Robotica (2025), **43**, pp. 2610–2643
doi:10.1017/S0263574725101884

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Arachne system: an accessible miniaturized mobile Multi-robot platform*

Ioannis Raptis[1] and Benjamin Sullivan[2]

[1]Department of Electrical and Computer Engineering, North Carolina A&T State University, Greensboro, NC, USA
[2]Department of Electrical and Computer Engineering, University of Massachusetts Lowell, Lowel, MA, USA
Corresponding author: Ioannis Raptis; Email: iraptis@ncat.edu

Received: 8 July 2024; **Revised:** 7 February 2025; **Accepted:** 23 May 2025; **First published online:** 8 July 2025

Keywords: swarm robotics; distributed robotic systems; microrobots; educational robotics

Abstract

This paper introduces the Arachne System, a scalable, cost-effective mobile microrobot swarm platform designed for educational and research applications. It details the design, functionalities, and potential of the Arachne Bots, emphasizing their accessibility to users with minimal robotics expertise. By providing a comprehensive overview of the system's hardware, sensory capabilities, and control algorithms, the paper demonstrates the platform's capacity to democratize and reduce entry barriers in mobile robotic swarms research, fostering innovation and educational opportunities in the field. Extensive experimental validation of the system showcases its broad range of capabilities and effectiveness in real-world implementation.

1. Introduction

Large cooperative robot swarms offer substantial advantages, including decentralization and cost-effectiveness, which generally surpass the challenges brought on by increased system complexity. Decentralization leads to systems that are not only robust against individual failures but also resilient and cost effective to maintain. Moreover, the per-unit cost of individual agents within a swarm is typically lower than that of singular robots designed for comparable tasks, leading to potential savings in both initial investment and long-term operational costs.

Multi-agent systems play a crucial role in mission-critical applications, enabling access to hard-to-reach areas, exploration of hazardous environments, and execution of rescue operations. In search and rescue missions, swarm robots navigate disaster zones, locate survivors, and relay critical information without endangering human responders. In environmental monitoring, they autonomously track pollution levels, map forest fires, and conduct underwater surveillance. The agriculture sector benefits from swarm robotics through precision farming, where coordinated robotic systems optimize planting, irrigation, and harvesting. In industrial automation, swarm robots streamline warehouse logistics, improving inventory management and material transportation. In space exploration, autonomous robotic swarms provide scalable solutions for planetary mapping, extraterrestrial construction, and resource extraction. These diverse applications highlight how swarm robotics enhances efficiency, scalability, and resilience in tackling real-world challenges.

However, the pace of progress in the field of swarm robotics is often hampered by financial and physical constraints. The development of an accessible, small-scale, low-cost, multi-agent robotics platform could significantly enhance research productivity and attract newcomers to the field. Such a

*An introductory overview video tutorial of the Arachne System is available at https://www.youtube.com/watch?v=2tZHDulS5Q8. The complete source code and documentation for the Arachne System are publicly available in our open-source GitHub repository at https://github.com/UMLARSL/Arachne-Bots.

© The Author(s), 2025. Published by Cambridge University Press.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

platform would serve both as an affordable introduction to swarm robotics and as a versatile testbed for sophisticated algorithms.

Our literature review focuses on a specific niche within distributed robotic systems [1] and swarm robotics [2], centering on small-scale (semi-)autonomous mobile multi-robot systems, hereafter referred to as *mobile microrobot swarms* for brevity. The defining features of these systems include their mobility, albeit potentially limited, and compact size, often just a few cubic inches. These robots, also called *agents* or *units*, form a collective, alternatively termed a swarm or group. Each unit is outfitted with a wireless microcontroller to facilitate inter-agent communication and potentially link with external swarm processing nodes, such as a central supervisory computer. The robots possess a degree of autonomy that enables the collective to undertake collaborative tasks or missions. Optionally, agents may be equipped with onboard sensors or actuators, granting them the ability to execute specialized functions, such as detecting chemical plumes or manipulating objects.

The list of systems, platforms, frameworks, and even products we review herein is not exhaustive. We merely aim to offer a historical overview of the technological benchmarks and research trajectories in the field of mobile microrobot swarms over time. The origin of frameworks for swarms of communicating miniature mobile robots dates back to the late 1980s, driven by the ambition to create hardware and software solutions that enable distributed robotic systems. This endeavor aimed to advance research and education in developing algorithms that facilitate cooperative behavior among mobile robots operating at scale. Inherent to microrobots are limitations due to the small size of the swarm units, affecting computational power, energy, mobility range, and the extent of sensing and communication abilities.

One of the pioneering efforts in creating a mobile micro-robot, sized at one cubic inch, was the development of Squirt in the late 1980s by researchers at the Massachusetts Institute of Technology's (MIT's) AI Lab [3–6]. This platform was motivated by educational goals in robotics. Similarly, the Khepera robot, initiated at the Microcomputing Laboratory (LAMI) of the Swiss Federal Institute of Technology in Lausanne (EPFL) in 1991, quickly became a favored tool for collaborative and distributed robotics research over the next three decades. The Khepera IV model [7], now discontinued, was priced at $3,200. The initial goal behind the development of both Khepera and Squirt was to advance the miniaturization of mobile robots with respect to sensors, computing, actuation, and power supply rather than focusing on the domain of robot swarms. A noteworthy mention is warranted for the marketable Lego Mindstorms platform [8, 9]. Although the Mindstorms "kit" doesn't strictly belong to the "micro" or swarm robotics category, serving instead as a standardized platform, it represents a significant commercial effort to make robotics educationally accessible and to incorporate robotics into the computing curriculum [9].

Carnegie Mellon University's Millibot comprised of a heterogeneous team of 7^3 cm^3 continuous track robots tasked to map and localize within its operational environment [10, 11]. Its design philosophy aimed for modularity on a small scale, namely, the ability to conveniently add different capabilities (mostly sensing and less actuating modules) for creating heterogeneous teams. Parallely, the University of Southern California's Robomote was specifically engineered to enhance power use and efficiency within large-scale, distributed, small-scale robotic systems and sensor networks [12, 13]. Its goal was to facilitate research into expansive mobile sensor networking by employing diminutive units, roughly 3.5^3 cm^3 in size, providing functionalities on par with those of larger, more expensive models. The term "mote" refers to low-power sensor nodes integral to distributed sensor networks, attracting substantial research interest in the early 2000s. The concept originates from the aspiring idea of "Smart Dust Motes" – millimeter-sized, low-power nodes designed for sensing and communication – put forward by researchers at the University of California, Berkeley (UCB) [14–16]; presumably, as pinpointing its exact origins proves difficult.

The attributes of a mote closely align with those of mobile microrobots, barring the elements of mobility and possibly actuation. The development of TinyOS [17], an operating system designed for the distinctive requirements of low-power sensing nodes, marks a notable progression derived from mote research at UCB. This example underscores the pivotal importance of specialized software solutions, including unique programming languages, operating systems, or simulation tools, in navigating the intricacies of swarm robotic systems [18, 19].

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

The MICAbot [20], developed in the early 2000s by the University of Notre Dame, leveraged the MICA platform [21] from UCB. MICA, a wireless embedded sensor system, served as the wireless microcontroller for these mobile robots. The MICA platform's appeal lay in its modular suite of sensors, which could be easily attached or detached from the main "mother" interface board.

Concurrently, at UCB's Sensor & Actuator Center, the CostBots project emerged [22, 23]. The developers recognized that the robotic systems available at that time were either prohibitively expensive or excessively specialized, posing challenges for replication. CostBots highlights a miniaturized mobile microrobot platform constructed from off-the-shelf components and open technology, addressing the need for affordable and accessible robotic systems.

Alice [24], a microrobot platform developed at the Autonomous Systems Lab (ASL) of EPFL, highlighted size as a critical design aspect. With a volume of just 2^3 cm^3, one of the most miniature systems signified the challenge of integrating locomotion, processing, sensing, and actuation within such compact dimensions. It also introduced the use of Zigbee [25], a communication radio module operating on TinyOS. Another notable platform is e-puck [26, 27], created by GCtronic and three EPFL labs, including the Laboratory of Intelligent Systems and ASL. With currently limited availability in Europe and Asia and a cost of $950, it features a 7 cm diameter for operating in small desks, a comprehensive sensor array, and an entirely open-source design. E-puck's "E-Puck Range and Bearing" board facilitates localization of nearby units.

Another example of pursuing miniaturization in communicating mobile robots is Jasmine, which occupies a volume of approximately 2.6^3 cm^3. This robot was developed in the early 2010s at the University of Stuttgart [28]. Similarly, Bebot, created at the University of Paderborn [29], also maintains a compact design with a volume of roughly 9^3 cm^3. A distinctive feature of Bebot is its overhead mechanical platforms, designed to enable collective load transportation through shared effort among swarm members. This capability demonstrates collaboration and swarm actuation, effectively addressing the limitations inherent in individual units.

Rice University's r-one system [30, 31] was explicitly developed for a problem-based curriculum and educational purposes. The value of this platform extends beyond its design to include resources that facilitate the development of a robotics curriculum. Concurrently, Harvard's Kilobot [32, 33] captured the robotics community's attention. Kilobot stands out with its distinctive vibrational tripod locomotion mechanism, a single sensor, and a simple yet effective infrared inter-agent communication system. Moreover, its manufacturing cost of $14 positions it as an attractive platform for fostering research in swarm robotics.

In the early 2020s, the Robotarium system from the Georgia Institute of Technology (GaTech) [34] emerged as a leading example of a remotely accessible multi-robot testbed, allowing users globally to implement and test their algorithms on a physical mobile robot swarm since 2017. The paper by Wilson et al. [35] delivers a comprehensive analysis and assessment of the Robotarium's user base, detailing demographics, types of experiments conducted, and users' anticipated outcomes. This study stands out as perhaps the sole contribution that documents and emphasizes the need for and importance of accessibility in mobile multi-robot platforms. Recently, the focus on miniature mobile robotics has pivoted toward transportation systems, with significant research directed toward autonomy and decision-making algorithms that address real-world transportation challenges. Notable ecosystems in this domain include MIT's Duckietown [36, 37], which also offered remote access, and the Scaled Smart City [38–40] at Cornell University, epitomizing the research transition towards autonomous transportation systems.

Since the early 2010s, the widespread availability and accessibility of PCB fabrication and 3D printing services, along with desktop printers, have empowered researchers, academic laboratories, and hobbyists to design and develop their own platforms. These platforms, distinguished by unique design, hardware, and software characteristics, often align with the domain of mobile microrobot swarms, either fully or partially. Notable examples include Carnegie Mellon's Finch [41], Arizona State University's Pheeno [42], Parallax's Scribbler [43], Pololu's 3pi [44], and iRobot's Create [45].

This comprehensive, albeit not exhaustive, overview of platforms encapsulates the defining notions and design traits of mobile microrobot swarms, enabling us to confidently derive certain observations,

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

insights, and conclusions. These conclusions are drawn from online resources and research literature corresponding to nearly three decades of development efforts by numerous institutions and small companies, primarily in North America and Europe.

In essence, the quintessential and indispensable attributes of a mobile microrobot swarm include compactness, affordability, mobility, and capabilities in communication and autonomy. At the system level, it is crucial for the agents (robots) to possess real-time localization within their operational environment (arena), and for the system to exhibit scalability, meaning it should seamlessly support a large number of units. Additionally, the platform must be accessible and thoroughly documented.

A significant body of research literature, which utilizes mobile multi-robot experimental testbeds to validate theoretical findings on topics pertinent to distributed control or estimation in multi-robot systems and swarm behavior, underscores the need for such platforms. Furthermore, from an educational standpoint, there is a global drive toward introducing robotics and programming to increasingly younger audiences.

Presently, only a single system (e-puck) is commercially available on a limited number of continents, while several platforms – such as Khepera and Kilobot – have been phased out from the market. This scenario suggests that these systems do not present a viable commercial opportunity, given that the demand, although present, is insufficient to sustain their production and distribution. Therefore, there is a palpable discrepancy between the availability and the demand for mobile microrobot swarm systems.

Moreover, literature suggests a preference among end-users either to purchase or to construct their own systems. The challenge of replication, particularly in the context of manufacturing, acts as a deterrent. Users generally favor purchasing affordable components over undertaking their own fabrication efforts. Despite the availability of numerous platforms offering detailed instructions and blueprints, actual replication is rare. This can be attributed to several factors, including limited access to fabrication tools (e.g., 3D printers) or the requisite expertise to operate them. Consequently, purchasing low-cost components, such as a robot chassis, emerges as a more viable and practical option. Additional barriers that render experimental testbeds for such systems inaccessible include limited laboratory space, a lack of expertise (e.g., in electronics, programming, CAD design), and issues related to affordability.

In light of the discussed need, significance, availability, barriers to accessibility, and end-user preferences, we introduce the Arachne[1] System: our proprietary open-access mobile microrobot swarm experimental testbed platform. This initiative, presented in this paper, represents the culmination of approximately ten years of research into developing computer-controlled, small-scale, affordable robotic platforms, targeting the educational engagement of high school and undergraduate students with mobile and swarm robotics. Notably, the platform was inclusively developed by first-year engineering and computer science undergraduate students, aiming to acquaint students with the then-emerging fields of networked control and cyber-physical systems. Through the development process, the platform serves as an introductory tool for students, merging concepts from communications, embedded programming, control systems, and robotics into a cohesive learning experience.

At the heart of the Arachne System are the Arachne Bots, centimeter-scale mobile robots characterized by their engaging design, minimal assembly time, low cost, and availability of parts. The system utilizes globally accessible off-the-shelf components, which can be sourced from both toy and electronics stores. In this paper, we pinpoint the essential building blocks required to construct a mobile microrobot swarm, ensuring that developers can substitute unavailable parts with equivalent alternatives. This approach allows for the replication of our design into customized systems that align with individual budgets and specifications.

Moreover, we demonstrate how straightforward electronic modifications can enhance the sensory capabilities of the Arachne Bots. Our contribution includes a suite of algorithms facilitating a range of behaviors, from basic tasks like set-point convergence and individual robot object manipulation to more complex collective actions such as collision avoidance and group object manipulation. We have

[1]The word Arachne, meaning "spider," originates from Greek and is rooted in a tale from Greek mythology.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

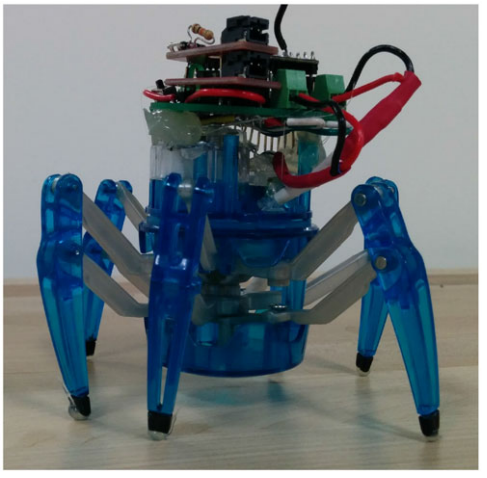


Figure 1. *Front view of the arachne bot. The main body is obtained from HEXBUG's toy robot, spider. A protoboard with a small footprint is placed on top of the robot's chassis. The protoboard mounts the wireless microcontroller. The latter generates the control signals that operate the robot's motion.*

articulated the control logic in an accessible manner, eliminating the need for an in-depth understanding of control theory or robotics.

This endeavor seeks to make research in mobile robotic swarms more inclusive, offering a platform that can be developed with little to no specialized knowledge. Our goal is to inspire the broader community in the creation of custom testbeds, original peripherals, and control algorithms for mobile microrobot swarms, thus democratizing the field for researchers, students, and enthusiasts alike.

The physical architecture and design of the Arachne Bots are explored in Section 2, where we detail its subsystems and components. The development of a miniaturized proximity sensor for the Arachne Bots is outlined in Section 3. Mathematical modeling principles governing the Arachne Bots are elucidated in Section 4, providing insight into their movement and control mechanisms. The same section is dedicated to the exposition and analysis of several control algorithms for various system tasks. An empirical assessment of the Arachne Bots' performance, demonstrating their practical capabilities, is presented in Section 5. Finally, Section 6 offers concluding observations on the platform.

2. Design and architecture

Arachne Bots, an overhead camera, and a laptop computer comprise the Arachne System. The overhead camera serves as the main motion sensor, while the supervisory controller is executed on the laptop computer. The Arachne Bots themselves consist of four basic subsystems: a chassis and transport mechanism, motion sensors, a microprocessor, and a radio component. The four subsystems constitute the essential, bare minimum, structural components required to assemble a mobile and cooperative multirobot system. Each Arachne Bot takes roughly 90 min to assemble, costs approximately $80, and has a volumetric footprint of about 12 cm^3.

2.1. Body and motors

Mechanical components of the Arachne Bots were obtained from the commercially available Spider toy robot by HEXBUG [46]. The HEXBUG's Spiders are six-legged arthropods that are powered using two DC motors. The Arachne Bots' controlled DC motors control forward motion and heading. Figure 1 shows the front view of an Arachne Bot. HEXBUG is a consumer robotics line developed by Innovation First International to complement the VEX Robotics Design System [47]. In January 2023, Spin Master Corp. acquired the HEXBUG brand, expanding its portfolio of creative robotic technology.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

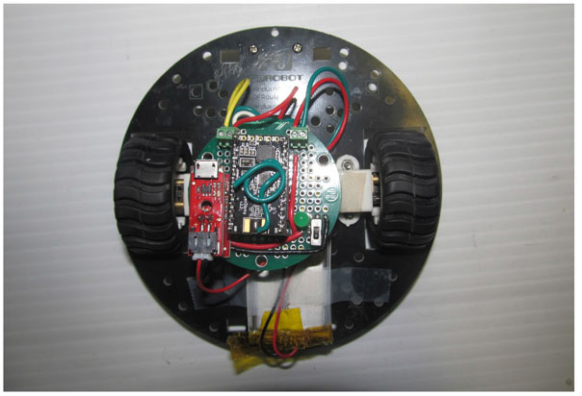


Figure 2. *The wireless microcontroller of the arachne bot is attached to a differential drive chassis.*

The frame of the HEXBUG Spider was selected due to its engaging bio-inspired appearance with the objective of stimulating the interest of younger audiences. The robot's hexapedal locomotion can accommodate mildly uneven terrains, exceeding many other centimeter-scaled, hard-wheeled, mobile robots. However, the system is versatile enough to support different types of drive systems, such as differential wheels or steering configurations. An alternative commercially available differential drive robotic chassis, that was used for the system, can be seen in Fig 2.

2.2. Wireless microcontroller

Each Arachne Bot is remotely controlled by the panStamp low-power wireless microcontroller [48]. The panStamp was selected due to its low cost, small footprint, and radio capabilities. It is programmed from the Arduino open-source Integrated Development Environment (IDE), which incorporates numerous libraries that can accommodate low-level operations. A significant volume of documentation and code is accessible online for the Arduino, which has been produced by the large community that supports this platform.

Adruino's easy-to-use tools and programming language significantly reduce the development time of the Arachne System, making it accessible to users with limited experience in computer engineering. The second significant advantage of the panStamp is its ability to function as a transceiver rather than only a transmitter or receiver. To this end, the control algorithms can be executed from a central supervisory controller or distributively between the wireless mobile nodes. The ability of the Panstamps to function as either simple transmitter-receiver pairs, or as a more complex network, is an essential strength of the Arachne System. These varied capabilities make the platform accessible for researchers or students with limited communication experience, while still allowing for complex communication networks.

Central to the panStamp is an 8 MHz Atmel Atmega 28P MCU. RF communication (frequency: 868/915 MHz, baud rate: 38400) is integrated using a Texas Instruments CC1101 RF interface with a line-of-sight range of up to 200 m.

2.3. Webcam-based agent tracking

Each Arachne Bot is tracked using Roborealm [49], a commercially available visual software application. The overhead camera captures the top view of the planar workspace containing the Arachne Bots. The software is able to track predetermined markers known as fiducials. The fiducial marker is placed on top of the rigid bodies (Arachne Bots or objects), parallel to the plane, in order to be fully visible by the overhead camera. The visual tracking system serves as a virtual Global Positioning System (GPS), localizing each Arachne Bot in the workspace (arena). The rate at which the pose (position and heading)

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

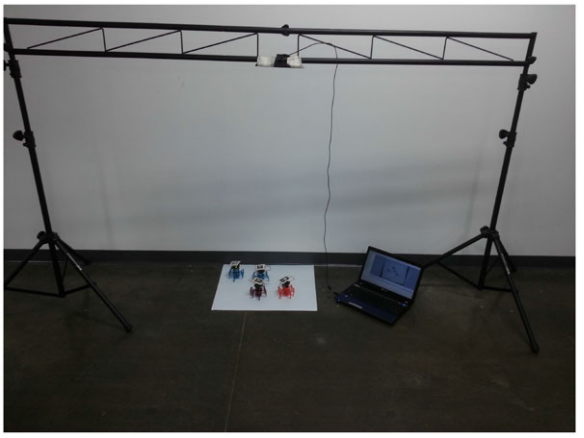


Figure 3. *Camera apparatus and testing area. In the existing system, the overhead camera is placed approximately* 1.6 *m from the floor resulting in a* 3 *m* × 1 *m visible arena.*

measurements are captured ranges between 20 to 30 fps. These data are then further broadcasted to each Arachne Bot at a rate of 100 Hz. The area Roborealm scans is approximately 3m × 1m. However, the size of the viewed area can be controlled by changing the camera's vertical position. The testing setup, including the camera and the robots, is shown in Fig. 3.

In the Arachne System, Roborealm transmits the pose measurements to each agent individually through a Python script that is executed to a central panStamp transmitter. The overhead video sensor used in the experiments was an inexpensive Logitech Photo ID webcam. A personal license for Roborealm costs $49.95, and an academic license is even less costly with a price of $39.95. The Roborealm software, overhead camera, and central panStamp transmitter were connected to a laptop computer. The central computer collects the pose data of each robot and transmits this information to each robot. In a centralized configuration (mission, scenario, or experiment), the central computer executes the supervisory controller controlling the individual agents.

2.3.1. Electronics

A Texas Instruments DRV8833 motor driver commands the motion of the Arachne Bots. This PWM-based motor driver is ideal due to the integrated over-current and over-temperature protection circuit, and its wide range of operational voltages. The motor driver controls both DC motors with a voltage range between 2.7 V and 10.8 V. The PWM inputs to the DRV8833 motor driver control the motor speed and direction. The Arachne Bots are powered by lithium-ion polymer (LiPo) batteries that incorporate an onboard charging station. The charging station also includes an LED power indicator light and a power switch.

The LiPo batteries used with the Arachne Bots output 3.7 V from a supply of 850 mAh, which offers approximately three to four hours of battery life. The battery chargers can accept power via a micro-USB port at either 3.3 V or 5 V. The batteries themselves can plug into the chargers using a JST and receive a maximum current of 100 mA. If on-board charging is unnecessary for a particular project, this circuit could be removed to reduce cost. Table I outlines the individual parts and their cost for a single Arachne Bot. The schematic for the standard Arachne Bot board is illustrated in Fig. 4.

Unlike the vast majority of existing small-scale multi-robot systems, the entire manufacturing process of the Arachne Bots is performed on a simple through-hole protoboard. Therefore, there is no need for machine assembly or extensive soldering experience. Any home user with access to a basic soldering kit could easily assemble an Arachne Bot without any additional training. This not only helps reduce the cost of construction but it also allows complete beginners to assemble Arachne Bots with ease.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Table I. *Electronic components and their cost for a single Arachne Bot.*

Component	Quantity	Price
panStamp	1	$18.95
HexBug Spider	1	$19.95
Female Headers	1	$1.50
LiPo 3.7 V 850 mAh battery	1	$8.95
spdt Switch	1	$0.75
3.7 V LiPo USB Charger/ 3.3 V DC Converter	1	$19.95
2” Round Proto Board	1	$2.95
Motor Driver	1	$4.95
3 mm Green LED	1	$0.35
Screw Terminals	2	$0.75
	Total	$79.80

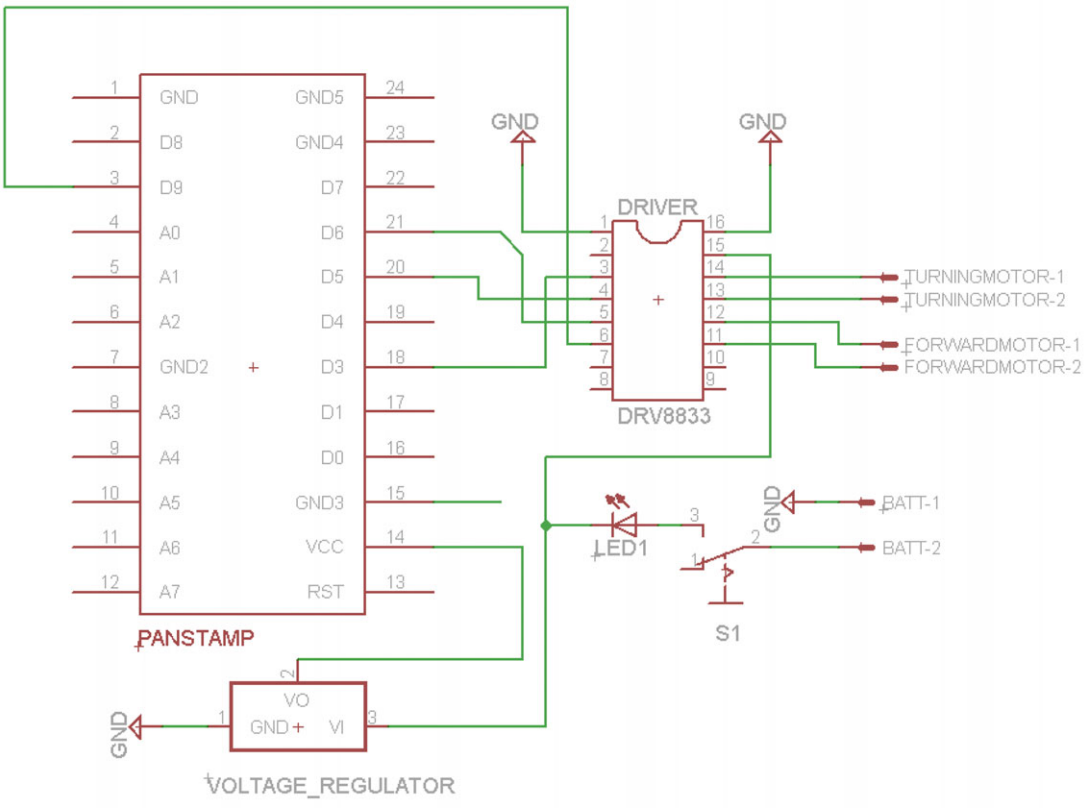


Figure 4. *Schematic of the standard Arachne Bot board. The main components are the panStamp wireless microcontroller, the DRV8833 motor driver, and the voltage regulator.*

3. Sensor Extension: the infrared beacon

The Arachne Bot can be configured to operate using different sensory or maneuverability attachments. One such extension is the InfraRed (IR) sensing beacon board. The beacon functions as an on-board perception module placed on each individual locomotive agent. It can be used to measure both the range and bearing of neighboring Arachne Bots or its surroundings locally. Its functionality resembles the operation of a primitive LiDAR system (Light Detection and Ranging). Compared to other available

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

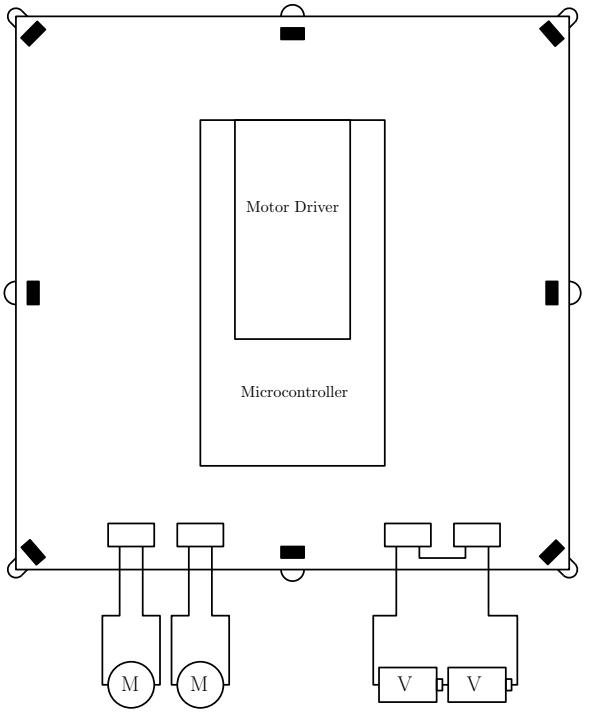


Figure 5. *Representation of the IR Beacon Board. In our design, the board acts as an interface shield, connecting the microcontroller to the power source and the two motors.*

multi-agent robotic systems, the Arachne Bot's IR Beacon (IRB) provides a very low-cost method of implementing cooperative control algorithms on mobile robotic collectives. This section describes how the standard board can be modified to incorporate an on-board IR perception module.

3.1. Hardware overview

The IRB board consists of eight emitter-receiver pairs operating at a wavelength of 940 *nm* arranged on top of, and below the board, at 45° intervals as shown in Fig. 5. The SEN-00241 receivers are straight-facing and have an operational field-of-view of approximately 45°. This allows for the beacons to detect 360° of infrared light (separated to eight sectors). When an infrared source is in range, exactly one receiver will detect the signal with little to no dead zones or overlap.

The 7OED-EL-1L2 emitters are high-powered infrared LEDs chosen due to their high intensity and low power consumption. With the microcontroller output of 3.3 V, the receivers can detect the emitters at distances of up to 15 cm, which is an ample range for a 3 cm by 3 cm robot. The emitters are installed under the beacon facing outwards. Since the receivers and emitters are placed on top of, and under the board, respectively, each pair can be mounted at the same orientation on the beacon without the risk of an agent detecting itself. This setup also allows the boards to detect collisions with reflective walls and surfaces, as well as the imminent occurrence of other agents.

The emitters are connected in parallel to one digital output pin of the microcontroller. Using the built-in analog voltage sensing capabilities of the unit, the sensor readings can range from 0 to 1023 with a sampling rate of 10 kHz. These measured values correspond to a voltage interval from 0 V to 5 V, meaning that each point is equivalent to approximately 4.9 mV. Low and high readings correspond to high and low-intensity signals, respectively.

The board is powered by two lithium-polymer batteries with terminal voltages of 3.7 V. The full 7.4 V supply is used to power the infrared receivers and motor driver which both require at least 5 V to operate

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

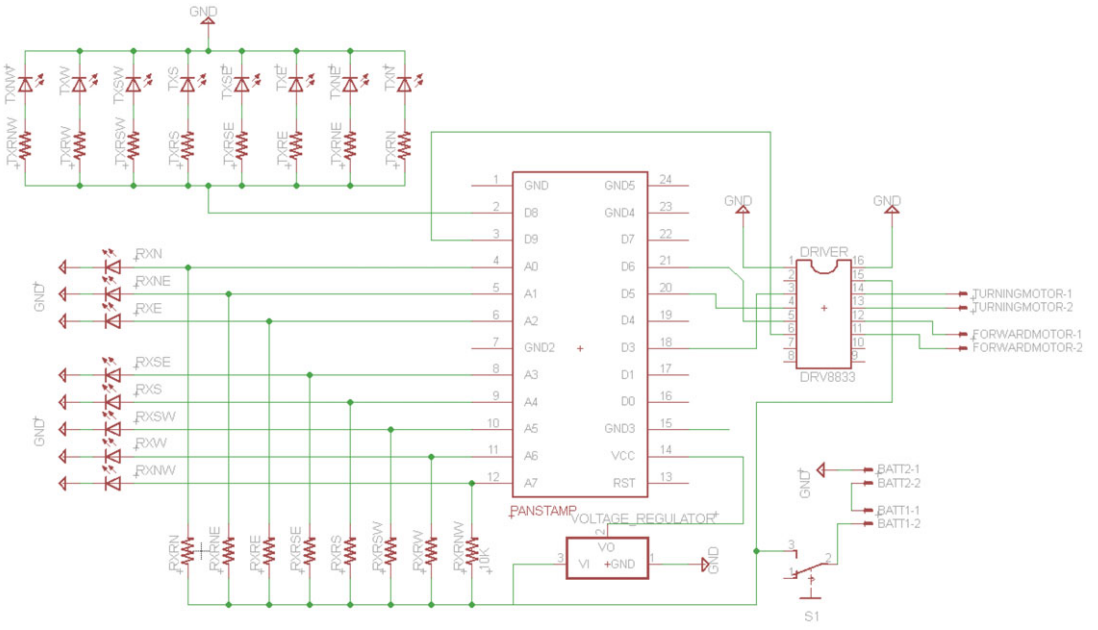


Figure 6. *Wiring schematic for the Arachne Bot IRB board. This extension module is equipped with emitter-receiver pairs that provide each Arachne Bot on-board perception capabilities. Each bot can detect adjacent obstacles and agents.*

accurately. The board also contains a voltage regulator which reduces the voltage to 3.3 V in order to power the microcontroller. The full wiring schematic for the IRB can be found in Fig. 6.

Even though printed electronics were used for the IRB board, the same circuit can be easily assembled with a simple through-hole protoboard (much like the standard Arachne Bot board) without sacrificing size or usability. Table II shows a price breakdown of each individual component of the Arachne Bots IRB.

3.2. Measurements

Three tests were performed to ensure the accuracy and usability of the IRB hardware. The first two tests investigate the localization capabilities of the IRB boards. The use of a local coordinate system requires two types of measurements: range and bearing. Range determines how far away a neighboring agent is located, and bearing determines its relative heading. A third test was also performed to determine the amount of noise or interference detected by the infrared receivers.

Each test was performed using two IRB boards on a flat surface. In order to measure range, an emitter beacon was kept stationary while a receiver beacon was moved apart from the emitter incrementally. The range measurements were performed at a constant bearing with the two North emitters of the two beacons facing each other. The receiving beacon was then moved back from 0 cm to 18 cm at increments of 1 cm. Figure 7 illustrates the experiment with corresponding results presented in Fig. 8. All graphical data are inverted for ease of viewing, meaning that a low reading corresponds to a low-intensity signal. The microprocessor uses a nonlinear quantized scale to determine the range in centimeters. Figure 9 shows the quantized values that the microprocessor uses along with the raw analog voltage reading attained.

The verification of the direction bearing was performed at a distance of 6 cm with the emitting board kept stationary, while the receiving board was rotated in 10° increments through a full rotation. The experiment procedure is illustrated in Fig. 10, and the receiver readings can be found in Fig. 11.

The final experiment was performed using a method that was very similar to the range test. The boards were placed on a level surface facing one another and starting at a distance of 1 cm. From there, they

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Table II. *Electronic components and their cost of an arachne bot with the IRB board.*

Part name	Quantity needed	Individual cost
IR Receiver	8	$1.95
IR Emitter	8	$0.35
Motor Driver	1	$4.95
Voltage Regulator	1	$1.95
Motor Connectors	2	$0.75
Battery Connectors	2	$0.16
Resistors	16	$0.05
Power Switch	1	$0.75
Battery	2	$12.95
HexBug Spider	1	$19.95
panStamp	1	$18.95
Total cost		$93.47

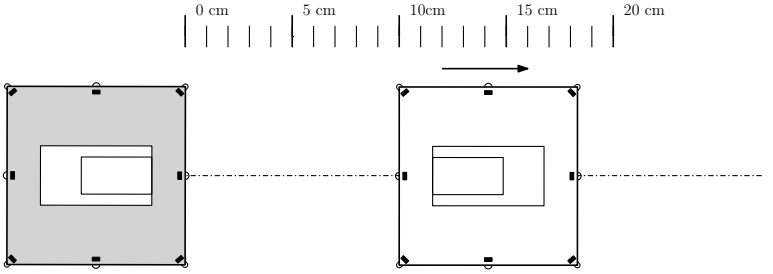


Figure 7. *Diagram of range test. The receiver beacon (white) is moving apart from the emitter beacon (grey), in increments of* 1 *cm while retaining constant heading.*

were gradually separated in intervals of 3 cm. At each of these distances, the analog voltage reading was recorded about 1500 times over a period of around 3 min. The results of this experiment prove that there is very little noise or interference. In fact, not even one point of deviation was measured over distances up to 13 cm. These results indicate that the beacons detect very little noise, removing the need to filter the infrared sensor signals.

4. Analytical foundations of the system: kinematic analysis, control principles, and task execution algorithms

This section describes some fundamental analytical background related to the motion of the Arachne Bots. In addition, it provides a series of engaging algorithms that involve both individual and collaborative behavior of the Arachne System. The objective is to provide an analytical description of the Arachne Bots motion and also set some benchmark routines that can be used for object manipulation and group-collaborative tasks.

We emphasize that the algorithms described herein are relevant to small-scale planar mobile robots across different locomotion configurations, extending beyond the Arachne System. Their design is intuitive, aiming to be accessible without necessitating an extensive knowledge of control systems theory.

4.1. Arachne Bot kinematics

The legs of the Arachne Bot are equivalent to a nonslipping driving wheel. Its steerability is controlled by the rotation of its torso. The Arachne Bot maneuverability is similar to a differential-drive

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

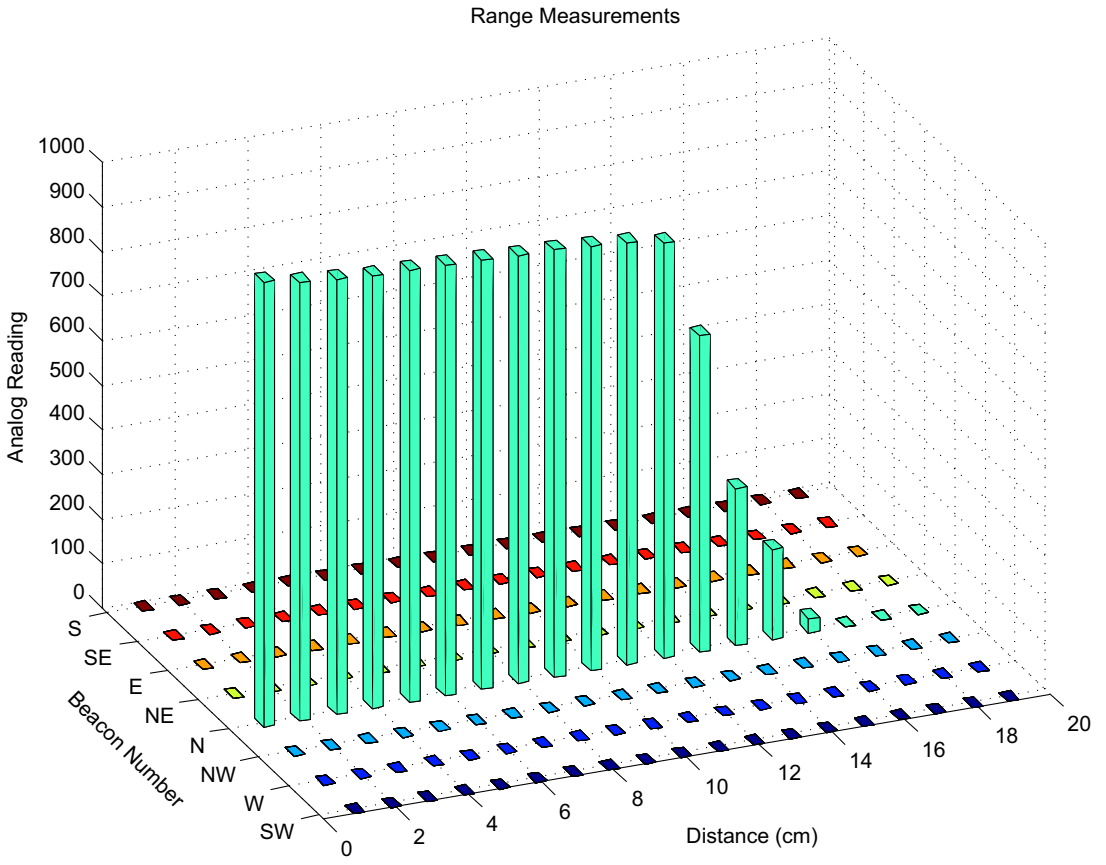


Figure 8. *Range measurements from the receivers of the IRB board.*

or unicycle-like robots. The robot's kinematics constitute a nonholonomic system that has a first-order, non-integrable constraint [50]. As a result, the Arachne Bots' motion cannot be described as omnidirectional.

Two reference frames are introduced to illustrate the Arachne Bots' planar kinematic equations. The reference frames are defined using the center and two orthogonal unit vectors. First, the fixed inertial (global) reference frame is $\mathcal{F}_I = \{O_I, \vec{i}_I, \vec{j}_I\}$. The second reference frame is a moving body-fixed reference frame, defined as $\mathcal{F}_B = \{O_B, \vec{i}_B, \vec{j}_B\}$. This reference frame origin, O_B, is the center of gravity body (CG) of the Arachne Bot. A traditional right-handed Cartesian coordinate system $\{\vec{i}_B, \vec{j}_B\}$ is constructed using a vector, $\vec{i}_B$, which points forward and a vector $\vec{j}_B$ which points to the left aft side as shown in Fig. 12.

The posture vector, $p = [x;y;\psi]$, fully defines the robot's planar 2-D motion. Here, x, y represent the inertial frame Cartesian coordinates, and the robot's heading is ψ. By definition, the two frames $\mathcal{F}_I$ and $\mathcal{F}_B$ are aligned when $\psi = 0$. The combination of angular velocity, ω, and linear velocity, v, controls the robot's locomotion. The standard unicycle-like robot kinematics are

$$\begin{aligned} \dot{x} &= v\cos\psi, \\ \dot{y} &= v\sin\psi, \\ \dot{\psi} &= \omega. \end{aligned} \tag{1}$$

The linear and angular velocities are controlled by regulating the voltage of the two DC motors on board the Arachne Bot. For simplicity, we assume a static map between the velocities of the robots and its motors, that is,

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

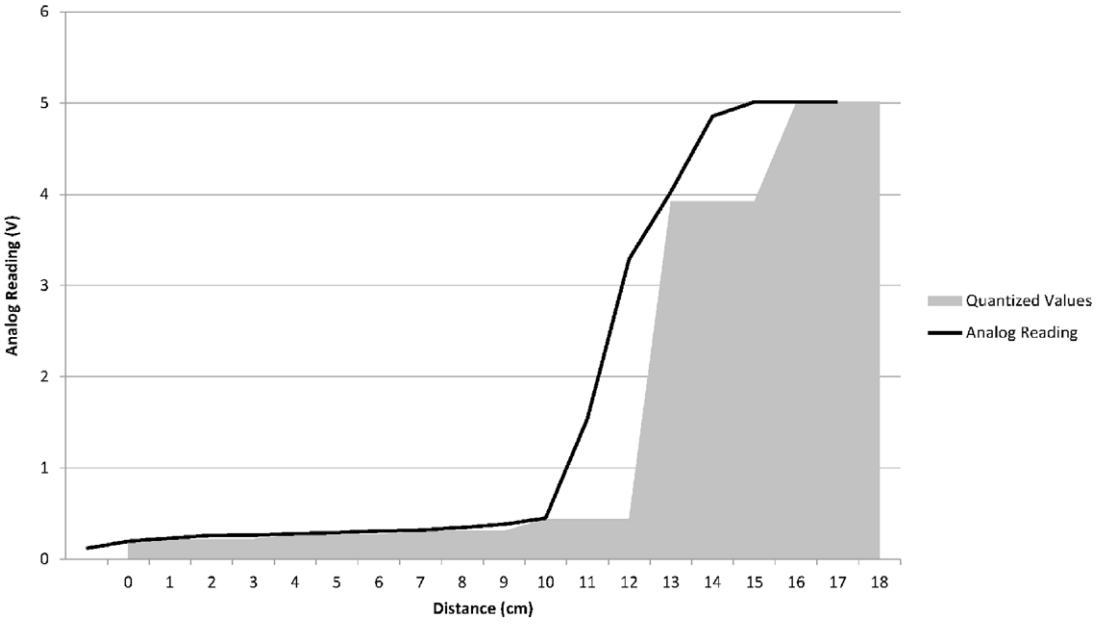


Figure 9. *Range map of the IRB board. The distance readings are not linear. In the chart, the dark line represents the analog voltage readings on the north pin, while the shaded areas mark the quantized distances which the microprocessor translates these readings from.*

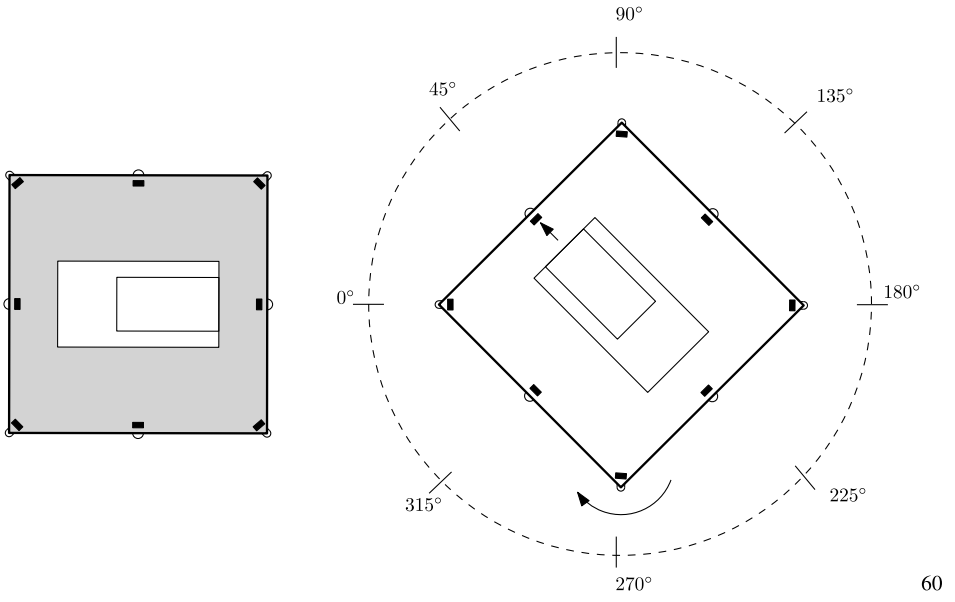


Figure 10. *Diagram of bearing test. The receiver board (white) is rotated in increments of* 10° *relative to the emitter board (gray). The relative range between the two boards is kept constant to* 6 *cm.*

$$v = K_v V_{legs}, \tag{2}$$

$$\omega = K_\omega V_{torso}, \tag{3}$$

where K_v and K_ω are positive gains, and V_{legs} and V_{torso} are the voltages that regulate the motion of the legs and the torso of the Arachne Bot, respectively.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

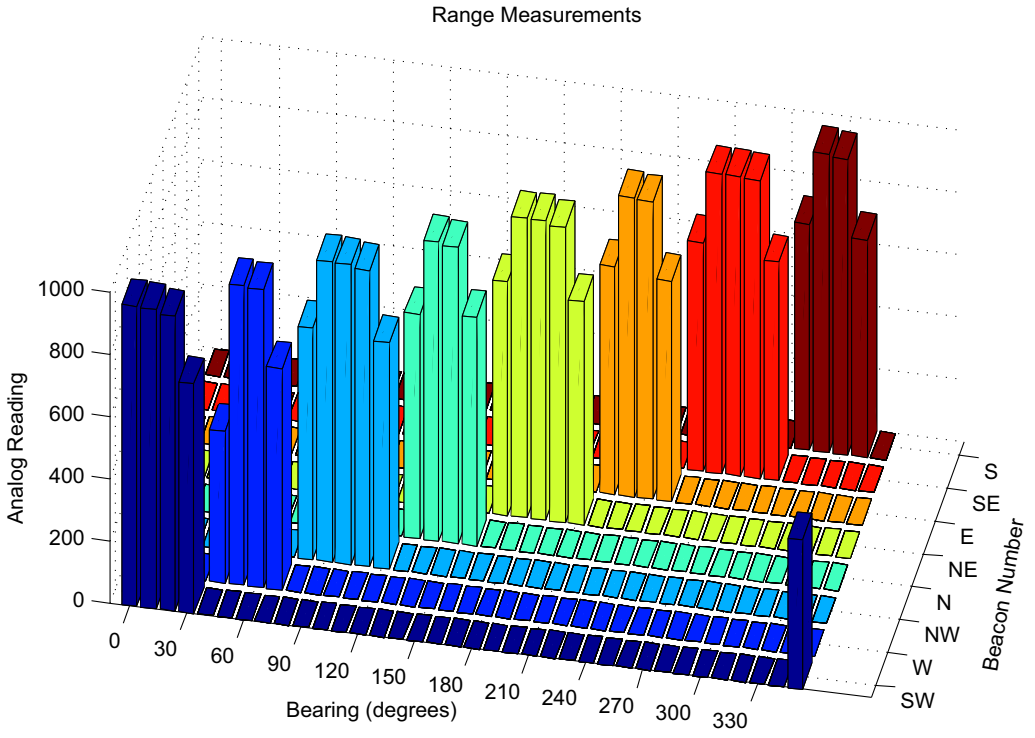


Figure 11. *Bearing measurements from the receivers of the IRB board.*

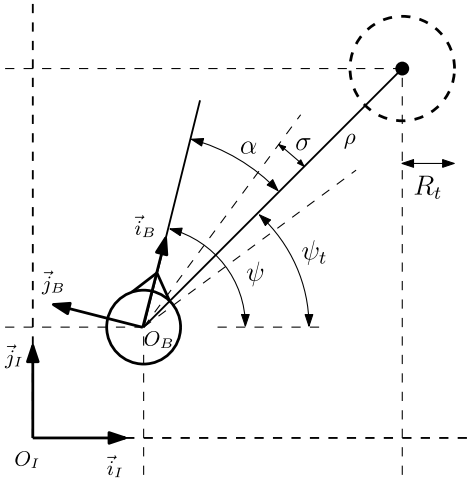


Figure 12. *The polar coordinate system, inertial and body-fixed reference frames, and the heading angles defined from cartesian to polar coordinate transformation. The angle σ represents the LOS of the robot.*

4.2. Set-point control (move to a target location)

The stabilization challenge of non-omindirectional robots has been thoroughly investigated [50–53]. The goal is to move the robot to the null posture $[0;0;2n\pi]$ with $n \in \mathbb{Z}$. Robot stabilization with non-holonomic constraints is presented in Kolmanovsky's survey paper [54].

In the context of the Arachne System, set-point control is considered a less analytically complex modulation of the stabilization law. Hereafter, we only need the inertial reference frame, $\mathcal{F}_I$, to define

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

coordinates. The goal is to have the robot navigate to a reference target location $(x_t, y_t) \in \mathbb{R}^2$ in the 2-D plane, referred to as the set-point or waypoint. The robot is not required to have a certain heading when it reaches the target set-point.

Our subsequent algorithms for collision avoidance and object manipulation use a high-level paradigm that calculates reference waypoints for robots[2] in a discrete-time or -event manner. This method is standard for the "task" or "mission" control layer, generating commands that define a velocity profile or waypoint. The subordinate set-point control algorithm ensures the robot converges on these waypoints within a practical timeframe by adjusting its linear and angular velocities. The lower-level motor and steering control layer, as delineated in (2) and (3) for the Arachne System, maps linear and angular velocity profiles into voltage signals for the robot's motor(s) and steering mechanism. This foundational layer aligns with the robot's electromechanical configuration that defines its maneuverability [55]. This hierarchical control architecture enables the high-level algorithms, specifically those for collision avoidance and object manipulation, to be not only applicable to the Arachne System but also adaptable and hardware-agnostic with respect to the mobile robot platforms on which they are implemented.

We adopt a modified version of the set-point control law presented in ref. [56]. Essentially, the algorithm is a formal version of the intuitive principle of "turn and move-towards-the-target" approach. The controller is expressed with respect to the polar coordinates, instead of the standard Cartesian form:

$$\rho = \sqrt{(x - x_t)^2 + (y - y_t)^2} \quad \psi_t = \operatorname{atan2}\left(y_t - y, x_t - x\right). \tag{4}$$

Here, the coordinates ρ and ψ_t represent the relative distance and heading, respectively, between the Arachne Bot and the target. The error between the robot's current heading and the target is $\alpha = \psi_t - \psi$. The three coordinate systems are illustrated in Fig. 12.

The set-point control problem requires that the coordinates (α, ρ) will asymptotically converge to one of the equilibrium points $(0, 2n\pi)$ with $n \in \mathbb{Z}$ as $t \to \infty$. This requires that the target is aligned with the heading of the robot $(\alpha \to 2n\pi)$, and that the robot converges to the target $(\rho \to 0)$. After the coordinates change (from x, y to ρ, α), the robot's kinematics become

$$\dot{\rho} = -\cos\alpha \cdot v, \tag{5}$$

$$\dot{\alpha} = \sin\alpha \cdot \frac{v}{\rho} - \omega. \tag{6}$$

A quick inspection of the kinematics illustrates that a singularity occurs when $\rho = 0$. This incidence takes place due to the ambiguity introduced by the definition of `atan2`$(\cdot, \cdot)$ when $(x, y) = (x_t, y_t)$. When the control law is implemented, this singularity is eliminated. Furthermore, in practical, real-world implementations, algorithm execution is terminated once the robot enters a circular disk of the set-point of radius R_t. The feedback control law is similar to that presented in ref. [35]:

$$\begin{aligned} v &= K_\rho \cos\alpha \cdot \rho, \\ \omega &= K_\rho \cos\alpha \cdot \sin\alpha + K_\alpha \sin\alpha, \end{aligned} \tag{7}$$

where K_ρ and K_α the positive valued gains. This differs from [35] because $K_\alpha \sin\alpha$ is used rather than the simpler proportional term $K_\alpha\alpha$. In our approach, all angles are channeled through trigonometric functions to eliminate the discontinuous jumps that take place by the definition of ψ_t and the measurement range of ψ.

The stability of the control law is established using the following Lyapunov function:

$$V(\rho, \alpha) = \frac{1}{2}\rho^2 + (1 - \cos\alpha)$$

[2]Henceforth, the terms 'Arachne Bot' and 'robot' or 'mobile robot' are considered synonymous. While these algorithms have been specifically developed for the Arachne System, their application can be feasibly extended to any mobile robotic platform.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Algorithm 1 Pseudocode of the set-point control algorithm.

function Set-point Control
inputs: $x, y, x_t, y_t, \sigma, R_t, K_\rho, K_\alpha$
outputs: v, ω

$\rho \leftarrow \sqrt{(x - x_t)^2 - (y - y_t)^2}$
$\psi_t \leftarrow \text{atan2}\,(y_t - y, x_t - x)$
$\alpha \leftarrow \psi_t - \psi$
if $\cos(\alpha) > \cos(\sigma)$ **then**
  if $\sin(\alpha) \geq 0$ **then**
    $(v, \omega) \leftarrow (0, -\omega_0)$
  else
    $(v, \omega) \leftarrow (0, \omega_0)$
  end if
else
  $v \leftarrow K_\rho \cos\alpha \cdot \rho$
  $\omega \leftarrow K_\rho \cos\alpha \cdot \sin\alpha + K_\alpha \sin\alpha$
end if
if $\rho \leq R_t$ **then**
  $(v, \omega) \leftarrow (0, 0)$
end if

which is positive definite for $\rho \neq 0$ and $\alpha \neq 2n\pi$ where $n \in \mathbb{Z}$. Taking the derivative of $V(\rho, \alpha)$ with respect to time along the system trajectories yields:

$$\begin{aligned}\dot{V} &= -\cos\alpha \cdot \rho \cdot v + \sin\alpha \cdot \left(\sin\alpha \cdot \frac{v}{\rho} - \omega\right) \\ &\overset{(7)}{=} -K_\rho \cos^2\alpha \cdot \rho^2 - K_\alpha \sin^2\alpha < 0 \\ &\text{for } \rho \neq 0 \text{ and } \alpha \neq 2n\pi,\ n \in \mathbb{Z}.\end{aligned}$$

The derivative of $V(\rho, \alpha)$ is negative definite. The resulting equilibrium points $(0, 2n\pi)$, with $n \in \mathbb{Z}$, are asymptotically stable.

When the heading error equals $2n\pi$, the robot aligns with the target. According to the control law in (7), the advancement v turns negative if $\cos\alpha < 0$. When that happens, the Arachne Bot reverses until its heading aligns with the target. To ensure forward movement only, conditional logic is applied, enabling positive advancement when the target falls within the robot's line-of-sight (LOS), represented by the angle $\sigma \in \left(0, \frac{\pi}{2}\right]$. In this case, pure rotation without advancement occurs if $|\alpha| > \sigma$ or $\cos(\alpha) > \cos(\sigma)$, with control velocities as follows:

$$v = 0; \quad \omega = \begin{cases} -\omega_0, & \sin\alpha \geq 0, \\ \omega_0, & \text{otherwise}, \end{cases}$$

where $\omega_0 > 0$ denotes a constant angular velocity. When $|\alpha| < \sigma$, the feedback control law reverts to (7), ensuring a positive (forward) advancement v as $\cos\alpha > 0$ in this heading error, α range. The pseudocode for this set-point control law is detailed in Algorithm 1.

4.3. Collision avoidance

Collision avoidance is frequently encountered in real-world multi-robot applications. Deconfliction algorithms should be used by each robot on the team to guarantee collision-free missions. These collision avoidance algorithms can take many forms including optimization and force fields [57]. We present an

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

intuitive, distributed deconfliction algorithm to validate the collaborative behavior and group constraints of Arachne Bots.

Consider a workspace containing n mobile robots (Arachne Bots), mathematically represented as the index set $\mathcal{N} = \{i = 1, \ldots, n\}$. A target location $\left(x_t^i, y_t^i\right) \in \mathbb{R}^2$, $i \in \mathcal{N}$, is prescribed to each robot. When two robots, i and j, are separated by a distance that is less than some critical value, R_c, a conflict has occurred:

$$\rho_{i,j} \leq R_c, \text{ where } \quad \rho_{i,j} = \sqrt{(x^i - x^j)^2 + (y^i - y^j)^2}.$$

An intermediate, reference location for each robot, $\left(x_r^i, y_r^i\right) \in \mathbb{R}^2$, $i \in \mathcal{N}$, is defined within the collision avoidance algorithm. The following equality $\left(x_r^i, y_r^i\right) = \left(x_t^i, y_t^i\right)$, holds true when no conflict occurs. At each execution instant, robot i determines the set of conflicting robots, which is defined as $\mathcal{D}_i = \left\{j \in \mathcal{N} - \{i\} : \rho_{i,j} \leq R_c\right\}$. In a real-world application, the set $\mathcal{D}_i$ can be identified by an onboard proximity sensor on robot i, such as the infrared beacon described in Section 3. If a single sensor measures the absolute position of each robot, the set $\mathcal{D}_i$ is determined by calculating the relative distance between each robot i and the remaining $n - 1$ robots in the swarm.

Robot i's relative heading angles with respect to (conflicting) robots belonging to $\mathcal{D}_i$ are

$$\psi_{i,j} = \text{atan2}\left(y^j - y^i, x^j - x^i\right). \tag{8}$$

The above definition yields $\psi_{j,i} = \psi_{i,j} - \pi$. Note that robot i's reference heading to the target (4) is by definition

$$\psi_t^i = \text{atan2}\left(y_t^i - y^i, x_t^i - x^i\right). \tag{9}$$

The algorithm partitions the radius R_c circular disk with origin at robot i, denoted by A_i, into S equal circular sectors, S_i^k, $k = 1, \ldots, S$, with central angles $s = 2\pi/S$. When a conflict arises, indicated by a nonempty set $\mathcal{D}_i$, the S circular sectors are sequentially scanned to locate the first one that is empty, meaning it does not contain any conflicting robots. The first sector, S_1, is defined such that its bisection is aligned with the line that connects robot i with its target location, and the reference location is set equal to the target location, i.e., $\left(x_r^i, y_r^i\right) = \left(x_t^i, y_t^i\right)$. Assuming that the first sector is empty of conflicting robots, robot i moves toward the target location. Otherwise, the second sector is scanned, and if, in turn, this sector is empty, the reference location becomes the point where the bisection of S_2 intersects the margin of A_i.

Overall, the scanning sequence continues until a sector S_i^k is found free of conflicting robots in $\mathcal{D}_i$. When this is the case, the reference point of robot i, $\left(x_r^i, y_r^i\right)$, is set equal to the intersection point of the bisection of the sector S_i^k with the circular margin of A_i, such that we have

$$x_r^i = x^i + R_c \cos\left[\psi_k^i\right] \qquad\qquad y_r^i = y^i + R_c \sin\left[\psi_k^i\right], \tag{10}$$

where

$$\psi_k^i = \psi_t^i + (k-1)\,\frac{2\pi}{S} \text{ for } k \in \mathcal{N}. \tag{11}$$

Figure 13 shows schematically the sector partitions of A_i. Segment S_k^i is deemed collision free when for $j \in \mathcal{D}_i$ we have

$$\cos\left(\psi_{i,j} - \psi^i\right) < \cos\left(\pi/S\right). \tag{12}$$

When a robot finds itself encircled by conflicting agents, meaning all circular sectors are occupied, it opts to remain stationary by setting its coordinates (x_r^i, y_r^i) to its current location (x^i, y^i). The choice of how many partitions S to divide the disk A_i into is a critical design parameter that influences the algorithm's behavior and performance. A higher number of partitions enhances the probability of identifying an unoccupied sector for the robot to move toward, yet it concurrently escalates the algorithm's computational load and run time.

Our approach delineates a specific movement strategy for the robots: each seeks the earliest unoccupied "corridor" to its right for navigation. This prescribed movement pattern induces a collective,

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

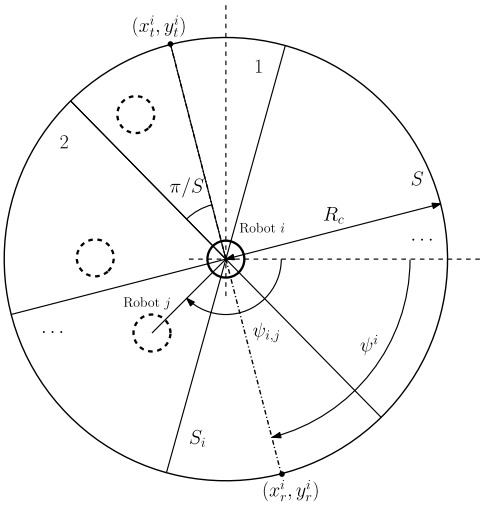


Figure 13. *The conflict disk, A_i, characterized by a radius R_c, is segmented into S circular sectors. During each iteration of the algorithm, these sectors are examined in sequence to locate a sector devoid of collisions. The sector identified as collision-free, denoted as S_i^k, where $k \in \{1, \ldots, S\}$, is designated to include the reference location (x_r^i, y_r^i).*

macroscopic behavior akin to the counterclockwise flow observed in a cyclone, leading the swarm, when observed as a whole, to exhibit rotational motion. Robots situated on the perimeter of this cyclonic formation are thus able to locate a clear path toward their designated targets. The pseudocode detailing this collision avoidance mechanism is presented in Algorithm 2.

The algorithm can be executed either centrally, in a supervisory controller, or distributively, on each robot. In the central execution, the supervisory controller receives the coordinates of all robots and determines the reference coordinates for each robot to ensure collision-free motion. In the distributed execution, each robot independently uses its global coordinates and sensory data, such as from a proximity sensor (e.g., the IRB board), to compute its desired reference coordinates.

In terms of complexity, the central execution has a time and space complexity of $O(nS)$, where n is the number of robots and S is the number of sectors. Conversely, the distributed execution reduces the complexity to $O(S)$.

4.4. Object manipulation

Incorporating object interaction capabilities into mobile robots significantly enhances their functionality and adaptability in diverse environments. We describe an algorithmic procedure that enables Arachne Bot to perform a form of object manipulation. Specifically, the algorithm allows a planar mobile robot to push a round object from any initial position to a designated target position, $\mathcal{T}$: $(x_t, y_t) \in \mathbb{R}^2$.

During object manipulation, the robot persistently re-positions itself so that the object lies between the robot and the target. It then walks toward the target, pushing the object as it goes. Let $\mathcal{O}$: $(x_o, y_o) \in \mathbb{R}^2$ denote the geometric center of the object and its coordinates. The contact point is determined by a position vector from the object to the target. This position vector, along with its unit vector, is given by

$$N_o = \left[x_o - x_t;\, y_o - y_t \right] \quad \text{and} \quad \hat{n} = \left[n_x;\, n_y \right],$$

where $\hat{n} = N_o / \|N_o\|$. The point of contact, $\mathcal{C}$, between the object and the robot is determined by the unity vector $\hat{n}$ as well as a positive coefficient μ_o, representative of the radius (or size) of the object. The position of $\mathcal{C}$, in the inertial reference frame is given by

$$\mathcal{C}\colon \left(C_x, C_y\right) = \left(x_o + \mu_o \cdot n_x, y_o + \mu_o \cdot n_y\right).$$

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Algorithm 2 Pseudocode of the collision avoidance algorithm.

function: Collision Avoidance
inputs: $n, R_c, x^l, y^l, x_t^i, y_t^i$ with $l = 1, \ldots, n$
outputs: x_r^i, y_r^i

$\psi_t^i \leftarrow \text{atan2}\left(y_t^i - y^i, x_t^i - x^i\right)$
for $j = 1{:}n$ **do**
  $\rho_{i,j} \leftarrow \sqrt{(x^i - x^j)^2 + (y^i - y^j)^2}$
  $\psi_{i,j} \leftarrow \text{atan2}\left(y^j - y^i, x^j - x^i\right)$
end for
Determine $\mathcal{D}_i = \left\{j \in \mathcal{N} - \{i\}{:}\rho_{i,j} \leq R_c\right\}$
where $\mathcal{N} = \{i = 1, \ldots, n\}$
if $\mathcal{D}_i = \emptyset$ **then** ▷ $\mathcal{D}_i$ is null
  $\left(x_r^i, y_r^i\right) = \left(x_t^i, y_t^i\right)$
else
  for $k = 1{:}S$ **do**
    $\psi^i \leftarrow \psi_t^i + (k-1)\,\frac{2\pi}{S}$
    Determine
$\mathcal{C}_i = \{j \in \mathcal{D}_i{:}\cos\left(\psi_{i,j} - \psi^i\right) > \cos(\pi/S)\}$
▷ Find all robots that lie in the circular sector k
    if $\mathcal{C}_i = \emptyset$ **then** ▷ $\mathcal{C}_i$ is null
      $x_r^i \leftarrow x^i + R_c \cos\left[\psi^i\right]$
      $y_r^i \leftarrow y^i + R_c \sin\left[\psi^i\right]$
    else
      $\left(x_r^i, y_r^i\right) = \left(x^i, y^i\right)$
    end if
  end for
end if

To circumvent an untimely collision with the object before reaching point $\mathcal{C}$, the robot's reference point is adjusted to first direct it toward an inline point, $\mathcal{I}$, prior to proceeding to $\mathcal{C}$. Similar to $\mathcal{C}$, $\mathcal{I}$ is located along the line that connects the target with the object. However, $\mathcal{I}$ is positioned further from the target compared to $\mathcal{C}$. The position of $\mathcal{I}$ is determined analogous to $\mathcal{C}$. The location of $\mathcal{I}$ is defined as

$$\mathcal{I}{:}\left(I_x, I_y\right) = \left(x_o + \mu_I \cdot n_x, y_o + \mu_I \cdot n_y\right),$$

where the coefficient μ_I, which is largest than μ_o, specifies its distance from the object.

When the robot commences the manipulation procedure from a position between the target and the object, it is instructed to execute a sidestep maneuver to one of two auxiliary reference points, $\mathcal{A}$ or $\mathcal{B}$, situated in the aft vicinity of the object. This maneuver is a precursor to the previously outlined procedure. Points $\mathcal{A}$ and $\mathcal{B}$ are defined with the same safe distance μ_I of $\mathcal{I}$ such that no premature collisions occur when the robot approaches the object. The waypoints $\mathcal{A}$ and $\mathcal{B}$ are given by

$$\mathcal{A}{:}\left(A_x, A_y\right) = \left(x_o - \mu_I \cdot n_x, y_o + \mu_I \cdot n_y\right)$$
$$\mathcal{B}{:}\left(B_x, B_y\right) = \left(x_o + \mu_I \cdot n_x, y_o - \mu_I \cdot n_y\right)$$

Figure 14 illustrates all the waypoints described so far. The sequence of waypoints that the Arachne Bot moves through is governed by the angle, β_o, which is determined by the difference between the line segment connecting the robot to the object and the one connecting the target to the object. This angle β_o is calculated as the difference between the relative angle of the object with respect to the robot, ϕ_o, and the relative angle of the object with respect to the target, θ_o. These angles are computed as follows:

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

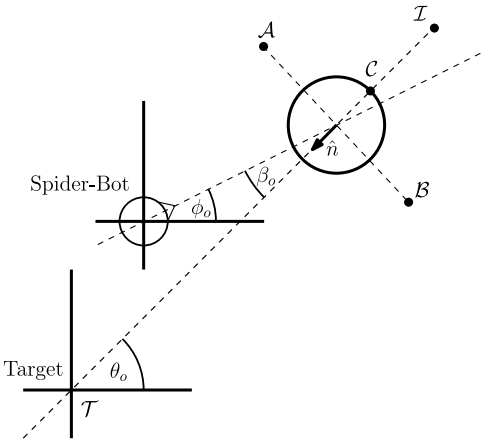


Figure 14. *Representation of waypoints and relative angles involved in the object manipulation algorithm.*

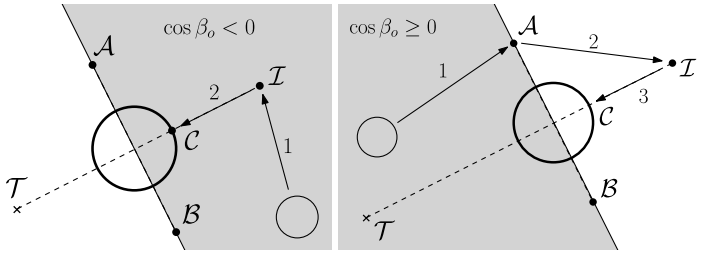


Figure 15. *Halfplanes and their corresponding conditions determining the sequence of waypoints for the object manipulation algorithm. The selection between waypoints $\mathcal{A}$ or $\mathcal{B}$ is dictated by the sign of* $\sin\beta_o$.

$$\theta_o = \text{atan2}\,(y_o - y_t, x_o - x_t)\,,$$
$$\phi_o = \text{atan2}\,(y_o - y, x_o - x)\,,$$

and

$$\beta_o = \phi_o - \theta_o.$$

The choice of the waypoint sequence – either $(\mathcal{I}, \mathcal{C})$ or $(\mathcal{A}$ or $\mathcal{B}, \mathcal{I}, \mathcal{C})$ – depends on the robot's initial position relative to the quadrants defined by the line through points $\mathcal{A}$ and $\mathcal{B}$ and the line connecting the target with the robot, as depicted in Fig. 15.

If the robot starts in the half-plane opposite the target, demarcated by $\mathcal{A}$ and $\mathcal{B}$, the condition $\cos\beta_o < 0$ applies, and the robot adheres to the sequence $(\mathcal{I}, \mathcal{C})$. Otherwise, it follows the sequence $(\mathcal{A}$ or $\mathcal{B}, \mathcal{I}, \mathcal{C})$. The choice between $\mathcal{A}$ and $\mathcal{B}$ as the initial waypoint hinges on the sign of $\sin\beta_o$; $\mathcal{A}$ is chosen if $\sin\beta_o > 0$; otherwise, the sequence starts with $\mathcal{B}$. The task is deemed completed when the relative distance between the object, $\mathcal{O}$, and the target, $\mathcal{T}$, falls below a tolerance threshold, ϵ_t.

The pseudo-code for this procedure is shown in Algorithm 3. At each step of the pseudo-code, a waypoint is generated. The set-point control algorithm, detailed in Section 4.2, is then utilized to guide the robot toward these waypoints.

4.5. Group object manipulation

Advancing from single-robot to multi-robot manipulation, this section presents an algorithm for a team of four Arachne Bots to transport a rectangular box with known dimensions. This algorithm enables

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Algorithm 3 Pseudo-code of the object manipulation algorithm.

function: Object Manipulation
inputs: $x_o, y_o, x_t, y_t, x, y, \mu_o, \mu_I$
outputs: Coordinates of $\mathcal{I}, \mathcal{C}, \mathcal{A}$ or $\mathcal{B}$

$N_o \leftarrow (x_o - x_t, y_o - y_t)$
$\hat{n} \leftarrow \frac{N_o}{\|N_o\|}$
$\mathcal{C} \leftarrow \left\{(x_o + \mu_o \cdot n_x), (y_o + \mu_o \cdot n_y)\right\}$
$\mathcal{I} \leftarrow \left\{(x_o + \mu_I \cdot n_x), (y_o + \mu_I \cdot n_y)\right\}$
$\mathcal{A} \leftarrow \left\{(x_o - \mu_I \cdot n_x), (y_o + \mu_I \cdot n_y)\right\}$
$\mathcal{B} \leftarrow \left\{(x_o + \mu_I \cdot n_x), (y_o - \mu_I \cdot n_y)\right\}$
$\theta_o \leftarrow \text{atan2}\,(y_o - y_t, x_o - x_t)$
$\phi_o \leftarrow \text{atan2}\,(y_o - y, x_o - x)$
$\beta_o \leftarrow \varphi_o - \theta_o$
if $\cos \beta_o < 0$ **then**
    move to $\mathcal{I}$
    move to $\mathcal{C}$
    move to $\mathcal{T}$
else if $\sin \beta_o \geq 0$ **then**
    move to $\mathcal{A}$
    move to $\mathcal{I}$
    move to $\mathcal{C}$
    move to $\mathcal{T}$
else
    move to $\mathcal{B}$
    move to $\mathcal{I}$
    move to $\mathcal{C}$
    move to $\mathcal{T}$
end if

the coordinated effort required for the robots to push the box to a specified target location and orientation, represented as $\mathfrak{T}$: $[\mathcal{T}, \theta_t] \in \mathbb{R}^2 \times [0, 2\pi)$ where $\mathcal{T}$: $(x_t, y_t) \in \mathbb{R}^2$, from any initial configuration of the robots and the box.

The specified challenge merges swarm robotics [58] with the classical computational challenge known as the "piano movers" problem [59, 60], which involves moving an object from one point to another within a plane containing geometric constraints, such as walls.

The core strategy involves the robots encircling the box in a calculated, equiangular arrangement to ensure effective *caging* and *transportation*. Caging, a method where robots form a perimeter around an object to control its movement, is fundamental for achieving the desired position and orientation [61–65]. In our approach, the movement toward the target is facilitated through waypoints, which guide the robots through three main phases: *pathing* (initial positioning), *translation* (moving towards the target location), and *rotation* (adjusting to the target orientation). In the subsequent analysis, it is necessary to define an object-fixed frame, denoted as $\mathcal{F}_B = \{\mathcal{O}, \vec{i}_B, \vec{j}_B\}$.

The origin of the frame, $\mathcal{O}$, is positioned at the geometric center of a rectangular box. The basis vectors $\{\vec{i}_B, \vec{j}_B\}$ are aligned with the symmetry axes of the box, with $\vec{j}_B$ preferably oriented along the symmetry axis corresponding to the length or longer side of the rectangular object.

Pathing.
The pathing phase entails the determination of the robots' initial approach waypoints in relation to the object, contingent upon the object's orientation. Initially, we define the orientation angle of the object, θ_b,

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

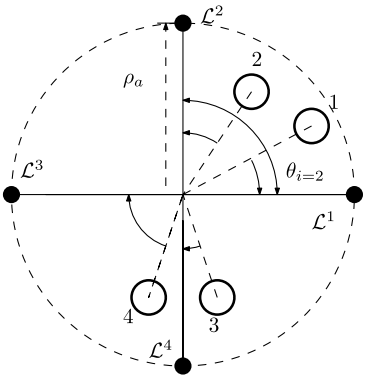


Figure 16. *Approaching robot waypoints, $\mathcal{L}^i$ with $i \in \{1, 2, 3, 4\}$, around the object during the pathing phase of the algorithm. The robots are distributed around the object in $\pi/2$ increments starting from the object's orientation θ_b in the reference frame.*

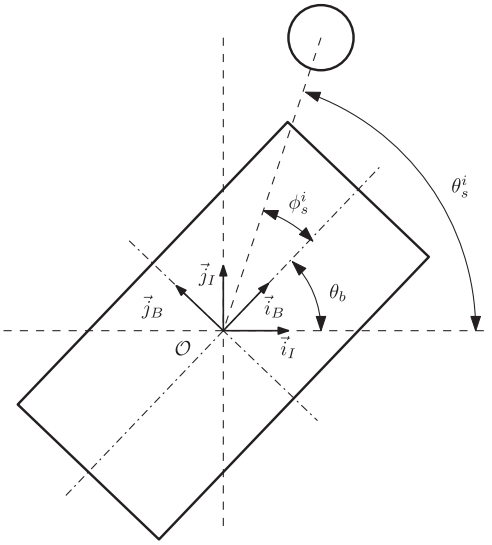


Figure 17. *Relevant angles for the calculation of the relative angles ϕ_s^i of the robots with respect to the box.*

within the inertial reference frame, leading to the segmentation of the plane into quadrants based on this angle. Subsequently, approach points are delineated at a fixed distance, ρ_a, from the object. These points are evenly distributed, with angular increments of $\frac{\pi}{2}$ radians commencing from θ_b. Figure 16 depicts the four waypoints used to position the robots around the box (object). The approach distance, ρ_a, is required to exceed half of the box's larger dimension – whether width or length – by a factor of at least two.

This configuration facilitates the robots' arrangement around the object, mitigating premature collisions and setting the stage for the initiation of the pushing phase. The calculation of the caging angles is articulated as follows:

$$\theta_s^i = \text{atan2}(y_s^i - y_o, x_s^i - x_o)$$
$$\phi_s^i = \theta_s^i - \theta_b,\ i \in \{1, 2, 3, 4\},$$

where θ_s^i and ϕ_s^i represent the relative angles of the robots with respect to the inertial and object-fixed frames, respectively. The graphical representation of these angles is depicted in Fig. 17.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Algorithm 4 Pseudocode of the object manipulation algorithm.

function: Arachne Bot Approach Waypoints Sorting
inputs: ϕ_s^i for $i \in \{1, 2, 3, 4\}$
outputs: Arachne Bot Location Assignments

Let $\mathcal{M}$ [1, . . . , 4] with $\mathcal{M}$[i]=i ▷ Index array of robots, where entry i represents robot i. Initially, the robots are indexed in an arbitrary manner.
Let $\mathcal{C}$ [1, . . . , 4] with $\mathcal{C}$[i]=cos ϕ_s^i ▷ Array with relative orientations of the four robots
Sort arrays $\mathcal{C}$ and $\mathcal{M}$ in descending order based on $\mathcal{C}$, ensuring that $\mathcal{C}[i] \geq \mathcal{C}[j]$ and $\cos \phi_s^{\mathcal{M}[i]} \geq \cos \phi_s^{\mathcal{M}[j]}$ for any $i < j$. ▷ This sorting procedure orders the robots based on their relative orientation, ensuring that $\cos \phi_s^{\mathcal{M}[1]} = \max \cos \phi_s^i$ for every i in $\{1, \ldots, 4\}$.
for $i = 1{:}4$ **do** ▷ Iterate over the four robots
  switch i **do**
    case 1
      Move robot $\mathcal{M}$ [66] to position $\mathcal{L}^1$
    case 2 ▷ Check if robot $\mathcal{M}[2]$ lies on the upper or lower halfplane defined by the line going through $\vec{i}_B$ (see Fig. 16)
      if $\phi_s^{\mathcal{M}[2]} > 0$ **then**
        Move robot $\mathcal{M}$ [46] to $\mathcal{L}^2$
      else
        Move robot $\mathcal{M}$ [46] to $\mathcal{L}^4$
      end if
    case 3
      if $\phi_s^{\mathcal{M}[3]} > 0$ and $\phi_s^{\mathcal{M}[2]} < 0$ **then**
        Move robot $\mathcal{M}[3]$ to point $\mathcal{L}_2$
      else if $\phi_s^{\mathcal{M}[3]} < 0$ and $\phi_s^{\mathcal{M}[2]} > 0$ **then**
        Move robot $\mathcal{M}[3]$ to point $\mathcal{L}_4$
      else
        Move robot $\mathcal{M}[3]$ to point $\mathcal{L}_3$
      end if
    case 4
      Move robot $\mathcal{M}[4]$ to the remaining point
end for

The ensuing task of pathing is to allocate the designated approach points among the four robots, considering their initial locations, with the aim of preventing collisions. Toward this end, Algorithm 4 presents the pseudocode of a sorting algorithm designed to achieve this objective. This procedure systematically assigns waypoints to each robot according to its relative orientation, ϕ_s^i, in relation to the other robots and the object.

Let $\mathcal{O}$: $\left(\mathcal{O}_x, \mathcal{O}_y\right) \in \mathbb{R}^2$ represent the coordinates of the box's geometric center, and let $\mathcal{L}^i$: $\left(\mathcal{L}_x^i, \mathcal{L}_y^i\right) \in \mathbb{R}^2$, for $i \in \{1, \ldots, 4\}$, denote the coordinates of the four approach points to the box, all expressed in the inertia frame. Employing straightforward geometrical principles, the coordinates of the waypoints for approaching the box are determined as follows:

$$\mathcal{L}^i\colon \left(\mathcal{L}_x^i;\mathcal{L}_y^i\right) = \begin{bmatrix} \mathcal{O}_x \\ \mathcal{O}_y \end{bmatrix} + \begin{bmatrix} \rho_a \cos\left(\theta_b + \frac{\pi}{2}(i-1)\right) \\ \rho_a \sin\left(\theta_b + \frac{\pi}{2}(i-1)\right) \end{bmatrix}, \; i \in \{1, \ldots, 4\}.$$

The pathing waypoints $\mathcal{L}^i$ position the robots safely along the symmetry axes of the object, at a distance ρ_a from its geometric center $\mathcal{O}$.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

Translation.

Upon all robots reaching their designated initial approach points, the algorithm progresses to its second phase, focusing on generating waypoints to facilitate the object's translation by the robots.

The translation phase is bifurcated into two parts: *caging* and *pushing*. In the caging part, robots surround the box to form a cross-shaped enclosure, positioning themselves at distances ρ_L and ρ_W along the box's length and width symmetry axes, respectively. The caging waypoints, during translation, are

$$\mathcal{N}^i\colon \left(\mathcal{N}_x^i, \mathcal{N}_y^i\right) = \begin{bmatrix} \mathcal{O}_x \\ \mathcal{O}_y \end{bmatrix} + \begin{bmatrix} \rho_L \cos\left(\theta_b + \frac{\pi}{2}(i-1)\right) \\ \rho_L \sin\left(\theta_b + \frac{\pi}{2}(i-1)\right) \end{bmatrix}, \text{ for } i \in \{1,3\};$$
$$\mathcal{N}^i\colon \left(\mathcal{N}_x^i, \mathcal{N}_y^i\right) = \begin{bmatrix} \mathcal{O}_x \\ \mathcal{O}_y \end{bmatrix} + \begin{bmatrix} \rho_W \cos\left(\theta_b + \frac{\pi}{2}(i-1)\right) \\ \rho_W \sin\left(\theta_b + \frac{\pi}{2}(i-1)\right) \end{bmatrix}, \text{ for } i \in \{2,4\}.$$

After the caging formation is established, the robots initiate a uniform, parallel motion, moving in unison toward the same direction, to advance the box toward its designated target location (x_t, y_t). This movement is facilitated by waypoints generated from a consistent directional vector and the caging waypoints. Adherence to these pushing waypoints by the robots orchestrates the collective movement of the object toward its goal.

The directional vector has a magnitude ρ_M and an angle θ_M, which is determined by the bearing from the object's center to the target location, calculated as follows:

$$\theta_M = \text{atan2}(y_o - y_t, x_o - x_t). \tag{13}$$

The pushing waypoints, denoted as $\mathcal{M}^i$, are derived by applying the directional vector to the corresponding caging waypoints $\mathcal{N}^i$ as

$$\mathcal{M}^i\colon \left(\mathcal{M}_x^i, \mathcal{M}_y^i\right) = \begin{bmatrix} \mathcal{N}_x^i \\ \mathcal{N}_y^i \end{bmatrix} - \begin{bmatrix} \rho_M \cos\theta_M \\ \rho_M \sin\theta_M \end{bmatrix}, \text{ for } i \in \{1,\ldots,4\}. \tag{14}$$

The depiction of the caging and pushing waypoints, along with the characteristics (magnitude and bearing) of the directional vector that delineates the pushing waypoints, is presented in Fig. 18. The translation phase, encompassing both caging and pushing, persists until the object arrives at its target location. This condition is mathematically satisfied when the distance ρ_t between the object and the target is reduced to less than a threshold value ϵ_t, which signifies the permissible distance error tolerance from the target. Once this condition is satisfied, the algorithm advances to its final phase: rotation.

Rotation.

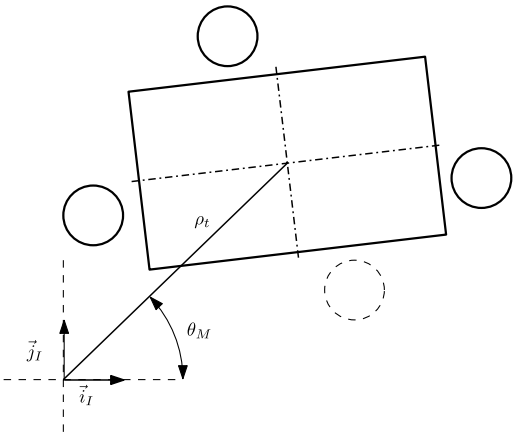
In the rotation phase, only two robots, lying in the opposing halfplanes defined by the box's length symmetry axis, participate in the object manipulation process, while the other two step back and track the box's orientation.

Assume that indexed units 2 and 4 are designated to the "long" sides of the box; they will be responsible for rotating the box. In this case, robots 1 and 3 do not participate in the manipulation process and observe the proceedings by adhering to the approaching waypoints $\mathcal{L}^1$ and $\mathcal{L}^3$.

The "wrenching" robots – 2 and 4 – are assigned waypoints, $\mathcal{W}^2$ and $\mathcal{W}^4$, located within the box's footprint (see Fig. 19). The waypoints are arranged along a line segment angularly displaced from the box's length symmetry axis by an angle θ_r, with their distance from the box's center denoted by ρ_r. To simplify, $\mathcal{W}^2$ and $\mathcal{W}^4$ are positioned on one of the box's diagonals. As a result, θ_r takes on values of either $-\pi/4$ or $\pi/4$, while ρ_r must remain below half the length of the main diagonal. This arrangement ensures that, as the robots follow these waypoints, their movement will enable a rotation aimed at minimizing the box's orientation error.

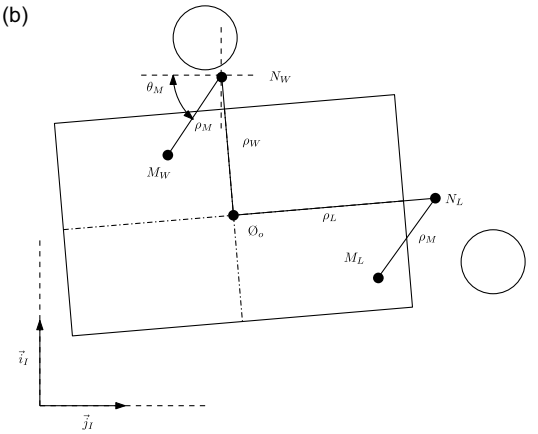
https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

(a)



Distance, ρ_t, and bearing angle, θ_M, between the object and the target. The points labeled N represent the caging waypoints, while the points labeled M correspond to the pushing waypoints. For notational simplicity, the subscripts L and W indicate robots assigned to the length and width symmetry axes of the object, respectively.

(b)



Caging and pushing waypoints for the box translation phase.

Figure 18. *Geometric configuration elements of the group object manipulation algorithm during the translation phase.*

The procedure of this phase involves comparing the angular error between the box's current orientation and its target orientation represented by the difference $\theta_b - \theta_t$. The sign of this error determines whether the box will be turned clockwise or counterclockwise. If it is positive, the offset angle θ_r is set to $\pi/4$, inducing a clockwise rotation of the box. Conversely, if the angular error is negative, θ_r is set to $-\pi/4$, resulting in a counterclockwise rotation of the box. The waypoints that enable the rotation are expressed as follows:

$$\mathcal{W}^2: \left(\mathcal{W}_x^1, \mathcal{W}_y^1\right) = \begin{bmatrix} \mathcal{O}_x \\ \mathcal{O}_y \end{bmatrix} + \begin{bmatrix} \rho_r \cos\theta_r \\ \rho_r \sin\theta_r \end{bmatrix} \tag{15}$$

$$\mathcal{W}^4: \left(\mathcal{W}_x^4, \mathcal{W}_y^4\right) = \begin{bmatrix} \mathcal{O}_x \\ \mathcal{O}_y \end{bmatrix} + \begin{bmatrix} -\rho_r \cos\theta_r \\ -\rho_r \sin\theta_r \end{bmatrix} \tag{16}$$

The waypoints are employed until the object achieves the desired angular orientation, specifically when the absolute angular error $|\theta_b - \theta_t|$ falls below a predetermined threshold ϵ_r. Should there remain

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

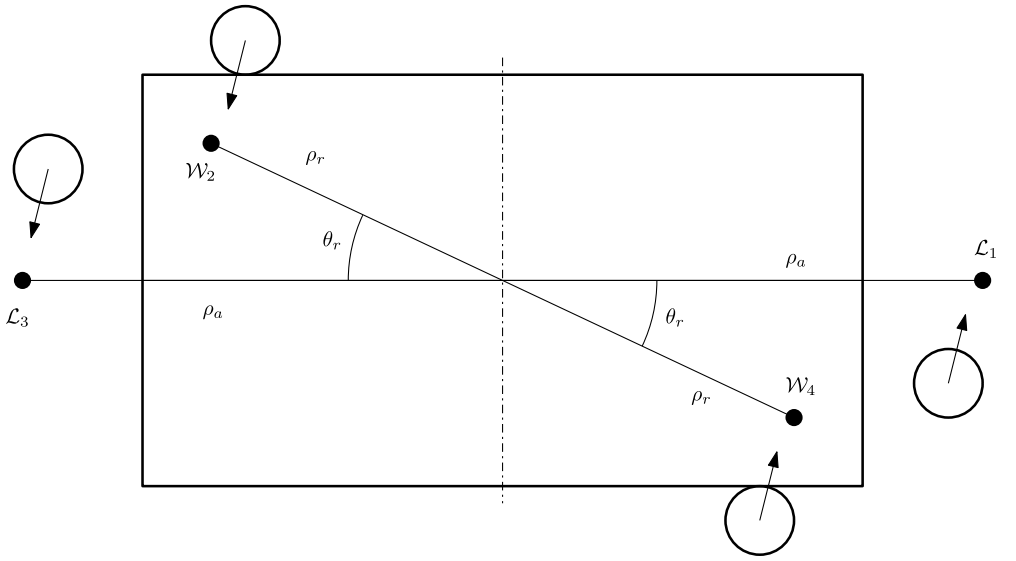


Figure 19. *Waypoint generation in the rotation phase. Two of the robots, 2 and 4, execute the box's turning maneuver, whereas robots 1 and 3 retract to a "spectator" configuration, following the box's manipulation. The wrenching waypoints lie within the area of the box.*

a positional error following the rotation phase, the process iteratively cycles through the translation and rotation phases until errors in both phases are reduced to within their respective predefined tolerance thresholds.

5. Experimental validation

This section presents a comprehensive overview and detailed description of the experiments conducted to validate the Arachne System's operation, functionality, and capabilities. Additionally, we offer several tuning guidelines and indicative coefficient values for the procedural algorithms, described in Section 4, and configuration procedures for optimizing the hardware setup.

The sequence of the algorithms tested includes set-point control, single-agent object manipulation, multi-agent object manipulation, standard collision avoidance, and infrared collision avoidance. For all experiments, raw video data were captured by an overhead camera with a 320×240 pixel resolution and analyzed at a high frequency of 100 Hz using Roborealm computer vision software. The processed video stream by Roboreal furnished the real-time coordinates of the Arachne Bots, which is essential for the execution of the majority of the task fulfillment algorithms. The unit of measurement for distance provided by RoboRealm is expressed in pixels. The camera's height affects the area represented by each pixel, thereby influencing the apparent size of objects in pixels captured in the video feed.

In evaluating the set-point control logic, described in Section 4.2, Arachne Bots were positioned at various starting points within the overhead camera's field of view. For each test, the Arachne Bot initiated its movement from locations within the inner boundary of a circle, assigned to reach the exact target location positioned at the circle's center. For this experiment, the starting positions were strategically chosen so that, for some of them, the target was positioned behind and to the aft of the robot. Figure 20 depicts a bot's posture at periodic time instances during multiple attempts of the set-point approaching assignment. This visualization underscores the Arachne Bots' adeptness in achieving concurrent advancement and rotation, a capability their unique walking mechanism facilitates. This functionality enables the bots to execute the fundamental operation of moving toward reference points in a configurable and efficient manner.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

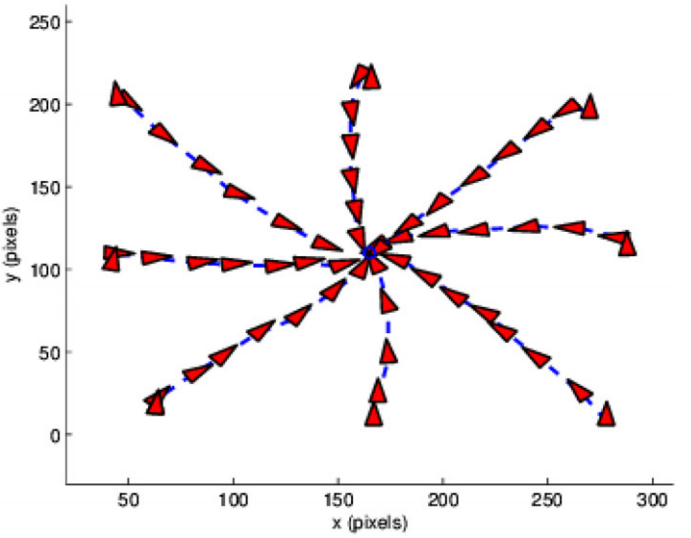


Figure 20. *Set-point reaching operation: an Arachne Bot was launched from different starting positions to converge on a predetermined target location. The feedback control law incorporates the Line of Sight (LOS) criterion, under which the robot performs a pure rotation until the target enters its LOS. Upon this condition being met, the robot progresses toward the target guided by the feedback law specified in (7).*

The disk radius R_t, asserting that the target has been reached and illustrated in Fig. 12, was fixed at 15 pixels. This radius is determined through empirical observation. The control logic includes the forward motion stipulation, as detailed in the pseudocode of Algorithm 1, which allows advancement towards the target only when it falls within the robot's LOS. The angular field of view σ, defining the extent of the LOS, was configured to 0.9 rad (51.6°). A larger σ angle results in a more curved trajectory. The two positive feedback gains, K_ρ and K_α, as specified in Eq. (7), are calibrated to ensure that the static mappings from velocities to voltages yield appropriate pulse width modulation (PWM) signals for actuating the motors of the legs and torso, respectively. In the case of our Arachne Bots, both gains were set at 127. Increasing the values of these gains results in a corresponding increase in the robot's rotational and translational speed.

Figure 20 depicts a bot's posture at periodic time instances during multiple attempts of the set-point approaching assignment. This visualization underscores the Arachne Bots' adeptness in achieving concurrent advancement and rotation, a capability their unique walking mechanism facilitates. This functionality enables the Arachne Bots to execute the fundamental operation of moving toward reference points configurable and efficiently. The subsequent algorithms evaluated are developed to create waypoints that, when adhered to, achieve specific objectives, such as object manipulation or collision avoidance. Positioned within the lower tier of the control system hierarchy, the set-point control logic is tasked with producing the linear and angular velocities required to reach the waypoints established by the "high-level" algorithm.

The next algorithm tested was the object manipulation task by a single Arachne Bot, explained in Section 4.4. The waypoint generation process for this task is outlined in Algorithm 3. For the object, we chose a plastic cylindrical hull with dimensions (height and radius) comparable to those of the Arachne Bot.

The radius of the object was experimentally evaluated to be approximately 35 pixels. The distance μ_o to the contact point is set equal to the object radius. An approximation of the robot's diameter is required to calculate certain waypoints within the algorithm, which was measured to be approximately 40 pixels. The distance μ_l, critical for defining the approach waypoints $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{I}$, is assigned based

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

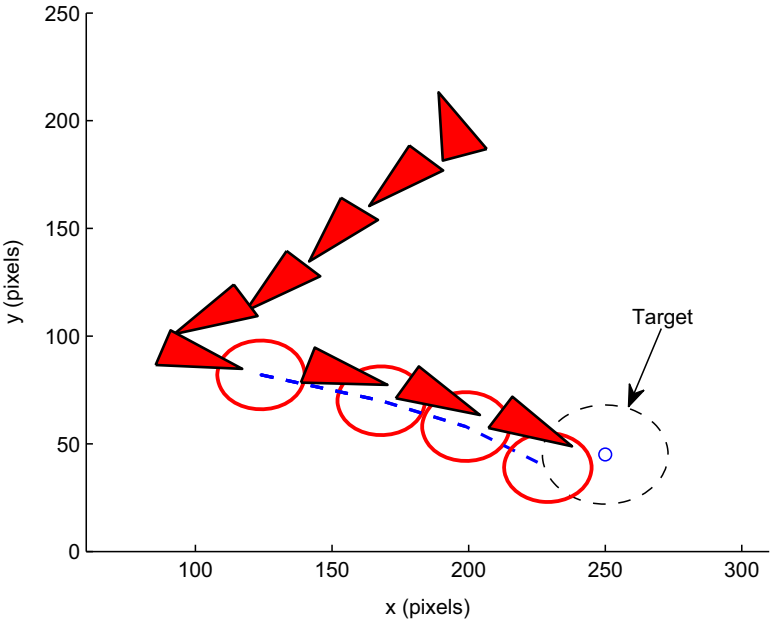


Figure 21. *Single-bot object manipulation: posture of the arachne bot and object location at intermittent time points during an experimental trial. The dashed circle marks the target location area.*

on the combined radius of the robot and the object, totaling 75 pixels. The tolerance threshold, ϵ_t, was fixed at 25 pixels.

Figure 21 demonstrates the experimental implementation of the single-bot object manipulation task. The actual and target coordinates of the object (from a distinct experimental trial) over time are depicted in Fig. 22. Despite the irregular shape of its legs interacting with the round object, the Arachne Bot successfully maneuvered the object.

The next algorithmic procedure that was validated is the box manipulation by a group of four Arachne Bots, described in Subsection 4.5. For brevity, in the subsequent description, it is implied that the index i takes values from the set $\{1, 2, 3, 4\}$. In our experiment, the dimensions of the box were (114 pixels) $\times$ (104 pixels). In our convention, we assume the box's length to be its larger dimension.

Initially, the distances ρ_L and ρ_W, which determine the caging waypoints $\mathcal{N}^i$ in the translation phase, are set to be equal to half the box's length and width, respectively, subtracting the robot's radius. For the current setup, ρ_L and ρ_W were established at 94 pixels and 84 pixels. This fitting is essential for the bots to surround the box effectively.

Reverting to the pathing phase, the distance ρ_a, defining the approach waypoints, should exceed the greater of ρ_L or ρ_W. In our experiments, ρ_a was set to $1.5\rho_L$, approximately 56 pixels. For devising the pushing waypoints $\mathcal{W}^i$, a directional vector in polar form, $\rho_M \angle \theta_M$, is appended to $\mathcal{N}^i$. This vector is intended to position the waypoints within the box for the bots executing the push, avoiding too small values that might hinder the box's forward movement by the leading bots or too large values placing waypoints outside the box. Considering the box's dimensions, ρ_M was adjusted to 22 pixels. The same value assigned to ρ_M was also used for ρ_r, denoting the distance that establishes the waypoints for the propelling bots responsible for inducing the box's rotation. The rationale was analogous: the waypoints for the wrenching bots must be within the box. Finally, the tolerance thresholds that mark the completion of the translation and rotation phases, ϵ_t and ϵ_r, were set at 25 pixels and 10° (approximately 0.174 radians), respectively.

Figure 23 illustrates nine execution snapshots acquired from data collected during an experimental trial. The execution time instance corresponding to each view is carefully chosen to ensure that the

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

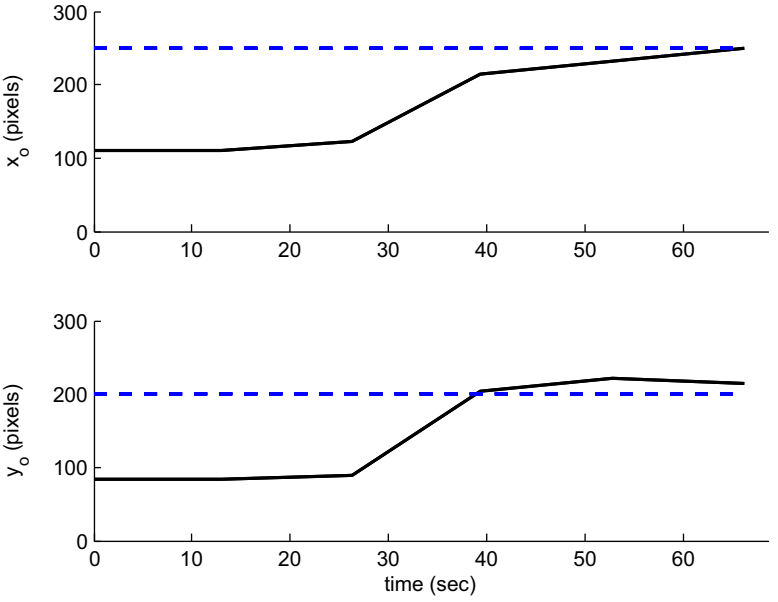


Figure 22. *The object's coordinates over time (black solid line) and its target location (dashed blue line).*

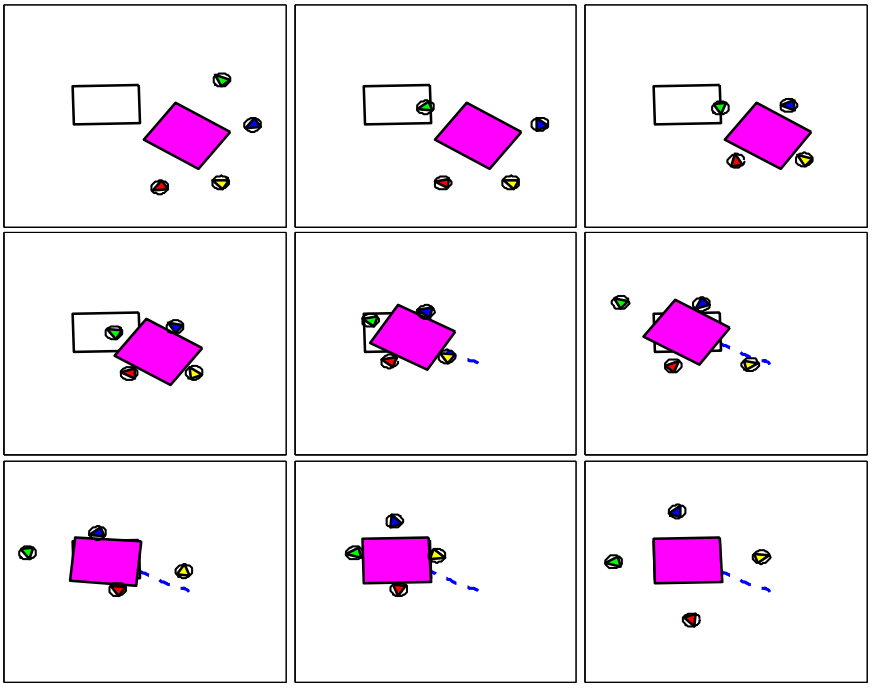


Figure 23. *Group object manipulation: each view displays the configuration of the bots and the box at selected execution time instances based on data obtained during an experimental run. The sequence of tiles is arranged to be read from top to bottom and left to right.*

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

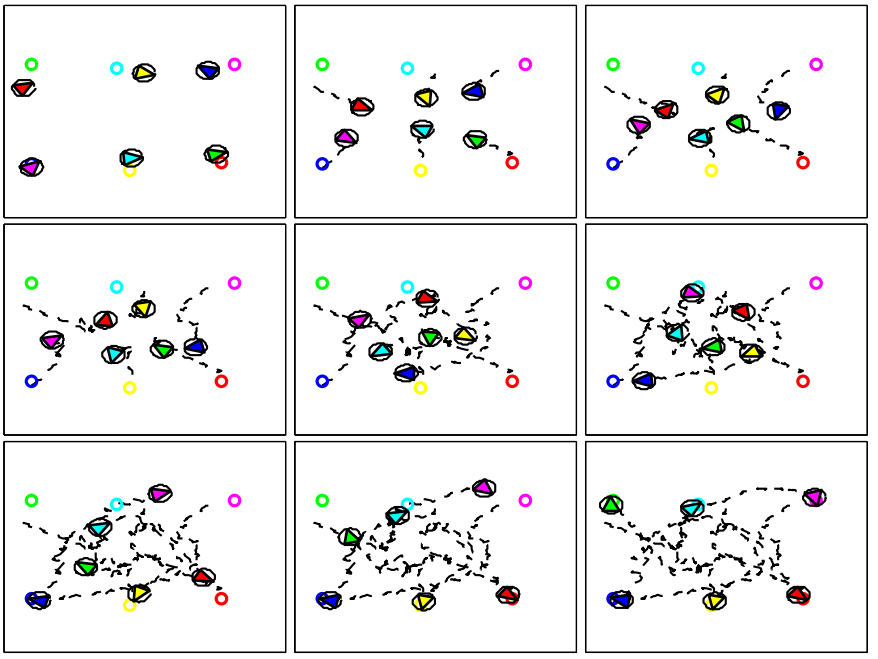


Figure 24. *Collision avoidance with the overhead camera: each tile portrays the paths and postures of six Arachne Bots at successive time instances throughout the execution of the collision avoidance algorithm in an experimental trial. The tiles are arranged to follow a viewing sequence from top to bottom and left to right.*

figure's gallery comprehensively showcases all phases of the group box manipulation task. The procedure systematically and reliably leads to the successful transportation of the box to its designated posture by the group of Arachne Bots.

The final two experimental setups are dedicated to exploring collision avoidance, utilizing two different proximity sensing techniques. Both implementations adhere to the algorithmic procedure outlined in Algorithm 2. The overhead camera system captures each bot's absolute posture (position and orientation). The key distinction between the two validation procedures lies in how the proximity between each robot and its neighboring units is detected. In the first scenario, the camera system provides this relative measurement (relative distance and bearing), whereas, in the second scenario, proximity detection is facilitated by the IRB, as detailed in Section 3. The key parameter for the collision avoidance algorithm, detailed in Section 4.3, is the collision radius, R_c, fixed at 52 pixels.

Figure 24 demonstrates the trajectories and orientations of six Arachne Bots engaged in collision avoidance over successive time intervals during an experimental trial. Despite navigating in a limited space with the clutter of six Arachne Bots, the swarm members arrived at their predetermined target locations without collisions. The effectiveness of two Arachne Bots equipped with the IRB for proximity sensing during a collision avoidance algorithm trial is demonstrated in Fig. 25. Echoing the format of previous validation results, the figure captures the bots' configurations in tiles corresponding to sequential time instances of the experiment, showcasing the IRB's capability to provide functional sensory information for the reliable and repeated completion of the collision avoidance task. With its modest cost, compact design, and simplicity, the IRB consistently proved to be a reliable sensor for collision avoidance, reaffirming its value across multiple experimental repetitions.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

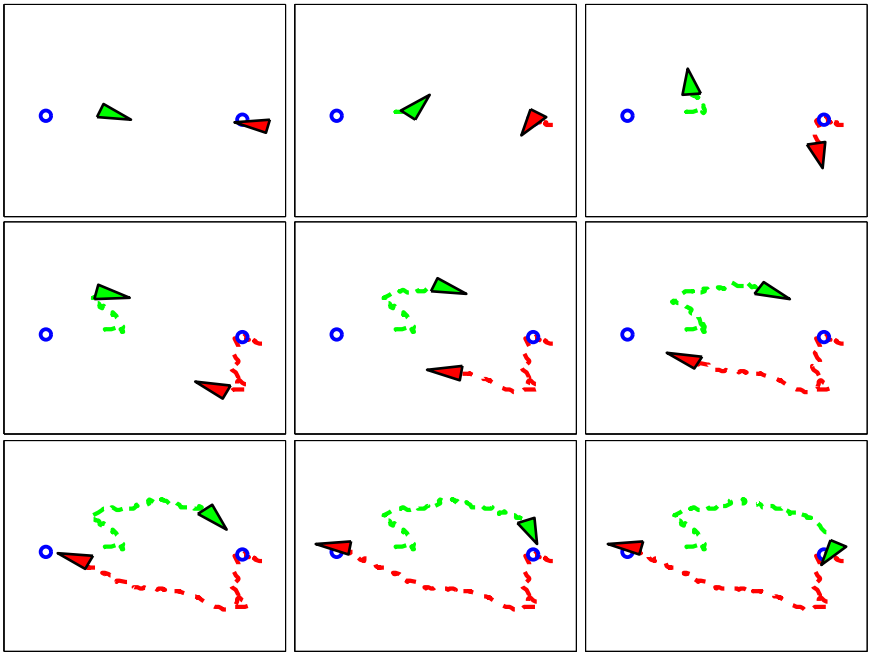


Figure 25. *Collision avoidance with the IRB: each tile portrays the paths and postures of two Arachne Bots at successive time instances throughout the execution of the collision avoidance algorithm in an experimental trial. Similar to the previous two illustration formats, the tiles are organized to follow a viewing sequence from top to bottom and left to right.*

6. Conclusion

This work introduces a new robotics platform that is both compact and inexpensive, requiring minimal time and resources for assembly. Moreover, its simplicity makes it accessible to users who may be unfamiliar with complex robotic systems. The platform's efficacy is evidenced by its application in testing various algorithms, including those for set-point control, single-agent object manipulation, and multi-agent object manipulation. Additionally, the platform's design facilitates customization, as demonstrated by the integration of an IRB. The primary goal of this new platform is to propel the field of swarm robotics forward by offering an easily reproducible, low-cost, and user-friendly system. Such a system is poised to enable researchers and educational institutions alike to engage directly with collective robotic behaviors.

Acknowledgments. The authors would like to thank David J. Willis for his invaluable help in editing the paper and insightful comments for improving it. Ioannis Raptis extends his gratitude to his beloved newborn daughter, his little spider, Alexandra Rapti (born 1/28/2024), who kept him company with her coos while writing this paper. Ioannis Raptis gratefully dedicates this work to his newborn daughter, Alexandra (born 1/28/2024) – his cherished "little spider" – whose gentle coos provided comfort throughout the writing of this paper.

Author contributions. IR conceived and designed the study, wrote and edited the article, and supervised BS in developing the Arachne Bots.

Financial support. This work was partially supported by the National Science Foundation under Grant No.DGE-2105701.

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

References

[1] L. E. Parker. Distributed Intelligence: Overview of the Field and its Application in Multi-robot Systems. **In:** *AAAI Fall Symposium: Regarding the Intelligence in Distributed Intelligent Systems* (2007) pp. 1–6.

[2] Iñaki Navarro and F. Matía, "An introduction to swarm robotics," *ISRN Robot.* **2013**, 1–10 (2013).

[3] R. A. Brooks, "Elephants don't play chess," *Robot. Auton. Syst.* **6**(1-2), 3–15 (1990).

[4] A. M. Flynn, R. A. Brooks, W. M. Wells and D. S. Barrett. The World's Largest One Cubic Inch Robot. **In:** *IEEE Micro Electro Mechanical Systems Proceedings, 'An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots'* (IEEE, 1989) pp. 98–101.

[5] A. M. Flynn, R. A. Brooks, W. M. Wells III and D. S. Barrett, "Intelligence for miniature robots," *Sens. Actuator* **20**(1-2), 187–196 (1989).

[6] A. M. Flynn, R. A. Brooks, W. M. Wells III and D. S. Barrett, Squirt: The prototypical mobile robot for autonomous graduate students (1989).

[7] J. M. Soares, I. Navarro and A. Martinoli. The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Coolbox. **In:** *Robot 2015: 2nd Iberian Robotics Conference: Advances in Robotics*, **416** (Springer, 2016) pp. 767–781.

[8] J. Gindling, A. Ioannidou, J. Loh, O. Lokkebo and A. Repenning. Legosheets: a Rule-Based Programming, Simulation and Manipulation Environment for the Lego Programmable Brick. **In:** *Proceedings of Symposium on Visual Languages* (IEEE, 1995) pp. 172–179.

[9] F. Klassner and S. D. Anderson, "Lego mindstorms: Not just for k-12 anymore," *IEEE Robot. Autom. Mag.* **10**(2), 12–18 (2003).

[10] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis and P. K. Khosla, "Heterogeneous teams of modular robots for mapping and exploration," *Auton. Robot.* **8**(3), 293–308 (2000).

[11] L. E. Navarro-Serment, R. Grabowski, C. J. J. Paredis and P. K. Khosla, "Millibots," *IEEE Robot. Autom. Mag.* **9**(4), 31–40 (2002).

[12] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal and G. S. Sukhatme. Robomote: Enabling Mobility in Sensor Networks. **In:** *Proceedings of the 4th International symposium on Information Processing in Sensor Networks* (2005) p. 55.

[13] G. T. Sibley, M. H. Rahimi and G. S. Sukhatme, "Robomote: A Tiny Mobile Robot Platform for Large-Scale Ad-Hoc Sensor Networks," **In:** *Proceedings of the 2002 IEEE International Conference On Robotics and Automation* ***2*** (IEEE, 2002) pp. 1143–1148.

[14] A. M. Agogino, J. Granderson and S. Qiu. "Sensor Validation and Fusion with Distributed Smart Dustmotes for Monitoring and Enabling Efficient Energy Use," **In:** *Proceedings of the AAAI 2002 Spring Symposium* (2002) pp. 51–57.

[15] B. Warneke, M. Last, B. Liebowitz and K. S. J. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *Computer* **34**(1), 44–51 (2001).

[16] Y.-J. Wen. *Smart Dust Sensor Mote Characterization, Validation, Fusion and Actuation. Master's Thesis* (University of California, Berkeley, 2004).

[17] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer and D. Culler, "Tinyos: An operating system for sensor networks," *Ambient intell.*, 115–148 (2005).

[18] C. Pinciroli and G. Beltrame. Buzz: An Extensible Programming Language for Heterogeneous Swarm Robotics. **In:** *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2016) pp. 3794–3800.

[19] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, D. Frederick and M. Birattari, "Argos: A modular, parallel, multi-engine simulator for multi-robot systems," *Lect. Notes Comput. Sci.* **6**(4), 271–295 (2012).

[20] M. B. McMickell, B. Goodwine and L. A. Montestruque, "Micabot: A Robotic Platform for Large-Scale Distributed Robotics," **In:** *2003 IEEE International Conference On Robotics and Automation (Cat No. 03CH37422)*, vol. 2, IEEE, 2003) pp. 1600–1605,

[21] J. Hill and D. Culler, A Wireless Embedded Sensor Architecture for System-Level Optimization. *Technical Report* (University of California at Berkeley, 2002).

[22] S. Bergbeiter, Costbots: An Off-Theshelf Platform for Distributed Robotics. *Master's Thesis* (University of California at Berkeley, 2004).

[23] S. Bergbreiter and K. S. J. Pister, "Cotsbots: An Off-the-Shelf Platform for Distributed Robotics," **In:** *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No. 03CH37453)*, vol. 2 (IEEE, 2003) pp. 1632–1637,

[24] G. Caprari and R. Siegwart. Mobile Micro-Robots Ready to Use: Alice. **In:** *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2005) pp. 3295–3300.

[25] S. C. Ergen. Zigbee/ieee 802.15. 4 Summary. *UC Berkeley, September* **10** (2004).

[26] Á. lvaro Gutiérrez, A. Campo, M. Dorigo, J. Donate, Félix Monasterio-Huelin and L. Magdalena. Open e-Puck Range & Bearing Miniaturized Board for Local Communication in Swarm Robotics. **In:** *Proceeding of the IEEE International Conference on Robotics and Automation* (2009) pp. 3111–3116.

[27] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano and A. Martinoli. The e-Puck, a Robot Designed for Education in Engineering. **In:** *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, vol. 1 (2009) pp. 59–65,

[28] S. Kernbach, "Swarmrobot. Org-Open-Hardware Microrobotic Project for Large-Scale Artificial Swarms. arXiv preprint arXiv:1110.5762 (2011).

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

[29] C. C. Loh and A. Traechtler, "Cooperative transportation of a load using nonholonomic mobile robots," *Procedia Eng.* **41**, 860–866 (2012).
[30] A. J. Lynch, Multi-Robot Behaviors with Bearing-Only Sensors and Scale-Free Coordinates. *Master's Thesis* (Rice University, 2012).
[31] J. McLurkin, J. Rykowski, M. John, Q. Kaseman and A. J. Lynch, "Using multi-robot systems for engineering education: Teaching and outreach with large numbers of an advanced, low-cost robot," *IEEE Trans. Educ.* **56**(1), 24–33 (2012).
[32] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera and R. Nagpal, "Kilobot: A low cost robot with scalable operations designed for collective behaviors," *Robot. Auton. Syst.* **62**(7), 966–975 (2014).
[33] M. Rubenstein, C. Ahler and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. **In:** *Proceeding of the IEEE International Conference on Robotics and Automation* (2012) pp. 3293–3298.
[34] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron and M. Egerstedt. The Robotarium: A Remotely Accessible Swarm Robotics Research Testbed. **In:** *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 1699–1706.
[35] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Syst. Mag.* **40**(1), 26–44 (2020).
[36] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, F. Yajun and Hoehener D. Duckietown: An Open, Inexpensive and Flexible Platform for Autonomy Education and Research. **In:** *2017 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2017) pp. 1497–1504.
[37] J. Tani, L. Paull, M. T. Zuber, D. Rus, J. How, J. Leonard and A. Censi, "Duckietown: An Innovative Way to Teach Autonomy," **In:** *Educational Robotics in the Makers Era*, vol. 1 (Springer, 2017) pp. 104–121.
[38] L. E. Beaver, A. M. I. M. Behdad Chalaki, L. Zhao, R. Zayas and A. A. Malikopoulos, "Demonstration of a time-efficient mobility system using a scaled smart city," *Veh. Syst. Dyn.* **58**(5), 787–804 (2020).
[39] B. Chalaki, L. E. Beaver, A. M. I. Mahbub, H. Bang and A. A. Malikopoulos, "A research and educational robotic testbed for real-time control of emerging mobility systems: From theory to scaled experiments [applications of control]," *IEEE Control Syst. Mag.* **42**(6), 20–34 (2022).
[40] A. Stager, L. Bhan, A. Malikopoulos and L. Zhao, "A scaled smart city for experimental validation of connected and automated vehicles," *IFAC-PapersOnLine* **51**(9), 130–135 (2018).
[41] T. Lauwers and I. Nourbakhsh. Designing the Finch: Creating a Robot Aligned to Computer Science Concepts. **In:** *Proceeding of the AAAI Symposium on Educational Advances in Artificial Intelligence* (2010).
[42] S. Wilson, R. Gameros, M. Sheely, M. Lin, K. Dover, R. Gevorkyan, M. Haberland, A. Bertozzi and S. Berman, "Pheeno, a versatile swarm robotic research and education platform," *IEEE Robot. Autom. Lett.* **1**(2), 884–891 (2016).
[43] C. H. Tran, Educational robotics: A case study with the parallax scribbler robot as a bot-mate (2010).
[44] B. Thursky and G. Gaspar, Using pololus 3pi robot in the education process (2010).
[45] N. Correll, J. Bachrach, D. Vickery and D. Rus. Ad-Hoc Wireless Network Coverage with Networked Robots that Cannot Localize. **In:** *Proceedings of the IEEE International Conference on Robotics and Automation* (2009) pp. 3878–3885.
[46] [Online]. HEXBUG. *Available at:* http://www.hexbug.com/.
[47] [Online]. VEX Robotics. *Available at:* http://www.vexrobotics.com/.
[48] [Online]. panStamp. *Available at:* http://www.panstamp.com/.
[49] [Online]. RoboRealm. *Available at:* http://www.roborealm.com/.
[50] Y. Kanayama, Y. Kimura, F. Miyazaki and T. Noguchi. A Stable Tracking Control Schemefor an Autonomous Mobile Robot. **In:** *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (1990) pp. 384–389.
[51] Z. P. Jiang, E. Lefeber and H. Nijmeijer, "Stabilization and tracking of a nonholonomic mobile robot with saturating actuators," *Syst. Control Lett.* **42**(5), 327–332 (2001).
[52] Z. P. Jiang and H. Nijmeijer, "Tracking control of mobile robots: A case study in backstepping," *Automatica* **33**, 1393–1399 (1997).
[53] C. Samson and K. Ait-Abderrahim. Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space. **In:** *Proceedings of the 1991 IEEE Conference on Robotics and Automation* (1991) pp. 1136–1141.
[54] H. Kolmanovsky and N. H. McClamroch, "Developmnets in nonholonomic control systems," *IEEE Control Syst. Mag.* **15**(6), 20–36 (1995).
[55] R. Siegwart and I. R. Nourbakhsh. *Introduction to Autonomous Mobile Robots* (MIT Press, 2004).
[56] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You and S.-R. Oh. A Stable Target-Tracking Control for Unicycle Mobile Robots. **In:** *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3 (2000) pp. 1822–1827.
[57] J. K. Kuchar and L. C. Yang, "A review of conflict detection and resolution modeling methods," *IEEE Trans. Intell. Transp.* **1**(4), 179–189 (2000).
[58] Z. Wang and M. Schwager, "Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication," *Int. J. Robot Res.* **35**(13), 1564–1586 (2016).
[59] J. Canny. A Voronoi Method for the Piano-Movers Problem. **In:** *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, vol. 2 (IEEE, 1985) pp. 530–535.
[60] J. T. Schwartz and M. Sharir, "On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Commun. Pure App. Math.* **36**(3), 345–398 (1983).

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press

[61] R. G. Brown and J. S. Jennings. A Pusher/Steerer Model for Strongly Cooperative Mobile Eobot Manipulation. **In:** *Proccedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots***3** (IEEE, 1995) pp. 562–568.
[62] G. Eoh, J. D. Jeon, J. S. Choi and B. H. Lee. Multi-Robot Cooperative Formation for Overweight Object Transportation. **In:** *2011 IEEE/SICE International Symposium on System Integration (SII)* (IEEE, 2011) pp. 726–731.
[63] C. R. Kube and H. Zhang. The Use of Perceptual Cues in Multi-robot Box-Pushing. **In:** *Proceedings of IEEE International Conference on Robotics and Automation* **3** (IEEE, 1996) pp. 2085–2090.
[64] D. Rus, B. Donald and J. Jennings. Moving Furniture with Teams of Autonomous Robots. **In:** *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots* **1**, (IEEE, 1995) pp. 235–242.
[65] S. Yamada and J.'ya Saito, "Adaptive action selection without explicit communication for multirobot box-pushing," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **31**(3), 398–404 (2001).
[66] [Online]. Arachne System repository. *Available at:* https://github.com/UMLARSL/Arachne-Bots.

Cite this article: I. Raptis and B. Sullivan (2025). "Arachne system: an accessible miniaturized mobile Multi-robot platform", Robotica **43**, 2610–2643. https://doi.org/10.1017/S0263574725101884

https://doi.org/10.1017/S0263574725101884 Published online by Cambridge University Press