, and refer to six case studies from prior work that point to its proven track record.

## Acknowledgments

This work was supported by ARL DCIST CRA W911NF-17-2-0181 and European Research Council (ERC) Project 949949 (gAIa). J. Blumenkamp acknowledges the support of the ‘Studienstiftung des deutschen Volkes’ and an EPSRC tuition fee grant. We also acknowledge a gift from Arm. We further thank all other members of the Prorok Lab for their suggestions and helpful discussions that have contributed towards topics reported in this article.

## References

1. Agilex Hunter. Agilex Hunter Website, 2024.
2. Agilex Limo. Agilex Limo Website, 2024.
3. Agilex Ranger Mini. Agilex Ranger Mini Website, 2024.
4. Agilex Titan. Agilex Titan Website, 2024.
5. Amado Antonini, Winter Guerra, Varun Murali, Thomas Sayre-McCord, and Sertac Karaman. The blackbird dataset: A large-scale dataset for uav perception in aggressive flight. In *International Symposium on Experimental Robotics*, pages 130–139. Springer, 2018.
6. Farshad Arvin, Jose Espinosa, Benjamin Bird, Andrew West, Simon Watson, and Barry Lennox. Mona: an affordable open-source mobile robot for education and research. *Journal of Intelligent & Robotic Systems*, 94:761–775, 2019. Publisher: Springer.
7. Farshad Arvin, John C. Murray, Licheng Shi, Chun Zhang, and Shigang Yue. Development of an autonomous micro robot for swarm robotics. In *2014 IEEE International Conference on Mechatronics and Automation*, pages 635–640, 2014.
8. Farshad Arvin, Khairulmizam Samsudin, Abdul Rahman Ramli, and others. Development of a miniature robot for swarm robotic application. *International Journal of Computer and Electrical Engineering*, 1(4):436–442, 2009. Publisher: International Association of Computer Science and Information Technology Press.
9. Bharathan Balaji, Sunil Mallya, Sahika Genc, Saurabh Gupta, Leo Dirac, Vineet Khare, Gourav Roy, Tao Sun, Yunzhe Tao, Brian Townsend, Eddie Calleja, Sunil Muralidhara, and Dhanasekar Karuppasamy. DeepRacer: Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2746–2754, 2020.
10. Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. In *Proceedings of the 16th International Symposium on Distributed Autonomous Robotic Systems, DARS '22*. Springer, 2022.
11. Matteo Bettini, Amanda Prorok, and Vincent Moens. BenchMARL: Benchmarking Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2312.01472*, 2023.
12. Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous Multi-Robot Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23*, pages 1485–1494, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. eprint: <https://dl.acm.org/doi/pdf/10.5555/3545946.3598801> event-place: London, United Kingdom.
13. Rainer Bischoff, Ulrich Huggenberger, and Erwin Prassler. KUKA youBot - a mobile manipulator for research and education. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.
14. Jan Blumenkamp, Qingbiao Li, Binyu Wang, Zhe Liu, and Amanda Prorok. See What the Robot Can’t See: Learning Cooperative Perception for Visual Navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7333–7340. IEEE, 2023.
15. Jan Blumenkamp, Steven Morad, Jennifer Gielis, Qingbiao Li, and Amanda Prorok. A Framework for Real-World Multi-Robot Systems Running Decentralized GNN-Based Policies. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8772–8778, 2022. eprint: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9811744>.

16. Jan Blumenkamp, Steven Morad, Jennifer Gielis, and Amanda Prorok. CoViS-Net: A Cooperative Visual Spatial Foundation Model for Multi-Robot Applications, 2024. [eprint: 2405.01107](#).
17. Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. TorchRL: A data-driven decision-making library for PyTorch. In *The Twelfth International Conference on Learning Representations*, 2024.
18. Gilles Caprari and Roland Siegwart. Design and control of the mobile micro robot alice. In *Proceedings of the 2nd International Symposium on Autonomous Minirobots for Research and Edutainment, AMiRE 2003: 18-20 February 2003, Brisbane, Australia*, pages 23–32. CITI, 2003.
19. Clearpath Robotics Inc. DINGO: INDOOR ROBOTIC PLATFORM. Technical report, Clearpath Robotics Inc., 2019.
20. Clearpath Robotics Inc. JACKAL Unmanned Ground Vehicle: Technical Specifications. Technical report, Clearpath Robotics Inc., 2020.
21. Clearpath Robotics Inc. RIDGEBACK: Omnidirectional Development Platform. Technical report, Clearpath Robotics Inc., 2020.
22. Clearpath Robotics Inc. BOXER: INDOOR ROBOTIC PLATFORM. Technical report, Clearpath Robotics Inc., 2022.
23. Clearpath Robotics Inc. Turtlebot 4: Robotics Learning Platform. Technical report, Clearpath Robotics Inc., 2022.
24. RoboMaster Organizing Committee (RMOC). RoboMaster 2023 University League Rules Manual. Technical report, DJI, T2, 22F, DJI Sky City, No. 55 Xianyuan Road, Nanshan District, Shenzhen, China, November 2022.
25. DirectDriveTech. Diablo: World’s First Direct-Drive Self-Balancing Wheeled-Leg Robot, 2024.
26. DJI RoboMaster S1. DJI RoboMaster S1 Website, 2024.
27. Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017. Publisher: IEEE.
28. FESTO Robotino. FESTO Robotino, 2024.
29. Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and others. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science robotics*, 7(67):eabl6259, 2022. Publisher: American Association for the Advancement of Science.
30. Zhan Gao, Guang Yang, and Amanda Prorok. Online Control Barrier Functions for Decentralized Multi-Agent Navigation. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 107–113. IEEE, 2023.
31. Wojciech Giernacki, Mateusz Skwierczyński, Wojciech Witwicki, Paweł Wroński, and Piotr Kozierski. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In *22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 37–42. IEEE, 2017.
32. J Gonzales, F Zhang, K Li, and F Borrelli. Autonomous drifting with onboard sensors. In *Advanced Vehicle Control*, pages 133–138. CRC Press, 2016.
33. Hangfa Compass Q2. Hangfa Compass Q2, 2024.
34. Hangfa Navigator Q2. Hangfa Navigator Q2, 2024.
35. Husarion Rosbot. Husarion Rosbot Website, 2024.



36. Nicholas Hyldmar, Yijun He, and Amanda Prorok. A Fleet of Miniature Cars for Experiments in Cooperative Driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3238–3244, 2019.
37. Sertac Karaman, Ariel Anders, Michael Boulet, Jane Connor and Kenneth Gregson, Winter J. Guerra, Owen Guldner, Mubarik Mohamoud and Brian Plancher, Robert Shin, and John Vivilecchia. Project-based, Collaborative, Algorithmic Robotics for High School Students: Programming Self-driving Race Cars at MIT. In *2017 IEEE Integrated STEM Education Conference (ISEC) (ISEC'17)*, Princeton, USA, March 2017.
38. Serge Kernbach, Ronald Thenius, Olga Kernbach, and Thomas Schmickl. Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259, 2009. Publisher: SAGE Publications Sage UK: London, England.
39. Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015. Publisher: Wiley Online Library.
40. Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU. *IEEE Robotics and Automation Letters*, 2(2):404–411, 2017.
41. Lynxmotion A4WD3. Lynxmotion A4WD3, 2024.
42. James McLurkin, Adam McMullen, Nick Robbins, Golnaz Habibi, Aaron Becker, Alvin Chou, Hao Li, Meagan John, Nnena Okeke, Joshua Rykowski, and others. A robot system design for low-cost multi-robot manipulation. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 912–918. IEEE, 2014.
43. Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.
44. Kartik Mohta, Michael Watterson, Yash Mulgaonkar, Sikang Liu, Chao Qu, Anurag Makineni, Kelsey Saulnier, Ke Sun, Alex Zhu, Jeffrey Delmerico, and others. Fast, autonomous flight in GPS-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120, 2018. Publisher: Wiley Online Library.
45. Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.
46. Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, Daniel Hoehener, Shih-Yuan Liu, Michael Novitzky, Igor Franzoni Okuyama, Jason Pazis, Guy Rosman, Valerio Varricchio, Hsueh-Cheng Wang, Dmitry Yershov, Hang Zhao, Michael Benjamin, Christopher Carr, Maria Zuber, Sertac Karaman, Emilio Frazzoli, Domitilla Del Vecchio, Daniela Rus, Jonathan How, John Leonard, and Andrea Censi. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504, 2017.
47. RDK X3. RDK X3, 2024.

48. RoboWorks Mecabot Pro. RoboWorks Mecabot Pro, 2024.
49. RoboWorks Mecabot X. RoboWorks Mecabot X, 2024.
50. Inkyu Sa, Mina Kamel, Michael Burri, Michael Bloesch, Raghav Khanna, Marija Popović, Juan Nieto, and Roland Siegwart. Build your own visual-inertial drone: A cost-effective and open-source autonomous drone. *IEEE Robotics & Automation Magazine*, 25(1):89–103, 2017. Publisher: IEEE.
51. Ajay Shankar, Sebastian G. Elbaum, and Carrick Detweiler. Freyja: A Full Multirotor System for Agile & Precise Outdoor Flights. In *IEEE International Conference on Robotics and Automation*, pages 217–223. IEEE, 2021.
52. Jorge M Soares, Inaki Navarro, and Alcherio Martinoli. The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox. In *Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Volume 1*, pages 767–781. Springer, 2016.
53. Turtlebot3. TurtleBot3 Website, 2024.
54. Hanlin Wang and Michael Rubenstein. Shape Formation in Homogeneous Swarms Using Local Task Swapping. *IEEE Transactions on Robotics*, 36(3):597–612, 2020.
55. Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.
56. Sean Wilson, Ruben Gameros, Michael Sheely, Matthew Lin, Kathryn Dover, Robert Gevorkyan, Matt Haberland, Andrea Bertozzi, and Spring Berman. Pheeno, A Versatile Swarm Robotic Research and Education Platform. *IEEE Robotics and Automation Letters*, 1(2):884–891, 2016.
57. Guang Yang, Mingyu Cai, Ahmad Ahmad, Calin Belta, and Roberto Tron. Efficient LQR-CBF-RRT\*: Safe and Optimal Motion Planning. *arXiv preprint arXiv:2304.00790*, 2023.
58. Ayberk Özgür, Séverin Lemaignan, Wafa Johal, Maria Beltran, Manon Briod, Léa Pereyre, Francesco Mondada, and Pierre Dillenbourg. Cellulo: Versatile Hand-held Robots for Education. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI '17*, pages 119–127, New York, NY, USA, 2017. Association for Computing Machinery. event-place: Vienna, Austria.