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

RobotPerf: An Open-Source, Vendor-Agnostic, Benchmarking Suite for Evaluating Robotics Computing System Performance

Víctor Mayoral-Vilches[1,2], Jason Jabbour[3], Yu-Shun Hsiao[3], Zishen Wan[4], Martiño Crespo-Álvarez[1], Matthew Stewart[3], Juan Manuel Reina-Muñoz[1], Prateek Nagras[1], Gaurav Vikhe[1], Mohammad Bakhshalipour[5], Martin Pinzger[2], Stefan Rass[6,2], Smruti Panigrahi[7], Giulio Corradi[8], Niladri Roy[9], Phillip B. Gibbons[5], Sabrina M. Neuman[10], Brian Plancher[11], Vijay Janapa Reddi[3]

***Abstract*— We introduce RobotPerf, a vendor-agnostic benchmarking suite designed to evaluate robotics computing performance across a diverse range of hardware platforms using ROS 2 as its common baseline. The suite encompasses ROS 2 packages covering the full robotics pipeline and integrates two distinct benchmarking approaches: black-box testing, which measures performance by eliminating upper layers and replacing them with a test application, and grey-box testing, an application-specific measure that observes internal system states with minimal interference. Our benchmarking framework provides ready-to-use tools and is easily adaptable for the assessment of custom ROS 2 computational graphs. Drawing from the knowledge of leading robot architects and system architecture experts, RobotPerf establishes a standardized approach to robotics benchmarking. As an open-source initiative, RobotPerf remains committed to evolving with community input to advance the future of hardware-accelerated robotics.**

I. Introduction

In order for robotic systems to operate safely and effective in dynamic real-world environments, their computations must run at real-time rates while meeting power constraints. Towards this end, accelerating robotic kernels on heterogeneous hardware, such as GPUs and FPGAs, is emerging as a crucial tool for enabling such performance [1–7]. This is particularly important given the impending end of Moore's Law and the end of Dennard Scaling, which limits single CPU performance [8, 9].

While hardware-accelerated kernels offer immense potential, they necessitate a reliable and standardized infrastructure to be effectively integrated into robotic systems. As the industry leans more into adopting such standard software infrastructure, the Robot Operating System (ROS) [10] has emerged as a favored choice. Serving as an industry-grade middleware, it aids in building robust computational robotics graphs, reinforcing the idea that robotics is more than just individual algorithms. The growing dependency on ROS 2 [11], combined with the computational improvements offered by hardware acceleration, accentuates the community's demand for a standardized, industry-grade benchmark to evaluate varied hardware solutions. Recently, there has been a plethora of workshops and tutorials focusing on benchmarking robotics applications [12–22], and while benchmarks for specific robotics algorithms [23, 24] and certain end-to-end robotic applications, such as drones [25–28], do exist, the nuances of analyzing general ROS 2 computational graphs on heterogeneous hardware is yet to be fully understood.

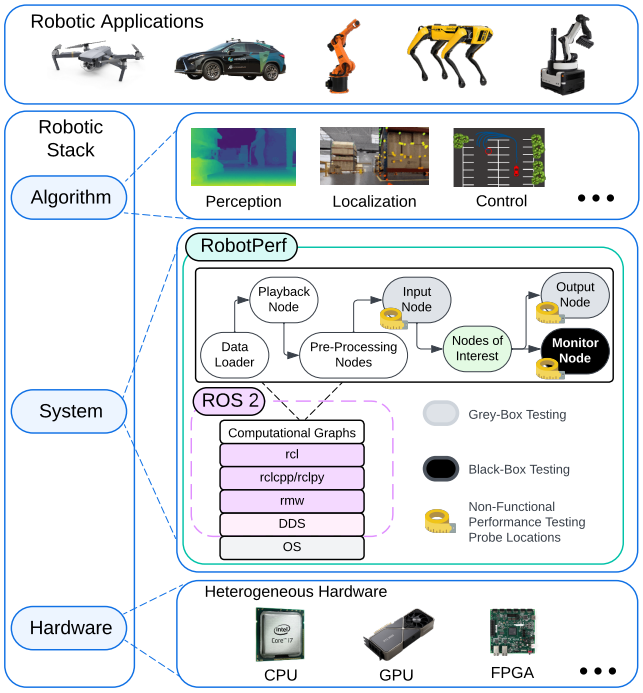

Fig. 1: A high level overview of RobotPerf. It targets industry-grade real-time systems with complex and extensible computation graphs using the Robot Operating System (ROS 2) as its common baseline. Emphasizing adaptability, portability, and a community-driven approach, RobotPerf aims to provide fair comparisons of ROS 2 computational graphs across CPUs, GPUs, FPGAs and other accelerators.

In this paper, we introduce *RobotPerf*, an open-source and community-driven benchmarking tool designed to assess the performance of robotic computing systems in a standardized, architecture-neutral, and reproducible way, accommodating the various combinations of hardware and software in different robotic platforms (see Figure 1). RobotPerf focuses on evaluating robotic workloads in the form of ROS 2 computational graphs on a wide array of hardware setups, encompassing a complete robotics pipeline and emphasizing real-time critical metrics. The framework incorporates two distinct benchmarking methodologies that utilize various

[1]Acceleration Robotics, Spain. [2]Alpen-Adria-Universität Klagenfurt, Austria. [3]Harvard University, USA. [4]Georgia Institute of Technology, USA. [5]Carnegie Mellon University, USA. [6]Johannes Kepler University Linz, Austria. [7]Ford Motor Company, USA. [8]AMD, USA. [9]Intel, USA. [10]Boston University, USA. [11]Barnard College, Columbia University, USA.

forms of instrumentation and ROS *nodes* to capture critical metrics in robotic systems. These approaches are: black-box testing, which measures performance by eliminating upper layers and replacing them with a test application, and grey-box testing, an application-specific measure that observes internal system states with minimal interference. The framework is user-friendly, easily extendable for evaluating custom ROS 2 computational graphs, and collaborates with major hardware acceleration vendors for a standardized benchmarking approach. It aims to foster research and innovation as an open-source project. We validate the framework's capabilities by conducting benchmarks on diverse hardware platforms, including CPUs, GPUs, and FPGAs, thereby showcasing RobotPerf's utility in drawing valuable performance insights.

RobotPerf's source code and documentation are made publicly available[1], and its methodologies are currently being used in the industry to benchmark industry-strength, production-grade systems.

II. Background & Related Work

A. The Robot Operating System (ROS and ROS 2)

ROS [10] is a widely-used middleware for robot development that serves as a *structured communications layer* and offers a comprehensive suite of additional functionalities including: open-source packages and drivers for various tasks, sensors, and actuators, as well as a collection of tools that simplify development, deployment, and debugging processes. ROS enables the creation of computational graphs (see Figure 1) that connect software processes, known as nodes, through topics, facilitating the development of end-to-end robotic systems. Within this framework, nodes can publish to or subscribe from topics, enhancing the modularity of robotic systems.

ROS 2 builds upon ROS and addresses many of its key limitations. Constructed to be industry-grade, ROS 2 adheres to industry Data Distribution Service (DDS) and Real-Time Publish Subscribe (RTPS) standards [29]. Based on the DDS standard, it enables fine-grained, direct, inter- and intra-node communication, enhancing performance, reducing latency, and improving scalability. Importantly, these improvements are also designed to support hardware acceleration [5, 30]. Over 600 companies have adopted ROS 2 and its predecessor ROS in their production environments, underscoring its significance and widespread adoption in the industry [11].

At the core of ROS 2's communication framework is the *rmw* (ROS Middleware) layer, which facilitates direct interaction with the DDS. Built on this foundation, the *rcl* (ROS Client Library) layer provides standardized APIs through language-specific client libraries, *rclcpp* and *rclpy*, to manage the scheduling and invocation of callbacks such as timers, subscriptions, and services. Without a ROS Master, ROS 2 creates a decentralized framework where nodes discover each other and manage their own parameters.

[1]RobotPerf's source code and documentation are available at: https://github.com/robotperf/benchmarks

TABLE I: Comparative evaluation of representative existing robotics benchmarks with RobotPerf across essential characteristics for robotic systems.

	Characteristics						
	Real-time Performance Metrics	Spans Multiple Pipeline Categories	Evaluation on Heterogeneous Hardware	Integration with ROS/ROS 2 Framework	Functional Performance Testing	Non-functional Performance Testing	Community Led
OMPL Benchmark [31]	✓	✗	✗	✗	✗	✓	✗
MotionBenchMaker [32]	✓	✗	✗	✗	✓	✓	✗
OpenCollBench [33]	✗	✗	✓	✗	✓	✗	✗
BARN [34]	✗	✗	✗	✓	✓	✗	✗
DynaBARN [35]	✓	✗	✗	✓	✓	✗	✗
MAVBench [25]	✓	✓	✓	✓	✓	✓	✗
Bench-MR [36]	✓	✗	✗	✗	✓	✗	✗
RTRBench [23]	✓	✓	✗	✗	✗	✓	✗
RobotPerf (ours)	✓	✓	✓	✓	✗	✓	✓

B. Robotics Benchmarks

There has been much recent development of open-source robotics libraries and associated benchmarks demonstrating their performance as well as a plethora of workshops and tutorials focusing on benchmarking robotics applications [12–22]. However, most of these robotics benchmarks focus on algorithm correctness (*functional* testing) in the context of domain specific problems, as well as end-to-end latency on CPUs [31–48]. A few works also analyze some *non-functional* metrics, such as CPU performance benchmarks, to explore bottleneck behaviors in selected workloads [23, 24, 49].

Recent work has also explored the implications of operating systems and task schedulers on ROS 2 computational graph performance through benchmarking [50–54] as well as by optimizing the scheduling and communication layers of ROS and ROS 2 themselves [55–62]. These works often focused on a specific context or (set of) performance counter(s).

Finally, previous work has leveraged hardware acceleration for select ROS Nodes and adaptive computing to optimize the ROS computational graphs [63–79]. However, these works do not provide comprehensive frameworks to quickly analyze and evaluate new heterogeneous computational graphs except for two works that are limited to the context of UAVs [25, 28].

Research efforts most closely related to our work include `ros2_tracing` [80] and RobotCore [5]. `ros2_tracing` provided instrumentation that demonstrated integration with the low-overhead LTTng tracer into ROS 2, while RobotCore illuminates the advantages of using vendor-specific tracing to complement `ros2_tracing` to assess the performance of hardware-accelerated ROS 2 nodes. Building on these

two specific foundational contributions, RobotPerf offers a comprehensive set of ROS 2 kernels spanning the robotics pipeline and evaluates them on diverse hardware.

Table I summarizes our unique contributions. It includes a selection of representative benchmarks from above and provides an evaluation of these benchmarks against RobotPerf, focusing on essential characteristics vital for robotic systems. We note that while our current approach focuses only on non-functional performance benchmarking tests, RobotPerf's architecture and methodology can be extended to also measure functional metrics.

III. RobotPerf: Principles & Methodology

RobotPerf is an open-source, industry-strength robotics benchmark for portability across heterogeneous hardware platforms. This section outlines the important design principles and describes the implementation methodology.

A. Non-Functional Performance Testing

Currently, RobotPerf specializes in non-functional performance testing, evaluating the efficiency and operational characteristics of robotic systems. Non-functional performance testing measures those aspects not belonging to the system's functions, such as computational latency, memory consumption, and CPU usage. In contrast, traditional functional performance testing looks into the system's specific tasks and function, verifying its effectiveness in its primary goals, like the accuracy of the control algorithm in following a planned robot's path. While functional testing confirms a system performs its designated tasks correctly, non-functional testing ensures it operates efficiently and reliably.

B. ROS 2 Integration & Adaptability

RobotPerf is designed specifically to evaluate ROS 2 computational graphs, rather than focusing on independent robotic algorithms. We emphasize benchmarking *ROS 2 workloads* because the use of ROS 2 as middleware allows for the easy composition of complex robotic systems. This makes the benchmark versatile and well-suited for a wide range of robotic applications and enables industry, which is widely using ROS, to rapidly adopt RobotPerf.

C. Platform Independence & Portability

RobotPerf allows for the evaluation of benchmarks on a variety of hardware platforms, including general-purpose CPUs and GPUs, reconfigurable FPGAs, and specialized accelerators (e.g., ray tracing accelerators [81]). Benchmarking robotic workloads on heterogeneous platforms is vital to evaluate their respective capabilities and limitations. This facilitates optimizations for efficiency, speed, and adaptability, as well as fine-tuning of resource allocations, ensuring robust and responsive operation across diverse contexts.

D. Flexible Methodology

We offer grey-box and black-box testing methods to suit different needs. Black-box testing provides a quick-to-enable external perspective and measures performance by eliminating the layers above the layer-of-interest and replacing those with a specific test application. Grey-box testing provides more granularity and dives deeper into the internal workings of ROS 2, allowing users to generate more accurate measurements at the cost of increased engineering effort. As such, each method has its trade-offs, and providing both options enables users flexibility. We describe each method in more detail below and highlight takeaways in Table II.

TABLE II: Grey-box vs. black-box benchmarking trade-offs.

Criteria	**Grey-Box**	**Black-Box**
Precision	Utilizes tracers from in-code instrumentation.	Limited to ROS 2 message subscriptions.
Performance	Low overhead. Driven by kernelspace.	Restricted to ROS 2 message callbacks. Recorded by userspace processes.
Flexibility	Multiple event types.	Limited to message subscriptions in current implementation.
Portability	Requires a valid tracer. Standard format (CTF).	Standard ROS 2 APIs. Custom JSON format.
Ease of use	Requires code modifications and data postprocessing.	Tests unmodified software with minor node additions.
Real-Robots	Does not modify the computational graph.	Modifies the computational graph adding extra dataflow.

1) Grey-Box Testing: Grey-box testing enables precise probe placement within a robot's computational graph, generating a chronologically ordered log of critical events using a tracer that could be proprietary or open source, such as LTTng [82]. As this approach is fully integrated with standard ROS 2 layers and tools through `ros2_tracing`, it incurs a minimal average latency of only 3.3 µs [80], making it well-suited for real-time systems. With this approach, optionally, RobotPerf offers specialized input and output nodes that are positioned outside the nodes of interest to avoid the need to instrument them. These nodes generate the message tracepoints upon publish and subscribe events which are processed to calculate end-to-end latency.

2) Black-Box Testing: The black-box methodology utilizes a user-level node called the `MonitorNode` to evaluate the performance of a ROS 2 node. The `MonitorNode` subscribes to the target node, recording the timestamp when each message is received. By accessing the propagated ID, the `MonitorNode` determines the end-to-end latency by comparing its timestamp against the `PlaybackNode`'s recorded timestamp for each message. While this approach does not need extra instrumentation, and is easier to implement, it offers a less detailed analysis and alters the computational graph by introducing new nodes and dataflow.

E. Opaque Performance Tests

The requirement for packages to be instrumented directly within the source code poses a challenge to many benchmarking efforts. To overcome this hurdle, for most benchmarks, we refrain from altering the workloads of interest and, instead, utilize specialized input and output nodes positioned outside the primary nodes of concern. This setup allows for benchmarking without the need for direct instrumentation of

TABLE III: RobotPerf `beta` Benchmarks.

Category	ID	Benchmark Name	Description
Perception	a1	a1_perception_2nodes	Graph with 2 components: `rectify` and `resize` [83, 84].
	a2	a2_rectify	`rectify` component [83, 84].
	a3	a3_stereo_image_proc	Computes disparity map from left and right images [85].
	a4	a4_depth_image_proc	Computes point cloud from rectified depth and color images [86].
	a5	a5_resize	`resize` component [83, 84].
Localization	b1	b1_visual_slam	Visual SLAM component [87].
	b2	b2_map_localization	Map localization component [88].
	b3	b3_apriltag_detection	Apriltag detection component [89].
Control	c1	c1_rrbot_joint_trajectory_controller	Joint trajectory controller [90].
	c2	c2_diffbot_diff_driver_controller	Differential driver controller [91].
	c3	c3_rrbot_forward_command_controller_position	Position-based forward command controller [92].
	c4	c4_rrbot_forward_command_controller_velocity	Velocity-based forward command controller [92].
	c5	c5_rrbot_forward_command_controller_acceleration	Acceleration-based forward command controller [92].
Manipulation	d1	d1_xarm6_planning_and_traj_execution	Manipulator planning and trajectory execution [93].
	d2	d2_collision_checking_fcl	Collision check: manipulator and box (FCL [94]).
	d3	d3_collision_checking_bullet	Collision check: manipulator and box (Bullet [95]).
	d4	d4_inverse_kinematics_kdl	Inverse kinematics (KDL plugin [96]).
	d5	d5_inverse_kinematics_lma	Inverse kinematics (LMA plugin [97]).
	d6	d6_direct_kinematics	Direct kinematics for manipulator [93].

the target layer. We term this methodology "opaque tests," a concept that RobotPerf adheres to when possible.

F. Reproducibility & Consistency

To ensure consistent and reproducible evaluations, RobotPerf adheres to specific common robotic data formats. In particular, it uses ROS 2 `rosbags`, incorporating both datasets developed in-house[2] and external third-party bags (e.g., the `r2b` dataset [98]).

To ensure consistent data loading and finer control over message delivery rates, we drew inspiration from [99]. Our computational graphs incorporate *modified and improved* `DataLoaderNode` and `PlaybackNode` implementations[3]. These enhanced nodes offer improvements that report worst-case latency and enable the reporting of maximum latency, introduce the ability to profile power consumption and so forth.

G. Metrics

We focus on three key metrics: latency, throughput and power consumption including energy efficiency. Latency measures the time between the start and the completion of a task. Throughput measures the total amount of work done in a given time for a task. Power measures the electrical energy per unit of time consumed while executing a given task. Measuring energy efficiency (or performance-per-Watt) captures the total amount of work (relative to either throughput or latency) that can be delivered for every watt of power consumed and is directly related to the runtime of battery powered robots [25].

[2]RobotPerf's ROS 2 rosbags available at: https://github.com/robotperf/rosbags

[3]RobotPerf's modified ros2_benchmark available at: https://github.com/robotperf/ros2_benchmark

H. Current Benchmarks and Categories

RobotPerf `beta` [100] introduces benchmarks that cover the robotics pipeline from perception, to localization, to control, as well as dedicated benchmarks for manipulation. The full list of benchmarks in the `beta` release can be found in Table III. Aligned with our principles defined above, each benchmark is a self-contained ROS 2 package which describes all dependencies (generally other ROS packages). To facilitate reproducibility, all benchmarks are designed to be built and run using the common ROS 2 development flows (`ament` build tools, `colcon` meta-build tools, etc.). Finally, so that the benchmarks can be easily consumed by other tools, a description of each benchmark, as well as its results, is defined in a machine-readable format. As such, accompanying the `package.xml` and `CMakeLists.txt` files required for all ROS packages, a YAML file named `benchmark.yaml` is in the root of each benchmark which describes the benchmark and includes accepted results.

I. Run Rules

To ensure the reliability and reproducibility of the performance data, we adhere to a stringent set of run rules. First, tests are performed in a controlled environment to ensure that performance data is not compromised by fluctuating external parameters. As per best practices recommended by `ros2_tracing` [80], we record and report settings like clock frequency and core count. Second, we look forward to the possibility of RobotPerf being embraced by the community and have results undergo peer review, which can contribute to enhancing reproducibility and accuracy. Finally, we aim to avoid overfitting to specific hardware setups or software configurations by encompassing a broad spectrum of test scenarios.

IV. Evaluation

We conduct comprehensive benchmarking using RobotPerf to evaluate its capabilities on three key aspects vital for a robotics-focused computing benchmark. First, we validate the framework's capacity to provide comparative insights across divergent heterogeneous platforms from edge devices to server-class hardware. Second, we analyze the results to understand RobotPerf's ability to guide selection of the optimal hardware solution tailored to particular robotic workloads. Finally, we assess how effectively RobotPerf reveals the advantages conferred by hardware and software acceleration techniques relative to general-purpose alternatives. All of our results and source code can be found open-source.

A. Fair and Representative Assessment of Heterogeneity

Assessing hardware heterogeneity in robotic applications is imperative in the ever-evolving field of robotics. Different robotic workloads demand varying computational resources and efficiency levels. Therefore, comprehensively evaluating performance across diverse hardware platforms is crucial.

We evaluated the RobotPerf benchmarks over a wide list of hardware platforms, including general-purpose CPUs on edge devices (e.g., Qualcomm RB5), server-class CPUs (e.g., Intel i7-8700), and specialized hardware accelerators (e.g., AMD Kria KR260). Figure 3 illustrates benchmark performance in robotics per category of workload (perception, localization, control, and manipulation) using radar plots, wherein the different hardware solutions are depicted together alongside different robotic workloads per category. Each hardware solution is presented with a different color, with smaller values and areas representing better performance in the respective category. Given our ability to benchmark 18 platforms (bottom of Figure 3), RobotPerf is capable of benchmarking heterogeneous hardware platforms and workloads, paving the way for community-driven co-design and optimization of hardware and software.

B. Quantitative Approach to Hardware Selection

The rapid evolution and diversity of tasks in robotics means we need to have a meticulous and context-specific approach to computing hardware selection and optimization. A "one-size-fits-all" hardware strategy would be an easy default selection, but it fails to capitalize on the nuanced differences in workload demands across diverse facets like perception, localization, control, and manipulation, each exhibiting distinctive sensitivities to hardware capabilities.

The results in Figure 3 demonstrate the fallacy of a "one-size-fits-all" solution. For example, focusing in on the latency radar plot for control from Figure 3 (col 3, row 1), we see that the i7-12700H (I7H) outperforms the NVIDIA AGX Orin Dev. Kit (NO) on benchmarks `C1`, `C3`, `C4`, and `C5`, but is 6.5× slower on benchmark `C2`.

This analysis indicates that each workload is unique, making it hard to generalize across both benchmarks and categories. By analyzing such data from the RobotPerf benchmarks, roboticists can better determine which hardware option best suits their needs given their specific workloads and performance requirements. Therefore, a rigorous analysis, guided by tools like RobotPerf, becomes essential to pinpoint the most effective hardware configurations that align well with individual workload requirements.

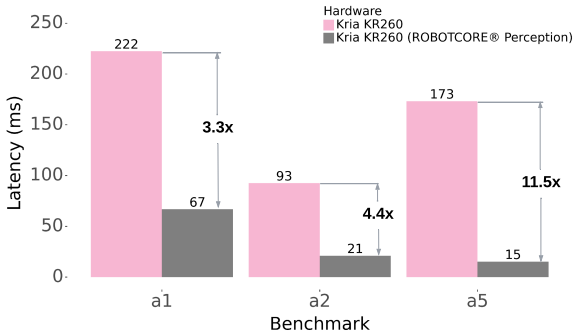

Fig. 2: Benchmark comparison of perception latency (ms) on AMD's Kria KR260 with and without the ROBOTCORE Perception accelerator. The benchmarks used are `a1`, `a2`, and `a5` as defined in Table III. We find that hardware acceleration can enable performance gains of as much as 11.5×.

C. Rigorous Assessment of Acceleration Benefits

In the rapidly advancing field of computing hardware, the optimization of algorithm implementations is a crucial factor in determining the success and efficiency of robotic applications. The need for an analytical tool, like RobotPerf, that facilitates the comparison of various algorithmic implementations on uniform hardware setups becomes important.

Figure 2 is a simplified version of Figure 3, depicting AMD's Kria KR260 hardware solution in two forms: the usual hardware and a variant that leverages a domain-specific hardware accelerator (ROBOTCORE Perception, a soft-core running in the FPGA for accelerating perception robotic computations). The figure demonstrates that hardware acceleration can enable performance gains of as much as 11.5× (from 173 ms down to 15 ms for benchmark `a5`). We stress that the results obtained here should be interpreted according to each end application and do not represent a generic recommendation on which hardware should be used. Other factors, including availability, the form factor, and community support, are relevant aspects to consider when selecting a hardware solution.

V. Conclusion and Future Work

RobotPerf represents an important step towards standardized benchmarking in robotics. With its comprehensive evaluation across the hardware/software stack and focus on industry-grade ROS 2 deployments, RobotPerf can pave the way for rigorous co-design of robotic hardware and algorithms. As RobotPerf matures with community involvement, we expect it to compare CPU, GPU and FPGA, exploring their power consumption and flexibility in augmenting real-world robotic computations. With a standardized robotics benchmark as a focal point, the field can make rapid progress in delivering real-time capable systems that will unlock the true potential of robotics in real-world applications.

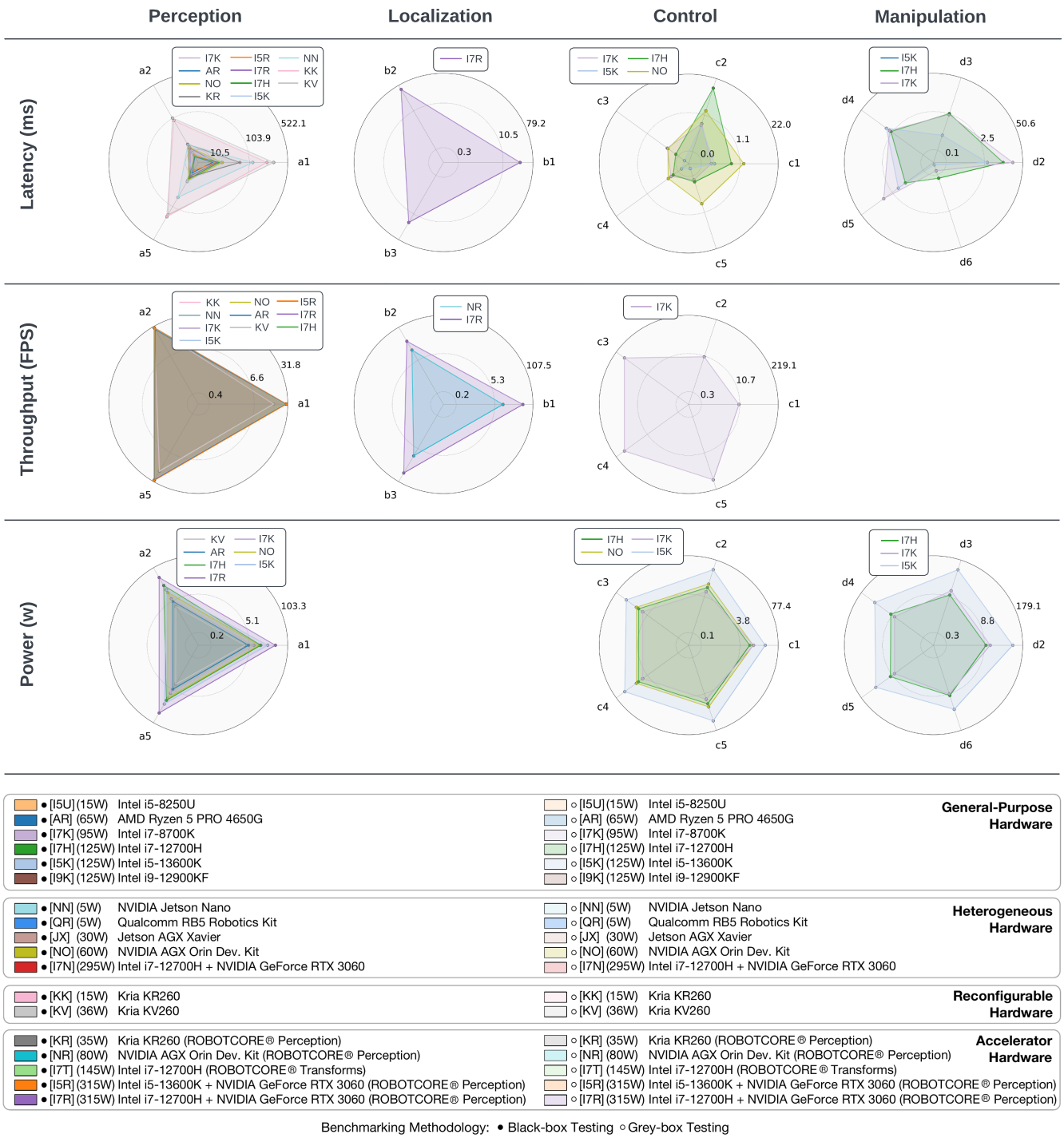

Fig. 3: Benchmarking results on diverse hardware platforms across perception, localization, control, and manipulation workloads defined in RobotPerf `beta` Benchmarks. Radar plots illustrate the latency, throughput, and power consumption for each hardware solution and workload, with reported values representing the maximum across a series of runs. The labels of vertices represent the workloads defined in Table III. Each hardware platform and performance testing procedure is delineated by a separate color, with darker colors representing Black-box testing and lighter colors Grey-box testing. In the figure's key, the hardware platforms are categorized into four specific types: general-purpose hardware, heterogeneous hardware, reconfigurable hardware, and accelerator hardware. Within each category, the platforms are ranked based on their Thermal Design Power (TDP), which indicates the maximum power they can draw under load. The throughput values for manipulation tasks and power values for localization tasks have not been incorporated into the `beta` version of RobotPerf. As RobotPerf continues to evolve, more results will be added in subsequent iterations.

References

[1] Sabrina M Neuman et al. "Robomorphic computing: a design methodology for domain-specific accelerators parameterized by robot morphology". In: *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2021, pp. 674–686.

[2] Weizhuang Liu et al. "Archytas: A framework for synthesizing and dynamically optimizing accelerators for robotic localization". In: *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*. 2021, pp. 479–493.

[3] Viktor Makoviychuk et al. "Isaac gym: High performance gpu-based physics simulation for robot learning". In: *arXiv preprint arXiv:2108.10470* (2021).

[4] Brian Plancher et al. "Grid: Gpu-accelerated rigid body dynamics with analytical gradients". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6253–6260.

[5] Víctor Mayoral-Vilches et al. "Robotcore: An open architecture for hardware acceleration in ros 2". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 9692–9699.

[6] Zishen Wan et al. "Robotic computing on fpgas: Current progress, research challenges, and opportunities". In: *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE. 2022, pp. 291–295.

[7] Shaoshan Liu et al. *Robotic computing on fpgas*. Springer, 2021.

[8] Hadi Esmaeilzadeh et al. "Dark Silicon and the End of Multicore Scaling". In: *Proceedings of the 38th Annual International Symposium on Computer Architecture*. ISCA '11. New York, NY, USA: ACM, 2011, pp. 365–376. ISBN: 978-1-4503-0472-6. DOI: 10.1145/2000064.2000108.

[9] Ganesh Venkatesh et al. "Conservation Cores: Reducing the Energy of Mature Computations". In: *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*. ASPLOS XV. New York, NY, USA: ACM, 2010, pp. 205–218. ISBN: 978-1-60558-839-1. DOI: 10.1145/1736020.1736044.

[10] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[11] Victor Mayoral-Vilches. *ros-robotics-companies*. https://github.com/vmayoral/ros-robotics-companies. [Accessed: July 9, 2023].

[12] *ICRA2021 workshop Cloud-Based Competitions and Benchmarks for Robotic Manipulation and Grasping*. June 2021. URL: https://sites.google.com/view/icra2021-workshop/home.

[13] *ICRA 2022 Workshop Determining Appropriate Metrics and Test Methods for Soft Actuators in Robotic Systems*. May 2022. URL: https://sites.google.com/andrew.cmu.edu/softactuatormetrics/.

[14] *ICRA 2022 Workshop on Releasing Robots into the Wild: Simulations, Benchmarks, and Deployment*. May 2022. URL: https://www.dynsyslab.org/releasing-robots-into-the-wild-workshop/.

[15] *IROS 2020 Workshop on Benchmarking Progress in Autonomous Driving*. Oct. 2020. URL: https://www.robotics.qmul.ac.uk/events/iros-2021-workshop/.

[16] *IROS 2021 Workshop - Benchmarking of robotic grasping and manipulation: protocols, metrics and data analysis*. Sept. 2021. URL: https://www.robotics.qmul.ac.uk/events/iros-2021-workshop/.

[17] *Evaluating Motion Planning Performance*. Oct. 2022. URL: https://motion-planning-workshop.kavrakilab.org/.

[18] *METHODS FOR OBJECTIVE COMPARISON OF RESULTS IN INTELLIGENT ROBOTICS RESEARCH*. Oct. 2023. URL: http://www.robot.t.u-tokyo.ac.jp/TCPEBRAS_IROS2023/index.html.

[19] *Benchmarking Tools for Evaluating Robotic Assembly of Small Parts*. July 2020. URL: https://www.uml.edu/research/nerve/assembly-workshop-rss-2020.aspx.

[20] *2021 RSS Workshop on Advancing Artificial Intelligence and Manipulation for Robotics: Understanding Gaps, Industry and Academic Perspectives, and Community Building*. July 2021. URL: https://sites.google.com/view/rss-ai-manipulationperspective/home.

[21] *Robot Learning in the Cloud: Remote Operations and Benchmarking*. July 2022. URL: https://sites.google.com/andrew.cmu.edu/cloud-robotics-benchmarking/.

[22] *Datasets and Benchmarking Tools for Advancing and Evaluating Robotic Manufacturing*. July 2023. URL: https://sites.google.com/view/rss-2023-nist-moad.

[23] Mohammad Bakhshalipour, Maxim Likhachev, and Phillip B Gibbons. "Rtrbench: A benchmark suite for real-time robotics". In: *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. 2022, pp. 175–186.

[24] Sabrina M Neuman et al. "Benchmarking and workload analysis of robot dynamics algorithms". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5235–5242.

[25] Behzad Boroujerdian et al. "Mavbench: Micro aerial vehicle benchmarking". In: *2018 51st annual*

IEEE/ACM international symposium on microarchitecture (MICRO). IEEE. 2018, pp. 894–907.

[26] Srivatsan Krishnan et al. "Automatic Domain-Specific SoC Design for Autonomous Unmanned Aerial Vehicles". In: *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE. 2022, pp. 300–317.

[27] Srivatsan Krishnan et al. "Roofline model for uavs: A bottleneck analysis tool for onboard compute characterization of autonomous unmanned aerial vehicles". In: *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. 2022, pp. 162–174.

[28] Dima Nikiforov et al. "RoSÉ: A Hardware-Software Co-Simulation Infrastructure Enabling Pre-Silicon Full-Stack Robotics SoC Evaluation". In: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 2023, pp. 1–15.

[29] Object Management Group (OMG). *Data Distribution Service (DDS) Interoperability Wire Protocol RTPS (Real-Time Publish-Subscribe) Protocol Specification Version 2.5*. https://www.omg.org/spec/DDSI-RTPS/2.5/. [Accessed: July 9, 2023].

[30] Víctor Mayoral-Vilches and Giulio Corradi. "Adaptive Computing in Robotics, Towards ROS 2 Software-Defined Hardware". In: *Xilinx, WP537* (2021).

[31] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. "The open motion planning library". In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82.

[32] Constantinos Chamzas et al. "MotionBenchMaker: A tool to generate and benchmark motion planning datasets". In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 882–889.

[33] Toni Tan, Rene Weller, and Gabriel Zachmann. "OpenCollBench-Benchmarking of Collision Detection & Proximity Queries as a Web-Service". In: *The 25th International Conference on 3D Web Technology*. 2020, pp. 1–9.

[34] Daniel Perille et al. "Benchmarking metric ground navigation". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2020, pp. 116–121.

[35] Anirudh Nair et al. "DynaBARN: Benchmarking Metric Ground Navigation in Dynamic Environments". In: *navigation* 7 (), p. 9.

[36] Eric Heiden et al. "Bench-MR: A motion planning benchmark for wheeled mobile robots". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4536–4543.

[37] Mark Moll, Ioan A Sucan, and Lydia E Kavraki. "Benchmarking Motion Planning Algorithms". In: ().

[38] Zachary Kingston and Lydia E Kavraki. "Robowflex: Robot motion planning with MoveIt made easy". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 3108–3114.

[39] Michael Ahn et al. "Robel: Robotics benchmarks for learning with low-cost robots". In: *Conference on robot learning*. PMLR. 2020, pp. 1300–1313.

[40] Jonathan Weisz et al. "Robobench: Towards sustainable robotics system benchmarking". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3383–3389.

[41] Angel P del Pobil, Rad Madhavan, and Elena Messina. "Benchmarks in robotics research". In: *Workshop IROS*. Citeseer. 2006.

[42] Olivier Michel, Fabien Rohrer, and Yvan Bourquin. "Rat's life: A cognitive robotics benchmark". In: *European Robotics Symposium 2008*. Springer. 2008, pp. 223–232.

[43] Adithyavairavan Murali et al. "Pyrobot: An open-source robotics framework for research and benchmarking". In: *arXiv preprint arXiv:1906.08236* (2019).

[44] Stephen James et al. "Rlbench: The robot learning benchmark & learning environment". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3019–3026.

[45] Jürgen Leitner et al. "The ACRV picking benchmark: A robotic shelf picking benchmark to foster reproducible research". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4705–4712.

[46] Yuke Zhu et al. "robosuite: A modular simulation framework and benchmark for robot learning". In: *arXiv preprint arXiv:2009.12293* (2020).

[47] Linxi Fan et al. "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark". In: *Conference on Robot Learning*. PMLR. 2018, pp. 767–782.

[48] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. "CommonRoad: Composable benchmarks for motion planning on roads". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 719–726.

[49] Jeffrey Delmerico and Davide Scaramuzza. "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 2502–2509.

[50] Michael Reke et al. "A self-driving car architecture in ROS2". In: *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE. 2020, pp. 1–6.

[51] Sinan Barut et al. "Benchmarking Real-Time Capabilities of ROS 2 and OROCOS for Robotics Applications". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 708–714.

[52] Lennart Puck et al. "Distributed and synchronized setup towards real-time robotic control using ROS2

on Linux". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2020, pp. 1287–1293.

[53] Yuqing Yang and Takuya Azumi. "Exploring Real-Time Executor on ROS 2". In: *IEEE International Conference on Embedded Software and Systems (ICESS)*. 2020, pp. 1–8.

[54] Abdullah Al Arafat et al. "Response time analysis for dynamic priority scheduling in ROS2". In: *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 2022, pp. 301–306.

[55] Yuhei Sugata et al. "Acceleration of publish/subscribe messaging in ROS-compliant FPGA component". In: *International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*. 2017, pp. 1–6.

[56] Takeshi Ohkawa et al. "High level synthesis of ROS protocol interpretation and communication circuit for FPGA". In: *IEEE/ACM International Workshop on Robotics Software Engineering (RoSE)*. 2019, pp. 33–36.

[57] Hyunjong Choi, Yecheng Xiang, and Hyoseung Kim. "PiCAS: New design of priority-driven chain-aware scheduling for ROS2". In: *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2021, pp. 251–263.

[58] Yuhei Suzuki et al. "Real-Time ROS extension on transparent CPU/GPU coordination mechanism". In: *IEEE International Symposium on Real-Time Distributed Computing (ISORC)*. 2018, pp. 184–192.

[59] Carlos San Vicente Gutiérrez et al. "Time-sensitive networking for robotics". In: *arXiv preprint arXiv:1804.07643* (2018).

[60] Carlos San Vicente Gutiérrez et al. "Real-time Linux communications: an evaluation of the Linux communication stack for real-time robotic applications". In: *arXiv preprint arXiv:1808.10821* (2018).

[61] Carlos San Vicente Gutiérrez et al. "Towards a distributed and real-time framework for robots: Evaluation of ROS 2.0 communications for real-time robotic applications". In: *arXiv preprint arXiv:1809.02595* (2018).

[62] Carlos San Vicente Gutiérrez et al. "Time Synchronization in modular collaborative robots". In: *arXiv preprint arXiv:1809.07295* (2018).

[63] Kazushi Yamashina et al. "Proposal of ROS-compliant FPGA Component for Low-Power Robotic Systems: case study on image processing application". In: *International Workshop on FPGAs for Software Programmers (FSP)* (2015).

[64] Kazushi Yamashina et al. "crecomp: Automated design tool for ros-compliant fpga component". In: *IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*. 2016, pp. 138–145.

[65] Ariel Podlubne and Diana Göhringer. "FPGA-ROS: Methodology to Augment the Robot Operating System with FPGA Designs". In: *IEEE International Conference on ReConFigurable Computing and FPGAs (ReConFig)*. 2019, pp. 1–5.

[66] Marc Eisoldt et al. "ReconfROS: Running ROS on Reconfigurable SoCs". In: *Drone Systems Engineering and Rapid Simulation and Performance Evaluation: Methods and Tools*. 2021, pp. 16–21.

[67] Christian Lienen, Marco Platzner, and Bernhard Rinner. "ReconROS: Flexible Hardware Acceleration for ROS2 Applications". In: *International Conference on Field-Programmable Technology (ICFPT)*. 2020, pp. 268–276.

[68] Daniel Pinheiro Leal et al. "Automated Integration of High-Level Synthesis FPGA Modules with ROS2 Systems". In: *International Conference on Field-Programmable Technology (ICFPT)*. 2020, pp. 292–293. DOI: 10.1109/ICFPT51103.2020.00052.

[69] Takeshi Ohkawa et al. "Architecture exploration of intelligent robot system using ros-compliant fpga component". In: *IEEE International Symposium on Rapid System Prototyping (RSP)*. 2016, pp. 1–7.

[70] Solod Panadda et al. "Low-Power High-Performance Intelligent Camera Framework ROS-FPGA Node". In: *Asia Pacific Conference on Robot IoT System Development and Platform*. 2020. 2021, pp. 73–74.

[71] J. Peña Queralta et al. "FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS". In: *IEEE SENSORS*. 2019, pp. 1–4. DOI: 10.1109/SENSORS43011.2019.8956928.

[72] Tapas Kumar Maiti. "ROS on ARM Processor Embedded with FPGA for Improvement of Robotic Computing". In: *International Symposium on Devices, Circuits and Systems (ISDCS)*. 2021, pp. 1–4. DOI: 10.1109/ISDCS52006.2021.9397897.

[73] Takeshi Ohkawa et al. "FPGA components for integrating FPGAs into robot systems". In: *IEICE Transactions on Information and Systems* 101.2 (2018), pp. 363–375.

[74] Daniel Pinheiro Leal et al. "FPGA Acceleration of ROS2-Based Reinforcement Learning Agents". In: *International Symposium on Computing and Networking Workshops*. 2020, pp. 106–112. DOI: 10.1109/CANDARW51189.2020.00031.

[75] Hayato Amano et al. "A dataset generation for object recognition and a tool for generating ROS2 FPGA node". In: *IEEE International Conference on Field-Programmable Technology (ICFPT)*. 2021, pp. 1–4.

[76] Yasuhiro Nitta, Sou Tamura, and Hideki Takase. "A study on introducing FPGA to ROS based autonomous driving system". In: *IEEE International Conference on Field-Programmable Technology (FPT)*. 2018, pp. 421–424.

[77] Kaiyuan Eric Chen et al. "FogROS: An Adaptive Framework for Automating Fog Robotics Deployment". In: *IEEE International Conference on Au-*

tomation Science and Engineering (CASE). 2021, pp. 2035–2042.

[78] NVIDIA. *NVIDIA Isaac ROS*. github.com/NVIDIA-ISAAC-ROS. Accessed 2022.

[79] Zishen Wan et al. "Analyzing and Improving Resilience and Robustness of Autonomous Systems". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[80] Christophe Bédard, Ingo Lütkebohle, and Michel Dagenais. "ros2_tracing: Multipurpose Low-Overhead Framework for Real-Time Tracing of ROS 2". In: (Accessed 2022). gitlab.com/ros-tracing/ros2_tracing.

[81] Yangdong Deng et al. "Toward real-time ray tracing: A survey on hardware acceleration and microarchitecture techniques". In: *ACM Computing Surveys (CSUR)* 50.4 (2017), pp. 1–41.

[82] Mathieu Desnoyers and Michel R Dagenais. "The lttng tracer: A low impact performance and behavior monitor for gnu/linux". In: *OLS (Ottawa Linux Symposium)*. Vol. 2006. Citeseer. 2006, pp. 209–224.

[83] ROS-Acceleration Community. *ros-acceleration*. GitHub repository. 2023. URL: https://github.com/ros-acceleration.

[84] ROS Perception Developers. *image_proc Subdirectory on Humble Branch of Image Pipeline*. GitHub Repository. 2023. URL: https://github.com/ros-perception/image_pipeline/tree/humble/image_proc.

[85] ROS Perception Developers. *stereo_image_proc Subdirectory on Humble Branch of Image Pipeline*. GitHub Repository. 2023. URL: https://github.com/ros-perception/image_pipeline/tree/humble/stereo_image_proc.

[86] ROS Perception Developers. *depth_image_proc Subdirectory on Humble Branch of Image Pipeline*. GitHub Repository. 2023. URL: https://github.com/ros-perception/image_pipeline/tree/humble/depth_image_proc.

[87] NVIDIA ISAAC ROS Developers. *isaac_ros_visual_slam*. GitHub Repository. 2023. URL: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_visual_slam.

[88] NVIDIA ISAAC ROS Developers. *isaac_ros_map_localization*. GitHub Repository. 2023. URL: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_map_localization.

[89] NVIDIA ISAAC ROS Developers. *isaac_ros_apriltag*. GitHub Repository. 2023. URL: https://github.com/NVIDIA-ISAAC-ROS/isaac_ros_apriltag.

[90] ROS Controls Developers. *joint_trajectory_controller Subdirectory in ROS 2 Controllers*. GitHub Repository. 2023. URL: https://github.com/ros-controls/ros2_controllers/tree/master/joint_trajectory_controller.

[91] ROS Controls Developers. *diff_drive_controller Subdirectory in ROS 2 Controllers*. GitHub Repository. 2023. URL: https://github.com/ros-controls/ros2_controllers/tree/master/diff_drive_controller.

[92] ROS Controls Developers. *forward_command_controller Subdirectory in ROS 2 Controllers*. GitHub Repository. 2023. URL: https://github.com/ros-controls/ros2_controllers/tree/master/forward_command_controller.

[93] MoveIt Maintainers. *MoveIt*. https://moveit.ros.org/. Official website. 2023. (Visited on 09/14/2023).

[94] Flexible Collision Library Developers. *Flexible Collision Library (FCL)*. https://github.com/flexible-collision-library/fcl. GitHub repository. 2023. (Visited on 09/14/2023).

[95] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021.

[96] MoveIt Documentation. *The KDL Kinematics Plugin*. MoveIt Documentation. Available online: https://moveit.picknik.ai/main/doc/examples/kinematics_configuration/kinematics_configuration_tutorial.html#the-kdl-kinematics-plugin. 2023.

[97] Moveit Documentation. *The LMA Kinematics Plugin*. MoveIt Documentation. Available online: https://moveit.picknik.ai/main/doc/examples/kinematics_configuration/kinematics_configuration_tutorial.html#the-lma-kinematics-plugin. 2023.

[98] Nvidia. *R2B Dataset 2023*. Apr. 2023. URL: https://catalog.ngc.nvidia.com/orgs/nvidia/teams/isaac/resources/r2bdataset2023.

[99] NVIDIA ISAAC ROS. *ROS2 Benchmark*. https://github.com/NVIDIA-ISAAC-ROS/ros2_benchmark. 2023.

[100] Robotperf. *Robotperf Benchmarks Repository*. GitHub repository directory. Year of access. URL: https://github.com/robotperf/benchmarks/tree/main/benchmarks.