2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

RASCAL: A Scalable, High-redundancy Robot for Automated Storage and Retrieval Systems

Richard Black, Marco Caballero†, Andromachi Chatzieleftheriou, Tim Deegan, Philip Heard, Freddie Hong, Russell Joyce, Sergey Legtchenko, Antony Rowstron, Adam Smith, David Sweeney, Hugh Williams

***Abstract*—Automated storage and retrieval systems (ASRS) are a key component of the modern storage industry, and are used in a wide range of applications, carrying anything from lightweight tape cartridges to entire pallets of goods. Many of these systems are under pressure to maximise the use of space by growing in height and density, but this can create challenges for the the robots that service them. In this context, we present *RASCAL*, a novel ASRS robot for small payload items in structured environments, with a focus on system-level scalability and redundancy. We describe the design objectives of *RASCAL* and how they address some of the limitations of existing robotic systems in this area, such as scalability and redundancy. We then demonstrate the viability of our design with a proof-of-concept implementation of a data centre storage media robot, and show through a series of experiments that its design, speed, accuracy, and energy efficiency are appropriate for this application.**

I. Introduction

The integration of robotics into different industries has significantly increased their productivity and efficiency, an effect that is expected to continue trending upwards [1]. A prominent example of this phenomenon is the evolution of Automated Storage and Retrieval Systems (ASRS), a collection of computer-controlled machines that move items between storage locations and fulfilment stations with varying degrees of human intervention. The increased accuracy, speed, and flexibility afforded by robotics-enabled ASRS power a wide range of industrial applications ranging from automated fulfilment centres [2] to data storage [3].

As digital economies continue to grow, the demand for efficient ASRS becomes more pressing [4]. This growing pressure places a premium on all resources, leading to ever-increasing height and density of storage racking [5] as well as the need for more agile and flexible robotic systems to service them [6]. However, existing robotics used in ASRS grapple with different limitations depending on their category (ground-based, fixed, aerial or hybrid).

Robots in *ground-based systems* [7], [8] move across shared floor space to access storage racks and perform item retrieval either directly, or at a secondary pick-up location after moving an entire rack there. The capability to navigate to any storage rack enables high flexibility of scheduling within the system, but at the expense of limiting either throughput or storage density. This is because maximising storage space is at odds with having aisles wide enough to allow robots to pass each other, otherwise congestion and reduced throughput would occur as the number of robots increases.

Fixed robotic systems [9]–[11] avoid the aforementioned congestion problems by using robots that are mounted on the storage racks themselves, or on infrastructure around them. This allows robots to handle heavier payloads at higher speeds, with access to tethered power and communications, but requires any drop-off locations to be close to the storage. Unfortunately, the partitioning that improves congestion in a shared space also significant limits the scalability and redundancy of a system, and the failure of a single robot can leave a large section of storage inaccessible until it can be repaired.

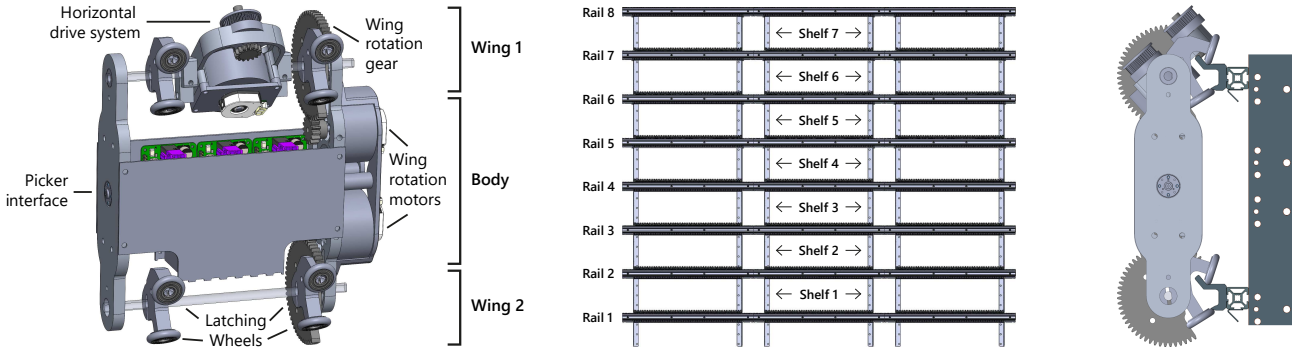
At the opposite end of the mobility spectrum we find *aerial robotic systems* [12], [13], a developing field in ASRS that seeks to fully exploit all three dimensions of a space. They offer several advantages over ground-based and fixed systems, such as the ability to operate in a very large environment, higher scalability, and lower traversal overhead. However, they face difficult challenges such as the need for high-precision navigation and manipulation systems, high power draw when flying, limited payload capacity, and complex safety considerations, especially when operating near humans.

Lastly, we consider *hybrid systems*, which combine the advantages of multiple types of ASRS while trying to minimise their disadvantages for a certain application. Examples include free-roaming wheeled robots that can transition to climbing both up and along the face of a storage rack on fixed rails, in order to access items on higher shelves [14]–[16]. One downside of such designs, however, is that the number of paths a robot can take to reach a destination can be limited by the places at which it can transition between different modes.

In the pursuit of a solution to these pressing challenges within structured environments, and given the existing limitations of the state of the art, this paper introduces *RASCAL* (*Rail-based Acrobatic System for Conveying in Automated Locations*), a novel untethered robot which aims to maximise scalability and redundancy in small payload ASRS. In the following sections, we discuss the design principles behind *RASCAL*, our experience in building a proof-of-concept implementation, and the results of a series of experiments that demonstrate its performance and viability for addressing the challenges of retrieving items from a dense storage system.

Authors are listed alphabetically, and all contributed to the work while at Microsoft Research, Cambridge, UK.

†Corresponding author: marco.caballero@microsoft.com



(a) View of robot from a storage shelf. (b) Storage racking with 7 shelves and 8 rails. (c) Profile of robot on rails.

Fig. 1: Design of *RASCAL*, a section of storage racking, and its novel rail geometry.

II. Design

RASCAL is an automated guided vehicle (AGV) designed to freely traverse the front of a vertical storage panel, and perform pick and place operations. A storage panel in this instance requires evenly spaced, parallel shelves, with a horizontal rail mounted along the front of each shelf, plus one additional rail above the top shelf. Other than this, the system requires no further physical infrastructure, and is agnostic to the type of items stored (within a certain size limit). Rail spacing is tied to the size of the robot, which must be able to span two adjacent rails. A typical deployment is likely to consist of many coordinated robots on each storage panel.

RASCAL operates in this environment using a combination of three subsystems: a *horizontal motion* system, which allows it to drive laterally along a pair of rails; a *vertical motion* system, which allows it to independently climb between shelves by pivoting the robot around one rail; and a *picker interface*, designed to accommodate multiple types of application-specific end effector, while keeping it in sync with the robot's motion. Fig. 1a shows a diagram of the main components of the robot. *RASCAL*'s high-level operations are typically monitored and controlled by a central computer that is responsible for planning the movements of all robots in the deployment, creating an autonomous system.

While we consider *RASCAL*'s design to be general enough to satisfy a wide range of storage applications, our primary motivation came from addressing the challenges of archival storage management in data centres, and the requirements of storing media generated by Project Silica [17]. Traditionally, archival storage in data centres is achieved through a combination of different media types, but relies heavily on offline magnetic tapes. As Project Silica adds a new storage medium [18], it also requires a new ASRS to manage the media library. Existing tape storage libraries comprise an array of slots filled with media which are accessed by robots that move vertically and horizontally using a fixed gantry system. While many systems use just a single robot, some libraries employ two robots for redundancy and increased throughput, however these still have strong limitations imposed by robots being unable to pass each other, and a single robot failure can limit access to a large portion of the library.

To achieve its desired goals, the design of *RASCAL* was driven by the following objectives:

- **Serviceability:** Robots should be easy to add and remove from the system, without requiring specialist tools or expertise.
- **Addressability:** Any robot should be able to access any item stored in the shelving.
- **Scalability:** A deployment should be able to scale its retrieval throughput by adding and removing robots. Likewise, it should be able to easily extend or reduce its storage capacity by adding or removing shelving without significant downtime.
- **Availability:** Failure of a robot should have a limited effect on the items that can be accessed, should not have a significant impact on routing, and should not obstruct other robots from continuing their operations.

To illustrate how our design satisfies these objectives, we start by defining the shared space on which the robots operate. *RASCAL* is designed to move horizontally and vertically along the front of storage panels, formed from an arbitrary number of contiguous storage racks like the section depicted in Fig. 1b. This racking is equipped with a set of passive (i.e. not powered) parallel rails mounted horizontally on the front at a fixed separation corresponding to the height of the robot (the rail pitch). This separation also serves as an upper bound for the size of stored objects, as they must be retrievable through the rails. As more racking is added, the rails are extended to cover the new section, creating a continuous set of rails that span the entire storage panel.

RASCAL takes advantage of a special rail geometry to latch on passively using a set of opposing wheels mounted on each end of the robot (Fig. 1c). This means the robot is held securely in place on the rails through gravity alone. The passive nature of this latching mechanism makes it very easy to add or remove robots from the rails, as it does not require any tools or power, satisfying the *serviceability* objective.

The *addressability* objective is fulfilled by the combination of *RASCAL*'s horizontal and vertical motion systems, which are enabled by the previously described rail system. These motion systems give the robot two degrees of freedom, actuated with a total of three motors, and enable it to move along and between rails. Two of these motors control

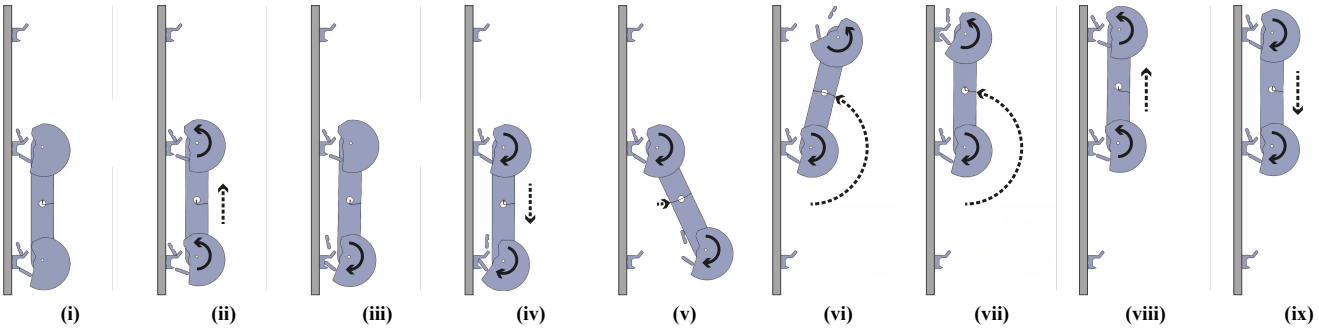


Fig. 2: Sequence of a climb-up manoeuvre between two shelves. Steps from left to right: (i) At rest, both wings latched on; (ii) Pull body upwards; (iii) Unlatch bottom wing; (iv) Lower body to stabilise top wing; (v) Swing body outwards, avoiding bottom rail; (vi-vii) Continue 180° rotation until upright; (viii) Lift body to latch on new top wing; (ix) Lower body to stable position.

the angle of the robot's 'wings' relative to the main body, which are the two rotating assemblies to which the wheels are rigidly attached (see Fig. 1a). Turning these motors in a choreographed sequence rotates the wings and produces the motion that powers *RASCAL*'s novel vertical climbing mechanism. The climbing manoeuvre (illustrated in Fig. 2) consists of unlatching one wing from its current rail while remaining firmly attached with the other; rotating the robot outwards from the storage rack around the attached rail and wing; and latching the free wing onto a new rail, two rails either above or below its original position. The third motor, mounted on one wing, is used to produce horizontal motion by turning a pinion gear that engages with teeth on the corresponding rail. These motion systems equip our robot with two key properties: independence and flexibility.

In this context, independence refers to the fact that a *RASCAL* does not depend on the state of any other robot or external motion system (such as an elevator) to perform its operations. Thus, an ASRS deployment can use as few or as many robots as it needs to satisfy its throughput requirements. This fact, coupled with the ability to arbitrarily extend the length of the storage racking horizontally or add more shelves vertically, shows our design accomplishes the *scalability* objective.

The flexibility of our motion system is better understood in the context of the limitations of other rail-guided robots. Whether operating on a vertical [14]–[16] or horizontal [19], [20] plane, rail-guided storage systems typically follow a grid pattern that only allows a robot to switch its direction of movement at specific points where rails intersect. Thus, a robot failure at a key intersection can severely limit the number of available paths between destinations. In contrast, while our design follows the aforementioned pattern for horizontal motion, vertical motion can happen at any point on a shelf (i.e. not just at intersections) resulting in many more routes to any given item. This also translates to a lower impact if a robot fails, as only items directly adjacent to the robot (behind it, and on the shelves immediately above and below it) are blocked, satisfying the last of our objectives, *availability*.

In the next section, we describe a proof-of-concept *RASCAL* that we built based on these design objectives, the architecture of the software that controls it, and the flexible picker interface that allows a variety of end-effectors to be attached to the robot for pick-and-place operations.

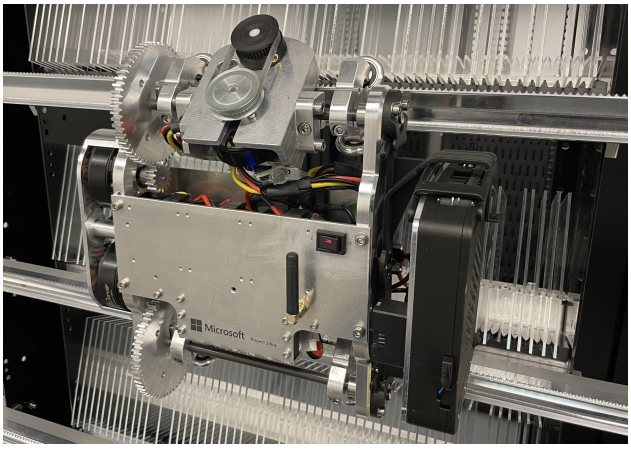


Fig. 3: A proof-of-concept *RASCAL* robot on a Silica rack.

III. Proof-of-concept implementation

The design presented in the previous section is the result of an iterative process in which we built and benchmarked several generations of *RASCAL* prototypes, along with associated rails and shelving. In this process, early prototypes aimed to test the feasibility of individual aspects of the design in isolation, such as the climbing mechanism or the horizontal motion system, while later iterations have focused on integration and refinement.

The proof-of-concept implementation seen in Fig. 3 and described in this section is the result of this process. It is heavily influenced by the specific use-case of archival storage management in data centres and is intended to demonstrate the feasibility of the design, rather than to be a production-ready system. Thus, the same design could guide the construction of a robot that uses different parts and materials for a different application, and consequently has different performance characteristics.

The proof-of-concept robot is around $240\,\mathrm{mm}$ wide, and sits $300\,\mathrm{mm}$ tall and $90\,\mathrm{mm}$ deep when mounted on the rails. The picker adds an additional $70\,\mathrm{mm}$ to the width, and increases the overall depth to around $150\,\mathrm{mm}$. When flipping, the robot extends a maximum of $270\,\mathrm{mm}$ from the structure, and $250\,\mathrm{mm}$ from the wing pivot point. The total mass of the robot (including picker and battery) is around $3.5\,\mathrm{kg}$.

The rest of the section focuses on the choices and trade-offs that our implementation makes, and how they affect the performance of the robot.

A. Localisation and sensing

The first thing a robot must do to be able to safely move in any space is to localise itself within it, a process that can be complex in a dynamic environment. In our system, this starts with initialising a robot in a specific safe region of the storage area we call the *loading bay*. The robot then tracks its own position using three independently updated levels of localisation (absolute, relative, and orientation), which are constantly cross-validated for consistency and to flag any errors. Importantly, our localisation system does not depend on any external infrastructure such as cameras and fiducial markers, which would limit scalability in deployment.

Absolute positioning is performed using radio-frequency identification (RFID) tags, which are placed at fixed intervals along the rails and read by the robot as it moves past them. Each tag has a unique identifier corresponding to a position within the panel, which the robot can use to calculate its location and correct for any drift that may have occurred. These tags also allow a robot to re-localise itself if the controller fails for any reason.

Fast *relative positioning* is the primary localisation method used by the robot, using a rotary encoder on the horizontal drive motor. This tracks the distance travelled along a rail, using the number of turns of the output pinion, which corresponds linearly with the robot's position on a shelf. Together, these two localisation layers enable millimetre precision in horizontal motion that should be sufficient for many applications. Additional precision for specific applications can be achieved by adding sensors to the picker for fine-tuning alignment in front of an object.

Finally, the robot's *orientation* is tracked using a MEMS accelerometer, which determines which wing is topmost when attached to the rails, while also serving to verify that the robot is at the correct angle during and after climbing. The accelerometer is additionally used for collision detection, allowing a robot to respond quickly to any unexpected obstacles encountered in its path.

B. Motor control and feedback

One of the biggest insights we obtained from testing early prototypes is related to motors and their control. Early iterations of the robot used a single stepper motor and worm-drive transmission to control the angle of both wings together, and an actively driven gripper on each wing to clamp onto the rails. This system makes sense in theory as it can solidly hold its position at any angle without power. In practice, however, it restricted the speed and flexibility of the climbing manoeuvre, as well as limiting the possible geometries of the robot, rails, and shelving.

In contrast, later iterations directly drive each 1:4 ratio wing spur gear with a high-torque brushless DC (BLDC) motor, allowing for fast independent control of each wing, while reducing overall cost and weight compared to a highly geared stepper motor. This increase in flexibility allowed the development of the passive latching design of the wings on the rails, at the expense of more complex control software for managing the relative wing angles and ensuring that power is never lost during a climb. Higher peak power draw is also required for these motors, which will scale with the weight and height of the robot, due to the motion of the climbing.

Horizontal movement is also performed using a BLDC motor, which simplifies the robot's firmware and electronics, and the sourcing of components. A single-turn absolute position rotary encoder is attached to each of the three BLDC motors, which provides accurate feedback to the motor controller, allowing for precise servo control of each motor's position and speed. The resulting performance of the motor control system is evaluated in Section IV.

C. Power management

An essential aspect of *RASCAL*'s design is its untethered nature, since that property is what enables the scalability and flexibility of the system. Consequently, the robot relies on being equipped with a battery capable of providing both the high peak power required by the motors when accelerating quickly or climbing, as well as sufficient run-time for the robot's operation. To this end, our proof-of-concept robot is equipped with a $1500\,\mathrm{mAh}$ six-cell lithium polymer battery, chosen for its high energy density and discharge rate. This battery provides enough capacity for the robot to run for multiple hours when idle (fully powered on but not moving).

While the current prototype is not optimised for battery life, the robot's energy usage, and therefore run-time, is a key factor in the viability of the design, so is evaluated in Section IV. While any production deployment of *RASCAL* would likely require a mechanism for onboard charging or hot-swapping of batteries, this is not implemented in the current prototype.

D. Picker interface

RASCAL is equipped with a *picker interface*, designed to allow for a variety of end-effectors that can be attached to the robot for pick-and-place operations. It is implemented as a geared servo motor with a turn range of approximately 200° to accommodate the necessary motions for climbing and pick-and-place operations, along with power and control wires to connect any additional hardware to the main robot PCB. A key property of the picker interface is that it rotates in alternating directions while climbing, which ensures the end-effector is always facing the shelving when stationary, while also preventing cables from winding up.

Our proof-of-concept shuttle is fitted with a picker to hold Project Silica [17] data storage platters, which are approximately the size of a DVD, but square, and spaced at a pitch of around $8\,\mathrm{mm}$ on the experimental shelving.

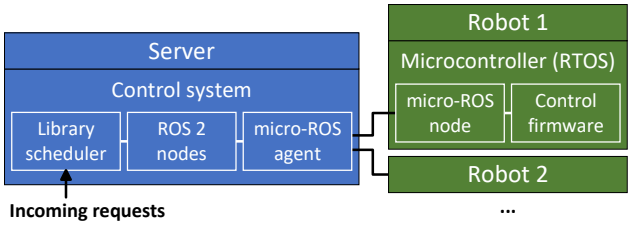


Fig. 4: Diagram of the proof-of-concept software architecture.

The proof-of-concept picker operates with a second, offset servo that turns in an opposing direction to the interface servo when picking, in order to create an approximately linear movement for extending forward to reach a shelf. Platters are then drawn into the picker (or expelled from it) using a conveyor mechanism that holds them along their top and bottom edges. Careful synchronisation of the picker's angle with the movement of the wing motors during a climb prevents collisions with the rails, despite close proximity to the storage shelves.

E. Software system

The robot's software system is implemented in a number of layers, split between microcontroller firmware on the robot, and monitoring and control software running externally (see Fig. 4). This layered approach allows for greater flexibility, as each layer can be specialised to a given deployment, and the system can scale across multiple devices. The software system and interfaces are also designed to scale to many robots, which may be controlled by one or multiple computers.

The monitoring and control software of the prototype is built using Robot Operating System 2 (ROS 2) [21], which provides useful tools and standardised interfaces to simplify development. The main robot firmware runs micro-ROS [22] on top of an RTOS, which allows each robot to exist as a first-class node in the ROS network. The robot can then communicate with other nodes using ROS 2 topics, services and actions, while maintaining the real-time guarantees required for low-level hardware control. The picker and motor controllers run on separate microcontrollers, to maintain performance and facilitate a modular construction.

System level scheduling is performed by a custom application on the control computer, which communicates with the robots over Wi-Fi. This is responsible for assigning incoming tasks to the robots, directing them along calculated paths, and monitoring their progress and health.

IV. Experiments and Evaluation

In this section we evaluate the performance of *RASCAL* through a series of experiments meant to determine the feasibility of our design, within the operational limits of our prototype. These experiments evaluate a single robot in a controlled environment, and are meant to gauge the energy cost and accuracy of its motion systems. Naturally, these measurements are not only a function of our design but also of the software implementation and hardware components, such as the motors, microcontrollers, and battery. Thus, the numbers presented here are not fundamental limits of the design, but rather a snapshot of the performance of our specific prototype.

Fig. 5: View of the experimental storage library setup, with two proof-of-concept *RASCAL* robots attached.

To facilitate this evaluation, we built a test storage panel consisting of 10 storage racks, each measuring $0.6\,\mathrm{m}$ wide by around $2\,\mathrm{m}$ tall (see Fig. 5). Each rack is fitted with 9 horizontal rails spaced $0.2\,\mathrm{m}$ apart, for a total of 8 different vertical levels (shelves) that a robot can be positioned at. This results in total traversable area of $6\,\mathrm{m}$ width by $1.6\,\mathrm{m}$ height.

A. Horizontal movement

When instructed to move horizontally, the robot calculates a trapezoidal motion profile based on the distance to the objective and requested trajectory parameters (maximum speed, acceleration, and deceleration), which it uses to drive the motor's proportional-integral (PI) controller to reach the target position. Consequently, evaluation of this system consists of measuring the robot's performance against the range of valid trajectory parameters that we found to work reliably (up to $2\,\mathrm{m\,s^{-1}}$ maximum speed, and $2\,\mathrm{m\,s^{-2}}$ acceleration and deceleration). In comparison, similar systems such as the SqUID [14], RackRacer [16], and Skypods [15] have maximum speeds of $0.55\,\mathrm{m\,s^{-1}}$, $1\,\mathrm{m\,s^{-1}}$, and $4\,\mathrm{m\,s^{-1}}$, respectively.

We performed an experimental sweep of trajectory parameters while measuring the effect on the accuracy and energy usage of the robot's motion. All three parameters were scaled together, i.e. a maximum speed of $x\,\mathrm{m\,s^{-1}}$ also corresponds to an acceleration and deceleration of $x\,\mathrm{m\,s^{-2}}$. For each trajectory, we issued the same set of 200 motion commands to the robot, with distance d sampled from a uniform distribution ($0.02\,\mathrm{m} \leq d \leq 5.46\,\mathrm{m}$), while sampling the energy consumption of the robot.

We normalise energy consumption values by the distance travelled to produce an "energy cost" metric of joules per metre of movement. These results are shown in Fig. 6, both with and without the baseline idle power draw of the robot hardware (measured at $7.05\,\mathrm{W}$) as *total energy cost* and *motion energy cost* respectively. They show us that in general, faster trajectories have a slightly higher average energy cost from driving the motor, although this does not hold for the very slowest trajectory tested. Interestingly, the

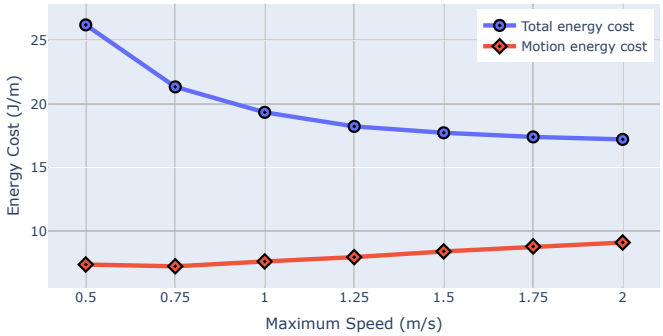


Fig. 6: Energy cost of horizontal movement at different trajectories, with/without baseline power draw. Acceleration and deceleration are scaled together with maximum speed.

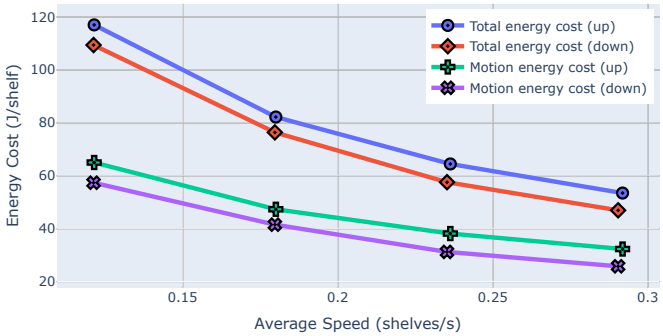


Fig. 7: Energy cost of vertical movement at different climbing speeds, with/without baseline power draw.

inverse is true when considering *total* average energy cost for a movement, which includes the baseline power draw of the robot hardware. This is because the robot spends less time performing the operation, and thus less time consuming energy. It is important to reiterate, however, that the current prototype was not optimised for energy efficiency, and the constant baseline power draw is therefore relatively high compared to the cost of using the motor.

Another insight to note from this data is that, if the robot has high utilisation, it is beneficial in terms of power to run it at high speeds. However the same would not hold if the robot will spend the rest of that time idle. Given the difference in motion energy cost is quite low, in practice the trajectory of the robot is likely to be influenced more heavily by other factors, such as the throughput and latency requirements of a workload, or the impact on hardware components.

Lastly, to determine motion accuracy, we instructed the robot to move repeatedly to the same location at different trajectories, and used a dial test indicator mounted to the rail to measure any deviation between each resulting position. From this, we found no statistical correlation between measured position offsets and trajectories, and all movements ended within $0.5\,\mathrm{mm}$ of the target.

B. Vertical movement

Similar to the horizontal experiments, we were interested in testing how climbing speed affects energy consumption with the vertical motion system. Each test sequence consisted of the robot climbing from the bottom shelf to the top shelf and back down three times at a given speed, without changing horizontal position. In this scenario, changing the speed means scaling the entire set of movements of each wing during a climb, including acceleration and deceleration. The speeds in these sweeps range from the lowest to the highest scaling at which the robot can reliably climb, without also adjusting the individual relative timing and angle values of the various wing rotations.

The results in Fig. 7 show the energy cost of climbing one shelf at different average vertical speeds, both with and without baseline power draw. Idle power draw for vertical movement (measured at $6.43\,\mathrm{W}$) is lower than its horizontal counterpart, as the former does not need to factor in the power required to hold the picker still. Contrary to intuition, the energy cost of vertical movement decreases as the speed increases, even when just considering the power draw of the motors. This is due to the robot using the largest amount of energy as it braces itself against gravity while midway through the climbing manoeuvre, and the faster the robot climbs, the less time it spends in this position. There is, however, a fundamental limit to this trend, as the robot's climbing speed is limited by the stability of the mechanism and the ability of the motor to drive the load.

Finally, it is important to compare the energy cost of vertical movement with that of horizontal movement, due to its implications for scheduling and routing in the system. Using the average motion energy cost of the fastest horizontal and vertical movement trajectories tested, the robot can move $3.19\,\mathrm{m}$ horizontally for every vertical level climbed ($2.92\,\mathrm{m}$ when including baseline power draw). Thus, an energy-aware scheduler would make use of this information (in addition to travel time and other factors) when planning missions.

V. Conclusions and Future Work

While ASRS robotics in large warehouse environments are developing at a rapid pace, there has been comparatively little innovation in systems used for small, high density payloads in structured environments, such as data centre tape library robots. We present a design for a scalable alternative to traditional gantry robot implementations, with a high level of redundancy and modularity, a focus on simplicity of the shelving, and all complexity contained in the robot itself. Through the building of several prototype iterations for a specific use case, and evaluating these with a series of experiments, we have demonstrated that the design is feasible, and that the robot is capable of performing tasks at practical speeds with a high degree of accuracy.

While we have initially validated this system with a small number of physical robots running concurrently, large-scale evaluation will more accurately inform us of our design's operating limits. Lastly, another direction to explore is with implementations for other ASRS use cases, such as the storage of different types of items, potentially with different robot size and payload requirements, or even exploring non-ASRS uses of the platform, such as a rail-based mobile sensor platform for environment monitoring.

REFERENCES

[1] D. Herr, M. Godel, R. Perkins, L. Pate, and T. Hall, "Robotics and autonomous systems: the economic impact across UK sectors, 2021," UK Department for Business, Energy & Industrial Strategy, BEIS Research Paper 2021/043, Oct. 2021. [Online]. Available: https://www.gov.uk/government/publications/robotics-and-autonomous-systems-the-economic-impact-across-uk-sectors-2021

[2] Í. R. da Costa Barros and T. P. Nascimento, "Robotic mobile fulfillment systems: A survey on recent developments and research opportunities," *Robotics and Autonomous Systems*, vol. 137, Mar. 2021.

[3] M. Mäsker, L. Nagel, T. Süß, A. Brinkmann, and L. Sorth, "Simulation and performance analysis of the ECMWF tape library system," in *Proc. International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*, Salt Lake City, UT, USA, Nov. 2016.

[4] N. Boysen, R. de Koster, and F. Weidinger, "Warehousing in the e-commerce era: A survey," *European Journal of Operational Research*, vol. 277, no. 2, pp. 396–411, Sep. 2019.

[5] J. J. Bartholdi III and S. T. Hackman, *Warehouse & Distribution Science*. Georgia Institute of Technology, Aug. 2019, release 0.98.1.

[6] T. Lamballais, D. Roy, and R. De Koster, "Estimating performance in a robotic mobile fulfillment system," *European Journal of Operational Research*, vol. 256, no. 3, pp. 976–990, Feb. 2017.

[7] Brightpick, "How to maximize order picking efficiency with autonomous mobile picking," White Paper, Feb. 2023. [Online]. Available: https://brightpick.ai/resources/white-paper-autonomous-mobile-picking/

[8] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, no. 1, pp. 9–19, Mar. 2008.

[9] Kindred. (2017) SORT: AI-powered intelligent pick robots for e-commerce automated putwall. [Online]. Available: https://www.kindred.ai/sort

[10] ——. (2022) INDUCT: Ultra high-speed, AI-driven robotic pick & place solution. [Online]. Available: https://www.kindred.ai/solutions/induct

[11] IBM Corporation. (2023) IBM TS4500 tape library. [Online]. Available: https://www.ibm.com/products/ts4500

[12] F. Betti Sorbelli, F. Corò, C. M. Pinotti, and A. Shende, "Automated picking system employing a drone," in *Proc. 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Santorini, Greece, May 2019.

[13] S. Proia, G. Cavone, A. Camposeo, F. Ceglie, R. Carli, and M. Dotoli, "Safe and ergonomic human-drone interaction in warehouses," in *Proc. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, Oct. 2022.

[14] BionicHIVE. (2023) BionicHIVE SqUID. [Online]. Available: https://bionichive.com/solution/

[15] Exotec. (2024) Skypod Robots. [Online]. Available: https://www.exotec.com/system/automated-warehouse-robots/

[16] G. Follert and W. Schroer, "Rackracer – versatile and flexible rack feeder for container racks," Product Sheet, 2014. [Online]. Available: https://www.iml.fraunhofer.de/content/dam/iml/en/documents/OE140/Produktblatt_RACKRACER_EN_2014.pdf

[17] P. Anderson *et al.*, "Project Silica: Towards sustainable cloud archival storage in glass," in *Proc. 29th ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*, Koblenz, Germany, Oct. 2023.

[18] A. Chatzieleftheriou, I. Stefanovici, D. Narayanan, B. Thomsen, and A. Rowstron, "Could cloud storage be disrupted in the next decade?" in *Proc. 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '20)*, Jul. 2020.

[19] M. Gooding. (2020, Nov.) Ocado Technology's robot warehouse a Hive of IoT innovation. [Online]. Available: https://techmonitor.ai/policy/digital-economy/ocado-technology-robot-hive-innovation

[20] AutoStore. (2024) What is AutoStore? [Online]. Available: https://www.autostoresystem.com/system

[21] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, May 2022.

[22] K. Belsare *et al.*, "Micro-ROS," in *Robot Operating System (ROS)*, ser. Studies in Computational Intelligence, A. Koubaa, Ed. Cham: Springer, Feb. 2023, vol. 1051, pp. 3–55.