Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities

Zishen Wan[1], Ashwin Lele[1], Bo Yu[2], Shaoshan Liu[2], Yu Wang[3],
Vijay Janapa Reddi[4], Cong Hao[1], and Arijit Raychowdhury[1]

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2] PerceptIn, Fremont, CA, USA
[3] Department of Electronic Engineering, Tsinghua University, Beijing, China
[4] School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA
{*zishenwan, alele9, callie.hao*}*@gatech.edu, arijit.raychowdhury@ece.gatech.edu*
{*bo.yu, shaoshan.liu*}*@perceptin.io, yu-wang@tsinghua.edu.cn, vj@eecs.harvard.edu*

***Abstract*—Robotic computing has reached a tipping point, with a myriad of robots (e.g., drones, self-driving cars, logistic robots) being widely applied in diverse scenarios. The continuous proliferation of robotics, however, critically depends on efficient computing substrates, driven by real-time requirements, robotic size-weight-and-power constraints, cybersecurity considerations, and dynamically changing scenarios. Within all platforms, FPGA is able to deliver both software and hardware solutions with low power, high performance, reconfigurability, reliability, and adaptivity characteristics, serving as the promising computing substrate for robotic applications. This paper highlights the current progress, design techniques, challenges, and open research challenges in the domain of robotic computing on FPGAs.**

I. Introduction

Robotic computing is on the rise. A myriad of robots such as drones, legged robots, and self-driving cars are on the verge of becoming an integral part of our life [1], [2]. Robotics is typically an art of system integration both in software and hardware (Fig. 1). The continuous proliferation of robots, however, face computing challenges, raised from the higher performance requirements, resource constraints, miniaturization of machine form factors, dynamic operating scenarios, and cybersecurity considerations. Therefore, it is essential to choose a proper computing substrate for robotic system that can meet real-time and power requirements and adapt to changing workloads.

CPUs and GPUs are two widely-used computing platforms, however, their performance and efficiency are still incompetent in real-time computation for complex robots. Take the motion planning task as an example, CPU typically takes a few seconds to find the collision-free trajectory [3], making it too slow for complex navigation tasks. GPUs can finish planning tasks in hundreds of milliseconds, still insufficient for many scenarios while at hundreds of watts cost [4]. ASICs are recently developed for specific robotic workloads with low power and high performance [5]–[7], but their fixed architecture has difficulty in adapting to rapid-evolving robotic algorithms and dynamic scenarios, and is vulnerable to cybersecurity threats.

As an alternative, we believe FPGA is the promising compute substrate for robotic applications. First, FPGA increases the performance with massive parallelism and deeply pipelined datapath, making it capable of meeting real-time requirements with high energy efficiency compared to CPUs and GPUs. Second, FPGA can adaptively generate custom architectures and update with the fast-evolving of robotic algorithms without going through re-fabrication as ASIC [8]. Third, FPGA is flexible in dealing with highly diverse robotic workloads, especially with partial reconfiguration allowing modification part of the operating board. Fourth, FPGA provides reliable design by leveraging reconfiguration to patch flows, compared to potential vulnerabilities detected in fixed architectures [9], which is especially essential in safety-critical scenarios [10]. Overall, FPGA has the potential to deliver high-performance, low-power, reconfigurable, adaptive, and secure features in robotic computing, and is booming in autonomous applications. However, several challenges, such as tedious development procedures, inefficient system support, and huge design space, remain in the FPGA-based robotic computing and impede the way ahead.

In this paper, we will discuss the current progress, challenges, and opportunities for FPGA-based robotic computing. Section II introduces the cross-layer stack of robotic system. Section III presents current FPGA accelerators and systems for robotic computing, with an emphasis on design techniques. Section IV discusses challenges and opportunities for FPGA-based robotic computing, and our view of the road ahead.

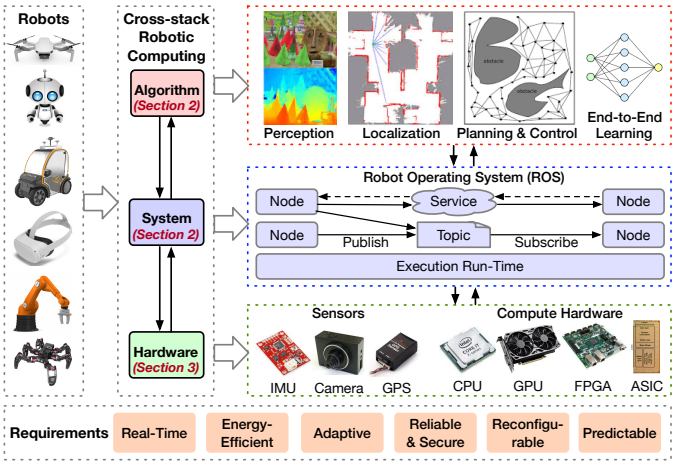
II. Cross-Layer Robotic Computing Systems

This section introduces the abstraction layers of the robotic computing stack. We traverse down Fig. 1 to explain robotic-specific algorithms and systems building blocks.

A. Robotic-Computing Algorithm Layer

Fig. 2 illustrates the representative algorithm building blocks in robotic computing, including sense-plan-act (perception, localization, planning, control) and end-to-end learning.

Perception. The goal of perception is to sense the dynamic surroundings and build a reliable and detailed representation based on sensory data (e.g., camera, IMU, GPS, LiDAR). Perception usually includes feature extraction, stereo vision, object detection, scene understanding, etc. In feature extraction, key points are usually detected using FAST feature and ORB



Challenges	Solutions & Research Opportunities *(Section 4)*
Dynamic changing workloads	① Reconfiguring robotic computing at run time
Unoptimized general solutions	② Modularizing robotic computing kernels design
Diverse hardware components	③ Mapping robotic computing on heterogeneous platforms
Inefficient ROS support	④ Connecting FPGA to ROS ecosystem
Large #algorithms and #hardware	⑤ Benchmarking robotic computing kernels
Tedious development procedure	⑥ Automating robotic computing design flow
	⑦ Building customized computing with the open-source framework
Inaccurate performance evaluation	⑧ Integrating robotic computing hardware in a simulation loop

Fig. 1: Cross-layer stack of the robotic computing system, requirements, research challenges, and open opportunities.

descriptor [11]. Compared with all image pixels, operating on feature points can improve the robustness and compute efficiency. Stereo vision is to obtain 3D structure information of the scene through disparity calculation. Local, semi-global, and global stereo matching algorithms are proposed based on operational scenarios [12]. Recently, advances in deep learning have exposed robotic perception systems to more tasks.

Localization. The goal of localization is to calculate the position and orientation of a robot itself in a given frame of reference. Knowing the position fundamentally enables robots to plan the trajectory and navigate, and knowing the orientation further helps robots stabilize. Simultaneous localization and mapping (SLAM) is a commonly-used algorithm where the robot simultaneously constructs a map of the environment while localizing itself [13], and one principled mathematical approach to solving SLAM is maximum a posteriori estimation. The filtering-based approach has recently been developed with Multi-State Kalman Filter-based algorithms such as MSCKF VIO [14] and OpenVINS [15].

Motion planning and control. The goal of motion planning is to find the optimal collision-free trajectory from the start position to the goal position, which is invoked during a robot movement to adapt to environmental changes. Motion planning is usually followed by a control module continuously tracking the differences between actual poses and poses on the pre-defined trajectory. Sampling-based solutions are widely used for motion planning, such as Probabilistic Roadmap (PRM) [16], Rapidly-exploring Random Tree (RRT) [17] and their variants, which generally contain three steps: roadmap construction, collision detection, and graph search.

End-to-End learning system. End-to-end algorithms enable skill learning directly from sensor input and perform all

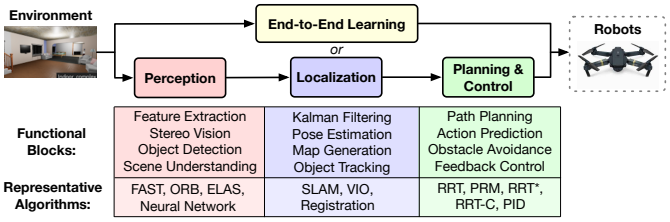


Fig. 2: Applications and algorithm building blocks in robotic systems.

the following cognitive robotic tasks using a single neural network model. Maps or separate planning stages are not required in end-to-end learning. The neural network model can be trained using reinforcement learning [18] or supervised learning [19]. The challenges of end-to-end learning include alleviating the model simulation-to-reality performance gap, designing optimal reward functions, and improving model explainability and robustness, which are actively explored.

B. Robotic-Computing System Layer

Robot Operating System (ROS). ROS is a commonly used operating system to provide tools, libraries, and package management for robotics development. It is a distributed framework of processes that enables executables to be individually designed and loosely coupled at runtime. Conceptually, the peer-to-peer network of ROS processes is called computation graph. The basic ROS computation graph includes nodes, topics, services, and masters, all of which provide data to the graph in different ways (Fig. 1). Each ROS node is a process used to perform a task. ROS nodes communicate with each other via topics or services. Topics allow one node to publish messages that multiple other nodes can subscribe. Services allow for creating a one-to-one communication between a service node and a client node. The ROS master is responsible for storing operating parameters and managing other nodes.

III. Current Progress and Design Techniques

This section presents our designs and current progress for FPGA-based robotic computing, with an emphasis on the design traits and techniques.

Perception on FPGAs. The perception typically contributes significantly to the end-to-end latency of robotic applications. Take the ORB perception module as an example, it usually accounts for 50%-80% compute latency of the whole localization scenario. To alleviate that, [20], [21] accelerate ORB-based perception on FPGA for both aerial and ground robots. The key design principles are to exploit task-level parallelisms by frame-multiplexing feature extraction, and customize on-chip memories to suit different types of data reuse. Another series of illustrative works is to accelerate stereo matching, which is the bottleneck of the stereo vision system. [22] implements local stereo matching algorithms on FPGA with characterized hardware-software partitions. [23] proposes a parallel 3D graph cut algorithm for accelerating global stereo matching, achieving 166× speedup to CPU. FP-Stereo [24] present streaming architecture and sampling-insensitive disparity algorithm on FPGA to accelerate semi-global stereo

matching. Recently, the Bayesian approach with generative probabilistic models facilitates efficient dense matching. An appealing example is iELAS [25], a hardware-friendly large-scale stereo algorithm implemented on FPGA. iELAS reforms the computational-intensive and irregular triangulation modules in a regular manner with intelligent points interpolation. Additionally, FPGA has been widely used in accelerating neural networks for robotic perception and end-to-end learning. Several techniques, including quantization, loop optimization, array partitioning, data reuse, and memory optimization, are proposed. Interested readers are pointed to [26] for details.

Localization on FPGAs. The backbone of SLAM is a complex non-linear optimization problem, bundle adjustment (BA), which consumes a significant amount of time and power. π-BA [27] designs a co-observation optimization technique to accelerate BA based on the key inspection that not all 3D points appear on all images in a BA problem. Pisces [28] co-optimizes SLAM power consumption and latency by exploiting inherent SLAM sparsity. By orchestrating sparse data, Pisces aligns correlated data and enables direct, parallel, and deterministic memory access. Going beyond point solutions, Archytas [29] presents a template hardware synthesis solution that automatically generates a SLAM accelerator given the hardware template and algorithm data-flow graph. To make the design adaptable to various environments, [30] dynamically optimizes the SLAM accelerator with an offline constructed lookup table and clock gating without sending new bitstreams to FPGA at run time. Typically, SLAM is suitable for unknown indoor environments, while Registration is used for known indoor environments and visual-inertial-odometry (VIO) functions well for outdoor environments. No single localization algorithm fits all scenarios. Interestingly, these algorithms share fundamental computation kernels amenable to matrix blocking. Eudoxus [21] implements SLAM, Registration, and VIO on FPGA by accelerating common matrix operations, with a lightweight runtime scheduler reducing variation.

Motion planning and control on FPGAs. Among the motion planning pipeline, the computation of collision detection is usually the bottleneck. Take RRT as an example, when it runs on CPU, 99% of the instructions are executed for collision detection, taking up 90% of total computation time. Recent efforts have proposed to accelerate motion planning kernels through algorithm-hardware co-design on FPGAs. [31] constructs robot-specific circuitry and architecture with roadmap pre-computation and massive path search parallelism, which is able to solve a motion planning query in 16 μs. [32] further presents a programmable dataflow architecture with a low-cost interconnection network, reducing the latency to 2.3 μs.

Several key design and optimization techniques are leveraged in FPGA-accelerated perception, localization, planning, and control. From the software aspect, hardware-friendly algorithms and data structures are proposed to promote parallelism with reduced intrinsic recursions and algorithm complexity. From the hardware aspect, robot-specific architecture, data sparsity, locality, parallelism, optimized interconnection networks, and reduced data movement contribute to high-performant and flexible motion planning design. These techniques can be generalized to other implementations, serving as a guide for future works.

Multi-robot collaboration on FPGAs. Going beyond single-robot applications, swarm robotics has been increasingly deployed in real-life scenarios where a team of robots collaboratively finish a task. Multi-robot workload typically demonstrates unique compute challenges. Several algorithm kernels may need to process the data at the same time, leading to hardware resources conflicts. Therefore, the FPGA accelerator should support multi-thread and dynamic scheduling. An intriguing example is INCAME [33], a single-core multi-robot exploration framework that supports dynamic multi-task scheduling with a virtual-instruction-based interrupt method. The perception and control tasks are assigned high priorities, while long-term decisions and optimization have low priorities. We envision that the multi-core multi-tasking FPGA accelerator will further improve the performance of the multi-robot system.

ROS on FPGAs. ROS-compliant FPGAs have recently been developed with ROS becoming increasingly common in robotics. ROS-compliant FPGAs must consider four functions: encapsulation of FPGA circuits, interface between ROS software and FPGA hardware, subscribe interface, and publish interface to ROS topic. Typically, large communication latency between ROS components is the bottleneck of offloading computing to FPGAs. [34] reduces the latency by implementing publish and subscribe messaging of ROS as hardware circuits, making the direct ROS-FPGA communication possible and efficient. Recently, [35], [36] propose tools and frameworks to offload and accelerate ROS computational graph on FPGA. However, the ecosystem of ROS on FPGAs is still in its infancy, better interface, automated tools, and whole ROS acceleration are to be developed.

IV. Research Challenges and Future Directions

This section discusses the research opportunities for FPGA-based robotic computing, and our view for road ahead (Fig. 1).

① **Reconfiguring robotic computing at run time.** Robots usually operate in highly dynamic environments, thus designing runtime-reconfigurable compute platforms is critical and can enable robots to be adaptive in various scenarios. Partial reconfiguration (PR) is a key feature of FPGA. Using PR, part of FPGA can be reconfigured at runtime without compromising the integrity of the applications running on those parts of the device that are not being reconfigured. Therefore, PR can allow various robotic computing kernels to time-share part of an FPGA, leading to high performance and energy efficiency, and making FPGA a more suitable computing platform for dynamic and complex robotic workloads.

② **Modularizing robotic computing kernels design.** The number of robotic algorithms is booming, but many algorithm variants share similar key computation blocks. It is thereby imperative to modularize the robotic computing kernel design. We can build optimized hardware acceleration blocks for these kernels as libraries or packages, while exploring their inherent

task-specific features such as sparsity, data flow, and memory access patterns. During the design phase, robotics practitioners can directly import these robotics-specific libraries and building blocks to build their FPGA design without delving into hardware engineering, which will greatly ease the design process. Modularizing the robotic algorithm design can help roboticists create custom accelerators for a kernel without hardware expertise.

③ **Mapping robotic computing on heterogeneous platforms.** One of the key technical challenges of designing robotic compute systems is to develop a suitable computer architecture, along with a software stack that allows computational flexibility. To improve the overall performance, FPGA-based System-on-Chip (SoC) solutions for robotic computing would be of the essence [8], which holistically integrates various computing technologies, including CPU, GPU, FPGA, and accelerators. The OpenCL framework can be used for programming and executing programs across heterogeneous platforms, and accelerator-level parallelism is expected to be explored [37]. By doing so, the SoCs are equipped with both software and hardware programmability, having the capability to deliver high performance, low power, adaptive, and reliable robotic computing.

④ **Connecting FPGA to ROS ecosystem.** With ROS increasingly utilized in robotics applications of all scales, robotic FPGA platforms need to be able to efficiently map ROS computational graphs on silicon. Going beyond the current work on accelerating specific ROS libraries, the inter-process and intra-process between ROS nodes also need to be accelerated [38]. It is worth noting that the hardware acceleration must be directly integrated into the ROS ecosystem to provide a seamless user experience for roboticists. A better interface between ROS and FPGA is expected to be delivered. Furthermore, through dynamically and efficiently mapping ROS to heterogeneous compute platforms, holistic hardware acceleration for robotic computing on ROS applications is expected to be achieved.

⑤ **Benchmarking robotic computing kernels.** Given the proliferation of robotic kernels and the rapid advances of hardware platforms, benchmarking these robotic algorithms and systems in a comparable, quantitative, and validatable manner is imperative. Such benchmarking comes into two folds, benchmarking a robotic algorithm across various hardware platforms, and benchmarking various robotic algorithms within the same hardware [39]. Particularly, benchmarks should consider the interactions of ROS and its computational graph. Benchmarking robotic computing will guide the robotics and hardware researchers to investigate the trade-offs in accuracy, performance, and energy efficiency of various robotic algorithms, and implement (or select) algorithms on FPGAs and other platforms in a performance-portable way.

⑥ **Automating robotic computing design flow.** Given the increasing complexity of robotic algorithms and the cross-stack nature, the development of robotic computing systems is becoming slow and tedious. Thus, building a push-button flow with robotic task requirements as input to automatically generate robotic accelerator design is critical [40]–[42]. We envision the agile framework will intelligently search the huge design space and automatically choose the optimal algorithm-hardware parameters with the help of modular kernels, benchmarking, and machine learning-assist methods. New robotic-centric electronic design automation (EDA) tool needs to be developed to convert the design to FPGA implementation. Automating the design follow will greatly facilitate the FPGA-based robotic computing development, and make FPGAs an ideal platform for fast prototyping and commercialization.

⑦ **Building customized robotic computing with the open-source framework.** The field of robotic computing is still in its infancy and fast-changing, and numerous opportunities still exist in task-specific acceleration. The open-source design framework with iteratively deployment, profiling, and optimization has recently been developed for machine learning applications [43], but it is still to be explored for robotic computing applications. Designers can build their custom specialized and optimized processors based on the RISC-V instruction set architecture (ISA). Defining and building an open-source FPGA-based RISC-V robotics-on-chip processor with open-source frameworks would considerably facilitate the design process and allow us to adapt to the rapidly changing landscape of robotic computing algorithms and accelerators.

⑧ **Integrating robotic computing hardware in a simulation loop.** The FPGA-accelerated kernels are usually part of the whole autonomy computing pipeline. The correlation among compute stages and other robotic cyber-physical components will impact the final robotic system performance and lead to inaccurate hardware evaluation [44], [45]. Thus, instead of isolated hardware development, adopting the hardware-in-the-loop (HIL) method is critical [46]. HIL requires plugging the hardware platforms into the simulation to understand how robots respond to stimuli on FPGA or other compute substrates. HIL can help designers quantify the FPGA real-time performance within the whole system and enable robust evaluation without risking real robots. Particularly, HIL can alleviate the FPGA hardware-induced gaps between training and deployment in learning-based systems. To perform faster performance evaluation at an earlier design stage, a closed-loop co-simulation framework of both FPGA architectural behavior (e.g., FireSim [47], SystemModeler [48]) and robotic environment simulator (e.g., AirSim [49]) is necessary.

The abundance of challenges raised above provides plentiful opportunities for research development at all levels. Endeavoring to solve these problems requires interdisciplinary approaches across all layers of computing stack, from algorithm and system to architecture, micro-architecture, and circuits.

V. Conclusion

Robotic computing is a rising area and critically depends on efficient, adaptive, and reliable compute substrates. This paper presents the cross-layer robotic computing stack and illustrates the current progress, along with FPGA design techniques. We conclude the paper by discussing the challenges, research opportunities, and roadmap for the next-generation FPGA-based robotic computing systems.

REFERENCES

[1] Z. Wan *et al.*, "A survey of fpga-based robotic computing," *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 48–74, 2021.

[2] S. Liu *et al.*, "Robotic computing on fpgas," *Synthesis Lectures on Computer Architecture*, vol. 16, no. 1, pp. 1–218, 2021.

[3] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 2951–2957, IEEE, 2015.

[4] J. Pan and D. Manocha, "Gpu-based parallel collision detection for fast motion planning," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 187–200, 2012.

[5] A. Suleiman *et al.*, "Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE Journal of Solid-State Circuits*, pp. 1106–1119, 2019.

[6] J.-H. Yoon and A. Raychowdhury, "Neuroslam: A 65-nm 7.25-to-8.79-tops/w mixed-signal oscillator-based slam accelerator for edge robotics," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 66–78, 2020.

[7] Z. Wan *et al.*, "Circuit and system technologies for energy-efficient edge robotics," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 275–280, IEEE, 2022.

[8] V. Mayoral-Vilches and G. Corradi, "Adaptive computing in robotics, leveraging ros 2 to enable software-defined hardware for fpgas," *arXiv preprint arXiv:2109.03276*, 2021.

[9] P. Kocher *et al.*, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1–19, 2019.

[10] Z. Wan *et al.*, "Analyzing and improving fault tolerance of learning-based navigation systems," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 841–846, IEEE, 2021.

[11] E. Rublee *et al.*, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, pp. 2564–2571, 2011.

[12] Z. Lu *et al.*, "A resource-efficient pipelined architecture for real-time semi-global stereo matching," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[13] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] K. Sun *et al.*, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[15] P. Geneva *et al.*, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4666–4672, IEEE, 2020.

[16] B. Ichter *et al.*, "Learned critical probabilistic roadmaps for robotic motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9535–9541, IEEE, 2020.

[17] S. M. LaValle *et al.*, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[18] A. Anwar and A. Raychowdhury, "Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning," *IEEE Access*, vol. 8, pp. 26549–26560, 2020.

[19] A. Loquercio *et al.*, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[20] Z. Wan *et al.*, "An energy-efficient quad-camera visual system for autonomous machines on fpga platform," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, IEEE, 2021.

[21] Y. Gan *et al.*, "Eudoxus: Characterizing and accelerating localization in autonomous machines industry track paper," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 827–840, IEEE, 2021.

[22] S. Perri *et al.*, "Stereo vision architecture for heterogeneous systems-on-chip," *Journal of Real-Time Image Processing*, pp. 393–415, 2020.

[23] R. Kamasaka *et al.*, "An fpga-oriented graph cut algorithm for accelerating stereo vision," in *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–6, IEEE, 2018.

[24] J. Zhao *et al.*, "Fp-stereo: Hardware-efficient stereo vision for embedded applications," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 269–276, IEEE, 2020.

[25] T. Gao *et al.*, "Ielas: An elas-based energy-efficient accelerator for real-time stereo matching on fpga platform," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, IEEE, 2021.

[26] K. Abdelouahab *et al.*, "Accelerating cnn inference on fpgas: A survey," *arXiv preprint arXiv:1806.01683*, 2018.

[27] Q. Liu *et al.*, "π-ba: Bundle adjustment hardware accelerator based on distribution of 3d-point observations," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 1083–1095, 2020.

[28] B. Asgari *et al.*, "Pisces: power-aware implementation of slam by customizing efficient sparse algebra," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2020.

[29] W. 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*, pp. 479–493, 2021.

[30] Q. Liu *et al.*, "An energy-efficient and runtime-reconfigurable fpga-based accelerator for robotic localization systems," in *2022 IEEE Custom Integrated Circuits Conference (CICC)*, IEEE, 2022.

[31] S. Murray *et al.*, "The microarchitecture of a real-time robot motion planning accelerator," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1–12, IEEE, 2016.

[32] S. Murray *et al.*, "A programmable architecture for robot motion planning acceleration," in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, vol. 2160, pp. 185–188, IEEE, 2019.

[33] J. Yu *et al.*, "Incame: Interruptible cnn accelerator for multi-robot exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[34] Y. Sugata *et al.*, "Acceleration of publish/subscribe messaging in ros-compliant fpga component," in *Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, pp. 1–6, 2017.

[35] D. P. Leal *et al.*, "Automated integration of high-level synthesis fpga modules with ros2 systems," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, pp. 292–293, IEEE, 2020.

[36] C. Lienen *et al.*, "Reconros: Flexible hardware acceleration for ros2 applications," in *2020 International Conference on Field-Programmable Technology (ICFPT)*, pp. 268–276, IEEE, 2020.

[37] M. D. Hill and V. J. Reddi, "Accelerator-level parallelism," *Communications of the ACM*, vol. 64, no. 12, pp. 36–38, 2021.

[38] C. Lienen and M. Platzner, "Reconros executor: Event-driven programming of fpga-accelerated ros 2 applications," *arXiv preprint arXiv:2201.07454*, 2022.

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

[40] S. Krishnan *et al.*, "Autopilot: Automating soc design space exploration for swap constrained autonomous uavs," *arXiv preprint arXiv:2102.02988*, 2021.

[41] S. M. Neuman *et al.*, "Robomorphic computing: a design methodology for domain-specific accelerators parameterized by robot morphology," in *26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 674–686, 2021.

[42] S. Krishnan *et al.*, "Autosoc: Automating algorithm-soc co-design for aerial robots," *arXiv preprint arXiv:2109.05683*, 2021.

[43] S. Prakash *et al.*, "Cfu playground: Full-stack open-source framework for tiny machine learning (tinyml) acceleration on fpgas," *arXiv preprint arXiv:2201.01863*, 2022.

[44] S. Krishnan *et al.*, "The sky is not the limit: A visual performance model for cyber-physical co-design in autonomous machines," *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 38–42, 2020.

[45] S. 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.

[46] B. Boroujerdian *et al.*, "Mavbench: Micro aerial vehicle benchmarking," in *2018 51st annual IEEE/ACM international symposium on microarchitecture (MICRO)*, pp. 894–907, IEEE, 2018.

[47] S. Karandikar *et al.*, "Firesim: Fpga-accelerated cycle-exact scale-out system simulation in the public cloud," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 29–42, IEEE, 2018.

[48] M. Acevedo, "Fpga-based hardware-in-the-loop co-simulator platform for systemmodeler," 2016.

[49] S. Shah *et al.*, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, pp. 621–635, Springer, 2018.