IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 September 2024, accepted 1 October 2024, date of publication 7 October 2024, date of current version 21 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3475584

RESEARCH ARTICLE

Distributed Control for Collaborative Robotic Systems Using 5G Edge Computing

DOMINIK URBANIAK[1], SEBASTIAN BRO DAMSGAARD[2], WEIFAN ZHANG[2], JAN ROSELL[1], RAÚL SUÁREZ[1], (Senior Member, IEEE), AND MICHAEL SUPPA[3]

[1]Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya, 08028 Barcelona, Spain
[2]Department of Electronic Systems, Aalborg University, 9220 Aalborg, Denmark
[3]Roboception GmbH, 81241 Munich, Germany

Corresponding author: Dominik Urbaniak (dominik.urbaniak@upc.edu)

This work was supported by European Union's Horizon 2020 Research and Innovation Program under the Marie Skłodowska-Curie under Grant 956670.

ABSTRACT Collaborative and mobile robotics for industrial environments promise to enable autonomous and flexible production processes. However, this vision also poses significant challenges to the robotic systems, requiring them to adapt to dynamic environments and ensure human safety by leveraging continuous image streams and sophisticated data processing. Edge computing allows offloading the computational load to edge servers, communicating the image data, generated on mobile robots, over fast and reliable private 5G networks. However, there are multidisciplinary and interdependent factors that influence the reaction time of the distributed system which are not well explored in the literature for real robotic use cases, but have a significant impact on safe robotic behavior and the effectiveness of edge computing. In this work, we implement a distributed control system that offloads the image processing to measure and analyze the effect of various factors on the reaction time of the system for collaborative robotics applications. Different values for the sensing rate, image resolution and compression and quality-of-service settings are evaluated for communication and computation times as well as for the task performance. To account for the safety requirements in collaborative robotics, we add a low-level control timeout in cases of large jitter and stop the robot in cases of frequently missed detections. A push and a teleoperation experiment evaluate the reaction times in real distributed control scenarios using 5G edge computing. All experiments are implemented using ROS 2 Humble. The code and videos of the experiments are available at https://github.com/DominikUrbaniak/ros2_distributed_control_system.

INDEX TERMS 5G, collaborative robotics, distributed control, edge computing, image processing, Industry 4.0, object tracking.

I. INTRODUCTION

Facing the market trend towards highly customized products, short product life-cycles and sustainability, production processes require high adaptability. Two directions appear promising to support that vision: 1) mobile manipulators that can perform object manipulation at different workstations, and, 2) collaborative robotics which is defined as a shared workspace for mobile manipulators and human workforce [1]. However, the implementation of these directions demand sophisticated autonomy of the mobile manipulators to be able to react quickly to a dynamic and uncertain environment, and to ensure human safety. Continuous monitoring using visual sensors allow the reaction and adjustment of the robot motion in such complex scenarios. However, the continuous processing of image data demands high computational effort and long computation times interfere with the ability of the robot to react quickly to changes. Edge computing provides large computational resources on-premise and utilizes a distributed network of computers. It enables small devices with limited resources to perform complex computations by offloading those to the

The associate editor coordinating the review of this manuscript and approving it for publication was Hadi Tabatabaee Malazi.

powerful computers in the network. Mobile manipulators could benefit greatly from edge computing, since their energy and computational resources are limited. Since they cannot be connected via Ethernet to stationary resources, fast and reliable communication over wireless networks is required [2].

In an environment where unforeseen changes can occur, the reaction time of the robotic system plays a crucial role to be able to assess performance and safety expectations. For instance, the implementation of the Speed and Separation Monitoring (SSM) safety mode depends on the reaction time [3]. In a distributed setting, the communication and computation latencies contribute significantly to it. They depend on different factors that can influence the effectiveness of an edge computing approach to reduce the reaction time. The communication latency depends on the network, the Quality of Service (QoS) settings, the size of the transmitted messages, uplink and downlink, and the communication rate. The communication jitter depends on the network and QoS settings and can result in unsafe robotic motion when not considered in the control loop. The computation latency depends on the computation hardware, algorithm type and configuration. Additionally, the robot control can add additional latency when robot motion is smoothed for human comfort. The interdependence must be considered since a low communication rate could be the bottleneck, and allocating more communication resources might not reduce the reaction time.

In this work, we analyze the latency impact of various interdependent factors from communications, computer vision and robotics where continuous image streams are processed on an edge computer using two types of algorithms, fiducial marker and human hand detections. To this end, we consider the reliability quality-of-service (QoS) feature provided by the Robot Operating System (ROS 2) and various combinations of communication technologies, sensing rates, image resolutions and image compression to relate the impact of latencies from different sources and evaluate them first in two isolated experiments (see Fig. 1). We do not consider different network slices, and do not focus on comparing computational hardware. Together with safety related findings, we apply the results from the isolated experiments in two real distributed control systems to visualize the latency impact. In the first experiment, a 6-Degree of Freedom (DoF) robot manipulator is controlled to push a cube with constant speed until a target pose is detected using ArUco markers as fiducial markers. This experiment allows the precise measurement of the latency impact by observing the placement error that results from the processing delay between the image capture and the robot low-level control. In the second experiment, we apply the same concept in a teleoperation scenario where the robot is controlled by the motion of a human hand, introducing the smoothing of the robot motion as additional factor on the reaction time. It illustrates the importance of the reaction time for collaborative applications.

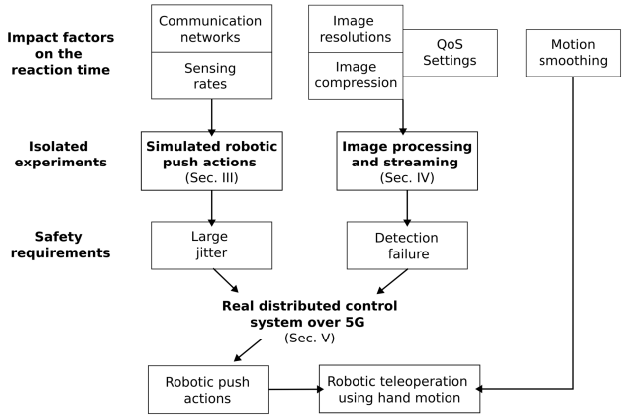


FIGURE 1. Hierarchical structure of the paper. Different impact factors on the reaction time are analyzed in isolated experiments (sections III, IV) and applied in real distributed control experiments with safety measures (section V).

Our contributions can be summarized as:

- The implementation of an entire distributed control system for collaborative robotics applications including wireless communication over a private 5G network, image processing on an edge server and closed-loop control of 6 DoF-manipulator velocities.
- The evaluation of the reaction time to assess the edge computing effectiveness based on various factors: communication network, sensing rate, image resolution and compression, computation time, ROS 2 reliability QoS profiles, and robot motion smoothing.
- The implementation of safety measures inside the robot control that consider jitter, detection failures and human interaction.

In the following section, related work considering latencies in edge computing is discussed. Section III assesses with numerous episodes of simulated push actions the impact of the sensing rate and the low-level control timeout on the comparison of communication technologies. Section IV deals with the integration of images for visual sensing which is required for the real distributed control experiments described in section V. Section VI discusses the results and gives an outlook on promising future work.

II. RELATED WORK

Collaborative robotics is expected to combine the human intuition and adaptability with the robotic precision and endurance to achieve higher throughput and product quality, and to reduce the risks of ‘‘occupational’’ diseases and injuries for human workers that repetitive, monotonous and tedious tasks foster [4]. Ensuring human safety at all times is a main challenge for collaborative robotic systems. The dynamic and uncertain environment require the ability to adapt quickly to unforeseen events. To this end, the ISO/TS 15066 defines safety modes [3]. Among them, the SSM mode adjusts the robot speed focusing on the directed velocities between human and robot and could allow the use of conventional industrial robots in a collaborative setting [5].

For this mode, the reaction time of the system plays an important role. Byner et al. [6] report a reaction time of $L_R = 210$ ms using a laser scanner to detect the human approaching perpendicularly on a linear trajectory. On the other hand, the Power and Force Limiting (PFL) safety mode is suitable for cobots which are lighter and slower [5]. Cobots are used for human-robot-interaction (HRI) such as gesture-based teleoperation which go beyond the collaboration in the sense of a shared workspace. Martin and Moutarde [7] teleoperate a Universal Robot UR3 detecting the 2D pose of a human operator using OpenPose [8]. The pose detection with 640×480 images runs at 30 Hz, the complete teleoperation control at 6 Hz, using a NVIDIA GTX 1080 Ti GPU.

OpenPose and other recent works [9] demonstrate the ability of deep neural networks to robustly detect human poses in images. However, methods that achieve near real-time processing often rely on powerful GPUs which could compromise the effective implementation on resource-constrained devices [9] such as mobile manipulators. Hence, for such devices, there are two options, either perform the tasks with their limited resources, or offload the computations to a powerful edge computer. To find the better option, Hayat et al. [10] propose a distributed control system in three scenarios. The first offloads the complete image processing to the edge, requiring the streaming of uncompressed images. In the second scenario, the images are pre-processed onboard and only a second processing step is performed on the edge computer. The third case does not offload any computations. Using two parameters, the factor of computational power between edge and onboard computer and the networks uplink data rate, the authors present a coherent answer to that question based on the simulation of various parameter combinations. Only at uplink data rates above 120 Mb/s, offloading the image processing completely improves the overall latency. The partial edge computing scenario improves the overall latency by 20% when the computation power on the edge computer is 5 times higher than the onboard computer and the uplink data rate is greater than 40 Mb/s.

Private 5G networks promise to offer powerful wireless communication for such edge computing scenarios. For instance, a 5G prototype is utilized to perform a robotic balancing task via edge computing [11]. The balancing task is performed by an Autonomous Mobile Robot (AMR) with 5-Degree-of-Freedom (DoF) manipulator and 3-DoF holonomic mobile base to keep a sphere in the center of a resistive touchscreen. The authors propose a distributed control architecture: the position of the ball is sensed by the touchscreen and sent wirelessly to an edge computer where the desired 3D pose of the touchscreen is computed by a high-level balance controller. This 3D Cartesian pose is translated into the 8-DoF coordinate frame of the AMR by applying inverse kinematics. The result is sent back to the AMR where the low-level control is performed. During the experiment, uplink and downlink packet sizes between 70 and 350 bytes are measured at a median frequency of around 100 Hz. In [12], it is shown that the navigation planning and docking control of an AMR can be reliably performed using 5G edge computing by offloading the motion planning to the edge. The authors measure different packet sizes from 64 bytes to $1,514$ bytes and data rates of 1.3 Mbit/s uplink and 1.9 Mbit/s downlink. They then compare the impact of the latency from private 5G and 5G Ultra-Reliable Low-Latency Communication (URLLC) and Ethernet communication in the navigation and docking tasks. Both experiments show a performance decrease with 5G communication, however, this is ''acceptable for reliable operation of the AMRs''.

In [13], the low-level control is offloaded to an edge server using 5G and a custom-build AMR for a navigation task. The authors justify the use of edge computing for such small computations with the ''ease of maintenance'' and ''improved resiliency to software and hardware failures''. Another work investigates Time-Sensitive-Networking (TSN) over 5G for industrial control systems that continuously perform ''sensing, computing, and actuation'' in a closed loop [14]. The authors set up a ball balancing task in which a touchscreen is tilted by a two-DoF servo motor to guide the ball along a predefined path. The sensor data of the touchscreen is transmitted as 64 bytes packet at 1000 Hz over the edge via 5G and back to the controller. A requirement of less than 20 ms must be constantly achieved to ensure a successful task performance. The TSN solution can prioritize the time sensitive communication in a network with additional data traffic, and therefore, achieve the latency requirement compared to a solution without TSN. In [15], the performance of 5G edge computing, 4G and 5G cloud computing, and onboard computing is compared for Unmanned Aerial Vehicle (UAV) control. The UAV state is published at 100 Hz and the control is performed at 40 Hz on the edge computer, sending the control command back to the UAV. The 5G edge computing solution achieves 20-30 ms latencies on average, which is significantly lower than the cloud computing solutions (larger than 100 ms on average). As a consequence, the control error is lower which results in a smoother circular trajectory and a successful avoidance of an obstacle that the cloud computing solutions are not capable of avoiding. Overall, the onboard solution achieves the best performance, but 5G edge computing shows ''similar behavior''. A comparison of 5G and Wi-Fi 6 networks is performed in [16] using the communication hardware in the loop with simulations of a robotic coordination and a teleoperation experiment. The data from the human motion capture system is mapped to a desired Cartesian end-effector pose at up to 250 Hz and converted to the robotic joint configuration using inverse kinematics before sending the result wirelessly to the low-level controller. This work shows that the smaller latency outliers of the 5G communication can improve the control accuracy while the Wi-Fi 6 network

generally achieves faster responses allowing an informed choice about the more suitable network depending on the priorities of the control application. In [17], the image processing is offloaded from a mobile manipulator to an edge computer in a robotic fruit picking application. RGB and depth images, at a resolution of 848 × 480 pixels, are transmitted at 30 Hz to the edge computer with a Nvidia RTX-2080 GPU. There, the detection of the fruits is performed with deep-learning based segmentation methods, and the labels are transmitted back to the robot at 50 Hz, where the motion planning is performed. The size of the packets uplink were 80 kB and downlink 16 kB. The processing times were sped up by more than 18 times compared to an onboard computer with a Nvidia Jetson Xavier NX, resulting in a significant reduction in the overall execution time.

In contrast to the related work, we analyze the significance of multidisciplinary and interdependent factors on the reaction time, providing extensive experimentation in simulation and with real hardware. Various communication technologies (Ethernet, private 5G, private 5G URLLC, private 4G, ideal Wi-Fi 5, loaded Wi-Fi 5) are compared for a distributed control scenario depending on different sensing rates of a robotic system in simulation. Additionally, we include the effect of image resolution and compression on the placement error and packet size, and perform long-term image streaming over Ethernet comparing these varying data rates and the ROS 2 *reliable* and *best-effort* QoS profiles. Finally, the holistic perspective over communications and robotics is assembled into two real experiments that offload the image processing to an edge server via a private 5G network to control a mobile manipulator in collaborative scenarios assessing the reaction time and its contributing elements.

III. SIMULATED ROBOTIC PUSH ACTIONS

The purpose of the simulated push experiment is the comparison of different communication technologies for varying communication rates, and to assess the timeout behavior at the low-level control for communication networks with large jitter. Using common camera frame rates as sensing rates and small packet sizes, this experiment simulates a hybrid edge computing scenario assuming local image preprocessing, as described in [10].

A. ROS 2 COMMUNICATION

The Robot Operating System 2 (ROS 2) [18] is an open source middleware that enables the development of robotic systems using various libraries and specific tools. ROS 2 nodes can be executed on different computers within a network and communicate via publisher/subscriber or service/client mechanism. The ROS 2 communication uses eProsima Fast DDS[1] (Data Distribution Service) which implements a RTPS (Real Time Publish Subscribe) protocol based on "unreliable transports such as UDP".

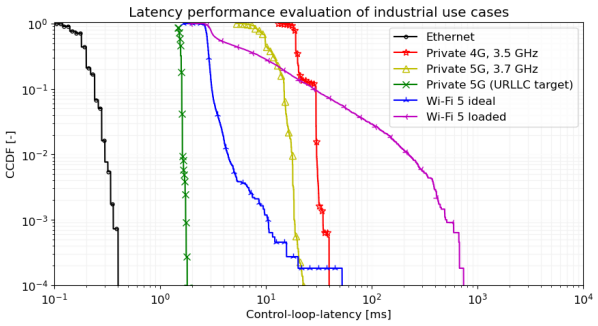


FIGURE 2. The complementary, cumulative distribution function (CCDF) of latencies generated from different communication technologies [19], used in the simulated robotic push actions.

B. LATENCY DISTRIBUTIONS

Different wireless communication technologies can enable mobile manipulators to communicate with an edge server. Their performance vary mainly in the latency and reliability. A comparison of several networks is illustrated in Fig. 2 [19], [20], with packet sizes around 100 bytes [21]. Ethernet serves as baseline to the wireless communication technologies. Compared to Wi-Fi 5, the telecommunication technologies keep the latency bounded in a smaller range, which makes their behavior more reliable. Especially, when other traffic in the network is simulated (loaded Wi-Fi 5), packet arrival is delayed by up to one second. On the other hand, Wi-Fi 5 communicates generally faster than private 4G/5G. The 5G URLLC latency data is simulated, since URLLC networks are not widely deployed yet.

C. METHODOLOGY

The simulated experiment is implemented in a Gazebo[2] simulation environment, running on a simulation computer. Its control architecture is illustrated in Fig. 3. The communication latency of a distributed control system is simulated by delaying the execution of the *Pose retrieval* node. To this end, a latency distribution is selected based on measurements of varying communication technologies (Fig. 2).

In the beginning of the experiment, a communication technology, the sensing rate f_S and the number of push episodes N are specified. Subsequently, the three ROS 2 nodes *Pose retrieval*, *High-level control* and *Low-level control* are executed at f_S until the goal pose $\mathbf{p}_g$ is detected and the robot moved backwards. Then, the simulation is reset and a new episode is started.

1) POSE RETRIEVAL

The ground truth pose of the cube $\mathbf{p}_k$ is sensed by using the *gazebo_ros_state* plugin (defined in the Gazebo world file) which offers a *get_entity_state* service. This service is called by a timer callback at a specified communication rate $f_S = 60$ Hz, with time steps k every $L_S = 17.7$ ms. Once the response with the current pose arrives, it is delayed by

[1]https://github.com/eProsima/Fast-DDS

[2]https://classic.gazebosim.org/

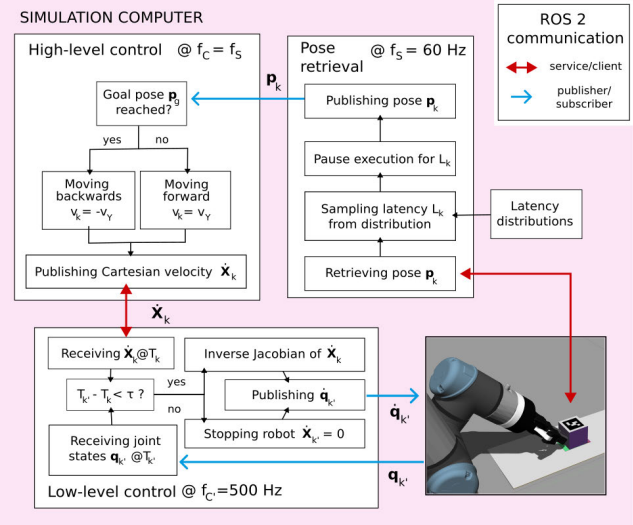


FIGURE 3. Control architecture of the simulated push actions allowing the visualization of the latency in the system regarding varying communication rates and networks.

a random sample L_k from a latency distribution by setting the `rclcpp::sleep_for(`L_k`)` function that pauses the execution of the thread for the specified duration L_k. Afterwards, the pose is published.

2) HIGH-LEVEL CONTROL

The high-level control is triggered in a subscription callback of the pose published by the pose retrieval node. The robot manipulator starts in a given initial configuration and starts progressing forward at a constant velocity v_y along the Y-axis. The initial pose and direction of motion are chosen, such that the gripper collides with the cube and, subsequently, pushes it forward. The 6D velocity $\dot{\mathbf{x}}_k$ is published on each time stamp, and the latest received pose $\mathbf{p}_k$ of the cube is compared with the goal value $\mathbf{p}_g$. When the cube reaches or surpasses $\mathbf{p}_g$, the robot starts moving backwards at $-v_y$.

3) LOW-LEVEL CONTROL

The low-level control is a velocity control that allows a smooth behavior under rapid direction changes of the robot motion and runs at 500 Hz with time step k' every 2 ms. It receives the Cartesian velocity $\dot{\mathbf{x}}_k$ and converts $\dot{\mathbf{x}}_k$ to joint velocities $\dot{\mathbf{q}}_k$ using the inverse of the Jacobian $\mathbf{J}$,

$$\dot{\mathbf{q}}_{k'} = \mathbf{J}_{k'}^{-1}(\mathbf{q}_{k'})\dot{\mathbf{x}}_k. \tag{1}$$

The Jacobian depends on the current joint configuration $\mathbf{q}_{k'}$. Note, that the low-level control runs independently of the high-level control. That means, once it receives $\dot{\mathbf{x}}_k \neq 0$, it keeps going at that velocity until it is reset to zero. To address this potential risk to the surroundings, the low-level controller measures the time since the last $\dot{\mathbf{x}}_{k-1}$ arrived. If that time exceeds the timeout τ, the low-level controller stops the robot until $\dot{\mathbf{x}}_k$ is received. Communications with large jitter can trigger this timeout, as large latency outliers can result in large latency between two consecutive messages.

D. IMPACT OF THE SENSING RATE

For each communication technology, the experiment is performed at three different sensing rates $f_S^{(1)} = 30$ Hz, $f_S^{(2)} = 60$ Hz, $f_S^{(3)} = 120$ Hz. The latency between publishing two consecutive images $L_S^{(1)} = 33.3$ ms, $L_S^{(2)} = 17.7$ ms, $L_S^{(3)} = 8.3$ ms, respectively. The resulting placement error e is shown in Fig. 4, where each box is obtained from $N = 200$ episodes. In general, the combination of faster network and higher sensing rate corresponds to lower errors. However, at a sensing rate of $f_S = 30$ Hz, there is not a significant difference between the communication technologies. Except for the private 4G, all other technologies produce the same median error of approximately 1.5 mm. This can be explained using the ratio $r = L_S/median(L)$ that relates the latency between two consecutive images and the median transmission latency of a communication technology. For the private 5G network, $r^{(1),5G} > 3$, for the private 4G network, $r^{(1),4G} > 1.7$. At 60 Hz, a performance decrease with private 4G and 5G is visible. Here, $r^{(2),5G} = 1.8$ and for the ideal Wi-Fi 5 network, $r^{(2),w5i} > 6$. This results in a median error of 0.8 mm, which is half compared to the results at 30 Hz and equal to the fastest technologies, Ethernet and private 5G URLLC. Only at 120 Hz, all communication technologies show a performance difference that would have been expected comparing the communication latency. Therefore, the sensing rate of the robotic system has to be taken into account when choosing a suitable network.

E. IMPACT OF THE LOW-LEVEL CONTROL TIMEOUT

For robotic systems that rely on a continuous stream of new information, networks with large jitter can pose safety risks. In our experiments the robot keeps running at the last received velocity. Large latency outliers can result in large errors when the reversed control command is delayed. When stopping the robot at such outliers, those large errors can be avoided. The experiment is performed at 60 Hz using the latency distribution of the loaded Wi-Fi 5. Figure 5 shows the dependency of placement error and execution time for four different timeouts τ that stop the robot at the low-level controller when a new packet does not arrive within τ. It can be observed that the timeout allows bounding the error, even narrowly to $e < 3$ mm, by reducing it to $\tau = 25$ ms. This, however, comes at the cost of an increased execution time, since the robot is stopped frequently and for longer duration. Nevertheless, the timeout can be specified such that the maximum error is reduced by more than 50% without impacting the execution time (see boxes at a timeout of $\tau = 400$ ms in Fig. 5).

F. CONCLUSION

This section showed that the sensing rate of the collaborative robotic system can have a significant impact on the reaction time and the resulting placement error depending on the magnitude of the communication latencies, such that a faster communication technology is only beneficial for the task

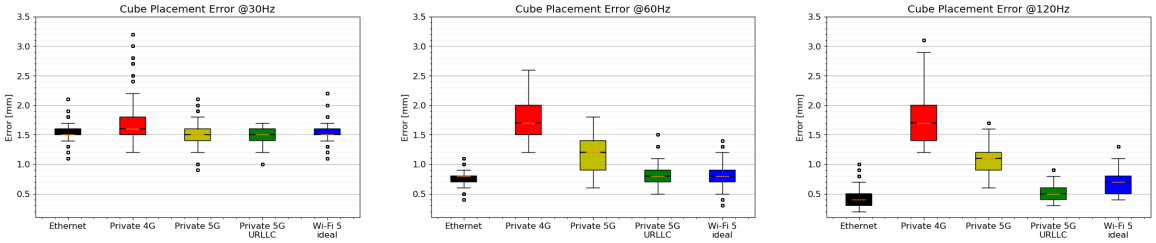


FIGURE 4. Impact of different sensing rates on the comparison of different communication technologies: only with higher sensing rates a difference of the push performance is significant.

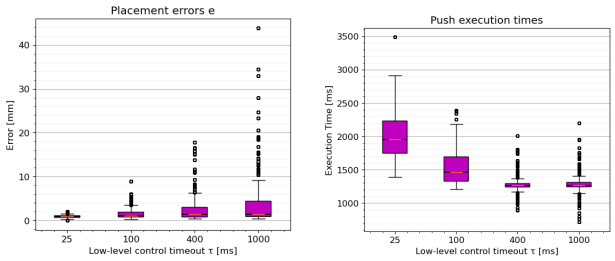


FIGURE 5. Impact of different low-level control timeouts τ on the placement error e and execution times for communication with large jitter (here the loaded Wi-Fi 5 measurements) at 60 Hz. A timeout of $\tau = 400$ ms balances both indicators well, decreasing the largest errors by half without increasing the execution time.

performance when the latency between two consecutive images is small enough. Additionally, a low-level control timeout is introduced which can prevent large performance drops due to large jitter without impacting the execution time.

IV. IMAGE PROCESSING AND STREAMING

Camera images are essential for the real experiments to track the position of the object during the push action or the hand of the human for the teleoperation. Different image settings have a significant influence on the reaction time in terms of computation and transmission times. On the other hand, the detection success is a vital prerequisite and high image resolution can improve the detection accuracy.

A. OpenCV

The image capturing and ArUco detection is performed using the OpenCV library [22]. The ArUco detection module takes an image of arbitrary resolution as input and outputs a list of all detected marker IDs and their respective poses. By defining a specific marker ID, it can be ensured that the same marker is found in consecutive images and noticed when it is not detected.

B. IMPACT OF IMAGE RESOLUTION AND COMPRESSION

The choice of image resolution and compression influences the detection success and precision, as well as the packet size and computation times of the ArUco detection. We run 1800 ArUco detections for each combination of: five different resolutions, compressed and uncompressed images, and three different distances (Fig. 6). The perspective is similar to the push actions experiment and allows to evaluate the magnitude

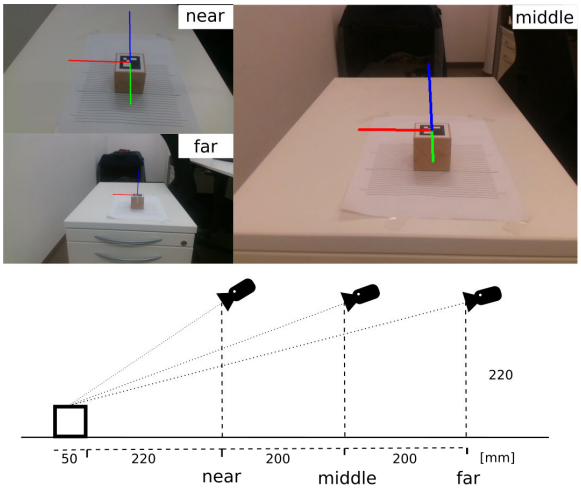


FIGURE 6. Setup for the ArUco detection comparison from three distances and different viewing angles.

TABLE 1. Pixel sizes at different image heights and distances between camera and ArUco marker.

Image resolution [px]	Pixel size [mm]			Detection rates [%]		
	near	middle	far	near	middle	far
320x240	0.8	2.0	5.2	100	62/94	0
640x480	0.4	1.0	2.6	100	100	25/4
960x540	0.4	0.9	2.3	100	100	92/100
1280x720	0.3	0.7	1.7	100	100	100
1920x1080	0.2	0.4	1.2	100	100	100

of the influencing factors. Table 1 states the pixel sizes of different resolutions and the rate of successful detections at all three views separately. It shows that the distance that a pixel represents in the real world depends on the image resolution and the relative size of the object in the image. Hence, the detection is more precise when the ArUco tag is closer to the camera and the resolution is high with each pixel representing 0.2 mm. The other extreme case results in a pixel representing 5.2 mm which is more than the 4.8 mm distance between two lines on the base of the cube.

If more precision is desirable, it comes at the cost of larger computation times and packet sizes, as shown in Table 2. The packet sizes of JPG-compressed images, on the other hand, decrease by an order of magnitude, but the computation times increase. The rates of successful detections do not differ between compressed and uncompressed images. To be able to make a final assessment, the impact of the packet size

TABLE 2. Packet sizes and computation times of ArUco detections for compressed (compr.) and uncompressed (uncompr.) images regarding varying resolutions. The bold values are used for the QoS measurements (see Table 3).

Image resolution [px]	Packet size [kB]		Computation times [ms]	
	compr.	uncompr.	compr.	uncompr.
320x240	17-**20**	245	1.9	1.3
640x480	59-**64**	**980**	4.9	2.3
960x540	103-**111**	1653	8.3	3.3
1280x720	169-**184**	**2938**	14.2	5.2
1920x1080	287-**308**	6610	35.7	11.7

TABLE 3. Communication latency (in milliseconds) for different packet sizes uplink and a constant 0.8 kB packet downlink comparing *reliable* and *best-effort* QoS profiles.

Packet size [kB] (Rate [Hz])	Reliable QoS latency		Best-effort QoS latency	
	Median	Max	Median	Max
20 (60)	4.05	105	4.07	28.6
64 (60)	7.68	176	7.60	32.1
111 (60)	10.1	74.1	9.80	33.5
184 (30)	17.3	121	16.6	41.5
308 (30)	21.7	136	19.9	43.2
980 (60)	40.9	459	33.2	57.9
2938 (30)	131	1230	99.9	150

on the communication latency is evaluated in the following subsection.

C. IMPACT OF ROS 2 QoS RELIABILITY PROFILES

ROS 2 allows setting QoS profiles, e.g. for the reliability. The default setting is the *reliable* communication. The publisher expects an arrival confirmation and retransmits missing samples. New packets are not accessible by the subscriber before older ones arrive with a default maximum blocking time of 100 ms [23]. Another reliability QoS profile utilizes *best-effort communication*. This profile does not make the subscriber wait for older packets which is expected to be beneficial for sensor feedback, as the system requires the latest packet to arrive as quickly as possible. Hence, in theory, the *best-effort* communication should be more suitable for reactive control experiments where only the latest packet represents the current state of the environment. To this end, not all packets are required to arrive if the latest packet arrives faster. To evaluate this approach, we perform *best-effort* and *reliable* communication measurements for different packet sizes. A constant packet size can be set arbitrarily by specifying the number of four-Byte elements of an integer vector within a custom ROS 2 message. In a 1000 Mbps Ethernet network, the integer vector is published from one device to an edge computer, which subsequently publishes a pose message of 0.8 kB that is received back at the device. The round-trip time is measured for 16 h in each configuration. The packet sizes are selected from Table 2 and are marked in bold font. The median and maximum round-trip latency for each packet size and QoS profile is given in Table 3. The median latency increases with an increasing packet size. The *best-effort* QoS consistently results in a lower maximum latency (less jitter). The median latency increases for the *reliable* QoS profile significantly only at larger packet sizes.

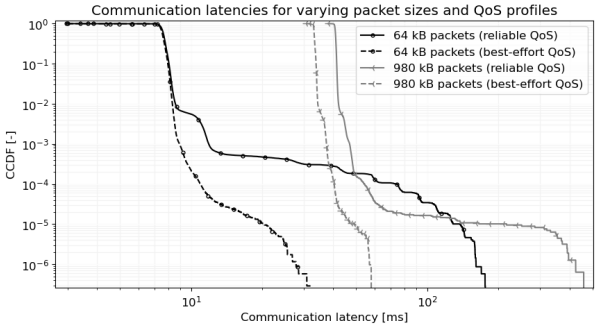


FIGURE 7. Comparing *reliable* and *best-effort* QoS profiles and packet sizes of compressed (64 kB) and uncompressed (980 kB) images in an Ethernet network. Communicating compressed images with a *best-effort* QoS profile results the shortest median latency and lowest jitter.

Figure 7 illustrates the distribution of the communication latency of 64 kB and 980 kB packets, which represent the compressed and uncompressed images with a resolution of 640 x 480 pixels. As shown in Table 3, the median communication latency is significantly lower for the 64 kB message (7.60 ms) compared to the 980 kB message, independent of the QoS profile. Communicating the 980 kB message with the *reliable* QoS profile, the median latency increases by 23% compared to the *best-effort* QoS profile. For the 64 kB message, the median latency only increases marginally by 1%. Comparing the QoS profiles, the *best-effort* communication reduces the jitter as expected. As a result, the *best-effort* QoS profile reduces the occasions where the low-level control timeout is triggered, resulting in a smoother robot motion.

D. CONCLUSION

This section showed the impact of image resolution and compression on communication times and detection success. The results indicate that it is meaningful to find the smallest image resolution where the object detection algorithm is reliable which depends on the object size in the image. Then, compress the image such that the combined communication and computation latency can be minimal. A best-effort QoS policy further allows reducing the jitter, as expected, since it reads the latest packet without waiting first for older ones.

V. DISTRIBUTED CONTROL SYSTEM OVER 5G

Compared to the simulated experiment, the setup of the distributed control architecture utilizes two computers, the onboard robot computer (1) and the edge computer (2) (see Fig. 8). The onboard robot computer is connected to the UR5e manipulator (3) via Ethernet and communicates with the edge server wirelessly via 5G modem (5). The 5G modem and the base station are in line-of-sight. The 5G modem and the RealSense camera (4) are connected to the robot computer via USB. The onboard robot computer is a laptop with an 8-core *Intel(R) i7-1165G7* CPU @ 2.80 GHz. The edge server performs computations on a 56-core *Intel(R) Xeon(R) E5-2680 v4* CPU @ 2.40 GHz.

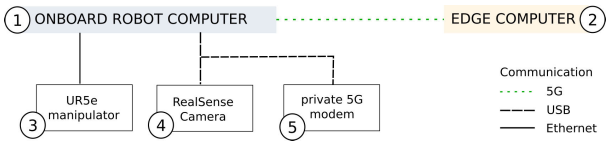


FIGURE 8. Schema of the experimental setup of the real distributed control experiments.

The most suitable parameters for the distributed control experiments are chosen with regards to the previous results, capturing images with a resolution of 640×480 pixels at 60 Hz and communicating them compressed with the *best-effort* QoS profile.

A. PRIVATE 5G NETWORK CONFIGURATIONS

The real distributed control experiments are performed at the *Smart Production Lab* at the Aalborg University (AAU) [19], [20]. It provides a private, standalone 5G network (release 15) with an on-site NDAC Mxie 5G core (medium+) and a Radio Access Network (RAN) that consists of a Nokia airscale baseband unit and 3 Nokia airScale indoor Radio (ASiR) configured with 100 MHz of bandwidth in the N78 band (3710 MHz), 3/7 uplink/downlink Time Division Duplex (TDD) ratio, and as a single frequency network, such that all 3 ASiR broadcast the same cell. This is an optimization for extremely fast handover when moving between coverage areas of the ASiR, at the cost of capacity, since all 3 ASiR share capacity. On the backhaul the 5G are routed through a 10 Gbps infrastructure.

B. ROS 2 COMMUNICATION VIA TWO NETWORKS

The private 5G network at the AAU Smart Production Lab is separate from the Ethernet/Wi-Fi network. As a consequence, the node that is running at the edge server cannot discover the 5G connected robot computer and vice versa. To deal with this, the eProsima Fast DDS allows setting up a peer discovery by specifying the IP addresses of both computers in an XML file [24].

C. ROBOTIC PUSH ACTIONS

A first distributed control experiment is the robotic push of a cube with an ArUco marker (Fig. 9). The control architecture is illustrated in Fig. 10. Compared to the simulated experiment, an additional node in the control loop reads the image $\mathbf{M}_k$ from the camera and publishes it at time $T_{1,k}$ for the edge computer via the private 5G wireless network. On the edge computer, $\mathbf{M}_k$ is received at $T_{2,k}$ and the ArUco detection is performed returning a list of poses $\mathbf{P}_k$ which is subsequently published at $T_{3,k}$ for the high-level control, onboard the robot computer, where $\mathbf{P}_k$ is received at $T_{4,k}$. The duration $L_{14,k} = T_{4,k} - T_{1,k}$ represents the combined communication and computation latency, and $L_{23,k} = T_{3,k} - T_{2,k}$ only the computation latency. The cube pose $\mathbf{p}_k \in \mathbf{P}_k$ corresponding to a specific tag ID is expected. Before running the push action, the goal pose $\mathbf{p}_g$ is recorded using a service call. Then, the cube is placed in front of

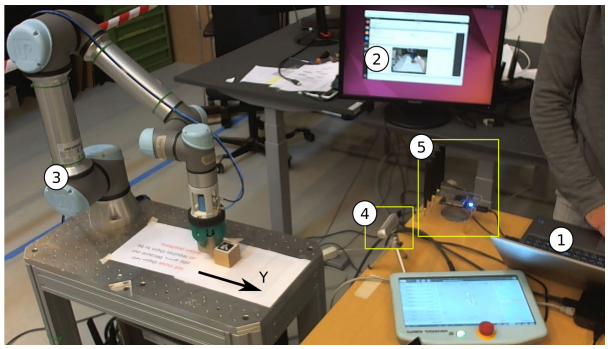


FIGURE 9. Setup of the distributed robotic control experiments to visualize the latency of the system with elements (1) to (5) as presented in Fig. 8.

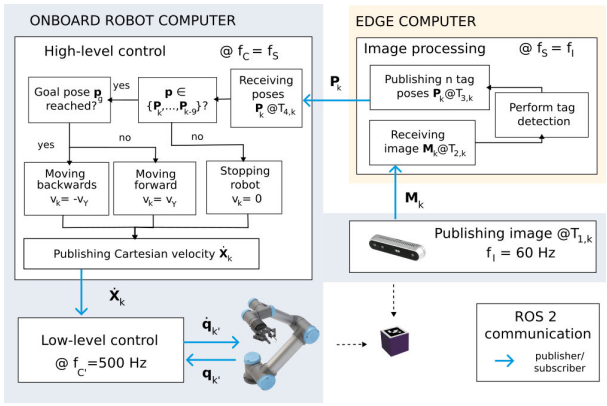


FIGURE 10. Distributed control architecture of real robotic push actions.

the gripper and another service starts the push experiment. The publisher/subscriber communication to the low-level control replaces here the service/client communication in the simulated experiment to be able to keep the best-effort communication in the entire control loop.

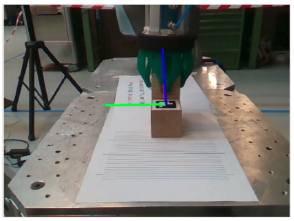
We run $N = 26$ push episodes with a low-level control timeout of 100 ms and a constant velocity $v_Y = 0.05$ m/s which is half compared to the simulated experiment.

1) HANDLING MISSING DETECTIONS

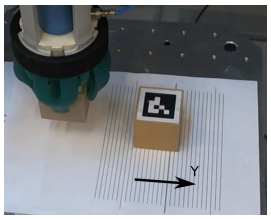
Unlike in the simulated experiment, the detection of the desired tag is not certain. Thus, a safety mechanism is added to stop the robot in case the expected tag is not among the last ten detected tags. Even though, the detection rate in the isolated experiment was 100% (see Table 1), the distributed control experiment challenged the detection algorithm especially due to the robot gripper that cast a dynamic shadow over the tag (Fig. 11). More details in the image also result in 50% larger packet sizes (91-99 kB) for the 640×480 pixel images, compared to the experiments in Fig. 6.

2) PLACEMENT ERROR

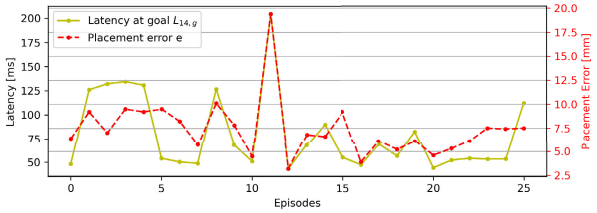
The placement error depends on the combined communication and computation latency of the first image that captures the cube passing the goal pose $\mathbf{p}_g$. This *Latency at goal*



(a) ArUco detection for the push experiment.



(b) Observation of the actual error $\hat{e}$ in episode 14.



(c) Relation of detected placement error e and latency $L_{14,g}$.

FIGURE 11. ArUco detection results and placement errors.

$L_{14,g}$ varies from 38.4 ms to 222 ms and is reflected in the placement error (Fig. 11c). The median computation latency is $L_{23} = 10.6$ ms. However, the correspondence between the detected placement error e and the latency is not always consistent, e.g. from episode 14 to 15, the latency decreases but the error increases. We suppose that the lighting conditions and the angled perspective not only increase the occasions of missing detections but also affect the detection accuracy. This can be observed when comparing e with the actual error $\hat{e}$, visually obtained by using another camera perspective (Fig. 11b). For instance, the actual error of twelve episodes (14-25) is five times within 1 mm, once larger than detected and six times smaller than detected. In episode 14, the detected placement error e is 2 to 3 mm less than the actual error $\hat{e}$ (see Fig. 11b), in episode 15, e is within 1 mm of $\hat{e}$. These results illustrate that in a real experiment there are uncertainties, like lighting conditions, that blur the effect of the latency on the placement error.

3) REACTION TIME

A detailed view of episode 18 is shown in Fig. 12 to evaluate the overall reaction time of the system. The triangular markers illustrate the time stamps $T_{1,k}$ when an image is published (A), the circles illustrate the time stamps $T_{4,k}$ when a pose is received (B). $L_{1,k}$ and $L_{4,k}$ represent the duration between two time stamps k and $k-1$ for T_1 and T_4, respectively. The time stamps at a missing pose detection are filled with red. The first detected pose reaching the goal pose is highlighted in green. At each received list of poses $\mathbf{P}_k$ ($@T_{4,k}$), the velocity $v_k = \pm v_Y$ is set for the linear motion along Y (C). This velocity eventually results in a position change of the end-effector (D). This experiment simulates a sudden change in a collaborative environment as the robot moves at a constant speed until one event reverses the robot motion. We define the duration from the image that displays the passing of the goal until the reversion of the motion as the overall reaction time of the system, L_R. It contains the combined communication

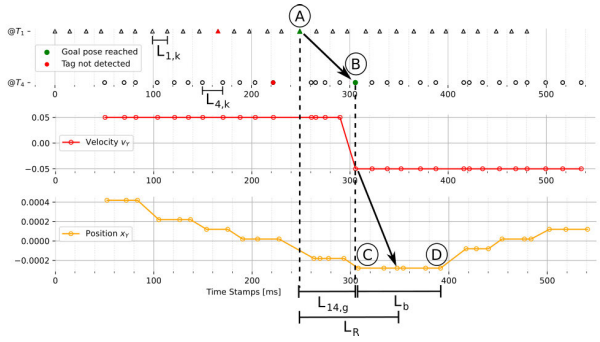


FIGURE 12. Tracing the reaction time L_R of one push experiment episode. Displayed are 30 time stamps around the target event (B) at $T_{1,k}$ and $T_{4,k}$ (see Fig. 10). The sensing rate f_S is represented by $L_{1,k}$, and $L_{4,k}$ indicates the duration between calls to the low-level control.

and computation time $L_{14,g}$ and half of the time for reversing the motion L_b which is taken between the last measure point forward and the first measurement backwards.

4) TIMEOUT BEHAVIOR

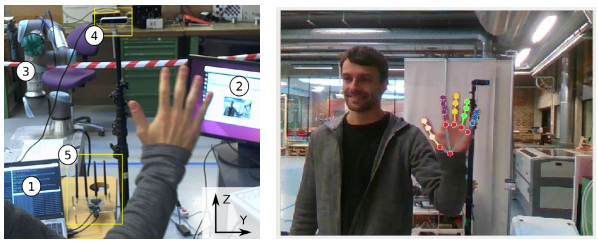
With $L_{1,k}$, the actual sensing rate $\hat{f}_S$ can be observed. In episode 18, the average over all $L_{1,k}$ is 16.7 ms ($\hat{=}$60 Hz) which equals the specified sensing rate $f_I = f_S = 60$ Hz. Even though, the average over all $L_{4,k}$ is also around 60 Hz, the individual variations vary slightly resulting from the jitter in the communication. Since the received pose is further processed onboard the robot computer, this jitter is transferred to the low-level control where the timeout is triggered in case the jitter exceeds a threshold $\tau = 100$ ms. During the 26 episodes, the timeout is not triggered once. However, the communication appears unstable in episode 11. In this episode, larger latencies result in a reduced sensing rate of about $\hat{f}_S = 40$ Hz and the largest placement error during the experiment, visible in Fig. 11c. To test the timeout trigger, we perform 9 additional episodes with a timeout set to $\tau = 50$ ms. Here, the robot is forced to stop in 8 out of 9 episodes between one and three times. These few and short stops are recognizable by the audible sound when the robot breaks but not by the motion speed. This finding supports the simulation results in Fig. 5 where a balanced low-level control timeout reduces the placement error outliers without affecting the execution time.

D. ROBOTIC TELEOPERATION USING HAND MOTION

In a second experiment, we teleoperate the robot such that the end-effector mirrors the human hand motion in 2D (Fig. 13). Compared to the push experiment, the same settings are used (JPG-compressed 640 × 480 pixel images at 60 Hz and *best-effort* QoS profile), but the detection algorithm and the high-level control node are different, as illustrated in Fig. 14.

1) MEDIAPIPE HAND LANDMARKS DETECTION

The MediaPipe hand landmarks detection is used [25] to track the human hand. It takes an image of arbitrary resolution as input and outputs a maximum of 21 hand landmark positions per hand, if an entire human hand is detected. It allows setting



(a) Teleoperation setup according to Fig. 8.

(b) Hand landmarks detection by MediaPipe [25].

FIGURE 13. Application of distributed control in the teleoperation experiment to make the manipulator mirror the hand motion in 2D.

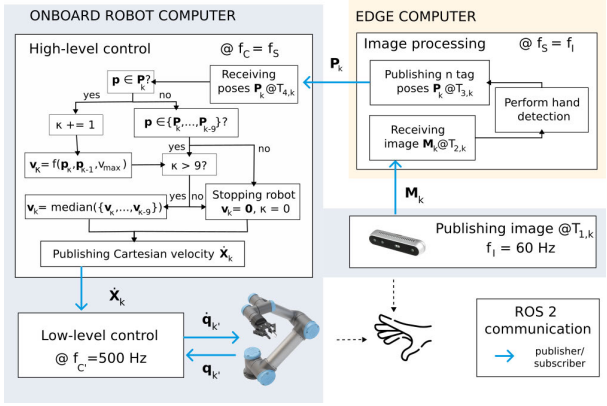


FIGURE 14. Distributed control architecture for the teleoperation experiment.

a maximum number of hands to be detected in an image, but there is no functionality that ensures the same hand is detected in consecutive images.

2) TELEOPERATION CONTROL STRUCTURE

After running the hand landmarks detection on the edge computer and setting the landmark positions in pose messages, the high-level control receives the detected poses. If the hand detection is complete (21 hand landmark positions), we add one to a counter κ and take the first pose $\mathbf{p}_k = \mathbf{P}_{k,0}$ which represents the hand landmark at the wrist. Based on the difference between two consecutive poses, we compute a velocity v_κ for each of the two directions Y and Z,

$$v_\kappa = \begin{cases} F(\mathbf{p}_\kappa - \mathbf{p}_{\kappa-1})/f_{C'}, & \text{if } v_\kappa < v_{\max} \\ v_{\max}, & \text{otherwise,} \end{cases} \tag{2}$$

where $F = 100$ is a factor that scales the speed of the robot motion, $f_{C'} = 500$ Hz is the low-level control rate and $v_{\max} = 0.2$ m/s bounds the robot velocity. To smooth the robot motion, we take the median of the last ten computed velocities if $\kappa > 9$. If the landmark is not detected within the last ten messages, the robot is stopped for safety reasons, similar to the push experiment, and κ is reset to zero.

3) REACTION TIME

Compared to the push experiment, the teleoperation reacts at each step k to the changes of the human hand, and the hand detection requires more than an order of magnitude

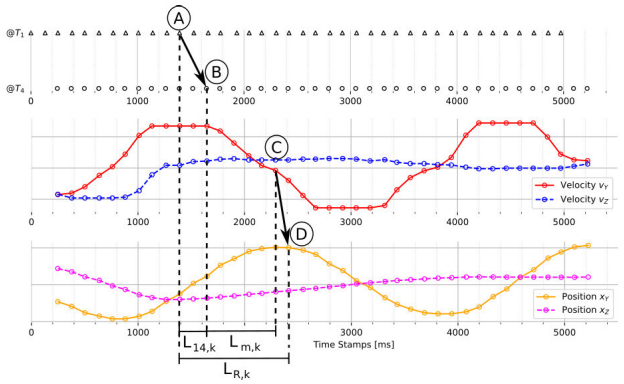


FIGURE 15. Illustration of the reaction time $L_{R,k}$ in the teleoperation experiment from image capturing (A), to pose receiving (B), to setting a high-level control velocity (C) to the actual motion of the manipulator's end-effector (D).

longer computation times ($L_{23} = 125$ ms). This seems to saturate the communication such that the specified frequency $f_I = 60$ Hz cannot be kept. As a consequence, the control loop runs only at $\hat{f}_I = 7.5$ Hz. Figure 15 presents a five-seconds section of the experiment where the human hand and the robot end-effector move right-left-right from the observation perspective (Fig. 13). The reaction time $L_{R,k} = 1$ s is approximated by empirically measuring the time between a frame where the human hand changes the motion direction (A) and the corresponding frame where the robot end-effector changes the motion direction (D). From here, the image processing time $L_{14,k}$ on the edge server is known and the time stamp where the velocity sign changes can be marked (C). In between, the duration $L_{m,k}$ results from the smoothing operation which requires five negative v_k to output a negative v_k due to the *median* operation. This appears reasonable as the output is descending for five points. This experiment shows that a vision-based teleoperation is a challenging application for a communication and computation infrastructure due to the requirements of communicating large image packets, performing sophisticated image processing and fast reaction times.

VI. DISCUSSION AND FUTURE WORK

The implementation of the real distributed control system revealed limitations that are discussed in the following, including various directions for future work.

A. ROBOTIC SIMULATIONS WITH COMMUNICATION HARDWARE IN THE LOOP

In the simulated robotic push experiment, we show the impact of the sensing rate on different communication technologies. When communicating small packets every 33 ms (30 Hz), the performance did not differ significantly among communication technologies. However, larger packets result in longer communication latencies L for all technologies. This decreases the ratio $r = L_S/median(L)$, such that it is expected that also at 30 Hz a performance difference can be observed. This relation can be investigated more realistically using communication hardware-in-the-loop [16] with robotic

simulations. As shown in this work, robotic simulations and communication experiments can be performed extensively, while real robotic experiments are more limited in the number of experiment rollouts.

B. LARGE-SCALE WIRELESS COMMUNICATION CHALLENGES

The image processing and streaming experiments indicate the benefit of the *best-effort* QoS for Ethernet networks. The bi-directional Ethernet connection allows for equal conditions in the long-term experiment. However, especially in less reliable wireless networks, the *best-effort* QoS profile is expected to be beneficial as it does not wait for each packet to arrive. In that sense, measuring the real impact on loaded Wi-Fi networks would be interesting, as well as a detailed comparison of Wi-Fi 6 and private 5G networks. Furthermore, the experiments show that the image compression using JPG-encoding reduce the network load by one order of magnitude without affecting the detection performance. However, the varying packet sizes of compressed images might complicate the resource allocation strategies, since the exact required data rate is not known beforehand. To further reduce the load, we consider implementing mechanisms to stop publishing images when they are not needed at all, or publishing them at a lower frequency when the application is not dynamic. The modular architecture of the experiments could enable a dynamic and flexible execution of tasks, completely on the edge or on the mobile device, or a hybrid solution, depending on the availability of communication and computation resources. This flexibility is expected to be beneficial particularly in large-scale scenarios where multi-robot deployments require multiple bi-directional communication instances between the robots and the edge computer. A suitable resource allocation strategy might then assign communication and computation resources dynamically to optimize the Quality of Experience (QoE) for all users [26]. For such large-scale scenarios, the utilized ROS 2 based unicast communication is not a practical solution, but the open-source middleware is developed towards better applicability in real industrial scenarios, for instance, a ROS extension, named Connected Robotics Platform (CROP), that allow the deployment in large-scale distributed systems has already been proposed [27].

C. NON-LINE-OF-SIGHT CONSIDERATIONS

The distributed control experiments were performed in a line-of-sight (LOS) scenario at the 5G Smart Production Lab at AAU where the 5G infrastructure was designed to avoid non-line-of-sight (NLOS) scenarios. Hence, most locations in the lab are in LOS. However, NLOS situations can occur in a dynamic and flexible production environment, and especially the use of higher frequency bands in the future will be more sensitive to NLOS. To further increase LOS conditions, reconfigurable intelligent surfaces (RIS) [28] can be installed in the industrial environment, or UAV-enabled relaying can be utilized [29].

D. EDGE COMPUTING VERSUS ONBOARD COMPUTING

In the distributed control experiments, we offload the image processing to an edge server. This makes sense, since it can result in significant time gains when the computation resources are larger on the edge computer than on the mobile device [10]. At the same time, it requires fast uplink data transmission of large images to avoid losing that time gain. In our case, the time gain of a few milliseconds using the edge computer was not significant enough to reduce the overall reaction time of the system due to the larger communication latency required to transmit images to the edge server. To achieve this gain in future works, we aim at using more complex detection algorithms and an edge computer with GPU support, as demonstrated in [17] and also connected it directly to the private 5G network. Another motivation for the use of edge computing could be the opportunity to run the most critical computations onboard, e.g. regarding human safety, and offload less critical tasks to the edge (e.g. autonomous object manipulation). In that case, the mobile manipulator can perform its task reliably without interruptions in case of approaching humans. Alternatively, external sensors could observe the shared workspace independently of the mobile robots and be connected by cable to the edge computer.

E. COLLABORATIVE ROBOTICS AS B5G/6G USE CASE

The teleoperation experiment illustrates the challenge of collaborative robotics requiring intensive computations and fast reaction times, which makes it a suitable use case for B5G/6G projects [30]. We used the same parameters for the reaction time factors as for the push experiment. Adjusting the parameters for each application might improve the performance, e.g. the detection success rate could be good enough with even smaller image resolution and the possible network saturation might be mitigated using a sensing rate slower than 60 Hz. Sensing, computing and communication are vital parts of our experiments which could make it an interesting application for the upcoming *Integrated Sensing, Computing and Communication* (ISCC) technology in 6G. The smoothing of the robot motion is a design choice that impacts the reaction time in our experiment significantly. However, it reduces sudden accelerations of the robot which addresses the ''psychological safety'' of humans [31]. Human motion prediction could allow mitigating the effect of the reaction time by anticipating the human behavior ahead of time [5], e.g. predicting the human pose 80 to 1000 ms into the future [32].

F. APPLICABILITY TO OTHER ROBOTIC USE CASES

This work focuses on collaborative robotics as an industrial application for 5G edge computing which requires the consideration of human safety in a shared environment with autonomous robots. Contact-rich object manipulation tasks, such as grasping and handling deformable objects, wiping, screwing, or cable routing also benefit from quick reaction times and require sophisticated perception and data

processing. The use of fiducial markers is convenient and the robustness to lighting conditions can be improved using a deep learning approach [33], however, occlusions and the requirement of placing a tag on the object make many use cases not feasible. Instead, recent advances in 6D pose estimators [34], [35] promise to increase the applicability to more robotic use cases. On the other hand, also learning-based robot control methods using deep reinforcement learning [36] have shown to significantly improve robotic skills, which have the potential to not only advance the automation of industrial processes but also to be applied in service robotics. Here, the comparison to Wi-Fi would be interesting again, since it is widely deployed in private households where GPU-accelerated gaming PCs could serve as edge computers to perform the image processing and high-level control. At the same time, it poses critical safety challenges, as private homes have more NLOS conditions and will likely not have camera surveillance in every room, so the PFL collaborative safety mode must be used.

VII. CONCLUSION

In this work, we consider multiple important factors that impact the reaction time in distributed collaborative robotic systems (sensing rate, communication technology, communication QoS settings, image resolution and compression, and computation latency). The experiments combine simulations and isolated communication and image processing experiments that allow the assessment of the statistical significance. As a result, we implement a complete distributed control system for collaborative robotics with 5G edge computing using the highest available sensing rate of 60 Hz and the lowest meaningful image resolution of 640×480 pixels which is transmitted to the edge server JPG-compressed using the ROS 2 best-effort reliability QoS profile. Additionally, the closed-loop control system includes safety mechanisms in cases of detection failures and large jitter showing that a balanced low-level control timeout avoids large errors without compromising the execution time. At the end, the evaluation results are thoroughly discussed with particular attention to the challenges of multi-robot deployments in industrial settings. All experiments are implemented with open-source libraries using ROS 2 Humble and the code is made publicly available.[3]

REFERENCES

[1] *Robotics—Vocabulary*, Standard ISO 8373:2021(en), International Organization for Standardization, 2021.

[2] D. Urbaniak, J. Rosell, and R. Suárez, "Edge computing in autonomous and collaborative assembly lines," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–4.

[3] *Robots and Robotic Devices—Collaborative Robots*, Standard ISO/TS 15066:2016, International Organization for Standardization, 2016.

[4] A. Vysocky and P. Novak, "Human-robot collaboration in industry," *MM Sci. J.*, vol. 9, no. 2, pp. 903–906, 2016.

[5] J. A. Marvel and R. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robot. Comput.-Integr. Manuf.*, vol. 44, pp. 144–155, Apr. 2017.

[3]https://github.com/DominikUrbaniak/ros2_distributed_control_system

[6] C. Byner, B. Matthias, and H. Ding, "Dynamic speed and separation monitoring for collaborative robot applications—Concepts and performance," *Robot. Comput.-Integr. Manuf.*, vol. 58, pp. 239–252, Aug. 2019.

[7] J. B. Martin and F. Moutarde, "Real-time gestural control of robot manipulator through deep learning human-pose inference," in *Proc. Int. Conf. Comput. Vis. Syst.*, Thessaloniki, Greece. Cham, Switzerland: Springer, Sep. 2019, pp. 565–572.

[8] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1302–1310.

[9] C. Zheng, W. Wu, C. Chen, T. Yang, S. Zhu, J. Shen, N. Kehtarnavaz, and M. Shah, "Deep learning-based human pose estimation: A survey," *ACM Comput. Surv.*, vol. 56, no. 1, pp. 1–37, 2023.

[10] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini, and D. Schnieders, "Edge computing in 5G for drone navigation: What to offload?" *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2571–2578, Apr. 2021.

[11] F. Voigtländer, A. Ramadan, J. Eichinger, C. Lenz, D. Pensky, and A. Knoll, "5G for robotics: Ultra-low latency control of distributed robotic systems," in *Proc. Int. Symp. Comput. Sci. Intell. Controls (ISCSIC)*, Oct. 2017, pp. 69–72.

[12] T. Raunholt, I. Rodriguez, P. Mogensen, and M. Larsen, "Towards a 5G mobile edge cloud planner for autonomous mobile robots," in *Proc. IEEE 94th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2021, pp. 1–5.

[13] M. Balogh, A. Vidács, G. Fehér, M. Maliosz, M. Á. Horváth, N. Reider, and S. Rácz, "Cloud-controlled autonomous mobile robot platform," in *Proc. IEEE 32nd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2021, pp. 1–6.

[14] K. Nikhileswar, K. Prabhu, D. Cavalcanti, and A. Regev, "Time-sensitive networking over 5G for industrial control systems," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–8.

[15] G. Damigos, T. Lindgren, and G. Nikolakopoulos, "Toward 5G edge computing for enabling autonomous aerial vehicles," *IEEE Access*, vol. 11, pp. 3926–3941, 2023.

[16] H. Lv, Z. Pang, K. Bhimavarapu, and G. Yang, "Impacts of wireless on robot control: The network hardware-in-the-loop simulation framework and real-life comparisons," *IEEE Trans. Ind. Informat.*, vol. 19, no. 9, pp. 9255–9265, Sep. 2023.

[17] U. A. Zahidi, A. Khan, T. Zhivkov, J. Dichtl, D. Li, S. Parsa, M. Hanheide, G. Cielniak, E. I. Sklar, S. Pearson, and A. Ghalamzan-E, "Optimising robotic operation speed with edge computing via 5G network: Insights from selective harvesting robots," *J. Field Robot.*, Jul. 2024. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/rob.22384

[18] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Sci. Robot.*, vol. 7, no. 66, May 2022, Art. no. eabm6074, doi: 10.1126/scirobotics.abm6074.

[19] I. Rodriguez, R. S. Mogensen, A. Schjørring, M. Razzaghpour, R. Maldonado, G. Berardinelli, R. Adeogun, P. H. Christensen, P. Mogensen, O. Madsen, C. Møller, G. Pocovi, T. Kolding, C. Rosa, B. Jørgensen, and S. Barbera, "5G swarm production: Advanced industrial manufacturing concepts enabled by wireless automation," *IEEE Commun. Mag.*, vol. 59, no. 1, pp. 48–54, Jan. 2021.

[20] I. Rodriguez, R. S. Mogensen, A. Fink, T. Raunholt, S. Markussen, P. H. Christensen, G. Berardinelli, P. Mogensen, C. Schou, and O. Madsen, "An experimental framework for 5G wireless system integration into Industry 4.0 applications," *Energies*, vol. 14, no. 15, p. 4444, Jul. 2021.

[21] R. S. Mogensen, I. Rodriguez, G. Berardinelli, A. Fink, R. Marcker, S. Markussen, T. Raunholt, T. Kolding, G. Pocovi, and S. Barbera, "Implementation and trial evaluation of a wireless manufacturing execution system for Industry 4.0," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–7.

[22] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools J. Softw. Tools Prof. Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

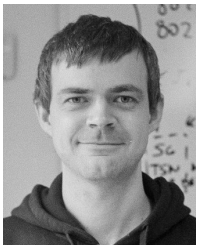
[23] (2019). *eProsima FastDDS Documentation*. eprosima.com. Accessed: Oct. 4, 2024. [Online]. Available: https://fast-dds.docs.eprosima.com/en/latest/fastdds/dds_layer/core/policy/standardQosPolicies.html#reliabilityqospolicy

[24] A. Mehmood. (2023). *Setting Up Node Discovery Across Multiple Systems in ROS2 Infrastructure*. medium.com. Accessed: Oct. 4, 2024. [Online]. Available: https://medium.com/@arshad.mehmood/setting-up-node-discovery-across-multiple-systems-in-ros2-infrastructure-a1a5c25f052f

[25] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe hands: On-device real-time hand tracking," 2020, *arXiv:2006.10214*.

[26] A. Sarah, G. Nencioni, and M. M. I. Khan, "Resource allocation in multi-access edge computing for 5G-and-beyond networks," *Comput. Netw.*, vol. 227, May 2023, Art. no. 109720.
[27] A. L. Ibañez. (2024). *Connected Robotics Platform*. Accessed: Oct. 4, 2024. [Online]. Available: http://wiki.ros.org/Connected%20Robotics%20Platform
[28] M. Di Renzo, A. Zappone, M. Debbah, M.-S. Alouini, C. Yuen, J. de Rosny, and S. Tretyakov, "Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 11, pp. 2450–2525, Nov. 2020.
[29] A. Ranjha, D. Naboulsi, M. El Emary, and F. Gagnon, "Facilitating URLLC vis-á-vis UAV-enabled relaying for MEC systems in 6-G networks," *IEEE Trans. Rel.*, early access, Feb. 9, 2024, doi: 10.1109/TR.2024.3357356.
[30] C. D. Alwis, A. Kalla, Q.-V. Pham, P. Kumar, K. Dev, W.-J. Hwang, and M. Liyanage, "Survey on 6G frontiers: Trends, applications, requirements, technologies and future research," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 836–886, 2021.
[31] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," *Found. Trends Robot.*, vol. 5, no. 4, pp. 261–349, 2017.
[32] T. Ma, Y. Nie, C. Long, Q. Zhang, and G. Li, "Progressively generating better initial guesses towards next stages for high-quality human motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 6427–6436.
[33] R. Berral-Soler, R. Muñoz-Salinas, R. Medina-Carnicer, and M. J. Marín-Jiménez, "DeepArUco: Marker detection and classification in challenging lighting conditions," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.* Cham, Switzerland: Springer, 2023, pp. 199–210.
[34] R. L. Haugaard, F. Hagelskjær, and T. M. Iversen, "SpyroPose: SE(3) pyramids for object pose distribution estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2023, pp. 2082–2091.
[35] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "FoundationPose: Unified 6D pose estimation and tracking of novel objects," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 35, Jun. 2024, pp. 17868–17879.
[36] I. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, "A review on reinforcement learning for contact-rich robotic manipulation tasks," *Robot. Comput.-Integr. Manuf.*, vol. 81, Jun. 2023, Art. no. 102517.

DOMINIK URBANIAK received the B.Sc. degree in engineering science and the M.Sc. degrees in robotics, cognition, and intelligence from the Technical University of Munich (TUM), in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with the Automatic Control, Robotics and Computer Vision Program, BarcelonaTech (UPC). He is an Early Stage Researcher (ESR) in the MSCA-ITN Project 5GSmartFact funded by EU, in which scope he completed an 18-month secondment with Roboception GmbH, Munich. His research interests include task and motion planning, computer vision, artificial intelligence, collaborative robotics, and wireless communication, and their interdisciplinary connection to enable the development of solutions for real-world problems.



SEBASTIAN BRO DAMSGAARD received the B.Sc. degree in internet technologies and computer systems and the M.Sc. degree in networks and distributed systems from Aalborg University, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree, with a focus on "5G Enabled Autonomous Mobile Robotic Systems." Previously, he was with Support and Deployment of IT Systems for Manufacturing, Grundfos. His research interests include the application of wireless communication for Industry 4.0, (edge) cloud computing for industrial applications, and optimizing communication for use in mobile manufacturing equipment.

WEIFAN ZHANG received the B.S. degree from Henan University, in 2014, and the M.S. degree from Fuzhou University, in 2018. Currently, he is pursuing the Ph.D. with Aalborg University, Aalborg, Denmark. From 2018 to 2020, he was with Huawei Technologies Company Ltd. His research interests include cloud computing and the IIoT communication.

JAN ROSELL received the B.S. degree in telecommunication engineering and the Ph.D. degree in advanced automation and robotics from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1989 and 1998, respectively. He joined the Institute of Industrial and Control Engineering (IOC), in 1992, where he has developed research activities in robotics. He has been involved in teaching activities in automatic control and robotics as an Assistant Professor, since 1996, and an Associate Professor, since 2001. His current technical areas include task and motion planning, mobile manipulation, and robot co-workers.

RAÚL SUÁREZ (Senior Member, IEEE) received the degree in electronic engineering from the National University of San Juan, San Juan, Argentina, in 1984, and the Ph.D. degree in electronic engineering from Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1993. He was responsible for the research line on process control with the Institute of Industrial and Control Engineering (IOC), from 1998 to 2003. He was the Deputy Director of IOC, from 2003 to 2009, and the Director, from 2009 to 2016. In 2016, the responsible of the División of Robotics, IOC. He was the Coordinator of the Ph.D. Program on Robotics, UPC, from 1995 to 2023, and the President of the IEEE RAS Spanish Chapter, from 2019 to 2023. His current position of a Research Supervisor with IOC, UPC. He is the co-author of about 100 scientific papers published in international conference proceedings and journals and he has participated in 27 competitive research projects (National and European) being principal investigator in 12 of them. His research activity is mainly focused on intelligent robotics, particularly in the areas of motion planning, human-like movements, robotized grasping, and dexterous manipulation.

MICHAEL SUPPA received the master's and Ph.D. degrees from the University of Hannover, Germany, in 2000 and 2007, respectively. From 2000 to 2009, he was employed as the Project Manager and a Researcher with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR). In 2009, he was appointed as the Head of the Perception and Cognition Department, Institute of Robotics and Mechatronics, and in 2014, he was additionally named the Deputy Institute Director. In 2015, he co-founded Roboception, a DLR spin-off company. He is currently the Co-Founder and the CEO of Roboception GmbH, a Munich-based start-up company providing comprehensive 3D perception solutions for robotic applications. He is a Honorary Professor with the University of Bremen. His key research interest includes perception-driven applied AI. He has been the PI in several European and national projects and was nominated for and received several best paper awards.