Towards Factory-Scale Edge Robotic Systems: Challenges and Research Directions

Amit Baxi, Mark Eisen, Susruth Sudhakaran, Fabian Oboril, Girish S. Murthy, Vincent S Mageshkumar, Michael Paulitsch, and Margaret Huang

Abstract

Mobile multi-robot systems are an integral component of highly automated factories of the future. Since mobile robots have limited on-board computing capability and battery capacity, there is increasing interest in exploring approaches that enable robots to effectively leverage wireless communications and Edge Computing solutions for perception, navigation, planning, coordination, and control. It is, however, a major challenge achieving precision, high-speed, co-ordinated actions between robots due to tight end-to-end latency, and safety requirements, especially while enabling time-sensitive data exchange over wireless networks and execution of computing workloads distributed across robots and the Edge system. The traditional approach of designing compute, communications, and control components in an Edge system as independent components, limits the capacity and scalability of computing and wireless resources and is therefore unsuitable to meet performance guarantees for energy and resource-efficient time-sensitive robotic applications. In this article, we discuss technical challenges in the context of two Edge Robotics use cases such as conveyer object pick-up and robot navigation, which are representative of time-critical control in IoT applications. We propose research directions grounded in an end-to-end system co-design paradigm and describe technology components such as virtualized robot functions, compute-communications-control co-design, Edge system co-simulation, safety and security aspects that are core to Edge Robotics.

We also briefly outline future research directions that are necessary to pave the path toward factory-scale Edge Robotics systems.

Introduction

Industrial IoT is going through a digital transformation toward connected, flexible and intelligent autonomous systems and the COVID-19 pandemic has accelerated this transformation [1]. Collaborative mobile robots, where robots work together as a team, can enable highly reconfigurable work cells that adapt to new processes and tasks such as joint assembly, packaging, and warehousing. Robots combining mobility and manipulation capabilities can achieve such reconfigurability, but they would need significant computing capability to run advanced perception, planning and control algorithms for autonomy and complex coordination, with safety and efficiency. Since mobile robots have limited computing capability due to battery life, size and cost limitations, there is a need to leverage emerging communication and Edge Computing technologies to offload compute-heavy algorithms and workloads to Edge servers [2], for advanced Artificial Intelligence (AI) capabilities as well as for multi-robot coordination and control.

These new capabilities are fueling a trend toward factory-scale Edge Robotic Systems, as shown in Fig. 1. In an Edge Robotics System, sensors in the infrastructure, robotic agents, and Edge computing servers, communicate over wireless and wired networks to perform autonomous system-level control and coordination, at lower latencies and with more predictable network performance, as compared to cloud-based solutions [3]. However, for the robots to effectively leverage Edge Computing, it is necessary to offload compute functions or workloads of multiple robots to the Edge server with the required time-sensitivity and determinism, and in a manner that efficiently utilizes wireless and computing resources, while also ensuring adequate task performance and safety. Distributed computing algorithms require synchronous or time-aware processing. Hence, communications and computing resources must be carefully managed, coordinated and allocated to ensure the robots can operate successfully and safely in the presence of imperfect or limited connectivity and varying availability of computing resources. Additionally, due to stringent safety and availability requirements in factories, backup mechanisms must be provided to handle degradations safely and gracefully.

Among the core technologies that can enable the vision of Edge Robotics at factory-scale, are the recent advancements in function virtualization. Function virtualization concerns the analysis and partitioning of services, such that they can be offloaded across a distributed computational infrastructure. Function virtualization has been applied to network services and has been a key driver in software-defined networks [4]. An Edge Robotic System can leverage similar tools in the virtualization of robotic functions, such as perception, mapping, planning, and control.

To support the compute and communication requirements of an Edge Robotic System, algorithms that jointly adapt network and compute resources with robotic control functions, otherwise called *co-design*, can enable time-critical robot performance while also achieving resource efficiency. In traditional Edge networks, compute, communications, and robotic control, are designed as independent components. However, such an approach leads to inefficient utilization of compute and com-

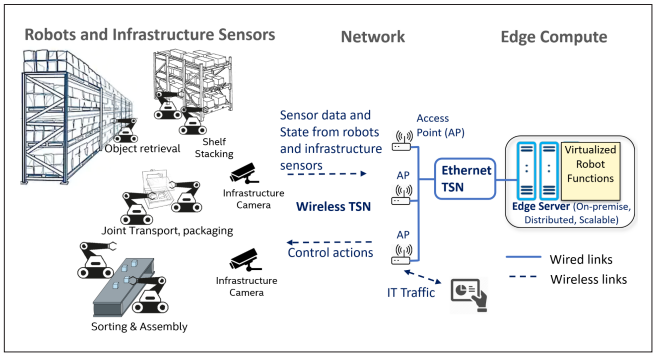

FIGURE 1. Factory scale Edge Robotics system.

Girish Srinivasa Murthy, Amit S. Baxi, and Vincent S. Mageshkumar are with Intel Labs, India.

Mark Eisen, Susruth Sudhakaran, and Margaret Huang are with Intel Labs, USA.

Fabian Oboril and Michael Paulitsch are with Intel Labs, Germany.

Digital Object Identifier: 10.1109/IOTM.001.2200056

2576-3180/22/$25.00 © 2022 IEEE

munication resources due to over-provisioning and it is difficult to meet latency/reliability guarantees when the system scales to many robots. A co-designed system, on the other hand, enables the exchange of robotic, computing, and network state information across layers in the Edge system stack to maximize resource efficiency and task performance across these system components in a contextual manner. We envision the co-design approach as a key enabler for Edge Robotic Systems, where computationally heavy robot functions are offloaded or virtualized on the Edge server with latency and reliability guarantees.

Complex interactions between robots, sensors, Edge computing, and the communications network make it necessary to construct a representative system for both development and evaluation. For factory-scale systems, it is infeasible to build a sufficient testbed, while existing simulation technology focuses on modeling each of these components individually. The third core technology thus involves the development of co-simulation tools that model the Robot-Network-Edge system end-to-end (E2E), that is, including sensing, communications, distributed computing and actuation control, at scale. Lastly, since such large, highly distributed systems may sometimes miss guarantees or malfunction, a security and safety architecture with system-level supervision and locally enforced safe modes is required.

We begin in this article by describing the factory-scale Edge Robotics System as a series of interacting sensing-perception-action-communication (SPAC) loops. We then describe challenges and research directions covering robot function virtualization on the Edge, co-design of communications, compute, and control, and Edge system co-simulation. Related to these are important considerations regarding safety and security. We conclude with a discussion on future directions relevant to Edge Robotic Systems.

Factory-Scale Edge Robotics

Achieving the vision of autonomous factory-scale Edge robotics, as illustrated in Fig. 1, requires the management of robotic perception-planning-control functions, the network and compute resources that comprise the Edge infrastructure, including infrastructure sensors. Each of the robotic agent/tasks can be individually represented as a time-sensitive SPAC loop, which comprises of

- Sensing the system state
- Communicating sensor data to the Edge server
- Processing sensor data via perception and planning functions
- Performing appropriate control action

Figure 2 and Fig. 3 illustrate core robotic SPAC tasks in a factory and are representative of time-sensitive control in many IoT applications. Figure 2 illustrates a simple conveyor belt object manipulation use case to highlight the challenges and opportunities in enabling Edge robotics at the factory-scale. Robots sense their environment through on-board sensors such as cameras, lidar, inertial sensors, etc., as well as through infrastructure sensors such as factory cameras. Sensor data is encoded and streamed to an Edge Server over wired/wireless networks. The data is processed at the Edge server to perceive the environment, for example, in Fig. 2, to estimate the 3D pose of the object on the conveyer belt, which is used by the robotic action primitive for grasping. Due to the high network data rate of camera images and large computational load of perception algorithms, this compute offloading process comes with non-negligible delay and potential for data loss in the wireless transmission. Each RGB-D camera generates ~50MBytes per second (assuming RGB at VGA resolu-tion, Depth at VGA and 24-bits/pixel resolution, both channels at 25 frames/sec) which need to be compressed and sent over wireless to the Edge server. Note that this high resolution is generally needed to detect small objects in the camera image. Perception algorithms can add a few to hundreds of milliseconds of compute latencies, depending on their complexity [5]. Additionally, due to uplink and downlink latencies, the sensing-to-actuation round-trip latency can easily exceed 100ms [6]. At a conveyer belt speed of 0.8m/s, 100ms latency translates into an error of ~8cm in estimation of object location on the belt, which can cause the object pick-up task to fail, unless complemented by the co-design techniques described later.

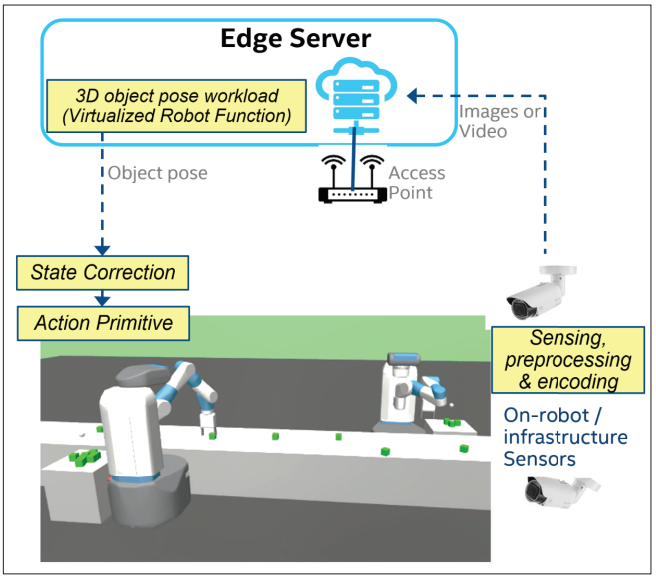

FIGURE 2. Edge Robotics: Conveyer object pick-up use case.

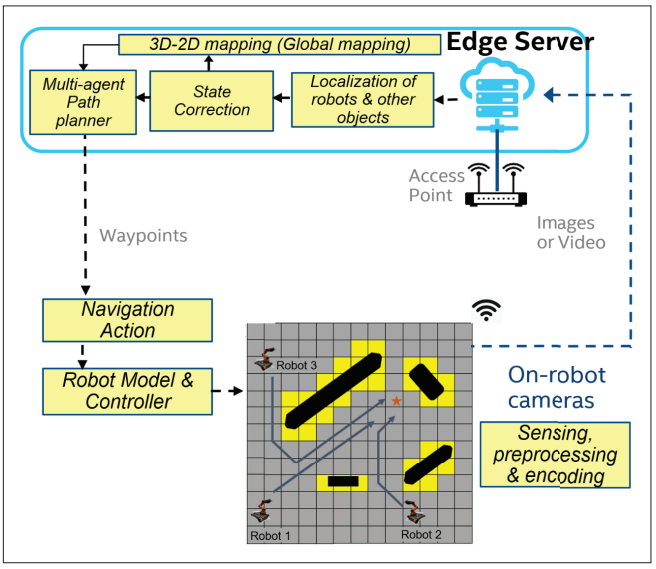

FIGURE 3. Edge Robotics based AMR navigation use case

Another key robotic task present in factory environments is the navigation of Autonomous Mobile Robots (AMRs), which is also represented with a SPAC loop in Fig. 3. Due to their mobility, AMR tasks fundamentally rely on wireless communications in the factory. For these tasks, data from on-robot 3D cameras must be processed at the Edge server for a variety of navigation functions, such as mapping, occupancy grid generation, localization, path planning and multi-robot coordination. This information is then used to compute waypoints, velocity and steering commands that are sent back to each AMR over the wireless channel. Like the manipulation example, large data rates over wireless and computational workloads can cause significant latency and data loss. For example, at navigation speed of 2m/s, a sensing-to-actuation latency of ~100ms, can result in AMR localization error of ~20cm, leading to path-planning errors and increase in potential collisions, driving the need for state correction.

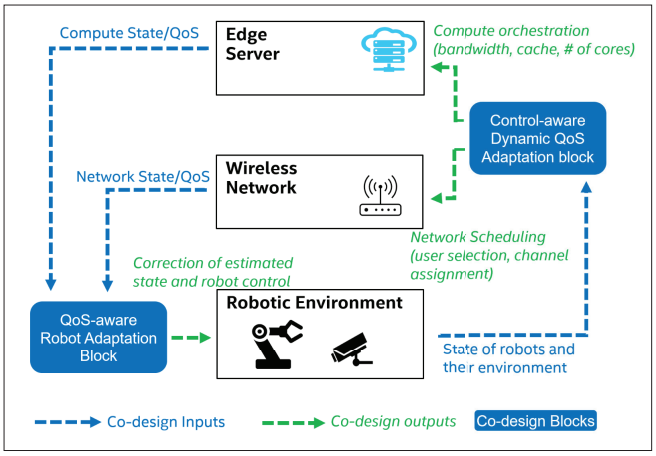

FIGURE 4. The co-design framework.

Safety is typically handled at the robot itself using dedicated safety sensors and computing hardware. However, to enable the increasing demand for flexibility, new dynamic safety concepts are required [7], which need additional compute. Hence, safety-related workloads may also need to be offloaded to the Edge server with latency awareness and time supervision.

At a factory-scale, large numbers of SPAC loops such as those illustrated in Fig. 2 and Fig. 3 will operate in parallel, while sharing a limited amount of network and compute resources. The proper allocation of such resources is necessary to manage the latencies and data loss that result from offloading compute functions, and the robotic control functions themselves must be able to adapt to latency and data loss.

Research Directions and Challenges

In this section we detail the core pieces of technology that can enable factory-scale Edge Robotics.

Robot Function Virtualization

Typically, robot functions for perception, planning and control are executed locally on the robot to ensure low-latency and deterministic control loops. In Edge Robotic Systems, Robot Function Virtualization (RFV) explores optimal partitioning of robotic functions or workloads across the Robot-Edge boundary, with the required time-sensitivity, determinism, while also improving robotic task performance, either in terms of improved sensing-to-actuation latency (task speed-up), improved energy efficiency (longer battery operating time) or improved task accuracy (better task success rate). It is possible to significantly accelerate compute functions and reduce computational latency via Edge offloading, however, offloading comes at a cost of increased communication latency and increased energy consumption due to communications. Edge offloading can also impact task accuracy due to information loss in encoding-decoding operations. RFV involves

- Modeling of the robotic tasks and analysis of sensing-perception-actuation pipeline
- Determining optimal Robot-Edge workload partitioning scheme to maximize compute offload to the Edge server, while also minimizing data transmission over wireless

Robot task modeling concerns the creation of the robots' simulation model, their environment, modeling sensors, physics of robots, and their movement. Robotic tasks can be achieved through traditional kinematics-control algorithms or can be learnt by techniques such as Reinforcement Learning (RL), the output of which is a control policy for the robot. Complex robotic tasks can be learned as a whole (end-to-end) or in the form of simpler robotic actions (Action Primitives) [8]. Robotic action primitives are workloads that are usually executed on robots' local compute resources.

In the distributed sensing-computing Edge system, it becomes crucial to model data flows between robots, infrastructure sensors and wireless networking and computing nodes, and achieve tight time-synchronization between these entities, to enable accurate data fusion, state estimation and robot control via the Edge server. This time synchronization may be achieved through Time Sensitive Networking (TSN) protocols [9].

Furthermore, it is also necessary to characterize algorithms and data processing building blocks in robots' SPAC pipeline. In the example shown in Fig. 2, this includes characterization of compute latencies, computing resource utilization (CPU/GPU utilization, memory utilization), energy utilization and bitrate over wireless communications for each of the processing blocks in the pipeline such as sensing, encoding-decoding, pose estimation, state correction and action primitive execution (actuation). Lack of tools to accurately model the compute latency and compute/energy/wireless resource utilization, on different Robot and Edge platforms with heterogeneous compute, is a key challenge in workload partitioning. The workload partitioning scheme must jointly optimize E2E latency (which includes computing and communication latencies), data volume over wireless and task accuracy (success rate). Techniques such as region-of-interest segmentation, saliency and novel encoding schemes can be used to minimize data volume over wireless. Rather than creating a static Robot-Edge workload partitioning scheme offline, a novel research direction to explore is *Dynamic Workload Partitioning*, where robots and Edge resources can jointly decide how to split workloads in real-time, based on available compute resources and network conditions. A library of parametrized and compute-communications co-optimized robot functions, virtualized on the Edge server, can be offered as microservices, which multiple robots can utilize.

Like the conveyer use case, navigation related workloads shown in Fig. 3, for AMRs, such as mapping, localization, path-planning, and obstacle avoidance can also be realized as virtual functions on the Edge server. Energy modeling and power characterization of workloads of battery-operated AMRs is an important capability that also drives the decision on workload parti-tioning based on power-performance tradeoffs.

Co-Design of Compute-Communications-Control

The RFV process enables compute offloading to the Edge server, however, this comes at the cost of large data rates over wireless. The resulting latencies and packet losses that can occur due to insufficient resources will lead to degraded performance in the robotic tasks. The adaptation of robotic control and optimal management of limited resources, called the co-design of compute, communications, and control, is thus an integral to the factory-scale Edge robotics solution.

A simplified co-design framework is shown in Fig. 4. Co-design refers to the joint decision-making processes of multiple layers in the Edge system, namely robotic control, network scheduling and compute orchestration. We focus on two aspects of co-design. The first component is to enhance robot controllers to adapt to latency/data loss effects to improve task success rate under unreliable network conditions and limited Edge resources. Here, the state of the network and/or compute resources is used by the robot to adapt its control policy for resilience to network effects such as delayed or dropped packets. This state information may be in the form of resource availability, wireless channel quality, or the current quality of service (QoS) being supplied by the network infrastructure. In the latter case, the QoS is typically represented as a latency, jitter, throughput, or packet loss probability [10]. Such adaptations may be in the form of state prediction [11] or robust control policies [12]. We refer to this in Fig. 4 as *QoS-aware Robot Adaptation*.

The second co-design component concerns the ability of Edge compute and communications resources to dynamically adapt in real-time based on control or task-level inputs from the robotic application. Here, robot state and environment

state information are used by the compute and network infrastructure to dynamically adapt its QoS needed to support the robotic tasks, and in a manner that improves overall resource utilization. The co-design of communication and control systems has been an active area of research for several years, often employing simplified communication and control models to aid in analysis [13]. These simplified models, however, do not well represent realistic factory environments, in which robotic control is highly non-linear and modern communication systems, such as 5G and Wi-Fi 6, that involve complex scheduling architectures. A more general and scalable approach is to instead dynamically adapt a high-level QoS parameter which can be used as an input in state-of-the-art communication and compute orchestration algorithms [14]. We refer to this as *Control-aware Dynamic QoS Adaptation*. Both components of the co-designed system can be derived using control theoretic analysis, heuristics, or machine learning tools.

To exemplify a practical implementation of compute-communications-control co-design, we describe its deployment in the manipulation use case illustrated in Fig. 2. Due to the communication delay in the camera uplink to the Edge server and the processing delay in executing the object detection workload, the object state information received by the robot is stale and will cause pick-up task fails at high belt speeds. The QoS-aware Robot Adaptation function at the robotic controller can correct for this information staleness by using either resource-level states, such as wireless channel conditions or compute availability, or by keeping direct track of the E2E delay experienced in the RFV pipeline. That is, the robot learns to predict the location of the object, compensating for round-trip latencies, before initiating any control actions. This makes the robot control policy more resilient to network and latency effects. That state correc-tion itself can be learned as a neural network using RL or designed via traditional control tools such as the extended Kalman filter.

We can make a similar analogy in case of AMRs moving in a factory, when localization, mapping and path-planning functions are offloaded to the Edge server, as in Fig. 3. In this case too, a state correction block can be used to precisely predict and correct poses of all mobile robots, thereby reducing potential collision incidents and improving path planning.

Referring again to the manipulation problem in Fig. 2, we describe a deployment of Control-aware Dynamic QoS adaptation. Here, a dynamic resource allocation policy reduces wireless and compute resource allocation to the robot when the object is beyond the grasping range and thus of limited time-sensitive importance to task success. Alternatively, dynamic resource allocation can also be done by limiting packet delivery rates (dynamically reducing the frame rate of the camera), or by limiting packet size or throughput (dynamically reducing the image reso-lution). Doing so prioritizes such resources for robotic tasks that require higher reliability and lower latency during the grasping stage, and this is a key enabler of scaling to factory-scale systems without requiring prohibitively large Edge resource requirements. The precise mapping of object position and information staleness to required QoS level can be derived through heuristics, or via optimization and RL methods.

In a more evolved co-design solution, robot adaptation and dynamic QoS modules can be jointly optimized, such that the dynamic QoS adapts in relation to the robot's state correction capabilities to relax QoS requirements even further. Moreover, there is significant potential in jointly orchestrating both the communications and computing resources such that E2E QoS requirements can be met across the network and compute layers with maximum resource efficiency. Doing so may require careful consideration of the scheduling constraints of network protocols, such as 5G and Wi-Fi, and of modern compute orchestrators, such as Kubernetes.

Creating an E2E simulation model of an Edge Robotics System entails simulation across domains of computing, physics, and communications, and therefore poses several challenges.

Edge System Co-simulator

Co-design approaches have the potential to improve robotic task performance and optimize resource usage as the system scales. However, an important question is — How do we design, evaluate, and verify co-design methods? It is impractical to physically create various configurations of multi-robot systems at scale to evaluate co-design methods, therefore, it becomes necessary to create their virtual models to emulate realistic behavior at scale. Creating an E2E simulation model of an Edge Robotics System entails simulation across domains of computing, physics, and communications, and therefore poses several challenges.

Modeling holistic behavior of such complex systems requires integration of virtual and physical parts of the system and modeling of interactions between them. Separate simulation models are available for individual subsystems including network, physics, and compute; hence, one approach is to apply co-simulation techniques to interconnect different simulators through well-defined software interfaces and enable data exchange between them. This would also entail coupling simulators to real-world prototypes or compute emulations to model the E2E system behavior. Such an approach would involve the following challenges:

- Enabling flow of state and control information from physics simulators into compute emulations
- Enabling data traffic from compute emulations into network simulators
- Synchronous execution and progression of state and time steps across network and physics simulators, and compute emulations
- Creating abstraction and well-defined software interfaces to multiple simulators, to achieve the above functionalities.

The Edge System Co-Simulator in Fig. 5 brings together multi-robot physics simulators, wireless network simulators for Wi-Fi and 5G, and compute emulation at the Robot and Edge system, through well-defined interfaces. It models interaction between these domains mimicking real-world compute and communication data flows. It combines virtual (simulation models) and real assets (software and hardware) for demonstrating and evaluating co-design concepts. The Co-simulator has three components:

Robot Physics Simulator simulates physics of robots, their environment and multi-robot tasks in a factory. Physical models emulate the physics associated with the entities such as robots, cameras and other sensors, and their states may include arm positions, locations, arm velocities, camera images and other sensor data used by the virtual world to perceive the physical world. As the simulation advances in time, the progression of states associated with these entities are shared with other simulators over well-defined interfaces. To reduce simulation-to-real gap, the simulator provides *hardware-in-loop* interface to connect real robots, control devices and sensors, ingests real-world states, which are processed in simulation domain and outputs control actions to devices in the real world.

Local and Edge compute emulation module interfaces with the physics model to perceive state of the environment and uses this perception to affect subsequent state changes to the physics model. The compute associated with perception and control can be modeled by a combination of real-world prototypes and virtual compute instances using compute containers or virtual machines. This enables executing algorithmic workloads or functions as software-in-loop emulation. Edge compute may execute functions such as vision processing, state estimation, navigation, planning and coordination of multiple robots.

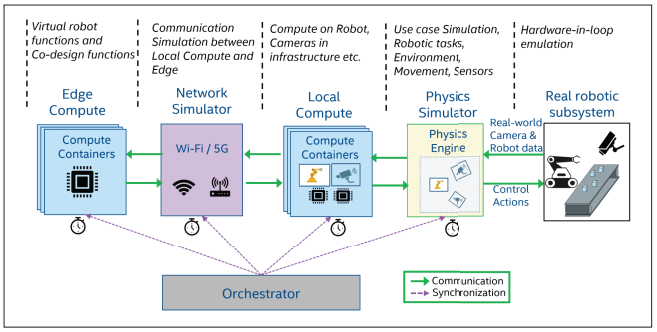

FIGURE 5. Edge System co-simulator architecture.

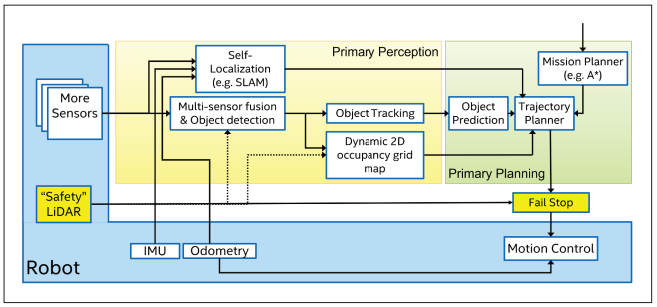

FIGURE 6. Illustration of state-of-the-art safety architecture.

Likewise, local compute may execute functions like sensor data pre-processing, compression, computing control primitives of robots etc.

Network Simulator integrates the traffic generated by the functions running at the local and Edge compute, with network simulation. It simulates communication delays and artifacts like jitter, packet loss, packet error rate and interference associated with the traffic, via interfaces to well-known network simulators such as ns-3 and OMNet++, to model Wi-Fi and 5G. Functions in local and Edge compute can exchange data and control packets with the robotic environment over standard interfaces such as Robotic Operating System (ROS) messages.

Safety and Availability

A common robot architecture with safety system is depicted in Fig. 6, which runs a primary perception and planning stage with a set of sensors and has a dedicated safety channel. Typically, safety channel uses a depth sensor, for example LiDAR, and scans the close vicinity of the robot for potential obstacles. If such an obstacle is detected, the robot is forced to stop (fail stop). Primary system must prevent safety layer stops, by avoiding coming too close to such obstacles.

In future systems, the safety layer will be required to perform more complex calculations to achieve the goals defined; for example, in the ISO TS15066 standard [7], which pertains to more elegant handling of dangerous situations than safety stops. To achieve this, comprehensive 3D environment models with object positions, heading, speed and behavior are required which add significant compute demand. The safety layer can be offloaded to the Edge server along with primary perception and planning workloads, however, when the Robot-Edge communication latencies exceed a given threshold, the robot will perform a fail-stop maneuver, impacting availability. In factory setups with inconsistent QoS, such an approach can significantly reduce robot availability and render the solution impractical. Hence, novel latency-aware safety approaches which can be executed on the Edge are required. A lightweight checker on the robot can allow flexible handling of latencies, while maintaining availability, when safety workloads are executed on the Edge server. The checker can validate if the environment model used to create the last safe actuation command (in the Edge system) is still valid using simple on-robot checks. If the checker indicates a valid environment, the command can be executed or continue to be executed, if no new command arrives within the expected latency. Otherwise, a fail-stop maneuver or degraded mode can be initiated. This would require the checker to receive a validated environment model from the Edge server, which is correlated against the most recent robot sensor data. If no safety-relevant change is detected in the environment, the robot can continue with its current motion, even if the expected communication latency is exceeded. Additionally, by combining the co-design approach with TSN-based time synchronization and system-level performance estimates, it may be possible to monitor temporal correlations between environmental perception, network state, Edge state, and robot state, and trigger additional system-level safe modes for the robot, if timeout violations or poor temporal correlations are observed. Finally, to achieve factory-wide safe operation, safety information must be exchanged between different machine modules to implement coordinated safety stop functions, via adoption of safety protocols such as EtherCAT and PROFISafe over wireless communications.

End-to-End Security

While security is vital to any cloud or Edge system, threat detection and safe recovery are extremely critical for time-sensitive systems such as Edge Robotics, since the faults injected by an adversary can have catastrophic results. The adversary may gain physical or remote access to the on-robot sensors, system or controller and alter the control dynamics of robot. The wireless network and network infrastructure are particularly vulnerable, where the adversary can slowly disrupt time-synchronization or inject delays between various entities in the system, for example between infrastructure cameras and the Edge, which may delay control signals to the robots, adversely affecting the behavior of a robot or of a collaborative task. The Edge computing infrastructure is also vulnerable, where the adversary may attack the orchestration or resource allocation modules, delaying or starving the system of vital communication or computing resources or cause computing errors that impact state estimation, planning and control. Therefore, beyond standard security mechanisms for Edge Computing assisted IoT [15] such as authentication, cryptographic features, root of trust, secure boot execution and tamper detection, an integrated approach towards safety and security analysis, design and verification is required, which considers time-criticality as important as data integrity.

Future Directions

Concepts of RFV, co-design, co-simulation, safety and security lay the foundation for factory-scale Edge Robotics. However, there remain areas of research to explore in further improving the scalability and efficiency of such systems.

Factory-scale environments feature complex coordination of multiple robots. It is necessary then to scale the RFV process to these multi-agent functions and scenarios. Latency-aware multi-agent RL approaches can be explored for achieving this complex coordination. Secondly, time-synchronized sensor streams from robots can be combined with sensors in the infrastruc-ture to provide occlusion-free and non-line of sight to AMRs, reduce the compute complexity on robots, and improve energy efficiency and cost.

The concepts of co-design between different elements of the E2E system can also be extended. The encoding used by sensors to send data to the Edge server can dynamically adapt to the state of the robotic system or capacity of the wireless channel. As previously mentioned, machine learning and RL may be prominently used to derive control and resource allocation policies. To run these algorithms online for dynamic learning, it is necessary to consider the effects of network/compute delay and capacity on the learning process itself. Hence, co-design principles can be leveraged for designing learning algorithms over networks.

Co-simulation can be extended to enable overlaying of the wireless network infrastructure over simulated robotic environment to model AMR mobility, hand-off between wireless Access Points (APs), including multi-AP resource scheduling and coordination. Modeling the Edge robotics factory as a Digital Twin would enable further validation of RFV and co-design concepts before real-world deployments.

Finally, safety and security must be tightly integrated in all aspects of system operation and it is necessary to create a design framework that integrates safety and security mechanisms for a robust and resilient system architecture.

Conclusion

Edge Robotics is a promising direction to enable highly automated factories of the future. There are several challenges in achieving timeliness, determinism, resource efficiency and safety. Virtualization of robot functions, applying co-design and co-simulation techniques and incorporation of safety and security features are core concepts which address some of the above challenges. However, achieving time-critical task performance and resource efficiency at factory-scale needs further research on multiple vectors outlined in this article.

References

[1] S. Yang *et al.*, "Robots under COVID-19 Pandemic: A Comprehensive Survey," *IEEE Access*, vol. 9, 2020, pp. 1590–1615.

[2] V. K. Sarker *et al.*, "Offloading SLAM for Indoor Mobile Robots with Edge-Fog-Cloud Computing," *1st Int'l. Conf. Advances in Science, Engineering and Robotics Tech.*, 2019, pp. 1–6.

[3] K. Antevski *et al.* "Enhancing Edge robotics through the use of context information," *Proc. Wksp. Experimentation and Measurements in 5G*, 2018, Association for Computing Machinery, New York, NY, USA, 7–12.

[4] G. Baldoni *et al.* "Edge Computing Enhancements in an NFV-based Ecosystem for 5G Neutral Hosts," *IEEE Conf. Network Function Virtualization and Software Defined Networks*, 2018, pp. 1–5.

[5] Paul, SK *et al.*, "Object Detection and Pose Estimation from RGB and Depth Data for Real-Time, Adaptive Robotic Grasping," *Advances in Computer Vision and Computational Biology*, Springer, Cham, 2021, pp. 121–42.

[6] Tanwani, AK *et al.*, "A Fog Robotics Approach to Deep Robot Learning: Application to Object Recognition and Grasp Planning in Surface Decluttering," *2019 Int'l. Conf. Robotics and Automation (ICRA)*, 2019

[7] B. Matthias and T. Reisinger, "Example Application of ISO/TS 15066 to a Collaborative Assembly Scenario," *Proc. ISR 2016: 47st Int'l. Symp. Robotics*, 2016, pp. 1–5.

[8] S. Nasiriany *et al.*, "Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks," arXiv preprint arXiv:2110.03655 (2021).

[9] I. Val *et al.*, "IEEE 802.1AS Clock Synchronization Performance Evaluation of an Integrated Wired–Wireless TSN Architecture," *IEEE Trans. Industrial Informatics*, vol. 18, no. 5, May 2022, pp. 2986–2999.

[10] E. Budiman and O. Wicaksono, "Measuring Quality of Service for Mobile Internet Services," *2016 2nd Int'l. Conf. Science in Information Technology (ICSITech)*, 2016, pp. 300–05.

[11] Ibarra, Linda Patricia Osuna *et al.*, "Observers And Predictors For Wireless Time-Sensitive Control Loops," *16th IEEE Int'l. Conf. Factory Communication Systems (WFCS)*, 2020.

[12] F. Gao *et al.*, "Robust Control of Heterogeneous Vehicular Platoon with Uncertain Dynamics and Communication Delay," *IET Intelligent Transport Systems*, vol. 10.7, 2016, pp. 503–13

[13] P. Park *et al.*, "Wireless Networked Control System Co-Design," *Int'l. Conf. Networking, Sensing and Control*. 2011.10

[14] M. Eisen *et al.*, "Communication-Control Co-design in Wireless Edge Industrial Systems." arXiv preprint arXiv:2202.03976, 2022.

[15] A. Alwarafy *et al.*, "A Survey on Security and Privacy Issues in Edge-Computing-Assisted Internet of Things," *IEEE Internet of Things Journal*, vol. 8, no. 6, 2020, pp. 4004–22.

Biographies

Amit Baxi (amit.s.baxi@intel.com) holds M.S. degree in Digital Design and Embedded Systems from Manipal University, India. He is currently a Research Scientist with Intel Labs in Bengaluru. His research interests include sensing systems, wearable health technologies, IoT and Edge robotics.

Mark Eisen (mark.eisen@intel.com) holds a Ph.D. in electrical engineering and Masters in Statistics from the University of Pennsylvania. He works as a Research Scientist at Intel Labs in Hillsboro, OR. His research interests include machine learning, wireless communications, networked control systems, and statistical optimization.

Susruth Sudhakaran (susruth.sudhakaran@intel.com) holds M.S. degree in Electrical Engineering from Southern Illinois University, Edwardsville. He has over 15 years of experience in wireless systems. He works as a Research Scientist at Intel Labs, Hillsboro, OR and his research interests include Wireless TSN, determinism over wireless, and research testbeds.

Fabian Oboril (fabian.oboril@intel.com) holds a Ph.D. in computer science on the dependable design of microprocessors from Karlsruhe Institute of Technology, Germany. He works as a Research Scientist with Intel Labs in Germany, focusing on autonomous mobile systems and safety aspects of indoor and outdoor mobile systems.

Girish S. Murthy (girish.srinivasa.murthy @intel.com) is a Research Scientist with Intel Labs in Bengaluru, India. Before joining Intel, he worked at Texas instruments on wireless systems research. His research interests include image, video analytics, compression, and autonomous systems.

Vincent Mageshkumar (vincent.s.mageshkumar@intel.com) is a Research Scientist at Intel Labs, Bengaluru, India. His research interests include function virtualization, energy modeling, hardware design, sensing systems, robotics, and AI.

Michael Paulitsch (michael.paulitsch@intel.com) holds a Ph.D. in computer science and economics. He is a Principal Engineer at Intel Labs in Germany. His focus is on the dependability of systems and AI. During his career, he has worked in Intel, Honeywell, Airbus, and Thales on critical systems in automotive, aerospace, space, and railway.

Margaret Huang (margaret.huang@intel.com) is a Senior Principal Engineer at Intel Labs, in Santa Clara, CA. Before joining Intel, she held R&D positions in Hewlett-Packard and Motorola/Freescale. In Intel, she drives sensing and autonomous systems projects, including radar perception and Edge Robotics.